Enhancing Speech Coder Quality: Improved Noise Estimation for Postfilters

Cheick Mohamed Konaté

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

June 2011

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Engineering.

2011/06/29

Abstract

ITU-T G.711.1 is a multirate wideband extension for the well-known ITU-T G.711 pulse code modulation of voice frequencies. The extended system is fully interoperable with the legacy narrowband one. In the case where the legacy G.711 is used to code a speech signal and G.711.1 is used to decode it, quantization noise may be audible. For this situation, the standard proposes an optional postfilter. The application of postfiltering requires an estimation of the quantization noise. The more accurate the estimate of the quantization noise is, the better the performance of the postfilter can be.

In this thesis, we propose an improved noise estimator for the postfilter proposed for the G.711.1 codec and assess its performance. The proposed estimator provides a more accurate estimate of the noise with the same computational complexity.

Sommaire

ITU-T G.711.1 est une extension multi-débit pour signaux à large-bande de la très répandue norme de compression audio de UIT-T G.711. Cette extension est interoperationelle avec sa version initiale à bande étroite. Lorsque l'ancienne version G.711 est employée pour coder un signal vocal et que G.711.1 est utiliser pour le décoder, le bruit de quantification peut être entendu. Pour ce cas, la norme propose un post-filtre optionel. Le post-filtre nécessite l'estimation du bruit de quantification. La précision de l'estimation du bruit de quantification va jouer sur la performance du post-filtre.

Dans cette thèse, nous proposons un meilleur estimateur du bruit de quantification pour le post-filtre proposé pour le codec G.711.1 et nous évaluons ses performances. L'estimateur que nous proposons donne une estimation plus précise du bruit de quantification avec la même complexité.

Acknowledgments

I would like to thank my supervisor Prof. Peter Kabal for his guidance and support throughout the research process that led to the achievement of this thesis.

I would also like to thank all the students in the lab that helped make the work environment very enjoyable. I am especially thankful to Abdul Hannan Khan, Amr Nour-Eldin, Hafsa Qureshi, Joachim Thiemann, Mahmood Movassagh and Qipeng Gong.

Special thanks to my parents for their support through all the years. I thank them for all the wonderful opportunities that they have given me, their love and encouragement. I would also like to thank my sisters, all my other family members and my friends.

Contents

List of Figures

List of Tables

Chapter 1

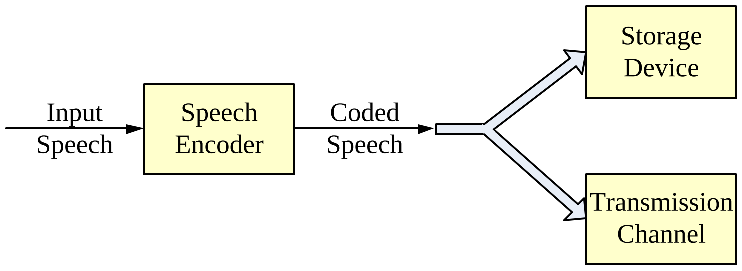
Introduction

1.1 Speech Coders

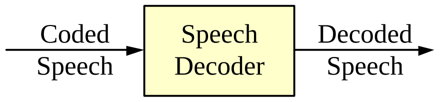
Speech coding is widely used today and it continues to be an important research topic. Many applications require it. They include but are not limited to mobile telephony, IP telephony and audio/video conferencing. Speech coding techniques mainly aim to compress the digital speech in an efficient manner for either storage or transmission. This first goal usually goes hand in hand with a second one: the quality of the decompressed speech signal (which is usually different from the original signal) has to be good. The component which compresses the speech signal in the coder is called an *encoder*. The component which decompresses the speech signal is called a *decoder*. Due to these two components, we often refer to speech coders as speech codecs. Fig. 1.1(a) shows a high-level speech encoding process. The input to the speech encoder is digital speech. The output is the coded signal. The latter usually has a smaller bit-rate than the input signal. This compressed signal is stored in a storage device or sent to another device through a transmission channel. Fig. 1.1(b) shows a high-level speech decoding process.

