UNIVERSITY OF CALIFORNIA

Santa Barbara

Recursively Adaptive Randomized Multi-Tree Coding (RAR MTC) of Speech with VAD/CNG

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in

Electrical and Computer Engineering

by

Hoontaek Oh

Committee in charge:

Professor Jerry D. Gibson, Chair

Professor Kenneth Rose

Professor Katie Byl

Professor Ramtin Pedarsani

Dr. John Treichler

Dr. David Detienne

June 2023

The dissertation of Hoontaek Oh is approved.

_____

Kenneth Rose

_____

Katie Byl

_____

Ramtin Pedarsani

_____

Dr. John Treichler

_____

Dr. David Detienne

_____

Jerry D. Gibson, Committee Chair

April 2023

Recursively Adaptive Randomized Multi-Tree Coding (RAR MTC)

of Speech with VAD/CNG

by

Hoontaek Oh

This dissertation is dedicated to:

my family, the most precious beings to me in the world.

My father, Si-Hak Oh

My mother, Hwa-Ja Lee

My older sister, Jinhee Oh

Acknowledgements

I would like to sincerely appreciate my advisor, Professor Jerry D. Gibson, for his warm support and insightful guidance throughout my doctoral research. Even though my research project was quite challenging at first because I had to follow up on the research decades ago, his feedback based on rich expertise enabled me to successfully finish it with the great result without losing direction. In addition, his endless passion for new research has strongly reminded me to think of the attitude a good researcher is supposed to have.

Furthermore, I am very thankful to Raytheon for supporting my research during these years. The support has motivated me to consistently keep focused on a quite challenging long-term research project as an international graduate student, and I can truly see how much Raytheon values the efforts of researchers pursuing performance beyond limits.

I thank my committee members, Prof. Rose, Prof. Byl, Prof. Pedarsani, Dr. John Treichler, and Dr. David Detienne for serving on my committee in a hard situation under COVID-19, and for their precious feedback on my dissertation.

Most of all, I would like to thank my family. My parents and sister are the reasons why I can live my life full of pleasure and happiness. Since my older sister and I have been living in the U.S. away from my parents in South Korea for a long time, my sister has always taken good care of me like my parents even though we lived in different cities. However, my sister and I would like to express most sorry to our parents for not being able to stand by them when they took big and small surgeries in South Korea for health issues. I sincerely appreciate my parents for raising us with devoted love and angel smiles, and now, it is my turn to pay back for their love. My dad, my mom, and my sister, I love you all.

Curriculum Vitae

Hoontaek Oh

April 2023

Fields of Study

Communication / Control / Signal Processing / Speech Coding / Machine Learning / Imaging System

Education

1) Doctor of Philosophy in Electrical and Computer Engineering — Apr. 2018 - Apr. 2023 (Expected)
   - Major: Communication / Control / Signal Processing (CCSP)
   - University of California, Santa Barbara, California, U.S.
   - Advisor: Jerry D. Gibson

2) Master of Science in Electrical and Computer Engineering — Sep. 2016 - Mar. 2018
   - Major: Communication / Control / Signal Processing (CCSP)
   - University of California, Santa Barbara, California, U.S.
   - Advisor: Jerry D. Gibson

3) Bachelor of Science in Electronic and Electrical Engineering — Mar. 2006 - Dec. 2012
   - Major: Analog & Digital Communication
   - Hong-Ik University, Seoul, South Korea

Professional Employment

1) Research Assistant (RA), University of California, Santa Barbara, California, U.S. — 2019 - 2022
   - Advisor: Prof. Jerry D. Gibson
   - Supporter: Raytheon, U.S.

2) Teaching Assistant (TA), University of California, Santa Barbara, California, U.S. — 2018 - 2023
   - TA for ECE 139 (Probability and Statistics), Spring 2023
   - TA for ENGR3 (Introduction to Programming), Spring 2022
   - TA for ECE 158 (Digital Signal Processing), Fall 2020
   - TA for ECE 158 (Digital Signal Processing), Fall 2018
   - TA for ECE 10C (Foundations of Analog and Digital Circuits and System), Spring 2018
   - TA for ECE 10B (Foundations of Analog and Digital Circuits and System), Winter 2018

3) Network Analyst, CJ Hellovision, Seoul, South Korea — Jan. 2012 - Oct. 2014
   - Service Management for Cable TV, Ethernet and VoIP services
   - Telecommunication Network Design
   - Network System Management at Data Center

4) Military Service — Jul. 2008 - Sep. 2010
   - Discharged as Avionics Engineer, Sergeant, ROKAF (Republic of Korea Air Force)

5) Private Tutoring for High Schoolers, Seoul, South Korea — Mar. 2006 - Jul. 2016
   - Math, Physics, and Chemistry

Publications

- H. Oh and J. D. Gibson, “Recursive Randomized Tree Coding of Speech”, 2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR), Aug. 2022

- J. D. Gibson and H. Oh, “A Reinforcement Learning Approach to Speech Coding”, Information 2022, 13, 331, MDPI, Jul. 2022

- H. Oh and J. D. Gibson, “Output Recursively Adaptive (ORA) Tree Coding of Speech with VAD/CNG”, 2020 54th Asilomar Conference on Signals, Systems, and Computers, Nov. 2020.

- J. D. Gibson and H. Oh, “Mutual Information Loss in Pyramidal Image Processing”, Information 2020, 11(6), 322, MDPI, Jun. 2020

- J. D. Gibson and H. Oh, "Characterizing Mutual Information Loss in Pyramidal Image Processing Structures", 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Mar. 2019

- J. D. Gibson, H. Oh, and K. K. Sivakumar, "Analysis of Cascaded Signal Processing Operations Using Entropy Rate Power", 2018 52nd Asilomar Conference on Signals, Systems, and Computers, Oct. 2018

Awards

- Outstanding UCSB Electrical and Computer Engineering (ECE) TA Award, University of California, Santa Barbara, California, U.S, 2021

- Outstanding UCSB Electrical and Computer Engineering (ECE) TA Award, University of California, Santa Barbara, California, U.S, 2019

- Outstanding UCSB Electrical and Computer Engineering (ECE) TA Award, University of California, Santa Barbara, California, U.S, 2018

Leadership

- A President of UCSB Korean Graduate Students Association (KGSA), University of California, Santa Barbara, California, U.S, 2018-2019

- A Leader (Drummer) of the Rock Band “Hello Band” of CJ Hellovision, Seoul, South Korea, 2013-2014

- A Student President of the Department “Electronic and Electrical Engineering”, Hong-Ik University, Seoul, South Korea, 2010-2011

- A Squad Commander (Leader) in the Republic of Korea Air Force (ROKAF), South Korea, 2010

- A Leader (Drummer) of the Rock Band “Aulim” of Hong-Ik University, Seoul, South Korea, 2007-2008

Abstract

Recursively Adaptive Randomized Multi-Tree Coding (RAR MTC)

of Speech with VAD/CNG

by

Hoontaek Oh

A new form of a tree codec for narrowband speech, “Recursively Adaptive Randomized Multi-tree Coding (RAR MTC) with VAD/CNG”, is developed based on a sample-by-sample analysis-and-synthesis linear predictive model by benchmarking and upgrading the tree coding models suggested by J. D. Gibson, W. Chang and H. C. Woo. in the 1990s. A simple structure of the Voice Activity Detection/Comfort Noise Generation (VAD/CNG) algorithm is newly applied to the prior speech tree coder to lower the average bit rate by increasing encoding efficiency. A backward adaptive all-pole short-term predictor, which was cascaded to a pitch-based long-term predictor, is replaced with a backward adaptive pole-zero predictor for better input waveform-tracking performance with higher accuracy of prediction. The RAR MTC encodes the initial samples of each voiced region by spanning a 5-level Pitch Compensating Quantizer (PCQ) tree, and then, our randomly interleaved 4-level and 2-level multitree (4-2 MTC) is used to encode the rest of voiced samples with a set of prediction parameters initialized by the 5-level tree coding. A newly developed gain control algorithm for a 2-level tree based on the polarity pattern of the past 5 excitation values advances its gain tracking performance.

In our simulations, the results show that those new features we have developed enable the RAR MTC codec to achieve very competitive performance with a lower delay and more

natural tone recovery compared to the widely used standard, AMR-NB, which is built on a CELP structure based on a block-based predictive model.

Table Of Contents

Chapter 1

Introduction

1. 1. Motivation

For a long time, speech coding research has continued to fuel the development of speech codecs for digital cellular and mobile radio communications. Researchers in this field have continuously pushed themselves to increase the performance threshold of existing speech codecs in diverse ways, and most of their efforts have focused on the block-based analysis-by-synthesis model resulting in the currently popular standardized speech codecs such as the Adaptive Multi-Rate (AMR) codec [1] and the Enhanced Voice Services (EVS) codec[2].

Unlike those speech codecs, there has been, also, the efforts to develop a new codec based on a sample-by-sample recursive model that behaves more closely to waveform-tracking with a lower delay than the block-based model. Especially, the designs of the tree coders developed by Iyengar and Kabal [3], Gibson and Woo [4], and Chang [5] pointed forward new non-block-based speech codec models by exploring multiple possible signal synthesis scenarios with a finite number of excitation candidates and finding the best scenario among them.

The research of Gibson and Chang [5] proved that we can even achieve a lower fractional rate for a tree coder by combining diverse structures of a tree unit with a good performance of gain adaptation algorithms; but at that time about 30 years ago, the multi-path exploration in

tree coding combined with backward adaptive parameter adaptation was too complex, particularly at the receiver, to produce competitive quality coded speech. With major advances in hardware technologies, we have renewed interest in their work and investigated a new tree codec, “Recursively Adaptive Randomized (RAR) Multi-Tree Coding (MTC) of Speech with VAD/CNG”. The codec is built on a new tree structure, a randomized 5-4-2 multi-tree, combined with a simplified VAD/CNG for a lower bit rate with a higher encoding efficiency. The proposed tree structure has never been explored before, and to optimize the gain adaptation for the tree, and especially to better adapt the gain of the 2-level tree composing the 5-4-2 multi-tree, we invented a new gain adaptation rule based on the polarity information of the past 5 excitations. The RAR MTC codec achieves the best performance ever produced by a non-block-based backward adaptive codec at rates below 8 kilobits/sec (kbps).

In this dissertation, we first introduce the common speech codec standards built on a block-based model, and then explain the details of our RAR MTC speech codec based on a sample-by-sample analysis model and compare our codec’s performance to the widely installed standardized block-based codec, AMR.

1. 2. Common Speech Codec Standards based on block-based Model

There are several key technical advancements that have greatly influenced the dominant designs of the most common speech coding standards. The analysis-by-synthesis model served as the main infrastructure of the most promising speech codecs for a narrowband speech by the early 1990s. Most research efforts have been made to create a good block-based

excitation codebook to improve reconstruction quality with a proper level of search complexity.

An early codebook design method was to use the Gaussian random codebook composed of Gaussian random sequences, but its encoding complexity was high because it must generate a sufficient number of random sequences to find a well-suitable sequence from more candidates. Another method is to use the multiple excitation types of codebooks which consist of a finite number of impulses placed throughout a speech frame, and each of them possibly has different polarity and amplitude [6]. This is complex to encode because it needs to iteratively and optimally place the impulses and set their proper amplitude. Later, Adoul and his colleagues proved that a sparse set of positive and negative impulses with the same amplitude would generate good quality of speech while running at a manageable level of complexity due to the sparseness of the impulses and a single common amplitude value for them [7, 8]. This sparse excitation codebook is called an algebraic codebook, which became the basis of the G.729 analysis-by-synthesis speech coding standard set by the ITU-T for speech coding at 8 kbps [9], and the technique was named Algebraic Code Excited Linear Prediction (ACELP) in G.729. The most widely used speech codecs are based on the block-based ACELP from the 2000s to date, and the Adaptive Multi-rate (AMR) codec and the Enhanced Voice Services (EVS) are the best-performing and most recently standardized codecs. We explore the properties of AMR and EVS in the next section.

1. 2. 1. AMR

The Adaptive Multi-Rate (AMR) speech codec [1] was first standardized for a narrowband speech signal by 3GPP in October 1999 and expanded to a wideband application later in 2001. It synthesizes the speech signal based on the linear prediction model using an ACELP codebook excitation. The parameters of the linear predictive model are converted to Line Spectral Pairs (LSP) and encoded for transmission. The overall design is Algebraic Code-Excited Linear Prediction (ACELP) built on a block-based analysis-and-synthesis model. The AMR Narrowband (AMR-NB) operates at the bit rates of 4.75, 5.15, 5.9, 6.7, 7.4, 7.95, 10.2, and 12.2 kbps while the AMR Wideband (AMR-WB) operates at the bit rates of 6.6, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.09, and 23.85 kbps.

Also, the AMR codec uses Discontinuous Transmission (DTX), Voice Activity Detection (VAD) and Comfort Noise Generation (CNG) to lower the average bit rate by reducing the bandwidth usage. AMR-NB uses a 160-sample frame for the narrowband input signal sampled at 8kHz, and its resultant algorithmic delay per frame is 20 ms with a 5 ms of algorithm look-ahead delay except for 12.2 kbps mode. These codecs have been used worldwide in 3rd generation digital cellular systems and have played an important role as the default codecs for VoLTE (voice over long-term evolution) in the 4th generation digital cellular. Further details of this codec can be found in the standard reference [1].

1. 2. 2. EVS

A newer speech codec, the Enhanced Voice Services (EVS) codec [2] was standardized in 2014 and is based on the ACELP structure and MDCT (Modulated Discrete Cosine Transform) and builds on components of AMR-WB. It shows enhanced voice quality with better coding efficiency for narrowband (NB), wideband (WB), Super-wideband (SWB), and Fullband (FB) with high robustness to packet loss and delay jitter by using channel-aware coding. Unlike AMR which mainly focuses on speech signals, EVS shows a good performance not only for speech, but also for music and mixed content too. The difference comes from the different roles of these two codecs. They both can detect the voice and other meaningful audio sources having a periodic tone like music, but EVS can additionally classify the meaningful audio source into a voice, music, or mixed, so depending on the classification result, EVS chooses the best encoding scheme. EVS can switch its operation bit rate every 20 ms based on using the standardized ERVC-NW (enhanced variable rate narrowband-wideband) codec, so it can operate at the bit rates from 5.9 kbps to 128 kbps depending on channel bandwidths, and the bit rate can be switched every 20 ms. Another advantage is that it is backward compatible with AMR-WB, and it has a great extent of new pre-processing and post-processing algorithms. Although the newer codec, EVS, is designed to work for a wider frequency range of audio sources with better performance than AMR in general, AMR-NB/WB was released earlier than EVS and is less complex and is still widely adopted in mobile voice communications.

Compared to these codecs based on the block-analysis-and-synthesis model, we discuss our efforts to build a speech codec based on the sample-by-sample approach in the next section.

1. 3. Speech Codec Research for Sample-by-Sample Approach

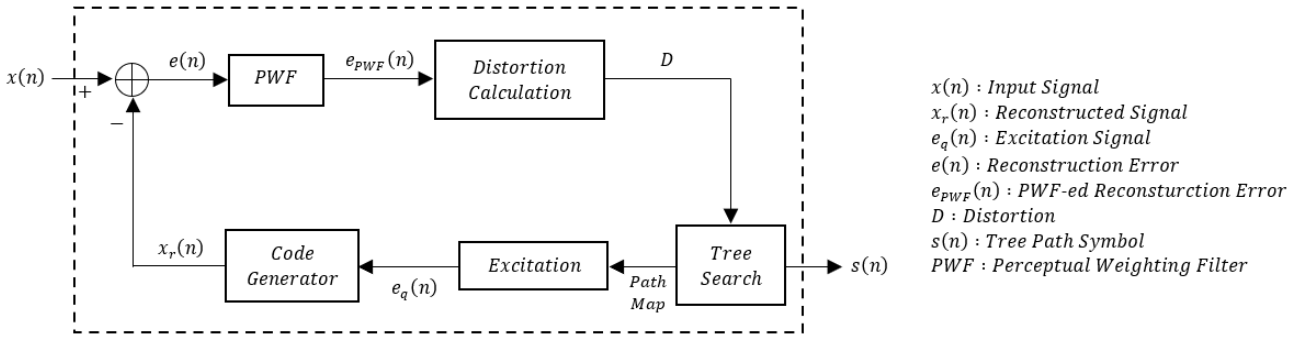
Unlike the most popular block-based speech codecs like AMR and EVS, there were also other early efforts to develop a new speech codec based on the sample-by-sample approach. The sample-by-sample approach encodes and decodes a signal sequentially in every sample so that a codec has a much simpler structure than block-based codecs with lower time delay. The classic speech coding technique, Pulse Code Modulation (PCM) [10], is one of the examples based on a sample-by-sample method, and when it is combined with predictive coding to quantize and encode the prediction error instead of the original signal's amplitude, it is called a Differential Pulse Code Modulation (DPCM) [11]. If we adapt the quantization step size from DPCM to increase SNR, we call it an Adaptive Differential Pulse Code Modulation (ADPCM) [12]. As a waveform coder, ADPCM processes the input on a sample-by-sample basis, so the algorithmic delay itself is just a single sample period, and its encoding and decoding algorithm can be much simpler than ACELP. A brief information about ADPCM can be found in Appendix.

The concept of Tree Coding can be considered a variation of ADPCM. In the following section, we describe the general concept of Tree Coding for speech and how it offers many performance benefits.

1. 3. 1. Tree Coding

The concept of Tree Coding in source coding is basically to generate multiple excitation sequences with different outputs and transmit the optimal excitation sequence generating the best output. Although a low look-ahead delay is necessarily accompanied, it allows existing

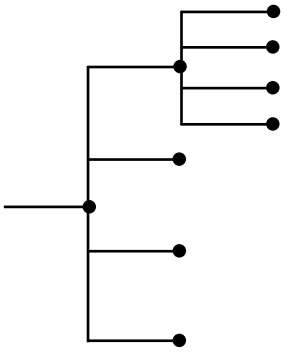
single-sequence-output encoders to have better performance, and we may design Tree Coding in myriad ways depending on the developer's imagination or requirements of a target system. Tree Coding of a speech signal was first suggested by Anderson and Bodie in the early 1970s [13], and in the 1990s, Gibson, Chang, and Woo published their research work about more advanced and promising structures of a speech Tree Coder based on adaptive predictive coding models [4] [5] while implying more potential of a Tree Coder to be explored in the future.



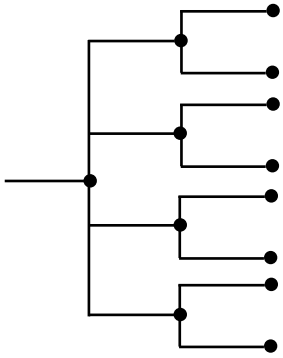
< Fig 1. Tree Coder >

Figure 1 shows a general block diagram of a Tree Coder for speech coding. The Tree Search module generates all possible excitation sequences to some depth and passes each of them through a code generator that reconstructs the signal. In our research model, we initially adopted the same predictive model for the code generator as [4] [5]. Once the error sequence between the original speech signal and the reconstructed signal is calculated, the distortion is measured by perceptually-weighted-filtering and mean-squaring the weighted encoding error for each of the candidate excitation sequences. Finally, the Tree Search module picks the best sequence with the minimum weighted distortion and releases one or multiple symbols of the best path. The symbol release rule is covered more deeply in a later chapter.

The tree can be designed in a myriad number of ways depending on the required performance for the target system. The rate of the tree code is calculated as $R = \frac{1}{\beta}\log\alpha \ bits/sample$ where α is the number of branches/node and β is the number of excitation values/branch. The 4-level tree in Figure 2 has $\alpha = 4$ and $\beta = 1$, and as a result, it provides $R = 2 \ bits/sample$.



< Fig 2. 4-level Tree >

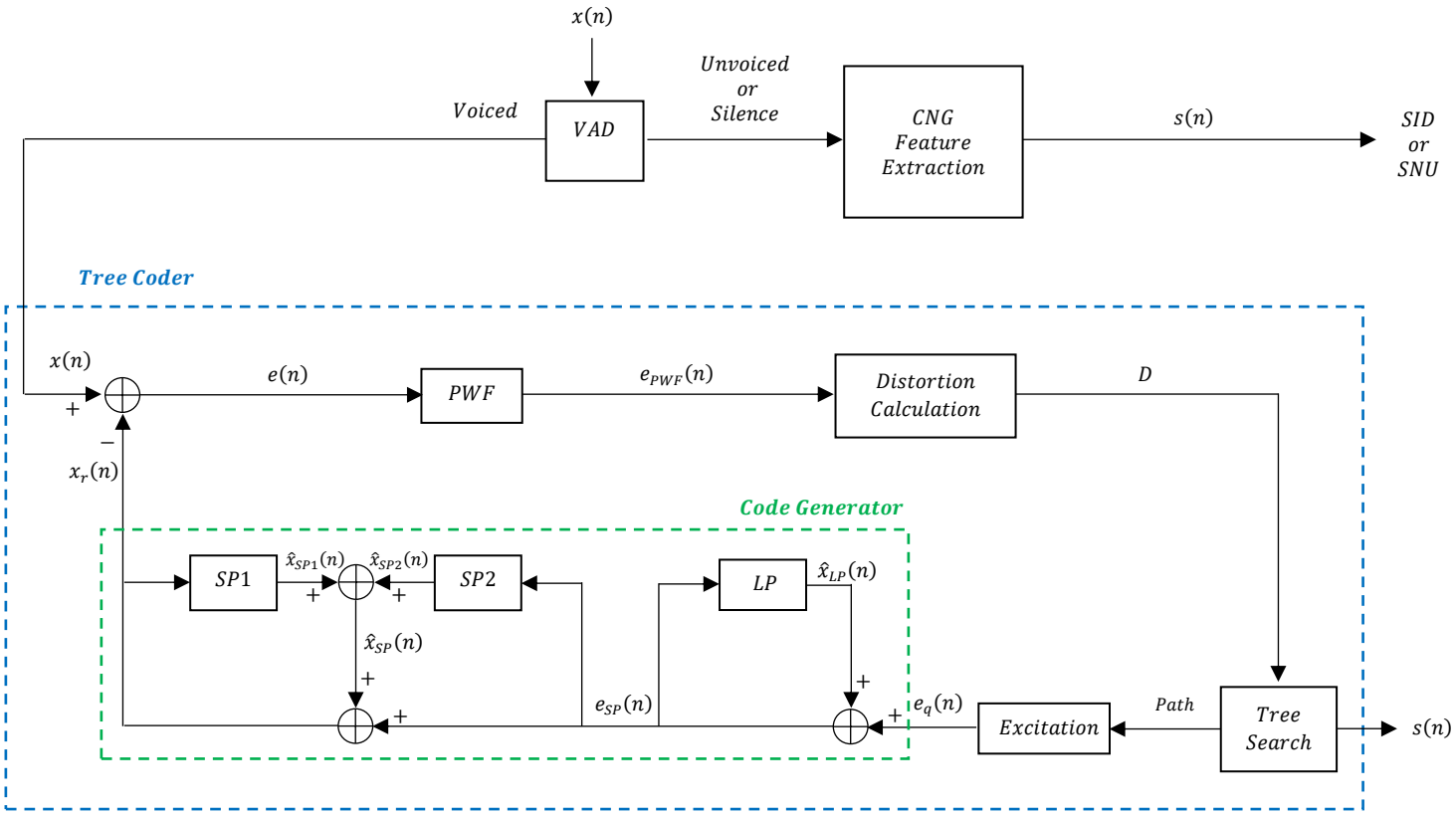


< Fig 3. 4-2 Multi-Tree >

A tree can have a fractional rate by setting α and β differently or combining multiple different shapes of trees. In Gibson and Woo's work [4], the tree is randomly populated by using a Gaussian codebook with $\alpha = 32$ and $\beta = 4$, so the rate is $R = 1.25 \ bits/sample$. Another example is a 4-2 multi-tree in Figure 3 which was explored in Gibson and Chang's paper [5]. The 4-2 multi-tree is a combination of a 4-level tree and a 2-level tree shown. Since the rate for a 4-level tree is 2, and the rate for a 2-level tree is 1, the resultant average R for a 4-2 multitree is 1.5 $bits/sample$.

1. 3. 2. Introduction of Recursively Adaptive Randomized Multi-Tree Coding (RAR MTC) of Speech with VAD/CNG

Gibson, Chang, and Woo's research work [4] [5] showed surprising performance with low delay by applying Tree Coding based on a predictive coding model with backward adaptation. However, as the requirement for low time delay was dropped by standards bodies in the 1990s, the block-based speech codecs have received more attention from researchers, and the research of tree coding of speech has not been performed at all for a long time until we restarted this research. Waveform tracking and low delay are the still big advantages of tree coding, and prior research on tree coding left many research directions unexplored.



< Fig 4. Block Diagram of RAR MTC with VAD/CNG >

First of all, we newly added our simplified version of Voice Activity Detection (VAD) / Comfort Noise Generation (CNG) to lower the average bit rates. For the structure of a tree, we created a 5-4-2 switchable Multi-Tree which is composed of three different tree units; 5-level, 4-level, and 2-level tree units. Also, we apply a variable symbol release rule to decrease the bit rates while minimizing the loss of sound quality for voiced speech. In addition, we devised a new gain adaptation rule for a 2-level tree for better waveform tracking. For a code generator inside tree coding, we used the same form of the backward adaptive short-term and long-term linear predictors in a cascaded form as Gibson, Chang, and Woo's papers [8] [9], but we reoptimized the controlling factors for their adaptation algorithms. The re-optimization of controlling factors has been done for the Perceptual Weighting Filter (PWF) too.

Our new speech tree coder shows over a 3.5 PESQ-MOS score which is the best ever achieved by recursive backward adaptive tree coding of speech. The performance is quite competitive to the AMR-NB codec with similar PESQ-MOS scores at 12.2 kbps in voiced regions. In the following chapters, we categorize our speech codec (RAR MTC) into several blocks based on their distinct functions in Figure 4 and go through the details of them to elaborate their roles and provide new insights.

Chapter 2

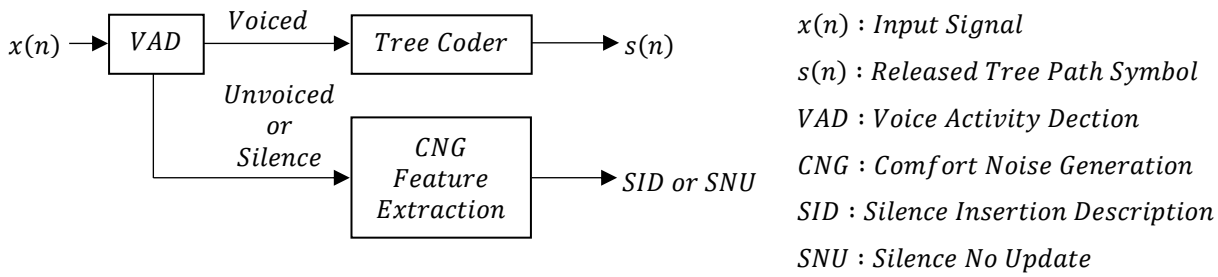
VAD/CNG

2. 1. Introduction

The most common speech codecs, AMR and EVS have their own state-of-art VAD/CNG modules to switch operational modes for high efficiency of encoding scheme depending on the information of the input signal. Since the prior tree coding works [4] [5] before our research did not apply the VAD/CNG to their research models, we incorporated a new version of the VAD/CNG module in our tree coder. Our VAD/CNG has a simpler structure than AMR or EVS, and it was intended to evaluate the benefits and algorithmic performance trade-off of a VAD/CNG module in a speech codec while mainly focusing on the advancement of the actual encoding scheme, a tree coder.

Once VAD/CNG is equipped, the encoder does not directly pass the segmentized input signal to the actual tree coder. Instead, the input signal is sent to VAD first to judge which mode will be used for encoding. In this chapter, we first cover the general role of the VAD module with the specific algorithms about how it categorizes the input and how we encode the input segments which are not encoded by a tree coder, and finally, how a decoder reconstructs the signal, using a Comfort Noise, from the transmitted information for the non-tree-coding data.

2. 2. Voice Activity Detection (VAD)



< Fig 5. Simplified Block Diagram of RAR MTC with VAD/CNG >

As we can see in the much-simplified block diagram of our codec, Figure 5, once the input signal, $x(n)$, comes in, it passes through a VAD module, and the speech signal is segmented for voice activity detection based on a block-based analysis. To detect the activity of a voice signal in each input signal segment, a Voice Activity Detection (VAD) module calculates several feature parameters from the input segment and classifies the input segment into 2 modes; "Voiced" (V) and "Unvoiced" (UV) or "Silence" (S). When the input is classified as "Voiced", the segment is transferred to and encoded by a tree coder. For the "Unvoiced" and "Silence" segments, the prediction coefficients and their prediction gain are calculated based on a block-based least squares method, and the noise parameters are quantized and transmitted to a receiver. Since transmitting noise parameters uses much fewer bits than encoding the input segment, the VAD substantially decreases the average bit rate while preserving sound quality. In the following section, the input classification algorithm and the feature extraction algorithm for "UV" and "S" modes are explored.

2. 2. 1. Mode Classification

The input signal is first segmented by N_{vad} samples in length for a block-based analysis for Voice Activity Detection (VAD). To classify the input segment, $x(n)$, the VAD calculates the power of the input segment, P_{total}, and zero-crossing rates, ZCR,

$$P_{total} = \frac{1}{N_{vad}} \sum_{n=0}^{N_{vad}-1} x^2(n) \quad where\ N_{vad} = 128 \tag{1}$$

$$ZCR = \frac{1}{2N_{vad}} \sum_{n=0}^{N_{vad}-1} |sgn[x(n)] - sgn[x(n-1)]| \quad where\ N_{vad} = 128. \tag{2}$$

As a next step, the VAD multiplies a Hamming window, $w(n)$, to the input segment for more accurate analysis on the original input signal's frequency spectrum

$$x_w(n) = x(n) \cdot w(n) \tag{3}$$

The Hamming-windowed input segment is Fourier-transformed to calculate a low-frequency-band power, $P_{F \le F_{th}}$, for the frequencies, $F \le F_{th}$. Finally, by using P_{total}, ZCR, and $P_{F \le F_{th}}$, we make a VAD decision based on the following algorithm

$$\begin{cases} If\ P_{total} \le P_{th},\ \ "Silence" \\ Else\ if\ P_{LPF} < P_{LPF\_th}\ and\ ZCR > ZCR_{th},\ \ "Unvoiced" \\ Otherwise, "Voiced" \end{cases} \tag{4}$$

From the above algorithms, we see that the block size N_{vad} impacts several system characteristics including algorithmic delay, average bit rate, and decision quality. As we increase the size of a VAD block, it increases the algorithmic delay for looking ahead, but it lowers the average bit rate because fewer silence information packets, SID or SNU, are transmitted in Silence or Unvoiced regions. The block size, also, affects the quality of the decision. The decision sharpness is substantially important for encoding the start and the end of Voiced regions. Depending on the size of the block, we may lose the chance to encode the initial voiced samples, or we may have to inefficiently encode too many Silence or Unvoiced

samples by using a Tree Coder. For a relatively short size of a block, a sharper decision can be made at the start and the end of the voiced regions. In our simulations, we tested $N_{vad} = 40, 60, 80, 120, 140, 160$ and found that $N_{vad} = 120$ is suitable to our audio sample files, but we slightly adjusted the value to $N_{vad} = 128$ which is the power of 2. As a result, the look-ahead time delay caused by our VAD is 16 ms.

In addition to the block size, we need to set proper values of threshold parameters for classification algorithms. First, we started optimization for the power threshold $P_{th} = 10^{-k} \ \ where \ k = 1, 2, \dots, 8$, and $k = 6$ showed the best quality for classifying Silence regions. To find a better value, we tested another set $P_{th} = m \times 10^{-6} \ where \ m = 1, 2, \dots, 8$, and finally, $P_{th} = 5 \times 10^{-6}$ was chosen as the best from our test values.

For the lower-frequency-band power P_{LPF_th}, it is calculated over the frequencies $F \leq F_{th}$, and the frequency threshold value $F_{th} = 1.5 \ kHz$ was chosen by observing the waveforms and the spectrograms of the Unvoiced signals we manually classified. $P_{LPF\_th} = 0.00544$ was searched by gradually increasing from 0.001 to 0.006, and then, the optimal zero-crossing rate threshold $ZCR_{th} = 0.25$ was found from the test set $ZCR_{th} = 0.01m \ where \ m = 5, 6, \dots, 40$.

Again, our VAD algorithm is just a simple basic level compared to state-of-art quality of VADs from AMR or EVS, and the performance may not be as outstanding as the standardized codecs utilizing the advanced algorithms based on more sophisticated signal analysis with a wider mode selection. Accordingly, our codec would be improved if we adopt the advanced VAD/CNG techniques from the recent standardized speech codecs.

2. 2. 2. CNG Parameters Extraction

Once the input segment is classified as “Silence”, the input segment is analyzed based on a linear prediction model as follows

$$x(n) = \sum_{k=1}^{L_{CN}} a_k x(n-k) + e_p(n) \tag{5}$$

$$a_k : Linear\ Prediction\ Coefficients$$

$$e_p(n) : Prediction\ Error$$

$$k = 1, 2, \dots, L_{CN}$$

The linear prediction coefficients, a_k, are calculated based on the autocorrelation method with the least-square error. The autocorrelation function can be obtained as following [14]

$$R(k) = \frac{1}{L_{CN}} \sum_{n=0}^{L_{CN}-|k|-1} x(n)\, x(n+|k|) \tag{6}$$

With $R(k)$, we use the following efficient recursive procedure to calculate a_k [14]

$$Let\ E^{(0)} = R(0) \tag{7}$$

$$p_k = \frac{R(k) - \sum_{l=1}^{k-1} a_l^{(k-1)} R(k-l)}{E^{(k-1)}}, \quad 1 \le k \le L_{CN} \tag{8}$$

$$a_k^{(k)} = p_k \tag{9}$$

$$a_l^{(k)} = a_l^{(k-1)} - p_l\, a_{k-l}^{(k-1)}, \quad 1 \le l \le k-1 \tag{10}$$

$$E^{(k)} = (1 - p_k^2)\, E^{(k-1)} \tag{11}$$

From the above procedures, p_k is the set of Partial Correlation (PARCOR) coefficients or reflection coefficients, and $E^{(k)}$ is the mean squared prediction error for the k-th order prediction. PARCORs have the property that if $|p_k| \le 1\ for\ all\ k\ values$, the synthesis process is stable. A stable set of linear prediction coefficients can be turned into an unstable

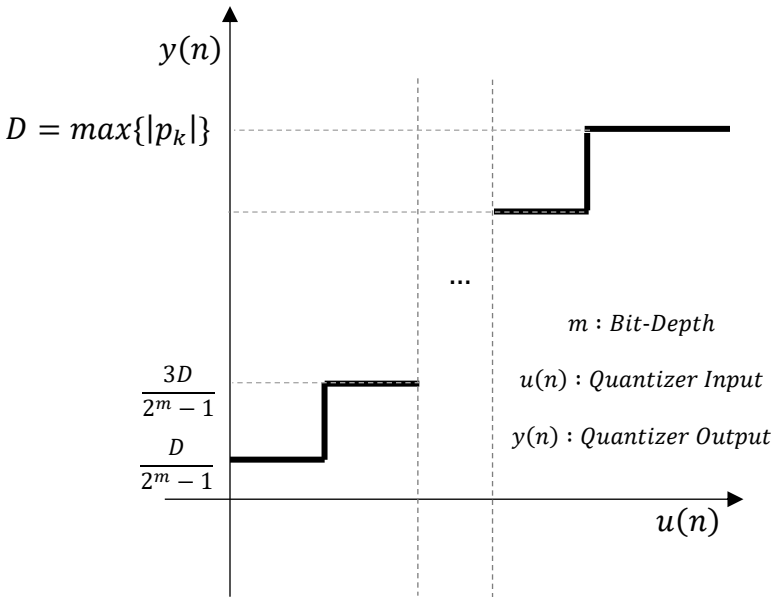
set by quantization or other transmission coding, but PARCOR coefficients are relatively less sensitive to quantization. For these reasons, we transmit the PARCORs instead of the linear prediction coefficients (LPC), with the prediction noise power, $E^{(L_{CN})}$.

2. 2. 3. Quantization of CNG Parameters

To transmit PARCORs and the prediction noise power as a bit stream, the quantities are quantized. For PARCORs, a different dynamic range and bit allocations of symmetric mid-rise uniform quantizers in Figure 6 were used for each of the PARCORs as in Table 1.

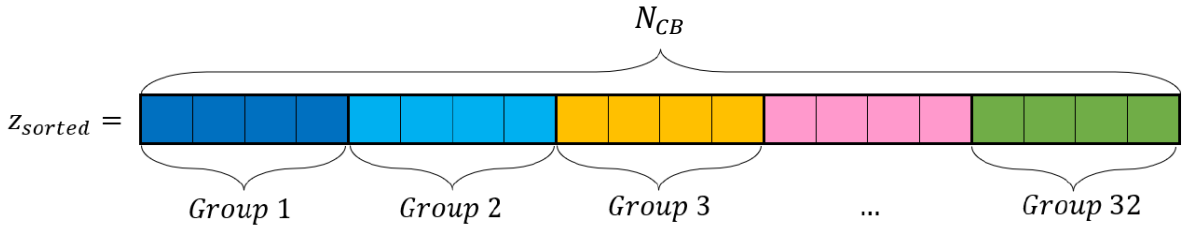
Table 1. Dynamic Range and Bit Allocations of Symmetric Mid-rise Uniform Quantizers for PARCORs

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
Dynamic Range $\max\{\lvert p_k \rvert\}$	0.97	0.97	0.82	0.92	0.74	0.82	0.58	0.73
Quantization Bit-Depth	8	8	6	6	5	5	5	5



< Fig 6. Symmetric Mid-rise Uniform Quantizer for PARCORs >

Their dynamic ranges are set by observing the unquantized PARCORs in "Silence" regions for our sample files, and the quantization bit-depths are chosen depending on the sensitivity of each order's PARCOR coefficient.



z_{sorted} : *Sorted Training Data Set*

N_{CB} : *Total Number of Training Data*

$\therefore$ *Initial CB* : y_k = *Mean of Group* "k"

< Fig 7. Initial Codebook from Training Data Set for Non-uniform Quantizer >

For the prediction noise power, we used a non-uniform quantizer with 5 bits. To optimize a codebook of the quantizer, we build up the training data set by collecting unquantized prediction error power quantities in each segment of "Silence" regions for all our sample files. The training data set is passed through the non-uniform quantizer with the initial code book in Figure 7, $CB = \{y_1, y_2, \dots, y_{32}\}$, based on the nearest neighbor rule, and by averaging each group of inputs which produce the same output, the CB is updated. This process is repeated 500 times to get the final CB. To get the initial CB, we sorted the training data set in an increasing magnitude order and divide the sorted data sequence into the same size of total 32 groups except for the last group, which is the remainder after grouping the first 31 groups.

2. 2. 4. SID/SNU

The quantized CNG parameters are not always transmitted to a receiver. If the parameters for the current input segment are close enough to the ones for the previous input segment, the transmitter sends a "Silence No Update (SNU)" packet which means that the receiver does not need to update its CNG parameters from the previous set. However, if the new parameters are quite distant from the older ones, a "Silence Information Descriptor (SID)" packet is transmitted including the new quantized CNG parameters. The exact algorithm for judging which CNG packet will be transmitted is shown below.

$$\begin{cases} If\ \left\|\underline{a}_{new} - \underline{a}_{old}\right\|^2 > \underline{a}_{th} = 0.05\ \ or\ \ \dfrac{E_{old}^{(L_{CN})} - E_{new}^{(L_{CN})}}{E_{old}^{(L_{CN})}} > E_{th}^{(L_{CN})} = 0.4, \quad send\ "SID" \\ Otherwise,\ \ send\ "SNU" \end{cases} \quad (12)$$

$$\underline{a}_{old} : Old\ CNG\ LPC\ Coefficient\ Vector$$

$$\underline{a}_{new} : New\ CNG\ LPC\ Coefficient\ Vector$$

$$E_{old}^{(L_{CN})} : Old\ CNG\ LPC\ Prediction\ Error\ Power$$

$$E_{new}^{(L_{CN})} : New\ CNG\ LPC\ Prediction\ Error\ Power.$$

The threshold values from the above algorithm are properly chosen through multiple simulations with different values, from 0.01 to 0.1 by 0.01 increment for $\underline{a}_{th}$, from 0.1 to 0.5 by 0.1 increment for $E_{th}^{(L_{CN})}$.

SID and SNU are classified by using a single header bit, and the size of SID is 53 bits = 48 bits (PARCORS) + 5 bits (Prediction Error Power). Although the header bit is used for classifying the two different noise packets, the usage of SNU decreases the average bit rate by removing unnecessary updates.

2. 2. 5. Comfort Noise Generation (CNG)

Once a decoder receives the SID or SNU frame, it updates the comfort noise parameters depending on the information in the frame and generates a comfort noise by using the updated noise parameters based on an autoregressive model as following

$$x_r(n) = \sum_{k=1}^{L_{CN}} a_k x_r(n-k) + e_{cng}(n) \tag{13}$$

$$x_r(n) : Comfort\ Noise$$

$$a_k : LPC\ converted\ from\ transmitted\ PARCORs$$

$$e_{cng}(n) \sim N\left(0, E^{(L_{CN})}\right)$$

$$E^{(L_{CN})} : Transmitted\ Prediction\ Power$$

For PARCORs, they are converted into linear prediction coefficients and applied to the synthesis model. This CNG process does not happen at the transmitter because an encoder has no need to recover the comfort noise signal, instead, it just delivers the required data for signal synthesis at a decoder. Furthermore, since we are not extending the concept of tree coding to this CNG process, an encoder has no reason to recover and analyze the signal.

It may achieve better sound performance when we increase the order of PARCORs and the quantization bit allocations while resulting in the higher average bit rates of the whole encoding process, so the specifications for these parameters can be varied depending on the requirement of the intended system.