1.2 Noise in speech coders

Imagine a situation where a person is speaking on a mobile phone. As the person is talking on the phone, he/she is walking downtown during a busy period of the day. Thus, the speech will certainly be affected by some external noise. This noise can include but is not limited to car horns, other people talks in the street, moving cars or some random person



(a) Speech Encoder



(b) Speech Decoder

Fig. 1.1 Speech Codec

whistling nearby. We will refer to the speech exiting directly the mouth of the speaker as “clean speech”. We will refer to the speech that enters through the microphone of the phone as “noisy speech” because that speech will have been affected by some of the external noise.

Imagine another situation where a person is talking on a mobile phone. Here, the person is talking in a closed room with barely any external noise. In this case, the speech that goes through the microphone is almost exactly the same as the clean speech. The speech coder in the phone then encodes the speech before it is sent to the phone of the listener. The type, the encoding process introduces some distortion in the speech. In waveform speech coders for example, the speech is coded sample-by-sample. Specifically, each sample is rounded (quantized) to some value. The difference between the original clean speech and the recovered speech (after the decoder) is the coding noise. For waveform coders, the coding noise is often referred to as “quantization noise”.

From the two situations described above, we see that noisy speech is inevitable in speech processing. The noisy speech can be affected with environmental noise and/or coding noise. The noise sometimes creates undesirable perceptual effects that can affect the quality of a conversation. For example, the noise can make it difficult for the conversation participants to properly hear each other. For these reasons, speech coding systems usually include processing stages to reduce the perceptual effects of the noise on the speech. This ends up making the conversation between the two parties more clear. Environmental and coding noise are different in nature. Consequently, the methods applied to reduce their respective effects have often been disjoint.

Environmental noise comes from the surroundings of the conversation parties. The noise can disturb the conversation in many ways. It could for example be so loud that the portions of the conversation becomes covered by it. The listener would not be able to hear the information given by the speaker clearly and this could lead to miscommunication. The noise could also be distracting to the listener. The environmental noise is available at the encoder and it is undesired. To avoid wasting bits encoding this unwanted noise, environmental noise reduction filters are typically applied before the encoding process. We refer to such an operation as *prefiltering*. On the other hand, the coding noise results from the distortion introduced during the coding procedure. Consequently, one can only reduce it after the signal has been decoded. We will refer to such an operation as *postfiltering*.

Typically, the environmental noise is estimated during non-speech periods. It is fair to assume that the talker is in the same environment when he/she resumes talking. The estimated noise can then be reduced during periods of speech. The filtering methods that are typically used to reduced the environmental noise are spectral subtraction and Wiener filtering [1]. The estimated noise is used to adaptively estimate the filters.

The coding noise tend to make the speech less periodic: the speech formants and speech harmonics are less prominent after coding. The postfilter attempts to reestablish the prominency of formants and harmonics. Historically, the coding noise was especially disturbing in low bit-rate coders. The parameters containing formant and harmonic information about the speech are usually available at the decoder in low bit-rate systems. These parameters have been commonly used to generate the postfilters. Thus, the coding-noise postfilters are typically based on a parametric representation of the speech spectrum.

Speech coders also use a technique on their encoding end to reduce the effect of the coding noise. This method is known as *noise shaping*. As the speech is encoded, the coding noise is perceptually shaped. Specifically, the coder takes advantage of the human auditory system masking property. It perceptually shapes the noise so that it is partially masked by the speech and becomes less audible to the listener. It is not always possible to completely mask the noise by shaping it so this method is usually augmented with a postfilter at the decoder end of the codec.

1.3 Thesis Description

ITU-T G.711.1 [2] is a multi-rate wide-band extension for the well-known ITU-T G.711 [3] pulse code modulation of voice frequencies. They are both high bit-rate waveform coders. G.711.1 is a multi-rate coder. It was designed such that it is fully interoperable with the legacy G.711 coder when it operates at 64 kbit/s. Specifically, at this bit-rate, a signal that is encoded with the legacy narrow-band G.711 can be decoded by G.711.1 and vice-versa.

The legacy G.711 supports two encoding laws: A-law and μ-law . The resulting quantization noise spectrum is flat. Perceptually, a flat coding noise is not optimal. Specifically, the noise energy sometimes exceeds that of the signal at certain frequency. It becomes audible in these cases and it can be annoying for the listener. In G.711.1, the quantization noise is shaped. Therefore, the perceptual effect of the flat noise that was present in the legacy coder is partly taken care of. However, for low energy signals, the noise shaping is not sufficient and some of the noise can still be heard.

An optional postfilter was proposed in the G.711.1 standard to reduce the coding noise present in signals that were encoded by the legacy coder. The parameters typically needed for implementing a conventional postfilter are not directly available at the decoder end in high bit-rate non-parametric coders such as G.711.1. Designing a conventional parametric postfilter in this case would be complex as these parameters would have to be estimated. The proposed postfilter is a low complexity filter. The quantization noise is estimated and acoustic background noise reduction methods are used to reduce it. The postfilter is therefore somewhat unconventional.

In this thesis, we will focus on the noise estimator in the postfilter. Clearly, the accuracy of the noise estimate plays an important role in the quantization noise reduction performance. In the postfilter proposed in G.711.1 the noise estimation is done through the exploitation of quantization laws properties. After analyzing this noise estimator, we realized that a more accurate estimator could be designed. We will propose the improved noise estimator of the coding noise generated by the legacy G.711 coder. We will additionally propose a noise estimator for signals that were encoded by G.711.1. As noted above, this noise is perceptually shaped but can still be heard for low energy signals.

1.4 Thesis Structure

In Chapter 2, we will review the main noise reduction rules used by most of the classical noise reduction filters. We will also explain the Two-Step Noise Reduction (TS-NR) algorithm. This algorithm is used in the realization of the G.711.1 proposed postfilter.

In Chapter 3, we will review the general approaches that have been used in the past to reduce coding noise and its perceptual effect. We will then discuss some of the main problems these approaches had and we will explain how they led to the development of the now known conventional postfilter.

In Chapter 4, we will briefly review the legacy G.711 codec. We will also explore some of the properties of the A-law algorithm. These properties are important to understand as they are used by the noise estimation systems we will see in this thesis.

In Chapter 5, we will give an overview of the G.711.1 codec. We will then explain how the coding noise is handled at the encoder to reduce its perceptual effect in this coder. Finally, we will explore the postfilter proposed in the standard to reduce the perceptual effects of the coding noise of signals coded by the legacy G.711 coder. We will see how this postfilter uses acoustic background noise techniques (specifically, the TS-NR method) to reduce the coding noise effects and how it uses the A-law properties to estimate the coding noise.

In Chapter 6, we will propose the refined postfilter for signals encoded by the legacy postfilter and we will propose a postfilter for signals encoded by G.711.1. We will also show our simulations results in this chapter and we will discuss them.

Finally, we conclude this thesis in Chapter 7.

Chapter 2

Acoustic Noise Suppression Techniques

Acoustic background noise reduction has been important research topic for a long time. It is still an active research field today. Two main applications were it is extensively used are automatic speech recognition (ASR) and voice communication systems.

In the mid 90's, Scalart and Vieira Filho [1] presented a unified view of the typical noise reduction techniques when only a single microphone is present – that is when a single noisy signal is available. They showed that for most classical methods used to enhance the noisy speech, one needs to compute the degraded signal Power Spectral Density (PSD) and an estimate of the clean signal PSD. They explained how using the decision-directed approach (proposed by Ephraim and Malah in [4]) to estimate the clean signal PSD can help greatly reduce the musical noise effect that older systems exhibit. The musical noise effect consists of audible tone bursts that one can hear in the enhanced speech. Such an effect is due to the fact that those older noise reduction systems use solely the degraded signal PSD. Specifically, sections of the signal that contains only noise have a big variance. That big variance is the main reason behind the musical noise effect.

In [5], Cappé analyzed the computation of the signal estimate by the decision-directed algorithm. He showed that the estimated signal followed the a degraded signal with one frame delay. This is mainly explained by the fact that the computation of the estimate heavily relies on the frame previous to the one being enhanced as we will see in Section 2.2. Consequently the noise reduction technique performance is degraded. Perceptually, Plapous *et al.* [6] reported that an unpleasant reverberation effect can be heard when the decision-directed method is used especially at transitions (from silent periods to speech periods and from speech periods to silent periods).