Unlike the unvoiced segment (UV), the silence (S) segment is considered the pure low-energy meaningless noise, so when VAD/CNG was first researched, the silence was once replaced with a pure zero signal, total silence, and there were several unexpected negative effects on the listener side. First, people are likely to think that their connection is lost during total silence. Secondly, the sudden start of speech signals after total silence often incurs uncomfortable and unnatural ear experiences, and it makes the sound very “choppy” so that it

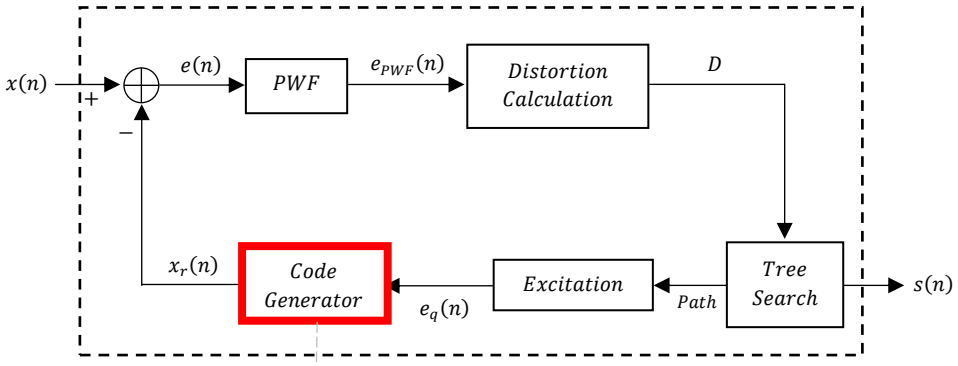
makes people hard to understand. To avoid these issues, CNG is substantially essential to a receiver in voice communication systems.

Chapter 3

Code Generator

3. 1. Introduction

When the input block is classified as “Voiced”, the input is handed over to the Tree Coder. The “Tree Search” module in Figure 1 generates multiple excitation sequence candidates and passes them to a code generator, and then for each of the excitation sequences, the code generator synthesizes output signals.



< **Fig 1. Tree Coder** >

$x(n)$: *Input Signal*

$x_r(n)$: *Reconstructed Signal*

$e_q(n)$: *Excitation Signal*

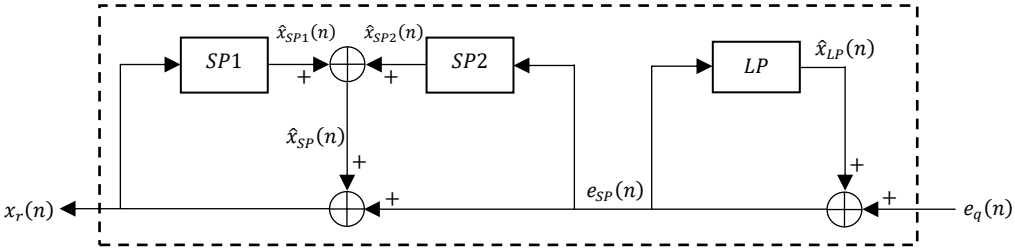
$e(n)$: *Reconstruction Error*

$e_{PWF}(n)$: *PWF-ed Reconsturction Error*

D : *Distortion*

$s(n)$: *Tree Path Symbol*

PWF : *Perceptual Weighting Filter*



< **Fig 8. Code Generator** >

$x_r(n)$: *Reconstructed Signal*

$\hat{x}_{SP1}(n)$: *Short-term Prediction (Poles Only)*

$\hat{x}_{SP2}(n)$: *Short-term Prediction (Zeros Only)*

$\hat{x}_{SP}(n)$: *Short-term Prediction* $\left(\hat{x}_{SP1}(n) + \hat{x}_{SP2}(n)\right)$

$\hat{x}_{LP}(n)$: *Long-term Prediction*

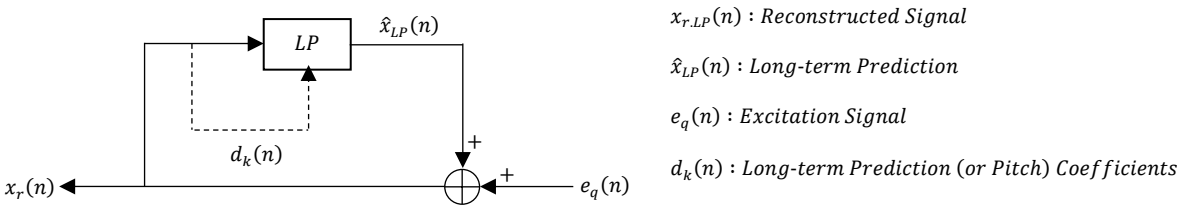
$e_{SP}(n)$: *Short-term Prediction Error*

$e_q(n)$: *Excitation Signal*

Based on the general concept of a Tree Coder, the structure of the code generator depends on a codec designer, but in our codec, we adopted a predictive coding structure as the predecessors did in tree coding of speech [4] [5].

The code generator in Figure 8 is composed of a long-term and short-term adaptive predictor in a cascaded form. When an excitation comes to a code generator, the first stage of signal synthesis is performed by the long-term predictor. The output of the first stage becomes the excitation to a short-term predictor which is the second stage of signal synthesis. The sequence of the two predictors has been set by interchanging their orders and finding the best one. In the following sections, the details of the adaptive predictors are introduced.

3. 2. Backward Adaptive Long-term Predictor



< Fig 9. Backward Adaptive Long-term Predictor >

The long-term predictor uses the feature of a speech signal, a pitch or long-term redundancy, for its prediction. When we plot the voiced regions of a speech signal in the time domain, we can see a certain shape of a waveform periodically repeats, and the period of the periodic signal is called a pitch period. By using the periodicity, we can intuitively predict the current input value by using a 3-tap of input values centered at the time a pitch period ago as follows [15]

$$\hat{x}_{LP}(n) = \sum_{k=-1}^{+1} d_k(n)\, x_{r.LP}(n - P - k) \tag{14}$$

$\hat{x}_{LP} : Long\text{-}term\ Prediction\ (LP), \quad x_{r.LP} : Reconstructed\ Singal\ from\ LP\ Syntheizer$

$P : Pitch\ Period, \qquad d_k(n) : LP\ Coefficients$

However, since our speech signal is composed of diverse sentences of speakers with different pitch values, it is nonstationary and time-variant. For this reason, the long-term predictor needs to constantly track the pitch period and LP coefficients in real time. To adapt these parameters, we use a hybrid algorithm composed of a block pitch parameters adaptation algorithm and a recursive pitch parameters algorithm. In the following sub-sections, we look at the details of each algorithm.

3. 2. 1. Block Pitch Parameters Adaptation

The block method adaptation algorithm uses the covariance for a block of past samples to find the pitch period and LP coefficients, and this algorithm was presented first in [16]. The covariance in this algorithm is defined as

$$q_n(u, v) = \sum_{m=1}^{N_{LP.Block}} x_{r.LP}(n - m - u)\, x_{r.LP}(n - m - v) \tag{15}$$

$q_n(u, v) : Covariance\ for\ two\ time\ lags, u\ and\ v.$

$N_{LP.Block} : Block\ Size$

By using the covariance function, we calculate the normalized correlation function $\rho_n(l)$ for the pitch lag l over the range of 16 ~ 160

$$\rho_n(l) = \frac{q_n(0,l)}{\sqrt{q_n(0,0)\, q_n(l,l)}} \tag{16}$$

The l minimizing $\rho_n(l)$ becomes the pitch period P for our long-term prediction, and once P is determined, we solve the following normal equations to find the LP coeffcients $d_k(n)$

$$\begin{bmatrix} (1+\gamma)\, q_n(P-1,P-1) & q_n(P-1,P) & q_n(P-1,P+1) \\ q_n(P,P-1) & (1+\gamma)\, q_n(P,P) & q_n(P,P+1) \\ q_n(P+1,P-1) & q_n(P+1,P) & (1+\gamma)\, q_n(P+1,P+1) \end{bmatrix} \times \begin{bmatrix} d_{-1}(n) \\ d_0(n) \\ d_1(n) \end{bmatrix} = \begin{bmatrix} q_n(0,P-1) \\ q_n(0,P) \\ q_n(0,P+1) \end{bmatrix} \tag{17}$$

Since it is possible to have mismatch in time or the above 3×3 matrix has determinant zero, the softening factor $\gamma = 0.001$ is added to the diagonal elements of the 3×3 matrix in the above normal equation to decrease the effect of time mismatch [16], [17].

3. 2. 2. Recursive Pitch Parameters Adaptation

Another long-term parameter update method recursively updates the prediction parameters based on gradient adaptation. The specific method was first introduced by Pettigrew and Cuperman [17], and their algorithm is given below

$$\hat{\sigma}^2_{x_{r.LP}}(n) = \lambda\, \hat{\sigma}^2_{x_{r.LP}}(n-1) + (1-\lambda)\hat{\sigma}^2_{x_{r.LP}}(n) \tag{18}$$

$$\hat{\rho}_n(k) = \lambda\, \hat{\rho}_{n-1}(k) + \frac{x_{r.LP}(n)\, x_{r.LP}(n-k)}{\hat{\sigma}^2_{x_{r.LP}}(n)} \tag{19}$$

$$Leakage\ Factor:\ \lambda = 0.95$$

First, it recursively tracks the variance of the LP synthesizer output $\hat{\sigma}^2_{x_{r.LP}}(n)$ and the auto-correlation function $\hat{\rho}_n(k)$ for $k = P(n)-1, P(n), P(n)+1$. Once these two quantities are calculated, the pitch period is updated based on the following algorithm [17]

$$\begin{cases} if\ \hat{\rho}_n(k+1) > \max(\hat{\rho}_n(k), \hat{\rho}_n(k-1), \hat{\rho}_{min})\ ,\ \ P(n+1) = P(n) + 1\ and \begin{bmatrix} \hat{\rho}_{n+1}(P-1) \\ \hat{\rho}_{n+1}(P) \\ \hat{\rho}_{n+1}(P+1) \end{bmatrix} = \begin{bmatrix} 0.3\ \hat{\rho}_n(P+1) \\ \hat{\rho}_n(P+1) \\ \hat{\rho}_n(P) \end{bmatrix} \\ if\ \hat{\rho}_n(k-1) > \max(\hat{\rho}_n(k), \hat{\rho}_n(k+1), \hat{\rho}_{min})\ ,\ \ P(n+1) = P(n) - 1\ and \begin{bmatrix} \hat{\rho}_{n+1}(P-1) \\ \hat{\rho}_{n+1}(P) \\ \hat{\rho}_{n+1}(P+1) \end{bmatrix} = \begin{bmatrix} \hat{\rho}_n(P) \\ \hat{\rho}_n(P-1) \\ 0.3\ \hat{\rho}_n(P-1) \end{bmatrix} \\ Otherwise\ ,\ \ No\ Change \end{cases} \quad (20)$$

The auto-correlation function vector is only for 3 time-lag values centered at the pitch period, so when the pitch period is changed by +1 or -1, its auto-correlation vector should be shifted up or down while filling up with the new autocorrelation factor, $0.3\ \hat{\rho}_n(P+1)$ or $0.3\ \hat{\rho}_n(P-1)$.

The pitch coefficients $d_k(n)$ are also recursively updated by tracking the long-term prediction error variance $\hat{\sigma}^2_{e_{LP}}(n)$ as follows [17]

$$\hat{\sigma}^2_{e_{LP}}(n) = \lambda\ \hat{\sigma}^2_{e_{LP}}(n-1) + (1-\lambda)\ e^2_{LP}(n) \quad (21)$$

$$d_k(n) = \lambda\ d_k(n) + \frac{\Delta_{LP}}{\hat{\sigma}_{e_{LP}}(n)\ \hat{\sigma}_{x_{r.LP}}(n)}\ \ where\ \ k = -1, 0, +1 \quad (22)$$

$$Leakage\ Factor:\ \lambda = 0.95$$

When we first designed a long-term predictor, the tap size was one, and then, it was expanded to 3 samples. Compared to a single-sample tap, a 3-sample tap shows better performance because it additionally uses more past input values centered at the time a pitch period ago, $n - P$ and compensates for mismatch between pitch and the sampling rate. The even number of tap size is not suitable because we cannot use the symmetry centered at $n - P$. The tap size longer than 3 was not explored because we did not want to increase the computation complexity of the hybrid adaptation algorithm. The leakage factor $\lambda = 0.95$ in a recursive adaptation algorithm is the best from the test set, $\{0.91, 0.92, 0.93, \ldots, 0.98, 0.99\}$ based on the PESQ-MOS performances.

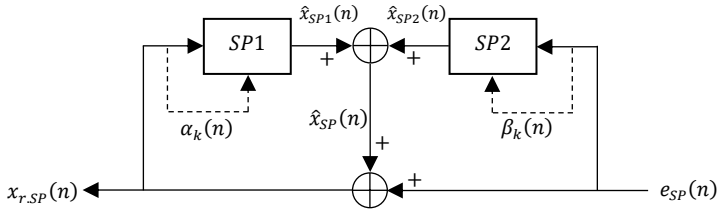
3. 2. 3. Hybrid Pitch Parameters Adaptation

The two prior methods have merits and demerits over each other. The block method is likely to generate better estimates by calculating the normalized correlation function for a wide range of time lags and directly solving the normal equations based on the past samples, but it is computationally burdensome if it has to repeat the whole progress every sample. Furthermore, it cannot prevent the pitch estimation from suddenly jumping to multiple times of the actual pitch period. On the other hand, the recursive method, as we can see, is much simpler than the block method because it tracks the autocorrelation functions for only 3 time delays centered at the pitch period based on the gradient descent algorithm. Moreover, if the pitch change is smooth enough, the pitch tracking works well too without a sudden big jump. However, if our speech has some drastic transition from one sound to another different sound, the recursive method is subject to be lost in tracking. Interestingly, the two methods complement each other's shortcomings, so they are mixed in a hybrid method [17].

Every 20 samples, the block method is used to update the pitch period and the pitch coefficients based on past output values. Otherwise, the recursive method updates them. The hybrid method performs better than the block method only or recursive method only.

3. 3. Backward Adaptive Short-term Predictor

The second-stage signal synthesis after the pitch synthesizer is performed by the adaptive short-term linear predictor which uses another feature of a voiced speech signal, the high correlation of consecutive samples.



$x_{r.SP}(n)$: *Reconstructed Signal*

$\hat{x}_{SP1}(n)$: *Short-term Prediction (Poles Only)*

$\hat{x}_{SP2}(n)$: *Short-term Prediction (Zeros Only)*

$\hat{x}_{SP}(n)$: *Short-term Prediction* $\left(\hat{x}_{SP1}(n) + \hat{x}_{SP2}(n)\right)$

$e_{SP}(n)$: *Short-term Prediction Error*

$\alpha_k(n)$: *Pole Coefficients*

$\beta_k(n)$: *Zero Coefficients*

< Fig 10. Backward Adaptive Short-term Predictor >

Figure 10 shows the block diagram of the adaptive short-term predictor we use, and the general model of the short-term predictor is shown below

$$x_{r.SP}(n) = \hat{x}_{SP} + e_{SP}(n) \tag{23}$$

$$\hat{x}_{SP} = \hat{x}_{SP1} + \hat{x}_{SP2} \tag{24}$$

$$\hat{x}_{SP1} = \sum_{l=1}^{N_{SP1}} \alpha_l(n)\, x_{r.SP}(n-l) \quad \text{where } N_{SP1} = 10 \ \ (Pole\ Part) \tag{25}$$

$$\hat{x}_{SP2} = \sum_{l=1}^{N_{SP2}} \beta_k(n)\, e_{SP}(n-k) \quad \text{where } N_{SP2} = 6 \ \ (Zero\ Part) \tag{26}$$

$$x_{r.SP}(n) = \sum_{l=1}^{N_{SP1}} \alpha_l(n)\, x_{r.SP}(n-l) + \sum_{k=1}^{N_{SP2}} \beta_k(n)\, e_{SP}(n-k) + e_{SP}(n) \tag{27}$$

The transfer function of the shot-term prediction synthesis is

$$\frac{X_{r.SP}(z)}{E_{SP}(z)} = \frac{1 + \sum_{k=1}^{N_{SP2}} \beta_k(n)\, z^{-k}}{1 - \sum_{l=1}^{N_{SP1}} \alpha_l(n)\, z^{-l}} \tag{28}$$

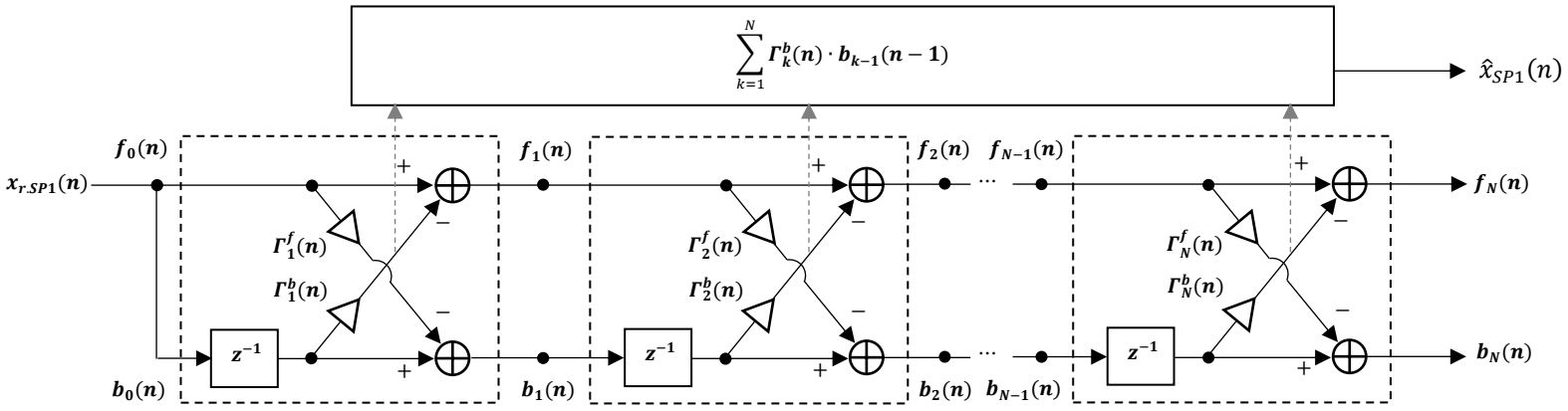
Compared to the pole-only predictor, the inclusion of zeros makes the estimation more accurate and can reduce the flattening of the estimated spectral envelope of the autoregressive model by coupling the backward adaptive predictor to the codec's input signals more directly. Rate distortion theory indicates that for optimal encoding using the mean squared error distortion measure, the short-term predictor should include zeros as well as poles when achieving small distortion [18].

3. 3. 1. Backward Pole Coefficients Adaptation

Instead of using the pole linear prediction coefficients $\alpha_k(n)$, we adopted the lattice-structure short-term predictor based on the reflection coefficients [19]

$$\hat{x}_{SP1} = \sum_{k=0}^{N-1} \Gamma_k^b(n) \cdot b_{k-1}(n-1) \;\; (RC\text{-}based\ Linear\ Prediction) \tag{29}$$

$\Gamma_k^b(n) : Backward\ Reflection\ Coefficients, \qquad b_{k-1}(n-1) : Backward\ Prediction\ Error$



$f_k(n) : The\ k\text{-}th\ Order\ Forward\ Prediction\ Error \qquad \Gamma_k^f(n) : The\ k\text{-}th\ Order\ Forward\ Reflection\ Coefficients$

$b_k(n) : The\ k\text{-}th\ Order\ Backward\ Prediction\ Error \qquad \Gamma_k^b(n) : The\ k\text{-}th\ Order\ Backward\ Reflection\ Coefficients$

< Fig 11. The Lattice Short-term Predictor based on Reflection Coefficients >

Figure 11 shows the block diagram of the lattice predictor based on reflection coefficients, and for the adaptation of the reflection coefficients, the recursive least-square (RLS) lattice algorithm below is used [20]

$$f_0(n) = b_0(n) = x_r(n) \tag{30}$$

$$\xi_0(n) = 0 \tag{31}$$

$$E_1^f(n) = E_1^b(n) = \lambda_2 \cdot E_1^b(n) + f_0^2(n), \quad \lambda_2 = 0.99 \tag{32}$$

$$\Gamma_k(n) = \lambda_2 \cdot \Gamma_k(n-1) + \frac{f_{k-1}(n) \cdot b_{k-1}(n-1)}{1 - \xi_{k-1}(n)} \tag{33}$$

$$\Gamma_k^f(n) = \frac{\Gamma_k(n)}{E_k^f(n)} \tag{34}$$

$$\Gamma_k^b(n) = \frac{\Gamma_k(n)}{E_k^b(n-1)} \tag{35}$$

$$E_{k+1}^f(n) = \frac{E_k^f(n) - \Gamma_k^b(n) \cdot \Gamma_k(n)}{0.98^2} \tag{36}$$

$$E_{k+1}^b(n) = \frac{E_k^b(n-1) - \Gamma_k^f(n) \cdot \Gamma_k(n)}{0.98^2} \tag{37}$$

$$\xi_k(n) = \xi_{k-1}(n) + \frac{b_{k-1}^2(n-1)}{E_k^b(n-1)} \tag{38}$$

$$f_k(n) = f_{k-1}(n) - \Gamma_k^b(n) \cdot b_{k-1}(n-1) \tag{39}$$

$$b_k(n) = b_{k-1}(n-1) - \Gamma_k^f(n) \cdot f_{k-1}(n) \tag{40}$$

$\lambda_2 : Leakage\ Factor$

$f_k(n) : Forward\ Prediction\ Error$

$b_k(n) : Backward\ Prediction\ Error$

$\xi_k(n) : Gain\ Parameter$

$E_{k+1}^f(n) : Forward\ Prediction\ Error\ Variances$

$E_{k+1}^b(n) : Backward\ Prediction\ Error\ Variances$

$\Gamma_k^f(n) : Forward\ Reflection\ Coefficients$

$\Gamma_k^b(n) : Backward\ Reflection\ Coefficients$

The RLS lattice algorithm shows rapid adaptations, and since it is based on reflection coefficients, the magnitude of each coefficient less than 1 implies stability.