Plapous *et al.* [7][6] proposed a method called the two-step noise reduction (TS-NR). This technique uses the decision-directed approach to estimate the signal. However the estimate computation corresponds to the current frame rather than the previous one. Therefore, as in the original decision-directed method, the musical noise effect is reduced. The additional advantage of the TS-NR is the removal of the reverberation effect noted in the decision-directed method.

In this chapter, we will first review the general approach taken by the different strategies for acoustic background noise reduction. We will then briefly describe the decision-directed approach and analyze its effects. Finally, we will explain the TS-NR algorithm.

2.1 Acoustic Background Noise Reduction

In ASR and voice recognition systems, only one microphone is typically used by the speaker. Therefore, only one noisy speech is available at the receiving end of the system. This noisy signal generally consists of “clean” speech that has been degraded by uncorrelated additive noise. This lower quality speech signal is the input to a background noise attenuation system which attempts to reduce the contaminating background noise. It typically does so by estimating the noise during non-speech periods of the noisy signal. The noise reduction process is generally performed before the signal is encoded for storage or transmission. The advantage of doing so is that some of the noise that will end up being discarded will not have to be encoded.

Let $y(n)$ denote the degraded signal. Let $x(n)$ denote the clean signal and let $b(n)$ denote the additive noise. We have $y(n) = x(n) + b(n)$. Let $X(p, k)$, $B(p, k)$ and $Y(p, k)$ denote the k^{th} spectral component of a frame p of $x(n)$, $b(n)$ and $y(n)$ respectively. Quasi-stationarity of the speech signal is assumed over the frame. The noise suppression system estimates a spectral gain $G(p, k)$ that it then applies to $Y(p, k)$ to reduce its noise. The spectral gain is optimized based on a selected approach. Different approaches have been used and are available in the literature. Some popular ones are power spectral subtraction, Wiener filtering and Minimum Mean Square Error (MMSE). In [1], Scalart and Vieira Filho presented a unified view of the typical noise reduction techniques when only a single microphone is present. They explained that for most of the chosen approaches, one has to evaluate:

- the degraded signal PSD $|Y(p,k)|^2$
- an estimate of the clean signal PSD $E\left(|X(p,k)|^2\right)$
- an estimate of the noise PSD $E\left(|B(p,k)|^2\right)$

where $E(\cdot)$ is the expectation operator.

One method used to estimate the signal PSD is the Decision-Directed method which we explain in the next section. The gains of some noise reduction systems are summarized in Table 2.1. The systems are all adaptive as the filter gains are computed on a frame-by-frame basis.

Table 2.1 Conventional Speech Enhancement methods

Method	Noise Suppression Gain Function
Power Estimation	$G_k^{\mathrm{PE}} = \sqrt{\dfrac{E\left(\|X(p,k)\|^2\right)}{E\left(\|X(p,k)\|^2\right)+E\left(\|B(p,k)\|^2\right)}}$
ML Estimate	$G_k^{\mathrm{ML}} = \dfrac{1}{2}\left(1+\sqrt{\dfrac{E\left(\|X(p,k)\|^2\right)}{E\left(\|X(p,k)\|^2\right)+E\left(\|B(p,k)\|^2\right)}}\right)$
Wiener Estimate	$G_k^{W} = \dfrac{E\left(\|X(p,k)\|^2\right)}{E\left(\|X(p,k)\|^2\right)+E\left(\|B(p,k)\|^2\right)}$

2.2 Decision-Directed Approach

2.2.1 Decision-Directed Algorithm

Ephraim and Malah proposed a *Decision-Directed* estimation algorithm in [4] to estimate the signal PSD. This algorithm is also used by Scalart and Vieira Filho [1]. The algorithm assumes that an estimate of the noise PSD $|\hat{B}(p,k)|^2$ has already been obtained. The degraded signal PSD is first computed as $|Y(p,k)|^2$. The signal PSD is then estimated as:

$$|\hat{X}(p,k)|^2 = \beta|\hat{X}(p-1,k)|^2 + (1-\beta)\max(0, |Y(p,k)|^2 - |\hat{B}(p,k)|^2). \tag{2.1}$$

The estimator used in Eq. (2.1) is the decision-directed estimator. A typical value used for the parameter β is $\beta = 0.98$.