When we first designed the pole predictor, we tested the linear prediction order from 4 to 60 for a short-term pole predictor only in a tree coder. Although a higher order produced a higher PESQ-MOS score, as the prediction order increased, the performance increased less. Since the adaptation algorithm of a short-term predictor is based on a sequential computation, the adaptation algorithm cannot be executed in parallel. As a result, the computational time is proportional to the prediction order. Considering these tradeoffs, the 10-th order short-term pole predictor was finally chosen. Like a long-term predictor, the leakage factor of the short-term pole coefficient adaptation algorithm, λ_2 , was optimized from the test set, $\{0.91, 0.92, 0.93, \dots, 0.98, 0.99\}$ based on the PESQ-MOS performances.

3. 3. 2. Backward Zero Coefficients Adaptation

The zero coefficients $\beta_k(n)$ are recursively updated by using the polarities of the current and past prediction errors for its step size based on the gradient approach as following [21]

$$\beta_k(n+1) = \lambda_3\, \beta_k(n) + (1-\lambda_3) sign\big(e_{SP}(n)\big)\, sign\big(e_{SP}(n-k)\big) \tag{41}$$

$$Leakage\ Factor:\ \lambda_3 = 127/128$$

For the zero predictor, we just directly used a 6-th order without any further test with different values, and the leakage factor λ_3 for the adaptation of zero coefficients was the best from the different test set $\left\{\frac{2^k-1}{2^k}\middle| k = 4,5,6,7,8,9\right\}$. The set of test values was motivated by the reference paper of the adaptation algorithm [21].

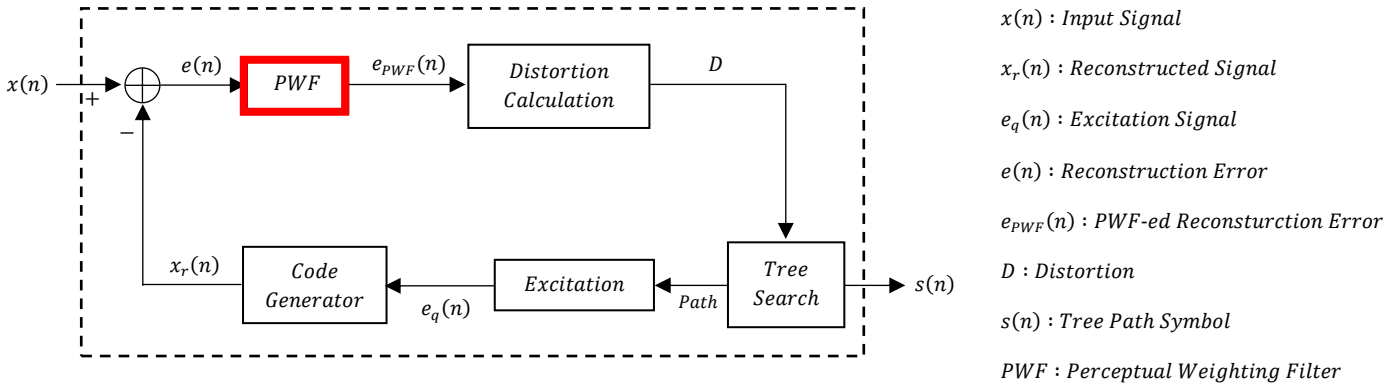
The use of both poles and zeros in a linear predictive model has several advantages compared to the pole-only model. Generally, an Infinite Impulse Response (IIR) filter based on poles and zeros can use a fewer number of pole and zero coefficients by using feedback based on a recursive nature than a Finite Impulse Response (FIR) filter. Furthermore, once all-pole predictor loses its tracking, it tends to track itself rather than changes in the input signal transfer function, but by adding zeros, a pole-zero predictor makes its prediction dependent upon the input signal transfer function to better track dynamic changes of the input with better accuracy of signal reconstruction [22].

Chapter 4

Perceptual Weighting Filter

4. 1. Introduction

As the Code Generator reconstructs a signal $x_r(n)$ for each excitation sequence candidate, a reconstruction error signal is obtained by subtracting the recovered signal from the original signal $x(n)$. Although the predictive coding puts the major priority on minimizing the root-mean-squared error (RMSE) distortion for the synthesized signal, the lower RMS distortion does not sound always better to our ears. It implies that the speech perception mechanism of our ears is not based on only the RMSE-based approach. Another useful approach is to shape a noise signal by using the Perceptually Weighting Filter (PWF) [15].



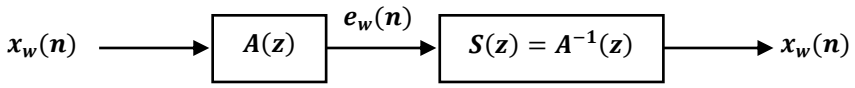
< Fig 1. Tree Coder >

In Figure 1, the reconstruction error $e(n)$ is passed through the PWF, and the RMSE is calculated from the filtered output for distortion comparison in a tree search algorithm.

In this chapter, the transfer function of PWF we used in our codec is introduced and it is shown how to adapt the coefficients of the PWF and how the PWF affects the resultant sound quality.

4. 2. PWF Design

The PWF is designed by using the linear prediction coefficients from the original input. The PWF, first, scans a block of the original signal, which is composed of the past 50 samples and the future 50 samples, by using a Hamming window. From the block, the linear prediction coefficients are calculated based on the least-squared error approach, and its linear prediction model can be simplified as Figure 13.



$x_w(n)$: *Hamming-windowed Input*

$\hat{x}_w(n)$: *Linear Prediction with* d_k

$e_w(n)$: *Linear Prediction Error*

$A(z)$: *Analysis Filter*

$S(z)$: *Synthesis Filter*

< Fig 12. Block Diagram of Linear Prediction Model for PWF Design >

In Figure 12, the analysis and synthesis filters are expressed as (41) and (42), so $x_w(n)$ and $e_w(n)$ can be written as (43) and (44).

$$Analysis\ Filter:\ A(z) = 1 - \sum_{k=1}^{L} d_k \cdot z^{-k} \tag{42}$$

$$Synthesis\ Filter:\ S(z) = A^{-1}(z) = \frac{1}{1 - \sum_{k=1}^{L} \tau_k \cdot z^{-k}} \tag{43}$$

$$x_w(n) = \hat{x}_w(n) + e_w(n) = \sum_{k=1}^{L} d_k \cdot x_w(n-k) + e_w(n) \quad where\ L = 10 \tag{44}$$

$$e_w(n) = x_w(n) - \sum_{k=1}^{L} d_k \cdot x_w(n-k) \tag{45}$$

$$\tau_k : Linear\ Prediction\ Coefficients\ with\ k = 1,2,\dots,10$$

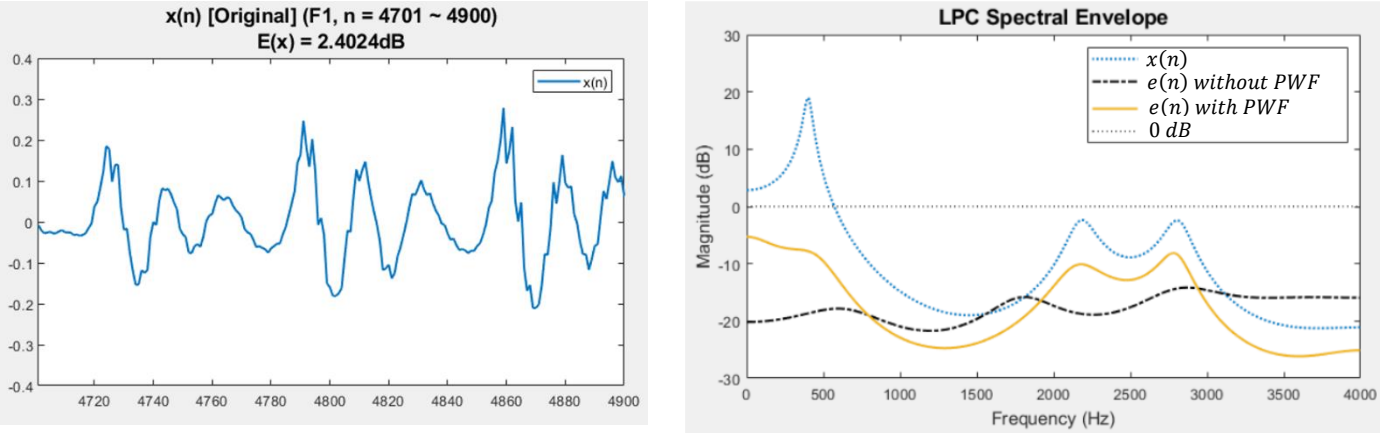
Based on the linear prediction coefficients (LPC), d_k given by Figure 12, the transfer function of the PWF has the following form [16]

$$W(z) = \frac{A(\frac{z}{\mu})}{A(\frac{z}{\gamma})} = \frac{1 - \sum_{k=1}^{L} \tau_k \cdot \mu^k \cdot z^{-k}}{1 - \sum_{k=1}^{L} \tau_k \cdot \gamma^k \cdot z^{-k}} \quad where\ L = 10, \quad \mu = 0.9, \quad \gamma = 0.4 \tag{46}$$

The LPC are periodically updated every 50 samples, and the constants, μ and γ, are the factors affecting the shape of the filter. In designing a PWF, the filter constants, μ and γ, were optimized by varying them separately from 0.1 to 0.9 by 0.1 increment. The values of μ and γ affect the positions of poles and zeros of $W(z)$, and as a result, they affect the shape of $W(z)$. They can be reoptimized on our test sequences if any change is made to a Tree Coder. Furthermore, the order of the filter coefficients, $L = 10$, is matched with the prediction order for the short-term pole predictor because we may, later, directly use the pole prediction coefficients from a short-term predictor as PWF coefficients.

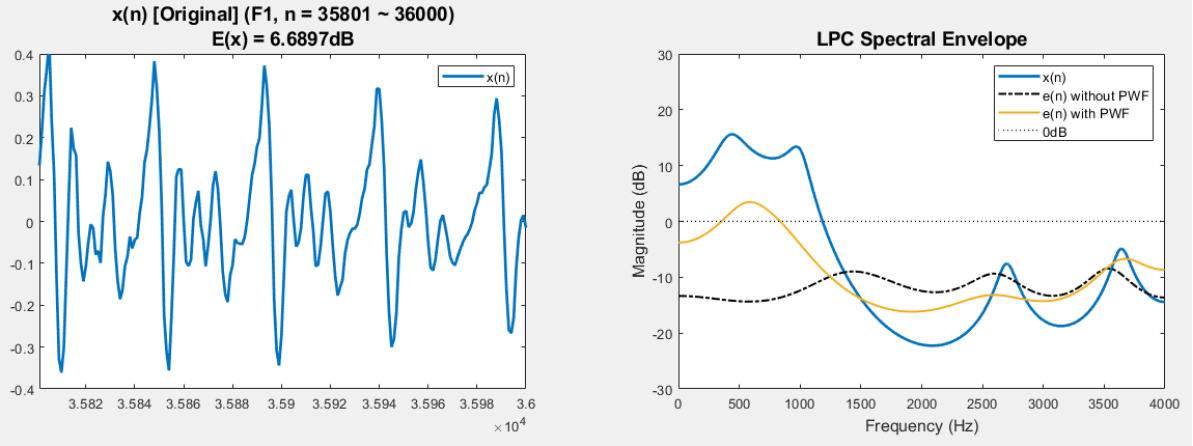
4. 3. Effect of PWF in Frequency Domain

The major role of the PWF is to shape a reconstruction error and accordingly result in better sound reproduction on average. To show how the filter ultimately shapes the noise, we picked up a certain input speech segment (n = 4701 ~ 4900) for the audio sample, F1, and analyzed the LPC spectral envelopes of the original signal, the reconstruction error signals without and with a PWF for our tree coder, and the results are shown in Figure 13.



< Fig 13. The Waveform of the Speech Segment (Left) and the LPC Spectral Envelopes (Right) of the Original Signal $x(n)$ and the Reconstruction Error $e(n)$ without and with PWF for the samples of F1 (n = 4701~4900) >

Without the PWF, the reconstruction error looks almost flat over the whole frequency range, and for some frequency bands, the error envelope is higher than the original speech envelope. This causes some audible distortions. However, when you look at the reconstruction error with the PWF, the error is shaped similar to the original signal's envelope and pushed down below the original envelope, and this shaping results in better reconstructed sound.



< Fig 14. The Waveform of the Speech Segment (Left) and the LPC Spectral Envelopes (Right) of the Original Signal $x(n)$ and the Reconstruction Error $e(n)$ without and with PWF for the samples of F1 (n = 35801 ~ 36000) >

The effect in Figure 13 may not happen for some voiced segments as shown in Figure 14 because the PWF's filter coefficients may not perfectly reflect the actual speech file's envelope. First of all, since the PWF coefficients are updated every 50 samples based on a block analysis, they may not perfectly match the actual real-time linear prediction coefficients in the very dynamic regions. Furthermore, since we fixed the shaping constants, μ and γ of the PWF transfer function, the set of values may not be suitable for some regions. We may get a better shaping result if we increase the order of the PWF to model more accurately the spectral envelope of the input signal, but this results in a longer filtering time, so the trade-off should be considered in actual application depending on the system's requirements.

Since the receiver has only a code generator and CNG module for signal synthesis for a single stream of an excitation signal, the PWF is not needed at a receiver because the effect of PWF has been already applied to the transmitted excitation signal which is generated by a tree coder at a transmitter.

Chapter 5

Tree Coding

5. 1. Introduction

In basic predictive coding with a quantizer, as a prediction error comes to a quantizer, a single output with the minimum distance to the input is generated, and the quantized error is directly transmitted to a receiver. In other words, only a single excitation sequence is explored by an encoder. The concept of Tree Coding is to explore multiple excitation sequence candidates while accepting some time delays to look ahead, and to send the best excitation sequence to a receiver.

Generally, Tree Coding allows myriad options in designing a tree structure, a tree search algorithm, a symbol release rule, and a code generator. In our Tree Coding, the tree is built by using switchable 5-level, 4-level, and 2-level trees constructed from adaptive scalar quantizers, so each tree branch is mapped to a single excitation value. As a tree grows, the number of tree branches to process rapidly increases, so the computational complexity goes beyond the system's capability soon. To control complexity, a full search of the tree must be avoided, and in our system, we adopt the (M, L) algorithm which limits the depth of the tree searching to L and the number of branches to keep for the next tree spanning to M. Finally,

for the best tree path with the minimum distortion, the number of symbols to be released is controlled by a symbol release rule.

In this chapter, the details of a tree design, a tree-complexity-control algorithm, and the symbol release rule we use in our Tree Coder are introduced.

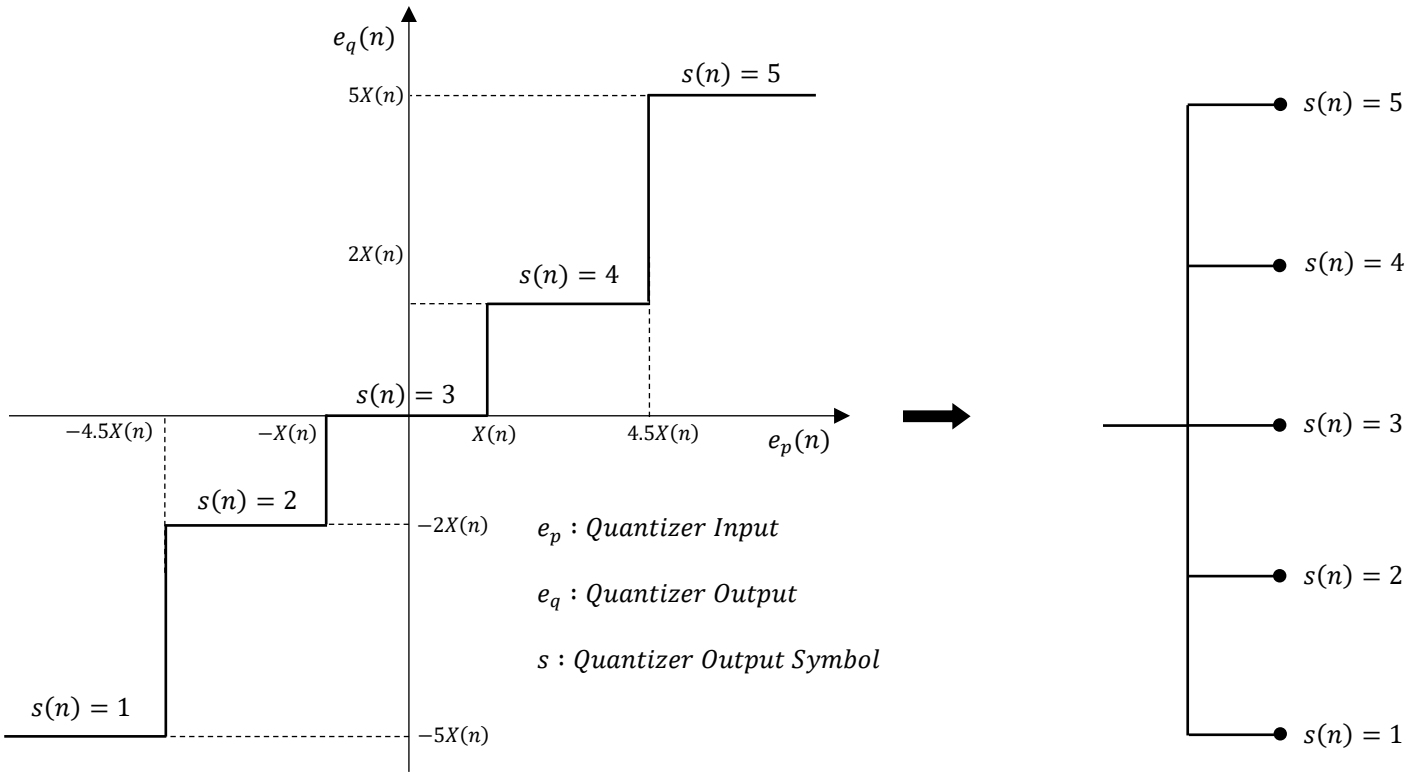
5. 2. Tree Design

A Tree Coder encodes the voiced speech segments classified as voiced by VAD. For the initial 140 samples (T_{SP}) of a voiced speech segment, a 5-level tree based on an adaptive Pitch-Compensating Quantizer (PCQ) [23] is used, and in this case, the code generator is composed of only a short-term predictor because there is no memory in the transition from Silence to Voiced Speech.

After 140 samples (T_{SP}), the remaining part of the voiced segment is encoded by the tree randomly interleaving 4-level and 2-level trees to excite a code generator with both long and short-term predictors. The following section shows how each tree is designed with a gain adaptation rule, and how we randomly order a 4-level and a 2-level tree.

5. 2. 1. 5-level Tree

We generate a 5-level tree modeled on a 5-level robust PCQ studied by Cohn and Melsa [23]. The PCQ rapidly responds to high peak factor (ratio of peak value to rms value) signals. For this reason, although it uses more levels than a 4-level or a 2-level tree, it tracks the initial part of a voiced segment with relatively sudden changes of the waveform causing a high peak factor. The 5-level PCQ is a mid-tread symmetric quantizer shown in Figure 15.



< Fig 15. 5-level PCQ (Left) mapping to 5-level Tree (Right) >

The quantity X(n) is recursively updated based on its current output as following [23]

$$X(n) = 2^{G(n)} \tag{47}$$

$$G(n) = GP(n) + C(n) + r \ \ where\, r = -11 \tag{48}$$

$$GP(n+1) = a_1\, GP(n) + f_1\big(s(n)\big) \ \ where\ a_1 = \frac{63}{64} \tag{49}$$

$$C(n+1) = a_2\, C(n) + f_2\big(s(n)\big) \ \ where\ a_2 = \frac{63}{64} \tag{50}$$

$$f_1\big(s(n)\big) = \begin{cases} -\dfrac{7}{128} & if\ s(n) = 3 \\ \dfrac{7}{64} & if\ s(n) = 2\ or\ 4 \\ -\dfrac{15}{64} & if\ s(n) = 1\ or\ 5 \end{cases} \tag{51}$$

$$f_2\big(s(n)\big) = \begin{cases} \dfrac{3}{4} & if\ s(n) = 1\ or\ 5 \\ 0 & Otherwise \end{cases} . \tag{52}$$

In the adaptation algorithm of a 5-level PCQ Tree, the fixed constants, r, a_1, a_2, $f_1(s(n))$, $f_1(s(n))$ are related to the adaptation performance, but r, a_1, a_2 from [23] did not perform well because of different dynamic ranges of audio sample files. For this reason, we re-optimized them from the sets $\{r = -15, -14, -5\}$, $\left\{a_1 = \frac{2^k-1}{2^k} \middle| k = 4,5,6,7,8,9\right\}$, $\left\{a_2 = \frac{2^k-1}{2^k} \middle| k = 4,5,6,7,8,9\right\}$ based on the PESQ-MOS performance.

In our actual simulation, we did not do source coding for each symbol, but we assume that ideal entropy coding is applied to the 5-level tree based on the empirical distribution of each symbol, $P_{PCQ}(s(n))$. Therefore, theoretically, the bits allocated to the symbol s(n) is $-P_{PCQ}(s(n)) \log_2 P_{PCQ}(s(n))$ bits, and accordingly, the average rate R is assumed to be the entropy of the distribution $P_{PCQ}(s(n))$

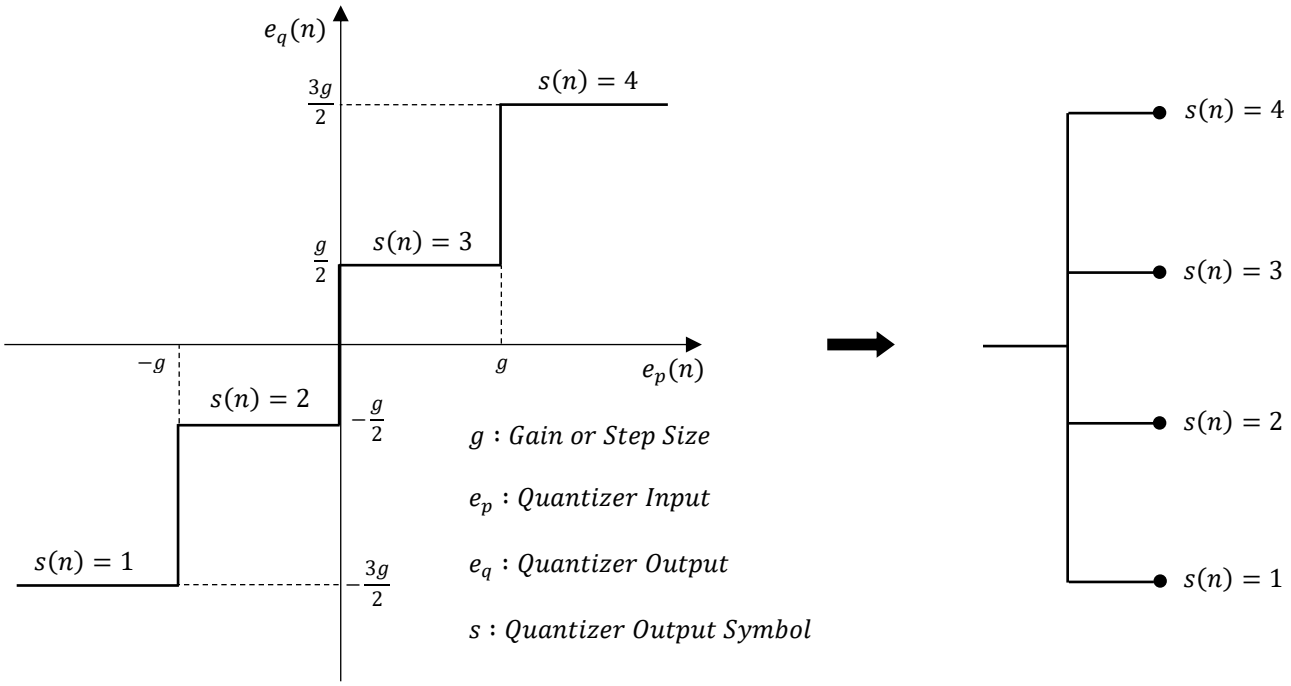
$$Average\ R_{5\text{-}level\ TC} = -\sum_{n=1}^{5} P_{PCQ}(s(n))\, log_2\, P_{PCQ}(s(n))\ \ bits/symbol \tag{53}$$

$$P_{PCQ}(s(n)) : Probability\ Distribution\ of\ s(n)$$

The 5-level tree at the beginning of Voiced segments not only responds quickly to a silence to Voiced transition but in turn it causes the backward adaptive parameter adaptation algorithms to rapidly begin to track the Voiced speech.

5. 2. 2. 4-level Tree

A 4-level tree is created by using a 4-level adaptive scalar quantizer [24] illustrated in Figure 16.



< Fig 16. 4-Level Adaptive Quantizer (Left) mapping to 4-level Tree (Right) >

The quantizer gain term $g(n)$ is also recursively updated by using a multiplier function $M(n)$ which takes the current output of a quantizer as an input [24]

$$g(n+1) = g^{\lambda_q}(n)\, M(n) \tag{54}$$

$$M(n) = \begin{cases} 0.8 & if\ s(n) = 2\ or\ 3 \\ 1.6 & if\ s(n) = 1\ or\ 4 \end{cases} \tag{55}$$

$$\lambda_q = \frac{127}{128}$$

If the quantizer's output is the inner level, it contracts by 0.8, and it expands by 1.6 if the output is the outer level. The product of the contraction multiplier 0.8 and the expansion multiplier 1.6 is 1.28 which is over 1 because the inner level has a higher probability. the leakage factor in the adaptation algorithm, $\lambda_q = \frac{127}{128}$, shows the best PESQ-MOS performance among the set $\left\{\lambda_q = \frac{2^k-1}{2^k} \middle| k = 4,5,6,7,8,9\right\}$.

Unlike the 5-level tree applying entropy coding to the source symbols, the 4 tree branch symbols are represented with 2 bits, so the rate is $R_{4\text{-}level\,TC} = 2\ bits/symbol$.

5. 2. 3. 2-level Tree

For the 2-level tree, we cannot use the same multiplier function as the 4-level tree because the only available information from the 2-level tree is the polarity of outputs. Therefore, we decided to use the polarity information of the past tree outputs for designing a gain adaptation algorithm. As the first step, we prepared a cascaded form of a short-term and a long-term predictor only without tree coding. By running the predictors for the voiced regions of our audio sample files, we generated a prediction error gain sequence based on the following equation

$$g_{No\,TC}(n) = \sqrt{\frac{e_p^2(n-3) + e_p^2(n-2) + e_p^2(n-1) + e_p^2(n)}{4}} \tag{56}$$

$$\begin{cases} If\ g_{No\,TC}(n) < g_{No\,TC}(n-1), & Contraction \\ If\ g_{No\,TC}(n) \geq g_{No\,TC}(n-1), & Expansion \end{cases} \tag{57}$$

From the prediction error gain, we calculate the empirical probabilities of contraction and expansion of the prediction error gain given the k-th polarity pattern group for the past 5 prediction error values $e_p(n-4), e_p(n-3), e_p(n-2), e_p(n-1), e_p(n)$.

Table 2. Probability of Contraction and Expansion of Gain for Prediction Error Polarity Sequence

Polarity Number	Group Index (k)	Prediction Error Polarity Sequence	Probability		
			P(k)	P(C|k)	P(E|k)
5 Polarities	1	+ + + + + - - - - -	13.1%	46.0%	54.0%
	2	- + + + + + - - - -	8.0%	37.5%	62.5%
	3	+ + - - - - - + + +	7.1%	52.5%	47.5%
	4	+ - + + + - + - - -	4.9%	48.4%	51.6%
	5	+ + + - - - - - + +	7.5%	41.6%	58.4%
	6	- + + - - + - - + +	6.3%	55.1%	44.9%
	7	+ + - + + - - + - -	6.4%	54.5%	45.5%
	8	+ - + - - - + - + +	4.4%	42.7%	57.3%
	9	+ + + + - - - - - +	8.0%	51.8%	48.2%
	10	- + + + - + - - - +	4.0%	52.8%	47.2%
	11	+ + - - + - - + + -	6.7%	51.9%	48.1%
	12	+ - + + - - + - - +	5.9%	49.0%	51.0%
	13	+ + + - + - - - + -	4.4%	51.7%	48.3%
	14	- + + - + + - - + -	6.3%	59.3%	40.7%
	15	+ + - + - - - + - +	4.4%	62.5%	37.5%
	16	+ - + - + - + - + -	2.6%	50.8%	49.2%

Table 2 shows the result for the statistics of the prediction gain contraction and expansion for each polarity pattern. In Table 2, the excitation polarity sequences with the same pattern of changes are grouped together, and $P(k)$ is the probability that the group "k" happens, and $P(C|k)$ and $P(E|k)$ mean the probability that the prediction error gain contracts and expands

given the polarity pattern group 'k'. From this table, if any higher probability of $P(C|k)$ and $P(E|k)$ is larger than 54%, we allocated the proper value of contraction or expansion multiplier value to the polarity pattern group "k" as shown in Figure 17. Otherwise, the multiplier is just set as 1.

$$Gain\ Update:\ g(n+1) = g^{\lambda_q}(n)\ M(n)$$

Pattern Group	Excitation Polarity Sequence					Multiplier Function M(n)
	n-4	n-3	n-2	n-1	n	
1	+	+	+	+	+	1.26
	-	-	-	-	-	
2	-	+	+	+	+	1.34
	+	-	-	-	-	
3	+	+	-	-	-	1
	-	-	+	+	+	
4	+	-	+	+	+	1
	-	+	-	-	-	
5	+	+	+	-	-	1.22
	-	-	-	+	+	
6	-	+	+	-	-	0.82
	+	-	-	+	+	
7	+	+	-	+	+	0.80
	-	-	+	-	-	
8	+	-	+	-	-	1.3
	-	+	-	+	+	
9	+	+	+	+	-	1
	-	-	-	-	+	
10	-	+	+	+	-	0.84
	+	-	-	-	+	
11	+	+	-	-	+	1
	-	-	+	+	-	
12	+	-	+	+	-	1
	-	+	-	-	+	
13	+	+	+	-	+	1
	-	-	-	+	-	
14	-	+	+	-	+	0.80
	+	-	-	+	-	
15	+	+	-	+	-	0.76
	-	-	+	-	+	
16	+	-	+	-	+	1
	-	+	-	+	-	

< Fig 17. Multiplier Function for 2-level Tree >

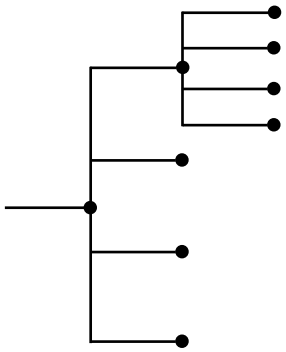
For the multiplier function table on the right side of Figure 17, the multiplier value for each group is empirically optimized in the range from 0.72 to 1 by 0.02 increment for contraction multiplier, from 1 to 1.38 by 0.02 increment for expansion multiplier. The leakage factor in

the adaptation algorithm, $\lambda_q = \frac{127}{128}$, shows the best PESQ-MOS performance among the set $\left\{\lambda_q = \frac{2^k-1}{2^k} \middle| k = 4,5,6,7,8,9\right\}$.

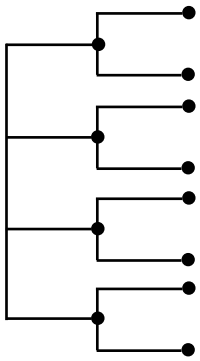
When we first built this algorithm, we started using only two past polarities as [5] [24], and as we increased the number of polarities from 2 through 5, we are likely to see the patterns with more unbalanced probability distributions making a multiplier function more meaningful. However, to keep the proper level of searching complexity, we decided to use 5 polarities without further increase. The rate for a 2-level tree is $R_{2\text{-}level\,TC} = 1\ bit/symbol$.

5. 2. 4. 4-2 Multitree

The application of the 2-level tree decreases the average rate of a tree coder, but we cannot use the 2-level tree only because the gain tracking performance is not good enough to track the prediction error alone. For this reason, we combine the 4-level and the 2-level tree as another tree structure unit and call the structure the 4-2 Multitree Coder (MTC) shown in Figure 3 [5].



< Fig 2. 4-level Tree >

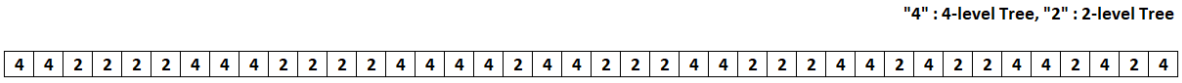


< Fig 3. 4-2 Multi-Tree >

For this 4-2 MTC, we can achieve the fraction rate $R_{4\text{-}2\,MTC} = \frac{2+1}{2} = 1.5\ bits/sample$.

5. 2. 5. Randomized 4-2 Multitree

We may simply repeat the 4-2 multitree for tree spanning with a fixed rate of 1.5 bits/symbol, but rather than imposing this periodicity of 4-2, we decide to randomly interleave the 4-level and the 2-level tree with equal probability to achieve the desired rate. Randomly choosing a 4-level or a 2-level tree every tree spanning for the full length of the utterances is not realizable in the actual application because the additional bits are needed to let a receiver know the random tree sequence to correctly update the gain term in the same way an encoder does. Instead, among all possible 40 sample tree sequences with a fixed rate of 1.5 bit/sample, we randomly chose 100 tree sequences and repeated until the end of the input speech signal and selected the best length 40 tree sequence with the highest average PESQ-MOS performance over our audio sample files. The best tree sequence of the 100 simulations is shown in Figure 18.



< Fig 18. The Best Tree Sequence (Length = 40) >

5. 3. (M, L) Algorithm

For a 4-level tree coder, the number of tree branches for depth k is 4^k which grows exponentially. Considering that a code generator adapts along each tree path, we need to limit the number of tree branches to an acceptable level of computational complexity. To do this, we apply a (M, L) algorithm to our tree coder [13]. According to the algorithm, when the number of the tree branches is larger than M, only the best M tree branches based on distortions survive for the next tree spanning, so other branches are pruned. In addition, when the tree depth reaches L, it releases the symbol(s) of the best tree path with the minimum distortion according to a chosen symbol release rule. For our RAR MTC codec, M = 8 and L=10 are used, and its resultant encoding time delay for L = 10 is 1.25 ms.

5. 4. Symbol Release Rule

The (M, L) algorithm lets the tree span up to the depth L while limiting the maximum tree branches to M, and once it reaches the depth L, it has to release the symbol(s) of the optimal path. In our initial experiment, we set our tree coder to release the first single symbol of the best path, but later, we adopted a variable symbol release (VSR) rule [25] to avoid undesirable path switching with a single symbol release (SSR) rule. According to [25], the VSR rule has the ability to distinguish between desirable and undesirable path switching and may avoid the latter by sending a variable number of symbols. Furthermore, by sending multiple symbols, we can expect a lower computational load compared to the SSR.

According to a VSR algorithm [25], a tree coder finds the best path with the minimum perceptually weighted distortion and releases the symbols mapping to the perceptually weighted reconstruction error $e_w(n), e_w(n+1), \ldots, e_w(n+j)$ until j satisfies

$$\frac{1}{j}\sum_{k=1}^{j} e_w^2(n+k) \le D_L = \frac{1}{L}\sum_{k=1}^{L} e_w^2(n+k) \quad where \;\; L\,(the\; Max.Search\; Depth) = 10 \qquad (58)$$

$$j \le \frac{L}{2}$$

In our actual simulations for our audio sample files with SSR and VSR, VSR was not always better than SSR, and it implies that VSR cannot perfectly filter the undesirable path better than SSR, but given that their average performances are quite close to each other, the VSR has the advantage for lower computation load.

Chapter 6

Performance of RAR MTC

6. 1. Introduction

Although our tree coder can be applicable for narrow-band or wide-band speech signals, the parameters of our tree coder have been optimized for narrow-band speech signals without background noise, but the final version of our codec was tested for the speech signals with multiple types of background noises too. Our speech codec performance was measured by two criteria, a PESQ-MOS score and an average bit rate. So in this chapter, we introduce what narrow-band speech signals and background noises have been used for our tests and the details of performance measurement criteria, and finally, we will see how well our codec performs well compared to a popular codec, AMR-NB. All simulations are performed by using MATLAB.

6. 2. Test Audio Files

For the design of our codec, we used speech files from Recommendation ITU-T P.501 [26] that includes a wide range of test signals recorded in different languages applicable for several purposes in telephony and other speech-based applications. Among the whole set of test files, we picked a set of the English-spoken test files sampled at 8 kHz as our narrow band speech inputs, and the list of the files is given in Table 3.

Table 3. Design Speech Sample Group from ITU-T P.501

	Original Reference File Name	Sentence
Female Voice	P501_C_english_f1_flat_08k.wav	These days a chiecken leg is a rare dish. The hogs were fed chopped corn and garbage.
	P501_C_english_f2_flat_08k.wav	Rice is often served in round bowls. A large size in stockings is hard to sell.
Male Voice	P501_C_english_m1_flat_08k.wav	The juice of lemons makes fine punch. Four hours of steady work faced us.
	P501_C_english_m2_flat_08k.wav	The birch canoe slid on the smooth planks. Glue the sheet to the dark blue background.

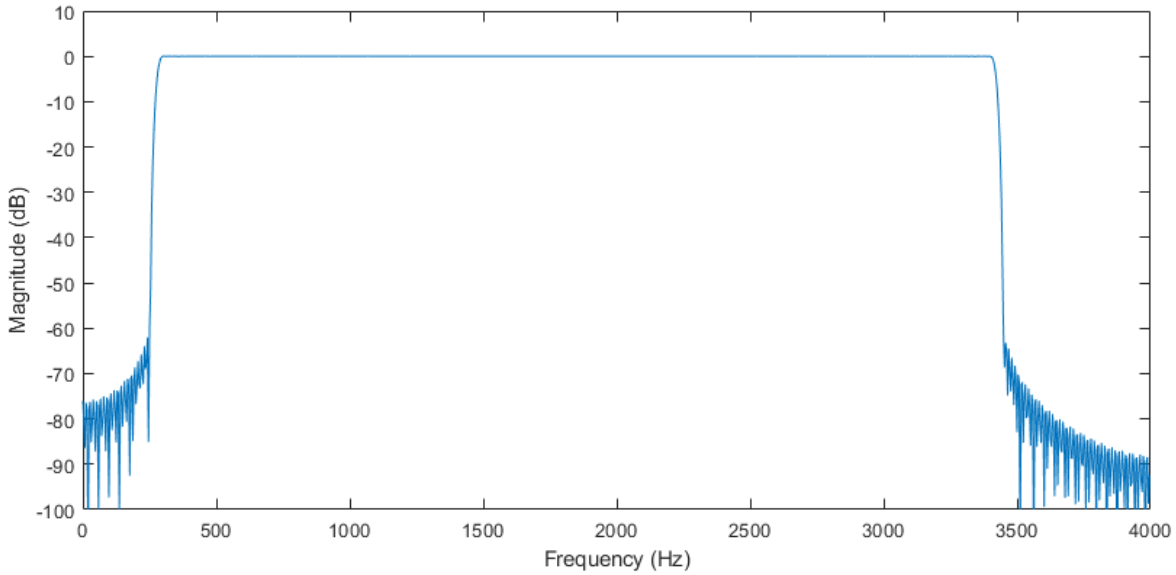
※ Reference : ITU-T P.501, Original Sampling Rate : 8 kHz, Bit Depth : 16 bits/sample, Bitrate : 128 kbps

BPF (300 ~ 3400 Hz)

Index	Modified File Name
F1	F1_BPF_og.wav
F2	F2_BPF_og.wav
M1	M1_BPF_og.wav
M2	M2_BPF_og.wav

Although the above sample files are sampled at 8 kHz, they have strong energy outside the desired narrowband telephone frequency band (300 ~ 3400 Hz), so we additionally pass them through a band pass filter (BPF) by using the default MATLAB BPF filter design function, “bandpass(x, fpass, fs)”. For the input values of the function, x has been sampled at a rate of fs Hz, and the two-element vector fpass specifies the passband frequency range of the filter in Hz. When we set fpass = [300, 3400] and fs = 8000, the filter is automatically designed as the 616-th order of FIR BPF with the passband frequency 300 ~ 3400 Hz, 0.1 dB of passband ripple, 60dB of stopband attenuation. The frequency response of the BPF is shown in Figure

19. The same BPF filter is applied to the output of our codecs too so that we can remove the unwanted frequency components arising from reconstruction process.



< Fig 19. Frequency Response of the MATLAB Band Pass Filter "bandpass(x,[300,3400],8000)" >

Various fixed parameters related to our codec were optimized for the band-pass-filtered sample files in Table 3, and then, we tested our speech codec on more audio samples from ITU-T P.501 outside of the design set to get more generalized result. Since we could not find more narrowband signals from the standard audio sample set, we chose a certain set of original high-definition English audio sample files sampled at 48 kHz and converted them into narrowband signals sampled at 8 kHz by passing them through the above BPF and down-sampling by 6. Table 4 shows the group of the test speech files we converted, 8 female voice files and 7 male voice files. Since the original reference files are composed of multiple sentences with a long period of silences between sentences, we segmented the whole sequence

into several pieces of a single sentence. Like we did for the first set of audio sample files, the output for the second set of these sentences are filtered by a BPF too.

Table 4. Non-Design Speech Sample Group from ITU-T P.501

	Original Reference File Name	Sentence
Female Voice	FB_female_conditioning_seq_long.wav	Port is a strong wine with a smoky taste.
	FB_female_conditioning_seq_long.wav	Pluck the bright rose without leaves.
	FB_female_conditioning_seq_long.wav	Clothes and Loding are free to new men.
	FB_female_conditioning_seq_long.wav	A cramp is no small danger on a swim.
	FB_male_female_single-talk_seq.wav	We frown when events take a bad turn.
	FB_male_female_single-talk_seq.wav	Rice is often served in round bowls.
	FB_male_female_single-talk_seq.wav	The hogs were fed chopped corn and garbage.
	FB_male_female_single-talk_seq.wav	He knew the skill of the great young actress
Male Voice	FB_male_conditioning_seq_long.wav	A pencil with black lead writes best.
	FB_male_conditioning_seq_long.wav	The shaky barn fell with a loud crash.
	FB_male_conditioning_seq_long.wav	They took the axe and the saw to the forest.
	FB_male_female_single-talk_seq.wav	The last switch cannot be turned off.
	FB_male_female_single-talk_seq.wav	It's easy to tell the depth of a well.
	FB_male_female_single-talk_seq.wav	Four hours of steady work faced us.
	FB_male_female_single-talk_seq.wav	Jazz and swing fans like fast music.