2.2.2 Decision-Directed Approach Analysis

Two effects can be observed from the decision-directed algorithm. They were interpreted by Cappé in [5] and we summarize them below:

- For large values of the $|Y(p,k)|^2 - |\hat{B}(p,k)|^2$ (much larger than 0 dB), the signal estimate PSD $|\hat{X}(p,k)|^2$ corresponds to a single frame delayed version of $|Y(p,k)|^2 - |\hat{B}(p,k)|^2$

- For small values of $|Y(p,k)|^2 - |\hat{B}(p,k)|^2$ (less than 0 dB), the signal estimate PSD $|\hat{X}(p,k)|^2$ corresponds to a greatly smoothed single frame delayed version of $|Y(p,k)|^2 - |\hat{B}(p,k)|^2$

The consequence of the smoothing for small values of $|Y(p,k)|^2 - |\hat{B}(p,k)|^2$ is a much smaller variance of $|\hat{X}(p,k)|^2$ compared to that of $|Y(p,k)|^2 - |\hat{B}(p,k)|^2$. This is the advantage of using this algorithm as it is the reason of the reduction of the musical noise effect. However, the frame delay that is introduced by the algorithm is a drawback especially at transient periods (speech to non-speech or non-speech to speech). Also, the gain estimation is biased due to the delay as it depends on the previous frame rather than on the current one. This degrades the attenuation performance and perceptually, a reverberation effect can be heard. To address this issue, Plapous *et al.* proposed the two-step noise reduction algorithm which we describe in the next Section.

2.3 Two-Step Noise Reduction Approach

2.3.1 TS-NR Algorithm

The Two-Step Noise Reduction (TS-NR) algorithm uses the decision-directed approach as a basis but this time, the filter gain $G(p,k)$ is estimated in a two-step procedure. The first step consists exactly of the decision directed algorithm. Specifically, a gain $G_{\text{DD}}(p,k)$ is computed as a function of the degraded signal PSD, the estimated signal PSD and the noise PSD. The gain from this first step is used to refine the estimated clean signal PSD:

$$|\hat{X}(p,k)|^2 = |G_{\mathrm{DD}}(p,k)|^2 |Y(p,k)|^2 \tag{2.2}$$

Using this new PSD of the signal, another spectral gain is computed in the second step. This second spectral gain is therefore a function of the degraded signal PSD, the estimated signal PSD from the first step of the algorithm and the noise PSD. The final enhanced speech obtained from the TS-NR algorithm is:

$$|\hat{X}(p,k)|^2 = |G_{\mathrm{TS-NR}}(p,k)|^2 |Y(p,k)|^2 \tag{2.3}$$

2.3.2 TS-NR Approach Analysis

Just as the decision-directed algorithm, the musical noise effect are highly reduced with the TS-NR algorithm because the variance of the estimated signal PSD is small when the $|Y(p,k)|^2 - |\hat{B}(p,k)|^2$ is lower or close to 0 dB. The advantage of the TS-NR algorithm over the decision-directed one is the absence of the bias due to the inherent delay in the decision-directed. Specifically, with the TS-NR method, the speech onsets and offsets are preserved.

Chapter 3

Adaptive Postfiltering

The idea of further processing decoded speech dates from back in the 1960's. Although different approaches suggest postfiltering as we will see in Section 3.1, it is easy to notice that any processed speech signal becomes affected by noise. This noise typically consists of *quantization noise* and *channel noise* (when the speech is propagated through a channel). It then becomes natural to attempt to enhance the reconstructed speech.

An early technique was proposed by Smith and Allen [8] in 1981. They applied their technique on a system using Adaptive Delta Modulation (ADM). Their enhancer consisted of a lowpass filter that was implemented by a short-time Fourier analysis/synthesis method. The cutoff frequency of the computed filter was adaptive: it was chosen so that all spectral content above it constituted only 1% of the total energy of the input signal. The selected cutoff frequency was obtained during encoding of the frame and was sent as side information. As a result, the high frequency noise was removed and a 16 kbit/s ADM with this enhancer was then qualitatively comparable to a 24 kbit/s ADM with no enhancement [8].