BPF (300 ~ 3400 Hz) & Downsample by 6

Index	Modified File Name
F3	F3_BPF_og.wav
F4	F4_BPF_og.wav
F5	F5_BPF_og.wav
F6	F6_BPF_og.wav
F7	F7_BPF_og.wav
F8	F8_BPF_og.wav
F9	F9_BPF_og.wav
F10	F10_BPF_og.wav
M3	M3_BPF_og.wav
M4	M4_BPF_og.wav
M5	M5_BPF_og.wav
M6	M6_BPF_og.wav
M7	M7_BPF_og.wav
M8	M8_BPF_og.wav
M9	M9_BPF_og.wav

※ Reference : ITU-T P.501, Original Sampling Rate : 48 kHz, Bit Depth : 16 bits/sample, Bitrate : 768 kbps.

After testing our codec for the above first and second groups of speech samples, we repeat the simulations with diverse background noises in Table 5 from the set of first responder background sounds [27] to test the robustness of our codec. Since these background noises are also HD audio files sampled at 48 kHz with 16-bit audio bit depth like the second speech sample group, we passed them through a BPF and down-sampled by 6 for conversion.

Table 5. Test Background Noise Files

Original Reference File Name
Fire_Truck_Siren_In_Cab_Yelp_106.5dBpk_99.5dBAvrg_@_5ft_from_Front_Console.wav
Fire_Truck_Siren_Outside_Wail_132dBpk_@_5ft_from_Siren.wav
K12_Saw_Cutting_Metal_Garage_Door_Sample_1_(loop)_120dBpk_107dBAvrg_@_5ft.wav
PASS_Alert_PASS_Alert_Scott_Airpack_x1_Backside_Closed_Room_111dBpk_@_6in.wav
Pneumatic_Chisel_Cutting_Metal_Door_6_Samples_123dBpk_108dBAvrg_@_5ft.wav
Scott_VibraAlert_25_Knocks_per_sec_-28dBov_16bit.wav
Water_Flow_1_3_4in_Hose_Fog_Nozzle_In_Building_106.5dBpk_95dBAvrg_@_6ft.wav

BPF (300 ~ 3400 Hz) & Downsample by 6

Index	Modified File Name
BN1	BN1_BPF_og.wav
BN2	BN2_BPF_og.wav
BN3	BN3_BPF_og.wav
BN4	BN4_BPF_og.wav
BN5	BN5_BPF_og.wav
BN6	BN6_BPF_og.wav
BN7	BN7_BPF_og.wav

※ Original Sampling Rate : 48 kHz, Bit Depth : 16 bits/sample, Bitrate : 768 kbps.

When we add the noise files to speech files for simulation, the noise sequence is cut to the same length as the speech file, and the power level of the truncated noise sequence is normalized depending on the speech file's power level so that the Speech-to-Background-Noise Ratio (SBNR) is equal to 3 dB

$$SBNR = 10\log\frac{P_x}{P_{BN_{norm}}} = 3\ dB \tag{59}$$

$$P_x : Power\ of\ Speech\ Signal$$

$$P_{BN_{norm}} : Power\ of\ Normalized\ Background\ Noise\ Signal$$

6. 3. Performance Evaluation Methods

To measure speech quality, we use a standardized software tool, Perceptual Evaluation of Speech Quality (PESQ) [28] which is an objective evaluation method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, and we adopted this method for our analysis. The perceptual model of PESQ is used to calculate a distance between the original and recovered speech signal which is described as a PESQ score, and the PESQ score is converted to a Mean Opinion Score (MOS)-like scale in the range of -0.5 to 4.5, so we simply call this output a PESQ-MOS score. The details of the structure of this software can be found in [28].

Along with the sound quality, another important criterion is the average bit rate of our codec. The best is to have a good sound quality with a lower bit rate. Since we adopt a VAD/CNG in our speech codec, the average bit rate is dependent on the ratio of "Silence" and "Unvoiced" regions over "Voiced" regions. Furthermore, we need to consider that, in

"Voiced" regions, our tree is populated with a 5-level tree for the initial 140 samples of a voiced segment with assumption that the 5-level tree symbols are ideally entropy-coded, and then the optimally randomized 4-2 multitree is used for the rest voiced samples. After considering all of these, the final equation for the average bit rate of the output is given below

$$Average\ Bitrate = \frac{N_{5\text{-}level\ TC}\ R_{5\text{-}level\ TC} + N_{4\text{-}2\ MTC}\ R_{4\text{-}2\ MTC} + S_{SID}\ N_{SID}}{N_{total}\ T_s} \qquad (60)$$

$$N_{5\text{-}level\ TC} : Total\ Number\ of\ Samples\ coded\ by\ a\ 5\text{-}level\ Tree$$

$$N_{4\text{-}2\ MTC} : Total\ Number\ of\ Samples\ coded\ by\ a\ randomized\ 4\text{-}2\ MTC$$

$$N_{SID} : Total\ Number\ of\ Transmitted\ SID\ Frames$$

$$S_{SID} : The\ size\ of\ a\ SID\ frame\ for\ a\ CNG\ parameter\ update\ (= 53\ bits)$$

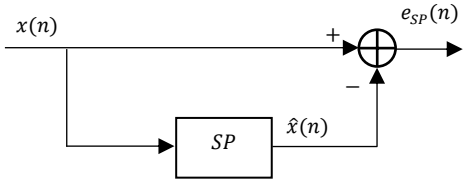
$$N_{Total} : Total\ Number\ of\ Speech\ Samples$$

$$T_s : Sampling\ Period\ \left(= \frac{1}{8000}\,\mathrm{s}\right)$$

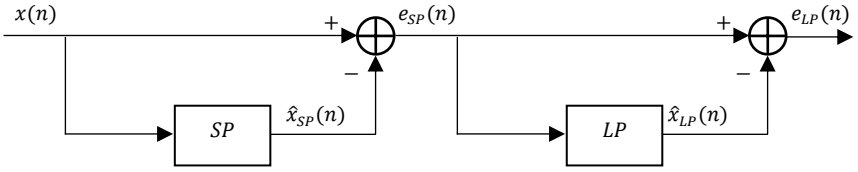
6. 4. Performance Analysis

6. 4. 1. Simulations for Short-term Prediction Order and Code Generator Sequence of a Forward Adaptive ADPCM without VAD/CNG

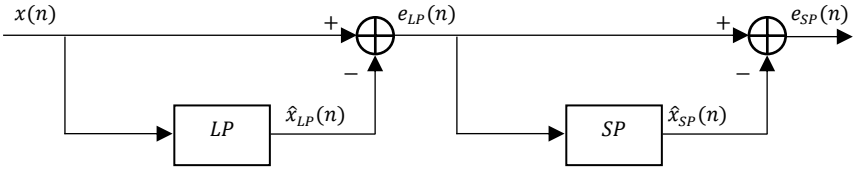
The initial simulation in this research was fulfilled to find the proper prediction order to be used in a code generator. For a linear predictor, although the higher prediction order generally gives a better Signal-to-Prediction-Error Ratio (SPER) performance, it costs a higher complexity of computation in the algorithms. Furthermore, it is needed to test how the addition of the long-term predictor increases the resultant prediction performance and how the sequence of the short-term and long-term predictors combined in a code generator affects the performance. For these reasons, we prepared three different models of ADPCM algorithms without VAD/CNG, Tree Coding, and a quantizer.



< Figure 20.1. SP Only without Quantizer >

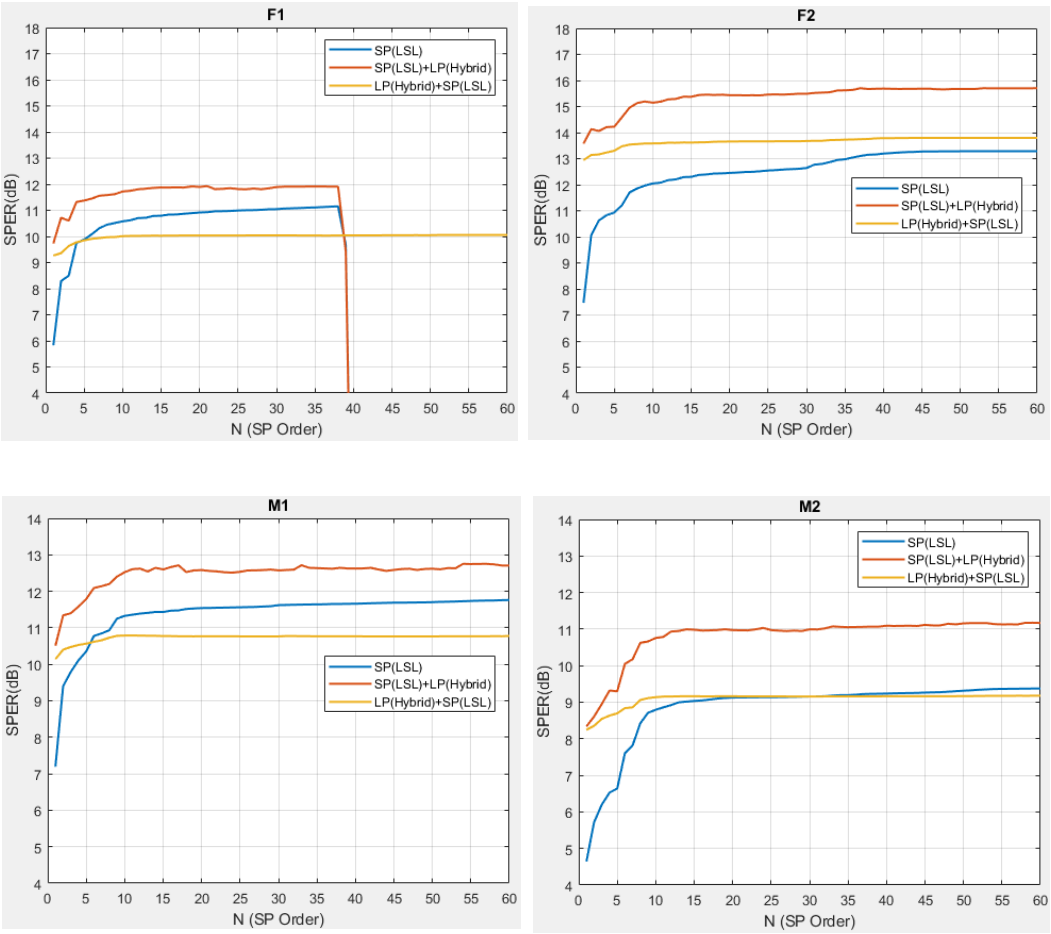


< Figure 20.2. SP + LP without Quantizer >



< Figure 20.3. LP + SP without Quantizer >

The first model in Fig. 20.1 is only the short-term predictor, and the second in Fig. 20.2 is the short-term predictor followed by the long-term predictor, and the Fig. 20.3, the third model, is the reverse of the second, the long-term predictor followed by the short-term predictor.



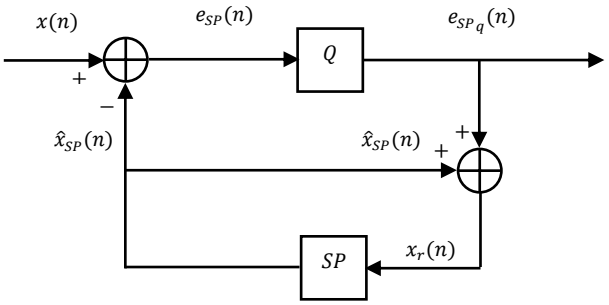
< Fig 21. SPER Performances for SP, SP+LP, LP+SP depending on SP Order >

For these models, we varied the short-term prediction order from 1 to 60 and calculated SPER (dB) performances, and the result is shown in Fig. 21. For all tested models, SPER performance increases as the prediction order increases, but the increment gradually decreases and flattens out. In addition, the second model, SP + LP in Fig. 20.2, shows the highest performance among the tested three models. Interestingly, the third model, LP + SP in Fig.

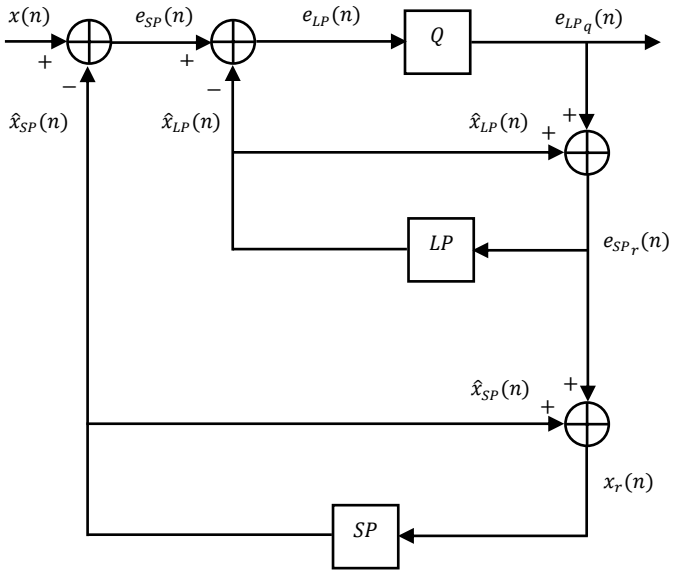
20.3, shows the second performance for a lower prediction order, but for F1, M1, and M2, as the prediction order increases, SP only in Fig. 20.1 gradually starts to outperform the third model, LP+SP.

6. 4. 2. Simulations for Short-term Prediction Order and Code Generator Sequence of a Backward Adaptive ADPCM without VAD/CNG

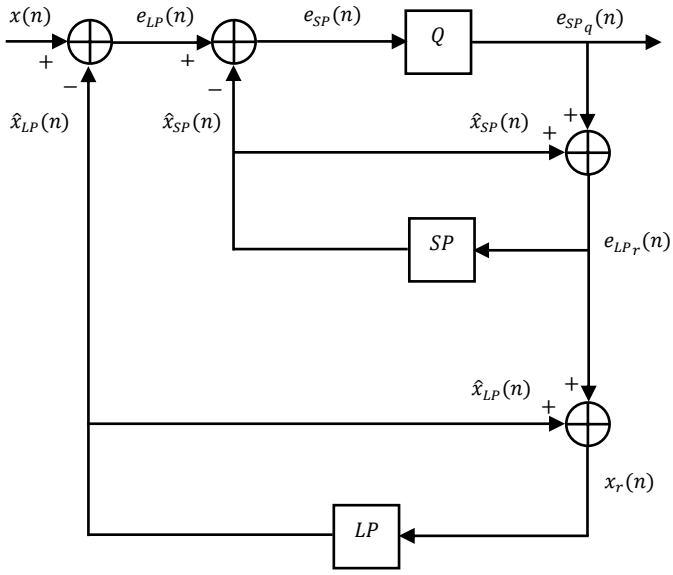
Since these models in Sec. 6.4.1 were based on pure predictors without a quantizer, we added a quantizer to these models.



< Figure 22.1. SP Only with Quantizer >

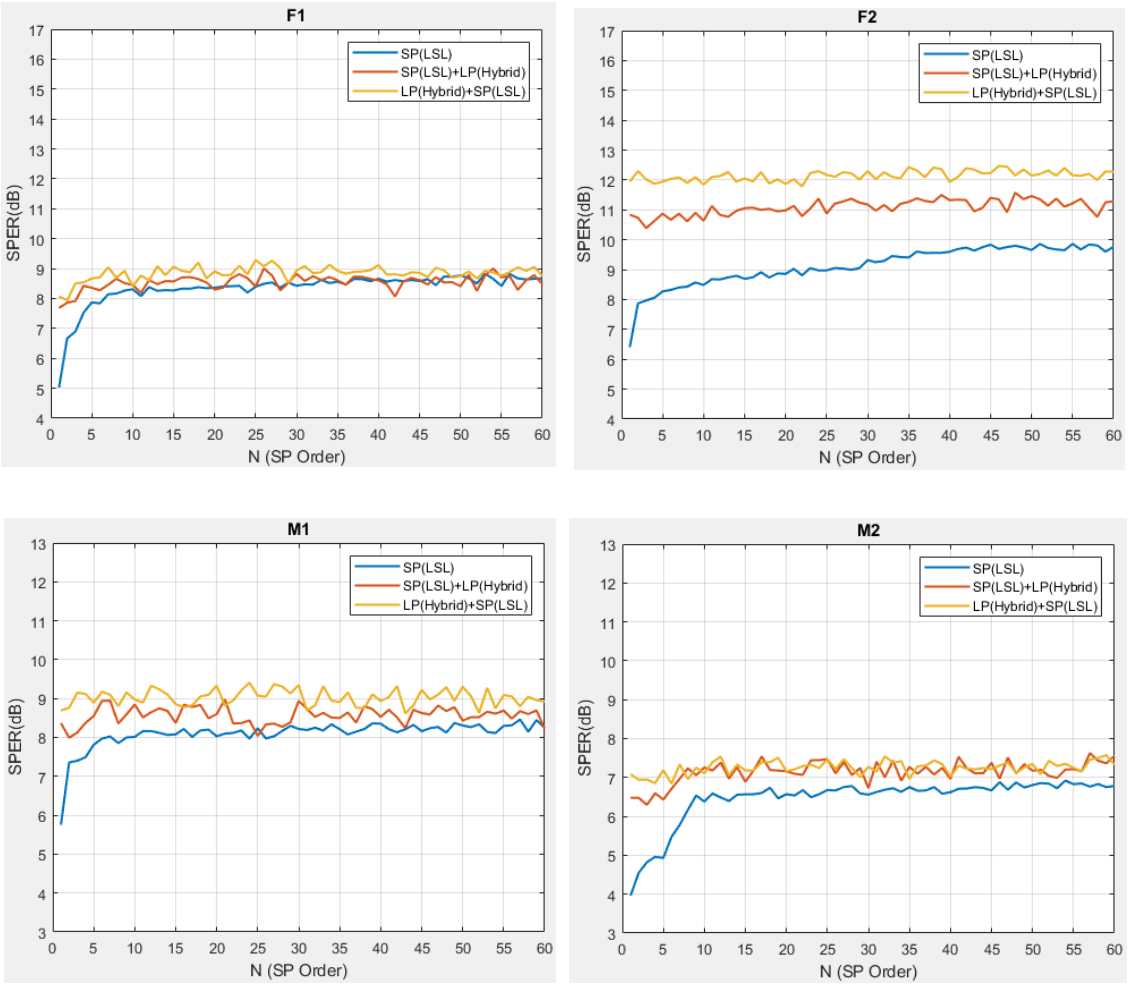


< Figure 22.2. SP + LP with Quantizer >



< Figure 22.3. LP + SP with Quantizer >

The block diagrams for the three models with a quantizer are shown in Fig. 22.1~22.3, and the same simulation for different prediction orders was performed.



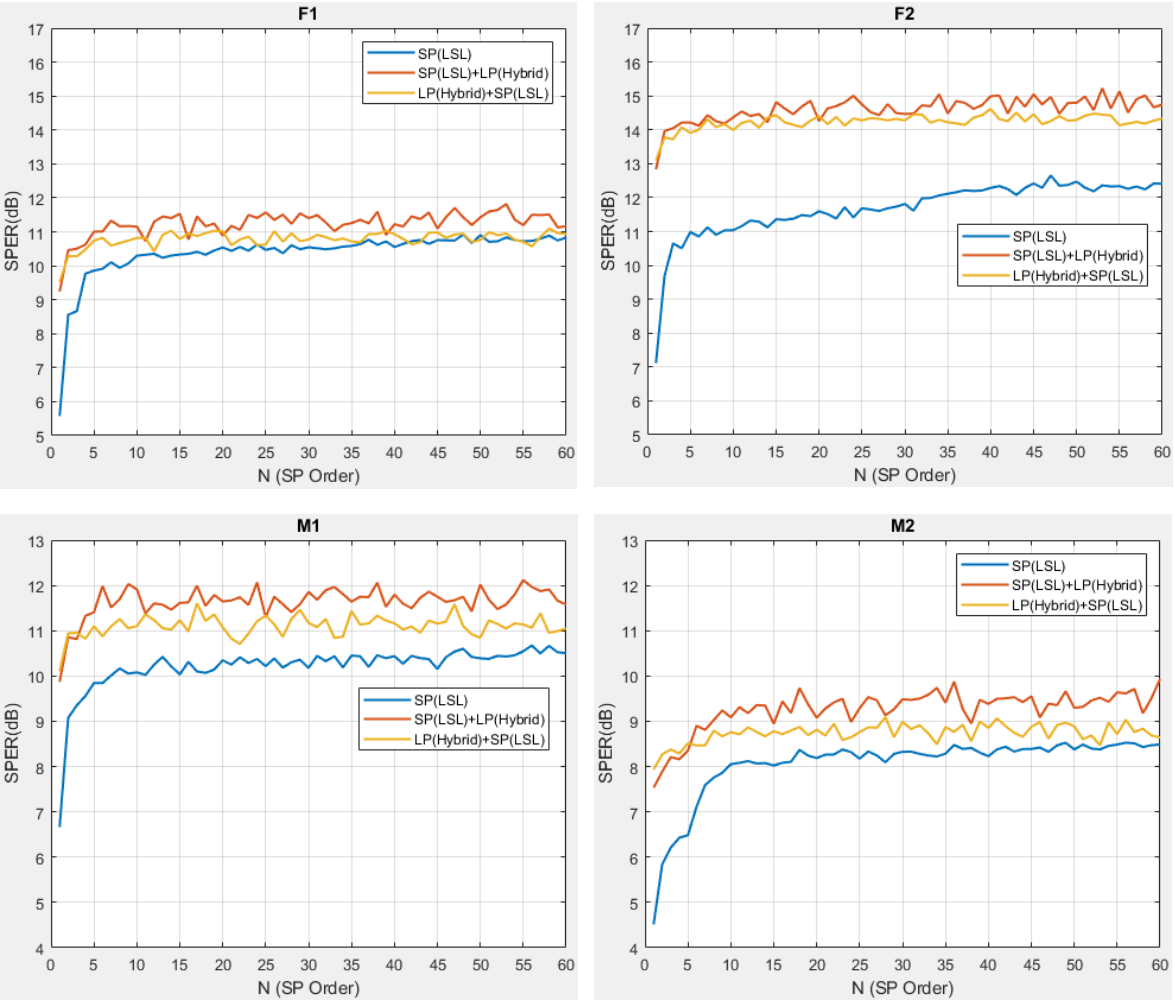
< Fig 23. SPER Performances for SP, SP+LP, LP+SP with a Quantizer depending on SP Order >

The simulation performance graphs with a quantizer are shown in Fig. 23, and it reveals several different results compared to the previous simulation without a quantizer. First of all, SP + LP or LP + SP, they are mostly better than a single SP. Furthermore, LP + SP performs better than SP + LP for all samples except for M2 which shows almost similar performances for higher SP orders. In common, the previous and the current simulation show that their

performances for all models get saturated as the SP order goes over around 10, and this led us to decide to use the 10-th order short-term predictor.

6. 4. 3. Simulations for Short-term Prediction Order and Code Generator Sequence of a Tree Coder without VAD/CNG

To make a better decision about whether we use SP + LP or LP + SP, we additionally applied tree coding based on only the 4-level tree with the rate, R = 2 bits/sample.



< Fig 24. SPER Performances for SP, SP+LP, LP+SP with Tree Coding depending on SP Order >

As shown in Fig. 24, the application of Tree Coding has increased their SPER performances over all values of a SP order compared to no Tree Coding, and unlike the previous result, SP + LP outperforms LP + SP for all audio sample files. Since Tree Coding is one of our codec's main features, SP + LP is finally chosen as our code generator structure.

6. 4. 4. Simulations of 5-4-2 Randomized Multi-Tree Coder with VAD/CNG and PWF with Non-Quantized CNG Parameters

As a next step, we upgraded the 4-level tree coder to the 4-2 randomized Multi-Tree Coder (MTC) with VAD/CNG and PWF to additionally lower the bit rate while keeping a good quality of sound. At this moment, when the 1-bit tree is used to span tree branches in the 4-2 randomized MTC, it did not adapt its gain value depending on the 1-bit tree's output, and when CNG parameters were generated, the encoder transmitted the parameters without quantization in the simulation. The codec at this stage is simply called Type 1. The main problem of the Type 1 codec is the poor encoding performance at the start of each voiced segment. The initial part of the voiced segment is quite short and low-energy so it is difficult for the adaptive predictors to quickly restart the adaptation of the prediction parameters. This led us to make our codec perform with only PCQ for the initial 120 samples of a voiced segment. Let's call this codec Type 2. Later then, the 1-bit tree not updating a gain term needed to be upgraded with a new gain adaptation algorithm, so we finally devised and applied the new gain adaptation algorithm for 1-bit tree based on the polarity sequence of the past five excitation values. This final version of codec is called Type 3.

Table 6. Performance for Type 1 ~ 3

	Test Model	Performance	F1	F2	M1	M2	Avg	Std Dev
Type 1	Random 4-2 MTC	PESQ-MOS	3.377	3.376	3.443	3.556	3.438	0.073
		Rate (kbits/s)	5.92	5.57	4.09	5.41	5.25	
Type 2	Random 4-2 MTC + 5-level PCQ	PESQ-MOS	3.425	3.515	3.544	3.569	3.513	0.054
		Rate (kbits/s)	6.03	5.66	4.15	5.49	5.33	
Type 3	Random 4-2 MTC + 5-level PCQ, 5-pol	PESQ-MOS	3.471	3.562	3.584	3.599	3.554	0.05
		Rate (kbits/s)	6.03	5.66	4.15	5.49	5.33	

We tested Type 1 ~ 3 for our design audio samples, F1, F2, M1, M2, which have almost 50 percent silence or unvoiced regions categorized by our VAD module, and the result is displayed in Table 6. The addition of the 5-level PCQ from Type 1 to Type 2 has increased PESQ-MOS. Moreover, the new 1-bit tree gain adaptation rule based on the 5-polarity adaptation rule in Figure 14 has additionally advanced our codec's performance as evident from Table 6.

Moreover, considering that voiced regions are encoded by a 4-2 tree coder at the bit rate of 12 kbps, VAD/CNG has dropped the average bit rate for whole utterances to about half of 12 kbps for our design audio samples, and the average bit rate is dependent on the portion of voiced regions for each input. As we mentioned earlier that our VAD/CNG has a simple structure to briefly explore its effect in a speech codec, our VAD/CNG's classification and noise generation performance might not be as good as other recent standardized speech codecs' VAD/CNG algorithms. To focus on the voice recovery quality in voiced regions only, we extracted the recovered speech signals for voiced regions only and concatenate them and pass the newly generated signal through the PESQ-MOS software. For a comparison to the AMR-NB codec, we also did the same extraction and concatenation process for the

reconstructed speech from the AMR-NB codec in the same voiced regions classified by our VAD module, and the final result is given in Table 7.

Table 7. Performance for Type 1 ~ 3 vs AMR in Voiced Regions Only

	Test Model	Performance	F1	F2	M1	M2	Avg	Std Dev
Type 1	Random 4-2 MTC	PESQ-MOS	3.646	3.581	3.784	3.699	3.678	0.074
		Rate (kbits/s)	12	12	12	12	12	
Type 2	Random 4-2 MTC + 5-level PCQ	PESQ-MOS	3.747	3.798	4.022	3.793	3.84	0.107
		Rate (kbits/s)	12.24	12.24	12.24	12.24	12.24	
Type 3	Random 4-2 MTC + 5-level PCQ, 5-pol	PESQ-MOS	3.766	3.827	4.039	3.819	3.863	0.104
		Rate (kbits/s)	12.24	12.24	12.24	12.24	12.24	
	AMR-NB	PESQ-MOS	4.04	4.001	4.089	4.063	4.048	0.032
		Rate (kbits/s)	12.2	12.2	12.2	12.2	12.2	

Among diverse running modes of AMR-NB, we used a 12.2 kbps of AMR-NB because is has the closest bit rate to our codec's bit rate, so Table 7 shows that AMR-NB's bit rate is slightly less than our type 3 model by about 0.04 kbps. From Table 7, we see the overall PESQ-MOS scores in voiced regions only have improved by almost 0.3 compared to the result for the whole utterances. Furthermore, we can see the same positive effect of the 5-level PCQ and the new gain control algorithm based on 5-polarity for the 2-bit tree like Table 6. Although the advancement for using the 5-polarity gain control algorithm is relatively small, it still shows plus effect for all tested audio samples. The most important result is that our final version of the model, Type 3, now has the PESQ-MOS scores higher than 3.5 which is considered Good, and even more than 3.7 up to slightly over 4.0 which is considered Very Good. Although the AMR-NB 12.2 kbps mode consistently shows the PESQ-MOS scores slightly over 4.0, for Male 1, our Type 3 codec has almost equivalent performance as the AMR-NB 12.2 kbps.

The Type 3 model has been optimized for the design set of audio samples, F1, F2, M1 and M2, and we expanded the same test in voiced regions for the samples outside the design set in Table 4.

Table 8-1. Performance for Type 1 ~ 3 vs AMR in Voiced Regions Only For Additional Female Samples

	Test Model	Performance	F4	F5	F6	F7	F8	F9	F10	F11	Avg	Std Dev
Type 1	Random 4-2 MTC	PESQ-MOS	3.826	3.543	3.597	3.577	3.602	3.54	3.634	3.669	3.624	0.087
Type 2	Random 4-2 MTC + 5-level PCQ	PESQ-MOS	3.878	3.769	3.632	3.512	3.721	3.677	3.624	3.787	3.7	0.106
Type 3	Random 4-2 MTC + 5-level PCQ, 5-pol	PESQ-MOS	3.861	3.772	3.639	3.514	3.733	3.68	3.638	3.79	3.703	0.102
	AMR-NB	PESQ-MOS	3.978	3.96	3.721	3.818	3.923	3.634	3.954	3.697	3.836	0.128

Table 8-1 shows the result for the additional female samples, and we see that the addition of 5-level PCQ has mostly increased the performance except for F7 and F10. From Type 2 to Type 3, the 5-polarity-based gain control algorithm have slightly increased the performance except for F4. Although AMR-NB shows better performance on average than our Type 3 by 0.133, but for F9 and F11, our Type 3 codec performs better.

Table 8-2. Performance for Type 1 ~ 3 vs AMR in Voiced Regions Only For Additional Male Samples

	Test Model	Performance	M4	M5	M6	M7	M8	M9	M10	Avg	Std Dev
Type 1	Random 4-2 MTC	PESQ-MOS	3.645	3.684	3.856	3.662	3.796	3.848	3.828	3.76	0.086
Type 2	Random 4-2 MTC + 5-level PCQ	PESQ-MOS	3.806	3.792	3.877	3.723	3.801	3.878	3.793	3.81	0.05
Type 3	Random 4-2 MTC + 5-level PCQ, 5-pol	PESQ-MOS	3.804	3.797	3.88	3.736	3.806	3.875	3.808	3.815	0.046
	AMR-NB	PESQ-MOS	3.984	4.091	3.973	3.902	3.824	3.745	4.052	3.939	0.114

Similar performance results are shown in Table 8-2 for the additional male samples. Type 2 outperforms Type 1 except for M10, and Type 3 slightly performs better than Type 2 except for M4 and M9. AMR-NB is still showing higher scores than Type 3 except for M9.

When we compare the results for the female and male samples, our codec and AMR show lower PESQ-MOS scores for female audio samples on average. Since female voice generally has a shorter pitch period, there are more pitch periods in a frame that are averaged together for fixed-length-block-based coders such as AMR. For our codec, the long-term redundancy from pitch is hard to precisely track by our backward adaptive algorithm, so additional pitch pulses in a shorter period of time create tracking difficulties [29]. However, it does not mean that the speech codecs always perform better for all male speakers because other factors except for pitch period may be involved with encoding performances.

Another fact in comparison with the design set of audio samples, the new set of audio samples show a little bit lower performance than the design set of samples for our codecs and AMR-NB 12 kbps.

6. 4. 5. Simulations of 5-4-2 Randomized Multi-Tree Coder with VAD/ CNG and PWF with Quantized CNG Parameters

As we mentioned earlier, the Type 3 codec in the previous section is not perfectly practical because it does not quantize the CNG parameters which are transmitted to a receiver for CNG process, so we apply a quantization scheme to the CNG parameters as a next step.

Table 9. Quantization of CNG Parameters for Type 3

Test Model		Design Set (Whole Utterance)				Avg.
		F1	F2	M1	M2	
Random 4-2 MTC + 5-level PCQ, 5-pol	1) Non-quantized PARCORs 2) Non-quantized Residual Energy	3.471	3.562	3.584	3.599	3.554
	1) Quantized PARCORs (Total 48 Bits) : Bit Allocation for PARCORs = [6, 6, 6, 6, 6, 6, 6, 6] : Dynamic Range = [1, 1, 1, 1, 1, 1, 1, 1] 2) Non-quantized Residual Energy	3.447	3.499	3.534	3.586	3.517
	1) Quantized PARCORs (Total 48 Bits) : Bit Allocation for PARCORs = [8, 8, 6, 6, 5, 5, 5, 5] : Dynamic Range = [1, 1, 1, 1, 1, 1, 1, 1] 2) Non-quantized Residual Energy	3.457	3.502	3.545	3.582	3.522
	1) Quantized PARCORs (Total 48 Bits) : Bit Allocation for PARCORs = [8, 8, 6, 6, 5, 5, 5, 5] : Dynamic Range = [0.97, 0.97, 0.82, 0.92, 0.74, 0.82, 0.58, 0.73] 2) Non-quantized Residual Energy	3.467	3.502	3.547	3.590	3.527
	1) Quantized PARCORs (Total 48 Bits) : Bit Allocation for PARCORs = [8, 8, 6, 6, 5, 5, 5, 5] : Dynamic Range = [0.97, 0.97, 0.82, 0.92, 0.74, 0.82, 0.58, 0.73] 2) Quantized Residual Energy : Bit Allocation = 5 bits	3.456	3.460	3.496	3.561	3.493

The CNG parameters are composed of 8 PARCORs and the residual energy. Rather than quantizing them all at once, we decided to quantize the PARCORs first by finding the proper quantization bit depth and dynamic range for each PARCOR. Table 9 shows the results for the sequential simulations to arrive at the final CNG parameter quantization scheme. From the non-quantized CNG parameters, we first applied the uniform quantizers for PARCORs with the same bit depth and dynamic range as shown in the second row of Table 9. Based on the

fact that the lower-order coefficients of PARCORs needs more bit-depth because of its higher variance, the new bit-depth scheme in the third row of Table 9 was applied next, and its result was slightly better than before in average. To enhance the performance more, we additionally adjusted the dynamic range for the quantization as shown in the fourth row of Table 9, so we finally settled at the uniform quantization setting for PARCORs. We did not do a broad range of simulations to find the optimal quantization scheme for PARCORs because the VAD/CNG algorithm is relatively not a core part of our codec as it has been built quite simply. To quantize the remaining parameter, the residual energy, we applied the non-uniform quantizer with 5-bit depth, and the final version of the quantization scheme in the 5-th row shows 0.061 lower PESQ-MOS scores than non-quantized CNG.

6. 4. 6. RAR MTC vs AMR

In order to check the relative performance of our codec, we ran the reference AMR codec for the design set of audio samples and replaced the signals in unvoiced and silence regions classified by our VAD/CNG algorithm with our codec's reconstructed comfort noise signals because it was difficult to substitute the AMR's VAD/CNG module to our VAD/CNG. There are several different modes for AMR codec, and among them, we chose the mode, AMR 12.2 kbps, with the bit rate closest to our codec running at 12 kbps in voiced regions.

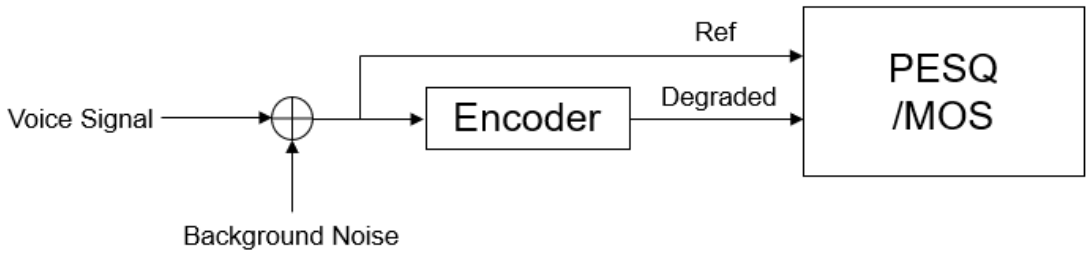
Table 10. RAR MTC vs AMR 12.2 kbps for the Design Set of Audio Samples

F1		F2		M1		M2	
Our Codec	AMR 12.2 kbps	Our Codec	AMR 12.2 kbps	Our Codec	AMR 12.2 kbps	Our Codec	AMR 12.2 kbps
3.456	3.418	3.460	3.456	3.496	3.399	3.561	3.587

As you can see Table 10, our codec shows slightly better performance except for M2, so it implies that our codec is competitive compared to the AMR codec, but the original AMR's outputs would be better than the above results because it uses a highly advanced VAD/CNG module.

6. 4. 7. RAR MTC vs AMR with Background Noises

Since our codec structure has been finalized based on the design set of 4 speech samples, we additionally found another set of audio samples and test it with background noises too. Since two codecs are using different algorithms for VAD/CNG, it is quite hard to do a perfectly fair comparison for actual voiced signal encoding, so we passed the input signals through our tree coder without our simple VAD/CNG which shows an acceptable level of classification performance for low-power white noise sounds of a voice signal, but the bad classification for the test background noises in emergent situations like a loud siren or sawing sounds. For this additional test with background noises, we prepared two methods of PESQ-MOS measurements.



< *Fig.* 25. *Option* 1 *for PESQ/MOS Measurement with Background Noise* >

One of the methods, Option 1 from Figure 25, takes the sum of the original voice signal and the background noise as a reference signal, and PSEQ/MOS is measured by comparing the encoder output to the reference signal. In order words, this measurement takes account of

the quality for both voice and background noise signals together, so the higher PESQ-MOS scores may not absolutely imply better voice quality because it is possible that the background noise is recovered better than the voice signal.

Table 11. RAR MTC vs AMR 12.2 kbps for Female Samples + Background Noise by Option 1 Measurement

AMR (12.2 kbps)

Added Noise	Design Set		Non-Design Set								Avg.
	F1	F2	F4	F5	F6	F7	F8	F9	F10	F11	
BN1	3.812	3.860	3.860	3.851	3.792	3.782	3.797	3.811	3.821	3.777	3.816
BN2	3.375	3.467	3.306	3.396	3.294	3.335	3.277	3.303	3.235	3.215	3.320
BN3	3.827	3.866	3.760	3.862	3.723	3.803	3.826	3.751	3.782	3.809	3.801
BN4	3.581	3.477	3.487	3.432	3.275	3.554	3.579	3.637	3.362	3.344	3.473
BN5	3.903	3.930	3.876	3.856	3.825	3.816	3.901	3.856	3.889	3.947	3.880
BN6	3.641	3.668	3.628	3.585	3.610	3.606	3.602	3.545	3.582	3.639	3.611
BN7	3.791	3.809	3.820	3.884	3.837	3.853	3.840	3.791	3.884	3.838	3.835

RAR MTC without VAD/CNG (12 kbps)

Added Noise	Design Set		Non-Design Set								Avg.
	F1	F2	F4	F5	F6	F7	F8	F9	F10	F11	
BN1	3.936	3.903	4.092	3.853	3.811	3.800	4.068	3.895	3.991	4.022	3.937
BN2	3.128	3.047	3.061	3.090	3.194	3.115	3.077	3.264	3.093	3.135	3.120
BN3	3.918	3.945	3.979	4.035	3.746	3.852	3.936	3.834	3.965	4.056	3.927
BN4	3.247	3.347	3.432	3.303	3.085	3.175	3.334	3.361	3.098	3.252	3.263
BN5	4.009	3.950	4.040	3.993	4.050	3.872	4.088	4.006	4.058	4.139	4.021
BN6	3.640	3.699	3.753	3.842	3.742	3.598	3.642	3.740	3.647	3.768	3.707
BN7	4.014	3.942	4.134	4.027	3.935	3.790	3.954	3.829	3.829	4.057	3.951

Table 11 above shows the result of the female audio samples with background noises for the Option 1 measurement. On average over voice samples, our codec performs better for the background noise BN 1, 3, 5, 6, 8. Especially for BN5, female voice samples except for F2, F5, and F7, show scores over 4.0 which means very good while AMR does not have any result over 4.0. Furthermore, our codec and AMR show poorer performances for BN2, 4 and 7 for

all voice samples. Considering that the codec's performance is dependent on the distribution of input signal's frequency components, BN2, 4 and 7 are particularly difficult sources for a predictive model.

Table 12. RAR MTC vs AMR 12.2 kbps for Male Samples + Background Noise by Option 1 Measurement

AMR (12.2 kbps)

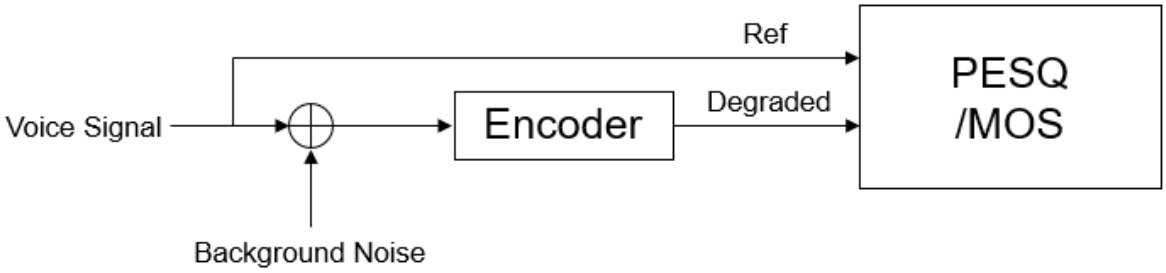
Added Noise	Design Set		Non-Design Set							Avg.
	M1	M2	M4	M5	M6	M7	M8	M9	M10	
BN1	3.849	3.806	3.855	3.775	3.851	3.844	3.833	3.838	3.865	3.835
BN2	3.571	3.536	3.445	3.198	3.456	3.338	3.176	3.461	3.530	3.412
BN3	3.858	3.846	3.889	3.799	3.920	3.855	3.774	3.879	3.946	3.863
BN4	3.576	3.543	3.615	3.692	3.495	3.788	3.410	3.538	3.700	3.595
BN5	3.923	3.889	3.856	3.810	3.856	3.844	3.796	3.774	3.911	3.851
BN6	3.698	3.662	3.620	3.535	3.568	3.422	3.627	3.606	3.692	3.603
BN7	3.817	3.812	3.844	3.924	3.903	3.799	3.747	3.869	3.888	3.845

RAR MTC without VAD/CNG (12 kbps)

Added Noise	Design Set		Non-Design Set							Avg.
	M1	M2	M4	M5	M6	M7	M8	M9	M10	
BN1	3.936	3.967	3.915	3.892	3.999	3.851	4.022	4.067	3.927	3.953
BN2	3.188	3.160	3.228	3.112	3.269	3.263	3.060	3.275	3.129	3.187
BN3	3.984	3.977	3.983	3.997	4.019	3.933	3.990	3.978	4.007	3.985
BN4	3.621	3.453	3.483	3.658	3.419	3.471	3.357	3.441	3.651	3.506
BN5	3.973	3.921	3.914	3.989	4.015	3.904	4.063	4.073	4.074	3.992
BN6	3.728	3.757	3.716	3.672	3.628	3.713	3.720	3.754	3.810	3.722
BN7	4.060	3.969	4.063	4.025	3.973	4.040	3.988	4.111	4.020	4.028

Table 12 is for the male voice samples with the same measurement, and we can see that our codec is better with the background noise BN 1, 3, 5, 6, 8, and that BN2, 4, and 7 are disturbing the overall encoding performances for both codecs like the female samples. As we already mentioned, the option 1 measurement is based on the speaker voice and background noise

together, these scores may not be directly connected to the actual voice recovery quality, so we prepared another PESQ-MOS measurement method, option 2.