In 1984, Jayant and Ramamoorthy [9] proposed a postfilter especially designed for Adaptive Differential Pulse Code Modulation (ADPCM). Conventional ADPCM operates at a bit rate of 32 kbit/s. Specifically, it codes a signal sampled at a frequency of 8 kHz with 4 bits per sample. The lower bit version operates at a bit rate of 24 kb/s, i.e. it codes a signal sampled at a frequency of 8 kHz with 3 bits per sample. A signal coded by the conventional ADPCM results in a signal of "telephone" quality. The low bitrate version produces speech with much lower quality because of the easily audible quantization noise. The proposed postfilter is a pole-zero filter based on the pole-zero predictor in the ADPCM system. Different scaling factors are applied to the coefficients of the predictor to form the coefficients of the postfilter. The filter moves poles and zeros to control the speech spectral envelope or more specifically its formants (the spectral peaks of the speech spectrum). Such a filter is called a *formant postfilter* or as we will see later, a *short-term postfilter*. Proper selection of the scalars weighing the coefficients determines the enhancement of the signal. This method reduces the *perceived* level of coding noise. It is important to note however, that when the coding noise level is high in such a system, the required postfilter tends to degrade the signal energy at high frequencies. This results in the speech sound becoming muffled.

In 1986, Yatsuzuka et al. [10] combined noise spectral shaping and adaptive postfiltering. On top of using a short-term postfilter, they proposed an additional *long-term postfilter* (also called a *pitch postfilter*) that was based on the periodicity of the pitch in speech. The role of this long-term filter is to reduce the noise between harmonics and emphasize the periodicity of the speech signal. Both the short and long term postfilters they used were all-pole filters. The resulting all-pole postfilter had the same muffling effect mentioned previously.

In 1987, Chen proposed yet another postfilter in his Ph.D thesis [11]. The latter had both long-term and short-term sections. An innovation in this postfilter was that the enhanced signal did not sound muffled. This is mainly due to the control of the *spectral tilt*. Chen described his postfilter in a US patent [12] in 1990 and he summarized his results in [13]. Since then, this structure has become a basic one for many researchers. We will often refer to this postfilter as the “conventional postfilter”.

3.1 Different Approaches

3.1.1 Theoretical Approaches

Different theoretical approaches have been investigated over the years. For example, the classical Wiener theory tells us how to generate an optimal filter that minimizes the noise power for a noisy signal. Let $x(n)$, $b(n)$, $y(n)$ and their spectral representations be defined as they were in Section 2.1. Note however that the noise $b(n)$ here is quantization noise as opposed to acoustic background noise. The quasi-stationarity of the speech signal is assumed over the frame. The optimal Wiener filter minimizes the Mean Square Error

(MSE) between the filter output and the original signal:

$$H(p,k) = \frac{|X(p,k)|^2}{|X(p,k)|^2 + |B(p,k)|^2}. \tag{3.1}$$

By dividing the numerator and denominator by the noise PSD $|B(p,k)|^2$, we can rewrite Eq. (3.1) in term of SNR:

$$H(p,k) = \frac{\mathrm{SNR}(p,k)}{\mathrm{SNR}(p,k) + 1}. \tag{3.2}$$

We can readily see from Eq. (3.2) that:

- in frequency bands where the SNR is high, the filter gain is approximately unity
- in frequency bands where the SNR is low, the filter gain is very small

It is important to note that such a filter can usually not be implemented in practice. The clean signal is unavailable at the decoder side and so the true SNR cannot be calculated. Estimates are used in order to approximate the filter. The quantization noise PSD estimate can not be obtained from non-speech frames. The Wiener filter gain function depends on the SNR at each frequency. Since the speech spectrum varies with time, the postfilter has to be adaptive. Specifically, a different filter had to be computed for each frame.

The performance objective should really be perceived quality rather than MSE or any other criterion. Even if one could compute these filters in practice, they still would not be perceptually optimal. Thus perceptual considerations tend to be made to find an effective trade-off between noise reduction and signal distortion resulting from the filtering operation.

3.1.2 Perceptual Approach

The perceptual approach was the route taken by Chen [13] when he designed his postfilter. It considers the properties of the human hearing system. More specifically, the concept of auditory masking is exploited.