< Fig. 26. Option 2 for PESQ/MOS Measurement with Background Noise >

As we can see in Figure 26, the reference signal is just the voice signal, so the option 2 is focusing on the quality of the voice signal only.

Table 13. RAR MTC vs AMR 12.2 kbps for Female Samples + Background Noise by Option 2 Measurement

AMR (12.2 kbps)

Added Noise	Design Set		Non-Design Set								Avg.
	F1	F2	F4	F5	F6	F7	F8	F9	F10	F11	
BN1	1.825	1.953	1.791	1.725	1.707	1.786	1.876	1.796	1.795	1.793	1.805
BN2	2.213	2.284	2.218	2.079	1.988	2.149	2.207	2.045	2.090	1.845	2.112
BN3	1.626	1.814	1.642	1.531	1.572	1.515	1.709	1.678	1.487	1.620	1.619
BN4	2.476	2.623	2.505	2.688	2.343	2.266	2.395	2.074	2.031	2.215	2.362
BN5	1.845	1.894	1.572	1.559	1.483	1.496	1.665	1.701	1.525	1.700	1.644
BN6	1.711	1.946	1.703	1.568	1.546	1.806	1.705	1.819	1.709	1.613	1.713
BN7	1.576	1.857	1.641	1.584	1.635	1.681	1.850	1.766	1.542	1.818	1.695

RAR MTC without VAD/CNG (12 kbps)

Added Noise	Design Set		Non-Design Set								Avg.
	F1	F2	F4	F5	F6	F7	F8	F9	F10	F11	
BN1	1.766	1.884	1.737	1.645	1.652	1.688	1.821	1.811	1.713	1.788	1.751
BN2	2.238	2.288	2.213	2.079	2.037	2.209	2.250	2.085	2.118	1.909	2.143
BN3	1.578	1.739	1.563	1.493	1.503	1.428	1.638	1.618	1.457	1.598	1.562
BN4	2.402	2.539	2.495	2.518	2.283	2.118	2.357	2.159	1.971	2.124	2.297
BN5	1.774	1.822	1.520	1.463	1.442	1.428	1.574	1.664	1.488	1.622	1.580
BN6	1.695	1.893	1.682	1.553	1.584	1.773	1.697	1.851	1.678	1.642	1.705
BN7	1.524	1.785	1.620	1.568	1.609	1.565	1.734	1.679	1.464	1.731	1.628

Table 13 is the result for female speakers with background noises based on the measurement option 2. For both codecs, compared to the results over 3.0 from the option 1, PESQ-MOS scores for the option 2 have substantially dropped to under 3.0 because the recovered background noises signal are the unwanted outputs for the option 2. Furthermore, the lowest scores are almost close to 1.5, and the highest scores are around 2.5 for both codecs. Unlike the option 1 shows worse performances for BN 2, 4, and 7, the option 2 adversely shows worse performances for the noises, BN 1, 3, 5, 6, 8 on average. In comparison of both codecs for option 2, AMR codec is slightly better in average over all background noises except BN 2.

Table 14. RAR MTC vs AMR 12.2 kbps for Male Samples + Background Noise by Option 2 Measurement

AMR (12.2 kbps)

Added Noise	Design Set		Non-Design Set							Avg.
	M1	M2	M4	M5	M6	M7	M8	M9	M10	
BN1	2.269	2.124	2.091	1.907	1.986	1.874	1.776	2.077	2.088	2.021
BN2	2.455	1.941	2.327	2.063	2.243	2.151	2.066	2.339	2.363	2.216
BN3	2.104	2.008	2.060	1.708	1.913	1.827	1.431	1.816	2.137	1.889
BN4	2.496	2.525	2.377	2.352	2.389	2.407	2.187	2.344	2.626	2.411
BN5	2.157	2.094	2.146	1.744	1.957	1.791	1.510	1.898	2.138	1.937
BN6	2.139	1.920	1.952	1.849	1.725	1.995	1.793	2.042	2.007	1.936
BN7	2.159	2.027	2.223	1.931	2.166	1.999	1.494	1.973	2.201	2.019

RAR MTC without VAD/CNG (12 kbps)

Added Noise	Design Set		Non-Design Set							Avg.
	M1	M2	M4	M5	M6	M7	M8	M9	M10	
BN1	2.198	2.096	2.118	1.887	1.968	1.874	1.695	2.049	2.070	1.995
BN2	2.508	2.521	2.410	2.091	2.266	2.194	2.109	2.370	2.419	2.321
BN3	2.036	1.972	2.039	1.691	1.846	1.776	1.391	1.756	2.108	1.846
BN4	2.494	2.478	2.321	2.295	2.278	2.386	2.199	2.354	2.603	2.379
BN5	2.145	2.074	2.088	1.718	1.903	1.776	1.510	1.888	2.046	1.905
BN6	2.147	1.967	1.916	1.791	1.618	1.917	1.704	2.096	1.979	1.904
BN7	2.092	1.975	2.185	1.907	2.099	1.949	1.493	1.892	2.120	1.968

When we look at Table 14 which shows the results for male speakers with the measurement option 2, the same data analysis from the female data can be drawn to male data although their exact data numbers are different.

Table 15. RAR MTC vs AMR 12.2 kbps for All Voice Samples + Background Noises By Option 1 & 2

Added Noise	Measurement Option 1				Measurement Option 2			
	AMR (12.2 kbps)		RAR MTC without VAD/CNG (12 kbps)		AMR (12.2 kbps)		RAR MTC without VAD/CNG (12 kbps)	
	Female	Male	Female	Male	Female	Male	Female	Male
BN1	3.816	3.835	3.937	3.953	1.805	2.021	1.751	1.995
BN2	3.320	3.412	3.120	3.187	2.112	2.216	2.143	2.321
BN3	3.801	3.863	3.927	3.985	1.619	1.889	1.562	1.846
BN4	3.473	3.595	3.263	3.506	2.362	2.411	2.297	2.379
BN5	3.880	3.851	4.021	3.992	1.644	1.937	1.580	1.905
BN6	3.611	3.603	3.707	3.722	1.713	1.936	1.705	1.904
BN7	3.835	3.845	3.951	4.028	1.695	2.019	1.628	1.968

By incorporating the average scores over female and male voice samples separately, we can get Table 15 above. The additional fact about this data is that the performances for both codecs are mostly better with male voices than female voices, and it implies that a longer pitch period of male voices are more likely to allow better prediction performances inside both codecs. Although it was hard to say which codec is dominantly better, as we change the measurement method from option 1 to option 2 with a huge drop of scores, the AMR looks slightly better than RAR MTC except for BN2, but considering that our RAR MTC is running at 12 kbps in voiced regions 0.2 kbps lower than AMR 12.2 kbps, we can say that their average performances are almost same. Even in the actual headphone sound tests, the differences are not easily audible.

6. 4. 8. Informal Listening Tests

PESQ-MOS is standardized to evaluate the objective sound quality as numeric scores based on extensive experiments. We also conducted the informal listening tests over headphones to complement the PESQ-MOS and to characterize the types of distortion present. For the listening test, we listened to the reconstructed signals of the RAR MTC and the AMR-NB 12 kbps mode from Table 10. Since the output for the AMR-NB 12 kbps from Table 10 has the signals in unvoiced and silence regions replaced with our RAR MTC codec's output, we could not hear any difference in the periods between voiced segments. For the voiced parts, the overall sound qualities for both codecs are very clear for all audio samples, so it is difficult to find any difference between the two codecs at first, but as we repeat the listening test with a deeper focus on the voiced parts, we can hear some differences. First of all, there are some audible, but very low level of hissing in the voiced parts for the RAR MTC codec, but the hissing cannot be easily heard especially for F1. The AMR-NB codec rarely has sounds hissing, but when we focus on the speakers' tone, their voiced regions sometimes have spectral distortion and sound like an artificial voice, whereas our codec reproduces very natural voice, and these experiences fit our intuition. Compared to the block-based AMR built on the CELP structure, our RAR MTC codec is more of a waveform-following type codec, so the tones of the reconstructed signals for our codec steadily sound natural over all voiced regions, but with a low level of hissing sound which could be similar to quantization noise. In summary, the PESQ-MOS scores from Table 7 and our informal listening tests show that the RAR MTC performs well in comparison to the AMR-NB codec which is a widely implemented international speech codec standard.

Chapter 7

Conclusions and Future Work

7. 1. Conclusions

We started this research by benchmarking the tree coders of Gibson, Chang, and Woo's research work [4] [5], and the potential of speech tree coding based on a sample-by-sample approach have led us to explore more possibilities based on the following new efforts.

- One of the efforts to realize the potential was to replace the all-pole short-term predictor of a code generator with a pole-zero predictor for better waveform-tracking performance.
- The addition of our simple VAD/CNG module enables our codec to have a lower average bit rate of close to half of 12 kbps for the input samples in Table 6 and Table 7. The average bit rate is dependent on the portion of voiced regions while resulting in some quality loss in unvoiced and silence regions, and this quality loss may be minimized with state-of-art VAD/CNG algorithms.
- Furthermore, by randomly interleaving 4-level and 2-level trees with a fixed rate of 1.5 bits/sample instead of repeating the unit trees one by one (ex : 4-2-4-2-4-2-…) and

selecting the best sequence for our design set of samples, our tree coder can be more stochastically optimized for the given design audio samples.

- The application of a 5-level PCQ tree for the initial samples of each voiced segment can help better initialization of predictive coding parameters of the randomized 4-2 tree coding for the remaining samples in voiced regions.
- Unlike the 4-level tree unit, the 2-level tree unit is quite tricky to adapt its gain because the tree generates only 2 outputs, but our new gain adaptation algorithm based on the 5-polarity of the past excitations has slightly increased the overall codec performances.
- The algorithmic time delay for the RAR MTC with VAD/CNG is composed of the look-ahead delay of VAD scanning (128 samples = 16 ms) and the tree encoding delay based on the tree depth (10 samples = 1.25 ms) for the (M = 8, L = 10) algorithm. The total delay is 17.25 ms which is lower than AMR-NB's delay, 25 ms.

As these new features are combined with a backward adaptive tree coder, our final version of tree coder, RAR MTC, can achieve the PESQ-MOS scores between 3.5 and 4.0 running at the bit rates of 4 to 6 kbps on average without background noises. Although AMR-NB codecs perform better than ours, considering that the AMR-NB is the result of long history of standardization meetings with hundreds of engineers and researchers, our research led by only two members for a much shorter time period shows excellent performances. Our RAR MTC codec achieves the best performance ever for the backward adaptive predictive coders based on a sample-by-sample analysis-and-synthesis model.

7. 2. Future Work

The RAR MTC codec still has more possibilities to be improved by upgrading each of the functional components composing a tree coder, so the following works should be considered in the future.

- When we compare the results of Table 6 and Table 7, we can see the performance drop in unvoiced and silence regions because of our VAD/CNG module while resulting in a lower average bit rate. The next step should include a newer version of VAD/CNG which can more precisely detect a voice tone or other information by using a filter bank like the AMR codec.
- For the pole-zero short-term predictor in a code generator, other promising adaptation algorithms need to be tested for the adaptation of the zero coefficients for better prediction performance. Applying the same lattice structure of adaptation algorithms to zero coefficients as pole coefficients may be a good choice for the efforts. In parallel, the prediction performance should be analyzed by varying the order of the zero coefficients.
- More efforts to find a better pitch parameters adaptation algorithm for a long-term predictor should be considered too. Especially, since the current recursive pitch parameter update rule can increase or decrease its pitch period estimate by only 1 each iteration, the pitch period tracking sometimes cannot quickly respond to the quick change of the actual pitch period during the regions where a recursive method should be used.
- For PWF, future work can consider the way to adapt the perceptual weighting filter's shaping constant, μ and γ, like its filter coefficients need to be considered. As

mentioned in Chapter 5, the PWF's shaping effect may not properly work in some regions, and we suspect that the fixed values of μ and γ may cause undesirable weighting effects in some regions.

- The efforts to find a better gain adaption algorithm for the 2-level tree unit should continue too. As one of the suggestions, using more polarity information than five in the current 2-level gain control rule can be considered. If a longer excitation polarity sequence than five shows a more skewed probability distribution for expansion or contraction, better adaptation performance can be expected with more diverse ranges of gain multiplication factors. Once any better performance of the 2-level gain control algorithm is developed, a tree coder can achieve a lower average bit rate with a little bit of sound quality loss by using more 2-level trees than 4-level trees in generating a tree sequence.
- Not limited to the combination of 4-level and 2-level trees, more diverse levels of tree units with different ratios should be analyzed to find a better structure of a tree code.
- Lastly, because of the good waveform-tracking performance of the RAR MTC codec, future work should consider expanding the evaluation for the first responder communication where the RAR MTC codec may outperform any other standardized speech codecs.

References

[1] 3GPP, “Mandatory Speech Codec Speech Processing Functions; AMR speech Codec; General description”, TS 26.071, 1999.

[2] 3GPP, “Codec for Enhanced Voice Services (EVS); General overview”, TS 26.441, 1999.

[3] V. Iyengar and P. Kabal, “A Low Delay 16 Kbits/sec. Speech Coder,” in Proc. I988 IEEE Int. Conf. Acoust., Speech, Signal Processing, pp. 243-246, 1988.

[4] H. C. Woo and J. D. Gibson, “Low Delay Tree Coding of Speech at 8 kbit/s”, IEEE Transactions on Speech and Audio Processing, Vol. 2, No. 3, July 1994.

[5] J. Gibson and W. Chang, “Fractional Rate Multitree Speech Coding”, IEEE Transactions on Communications, Vol. 39, No. 6, June 1991.

[6] B.S. Atal and M.R. Schroeder, “Stochastic Coding of Speech at Very Low Bit Rates”, In Proceedings of the International Conference on Communications, Amsterdam, The Netherlands, pp. 1610–1613, May 1984.

[7] J. P. Adoul, P. Mabilleau, M. Delprat, S. Morissette, “Fast CELP Coding based on Algebraic Codes.”, In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Dallas, TX, USA, pp. 1957–1960, April 1987.

[8] R. Salami, C. Laflamme, J. P. Adoul, A. Kataoka, “Design and Description of CS-ACELP: A Toll Quality 8 kb/s Speech Coder.” IEEE Trans. Speech Audio Process, Vol 6, 116–130, 1998.

[9] ITU-T Recommendation G.729, “Coding of Speech at 8 kbit/s using Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP)”, ITU-T, May 2012.

[10] Bell Lab., “Pulse Code Modulation”, Bell Lab Record, Vol 25, 1947.

[11] C. Chapin Cutler, “Differential Quantization of Communication Signals”, U.S. Patent 2605361, Jul. 1952.

[12] P. Cummiskey, N. S. Jayant, J. L. Flanagan, “Adaptive Quantization in Differential PCM coding of Speech”, The Bell System Technial Journal, Vol 52(7), 1105-1118, Sep. 1973.

[13] J. B. Anderson and J. B. Bodie, “Tree Encoding of Speech”, IEEE Transactions on Information Theory, Vol. IT-21, No. 4, July 1975.

[14] J. D. Gibson, T. Berger, T. Lookabaugh, R. Baker, D. Lindbergh, “Digital Compression for Media: Principles and Standards”, Morgan Kaufmann, Jan. 1998.

[15] B. S. Atal and M. R. Schroeder, “Adaptive Predictive Coding of Speech Signals,” Bell Syst. Tech. J., Vol. 49, pp. 1973-1986, Oct. 1970.

[16] B. S. Atal, “Predictive Coding of Speech at Low Bit Rates,’’ IEEE Trans. Commun., vol. COM-30, pp. 600-614, Apr. 1982.

[17] R. Pettigrew and V. Cuperman, “Backward Pitch Prediction for Low Delay Speech Coding,” in Proc. GLOBECOM, pp. 34.3.1-34.3.6, 1989.

[18] T. Berger, “Rate Distortion Theory: A Mathematical Basis for Data Compression”, Prentice-Hall, 1971.

[19] R. C. Reininger and J. D. Gibson, “Backward Adaptive Lattice and Transversal Predictors in ADPCM,” IEEE Trans. Commun., Vol. COM-33, pp. 74-82, Jan. 1985.

[20] W. W. Chang, “Analysis and Design of Low Rate Tree Coders,” Ph.D. Dissertation, Texas A&M University, College Station, TX, May 1989.

[21] ITU-T Recommendation G.726, “40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)”, ITU-T, Dec. 1990.

[22] J. D. Gibson, “Backward Adaptive Prediction as Spectral Analysis in a Closed Loop”, IEEE Trans. Acoust. Speech Signal Process, Vol 33, 1166–1174, 1985.

[23] D. Cohn and J. Melsa, “A Pitch Compensating Quantizer”, ICASSP '76. IEEE International Conference on Acoustics, Speech, and Signal Processing, Apr. 1976.

[24] N. S. Jayant and P. Noll, “Digital Coding of Waveforms: Principles and Applications to Speech and Video.”, Englewood Cliffs, NJ: Prentice-Hall, 1984.

[25] A. C. Goris and J. D. Gibson, “Incremental Tree Coding of Speech”, IEEE Transactions on Information Theory, Vol. IT-27, No.4, Jul 1981.

[26] ITU-T Recommendation P.501, “Test Signals for Use in Telephony and Other Speech-based Applications”, ITU-T, May 2020.

[27] Audio Library (https://its.ntia.gov/research-topics/audio-quality-research/public-safety-audio-quality/mrt-library/audio-source-files), NTIA/ITS.

[28] ITU-T Recommendation P.862, “Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for End-to-end Speech Quality Assessment of Narrow-band Telephone Networks and Speech Codecs”, ITU-T, Feb. 2001.

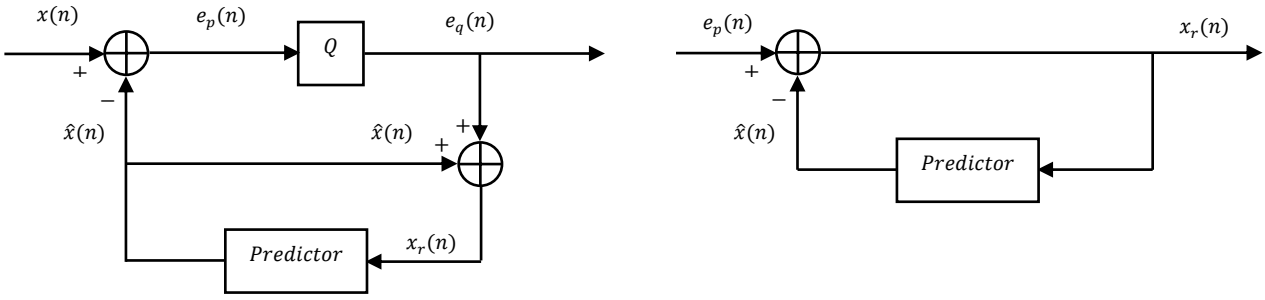
[29] J. D. Gibson and H. Oh, “A Reinforcement Learning Approach to Speech Coding”, MDPI, Jul. 2022.

[30] C. Chapin Cutler, “Differential Quantization of Communication Signals”, U.S. Patent 2605361, Jul. 1952.

[31] P. Cummiskey, N. S. Jayant and J. L. Flanagan, “Adaptive Quantization in Differential PCM coding of Speech”, The Bell System Technial Journal, Vol 52(7), 1105-1118, Sep. 1973.

Appendix. Adaptive Differential Pulse Code Modulation (ADPCM)

Differential Pulse Code Modulation (DPCM) [30] is the most common classical technique in signal processing to compress or encode data to represent audio or video signals by reducing correlated information between sequential data. DPCM is a form of predictive coding that takes advantage of the statistical correlation between adjacent samples in voiced speech segments. Rather than directly quantizing the current voice sample's amplitude, it predicts the current voice value first and quantizes the difference (or prediction error) between the prediction and the actual value. Supposing that we have a good predictor, the dynamic range of the prediction error is much smaller than the actual voice sample, so the encoder can achieve better Signal-to-Noise Ratio (SNR) performance with the same number of quantization bits. However, the fixed step size of a quantizer may show poor performance when the prediction gain dynamically changes, so if we add an algorithm to adapt the quantizer step size to allow smaller quantization error, the variant of DPCM is called Adaptive Differential Pulse Code Modulation (ADPCM) [31].



$<$ ***Fig. A. Encoder (Left) and Decoder (Right) of (A)DPCM*** $>>$

Once quantized prediction errors are encoded by binary codes, the encoded data are transmitted to a receiver, and a decoder reconstructs the signal with the transmitted prediction error with the same structure of the predictor as an encoder in Figure A.