It is generally believed that an overall masking function exists for a given speech frame. That is, if noise is added to the speech frame and its power spectrum strictly below the overall masking function at all frequencies, then it is inaudible. It is generally accepted that such a function tends to follow the spectral envelope of the speech in a given frame.

In order to push coding noise below the overall masking threshold function, many coders use noise spectral shaping during their encoding phase. An ideal encoder would be able to push the noise at all frequencies below the masking function. That would make the resulting speech perceptually optimal. In practice however, this is not always easy to achieve especially for low-bitrate coders where the usual average level of coding noise is quite high. As we push the noise level down at some frequencies we must accordingly bring the noise level up at other frequencies. Chen [13] metaphorically describes the situation as being “similar to stepping on a balloon”. As a result, noise shaping is usually not sufficient to make the noise imperceptible.

At the encoder, most spectral shaping algorithms shape the coding noise such that it is below the threshold function in formant regions of the speech and sacrifice the valley regions (the regions between formants). The reason behind this practice is that is that formants are perceptually more important than valleys. Thus, it makes sense that the noise is kept inaudible in formant regions.

Assume that the noise was shaped such that it is below the masking threshold function for all formants but over the masking function for spectral valleys. If no additional processing is done to this signal, most of the perceived noise will come from the spectral valleys including valleys between harmonics. This is mainly due to the absence of strong resonances in these regions to mask the noise. A postfilter is used to attenuate the valley components. In doing so, the speech component in the valley region gets attenuated as well. This distortion is perceptually acceptable however because distortions introduced in the valley regions are not easily detected by our ears [14]. The postfilter takes advantage of this fact.

3.2 Conventional Postfilter

The adaptive conventional postfilter consists of two cascaded filters: a short-term filter and a long-term filter. Its transfer function has the following general form:

$$H(z) = GH_S(z)H_L(z), \tag{3.3}$$

where H_S is a short-term filter, H_L is a long-term filter and G is an adaptive scaling factor. The role of the short-term filter is to emphasize speech formants and attenuate speech valleys without introducing any spectral tilt. The long-term filter's role is to emphasize the pitch harmonic peaks and attenuate the regions between them without any spectral tilt either. The role of the gain control G is to ensure that the energy of the signal is the same before and after postfiltering.

3.2.1 Short-Term Filter

Ideally, the frequency response of the short-term filter (or formant filter) should follow the formants and valleys of the spectral envelope of the speech without introducing any spectral tilt. The short-term filter is derived from an LP predictor as the LP spectrum gives the envelope of the speech. The LP parameters are typically available as side information in low-bit parametric coders. The general transfer function of a short-term filter is given by:

$$H_S(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)}(1 - \mu z^{-1}). \tag{3.4}$$

Let's explain Eq. (3.4) by writing the transfer function of the short-term postfilter as :

$$H_S(z) = H_{\mathrm{S0}}(z) H_{\mathrm{S1}}(z), \tag{3.5}$$

where $H_{\mathrm{S0}}(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)}$ and $H_{\mathrm{S1}}(z) = (1 - \mu z^{-1})$.

$H_{\mathrm{S0}}(z)$ is a pole-zero filter where $A(z)$ is an adaptive short-term prediction filter. γ_1 and γ_2 are *emphasis parameters*. They are chosen to be in $0 < \gamma_1 < \gamma_2 < 1$ and they control the degree of spectral emphasis of the filter. Specifically, the filter moves poles and zeros to control the peaks and the bandwidths of the spectral envelope. $H_{\mathrm{S0}}(z)$ has the same number of poles and zeros. The postfilter proposed by Jayant and Ramamoorthy [9] for ADPCM is a formant postfilter. However, their postfilter is a little different than $H_{\mathrm{S0}}(z)$ as it has two poles and six zeros. The short-term postfilter proposed by Yatsuzuka et *al.* in [10] only consisted of the second factor of $H_{\mathrm{S0}}(z)$, i.e. the all-pole filter $\frac{1}{A(z/\gamma_2)}$.

In dB, the magnitude response of $H_{\mathrm{S0}}(z)$ is given by:

$$\begin{aligned}|H_{\mathrm{S0}}(e^{j\omega})| &= 20 \log \left| \frac{A(e^{j\omega}/\gamma_1)}{A(e^{j\omega}/\gamma_2)} \right| \quad \text{[dB]}, \\ &= 20 \log \left| A(e^{j\omega}/\gamma_1) \right| + 20 \log \frac{1}{|A(e^{j\omega}/\gamma_2)|} \quad \text{[dB]},\end{aligned}$$

which we can rewrite as:

$$|H_{S0}(e^{j\omega})| = 20\log\frac{1}{|A(e^{j\omega}/\gamma_2)|} - 20\log\frac{1}{|A(e^{j\omega}/\gamma_1)|} \quad [\text{dB}]. \tag{3.6}$$

We see from Eq. (3.6) that the magnitude response in dB consists of the difference of the magnitude responses of two LPC synthesis filters. Therefore, with a good choice of γ_1 and γ_2, one can get some control on the response of $H_{S0}(z)$. The optimal choice for the two values depends on the speech and the bitrate. Thus, they can generally be determined empirically based on listening tests. Different LP synthesis filters (different values of γ_2) are shown in Fig. 3.1. For clarity, the different curves are shifted in the figure. The separation gap between subsequent curves is 30 dB. The general tilt mentioned earlier is clearly visible here.

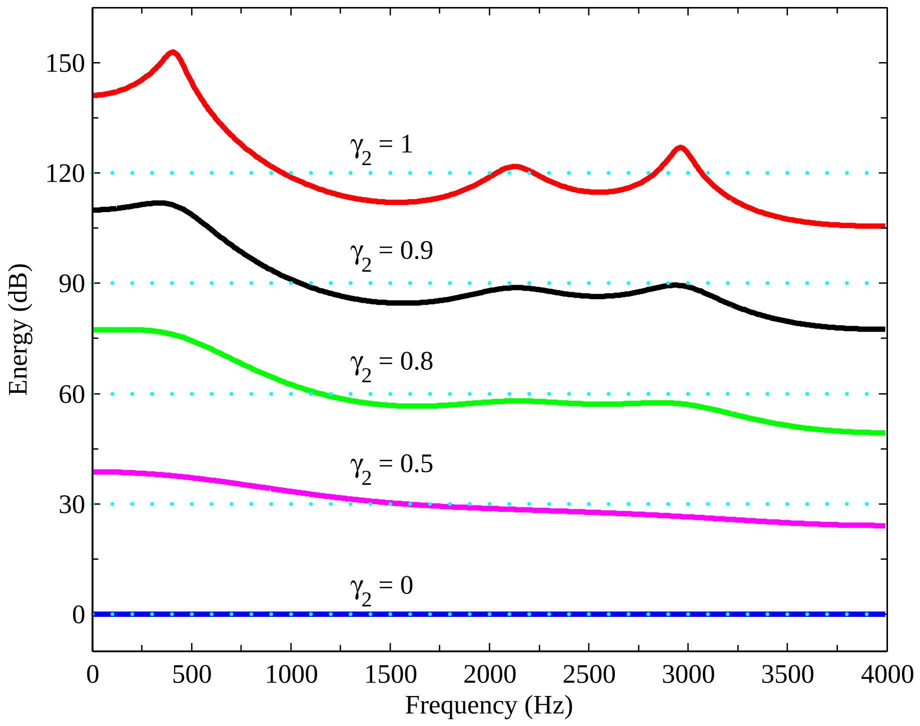


Fig. 3.1 LPC Synthesis filters $\frac{1}{A(z/\gamma_2)}$ with different values of γ_2. For clarity, the curves have been offset from each other by 30 dB

In [13], Chen and Gersho implemented this filter on a 4.8 kbits/s Vector Adaptive Predictive Coding (VAPC) system. They noticed that when $\gamma_2 = 0.8$, the LP filter has both a spectral tilt and smoothed formant peaks and that when $\gamma_2 = 0.5$, the LP filter only has a spectral tilt. They decided to set $\gamma_1 = 0.5$ and $\gamma_2 = 0.8$ in $H_{\mathrm{S0}}(z)$. Doing so, we see from Eq. (3.6), that most of the tilt in the LPC with $\gamma_1 = 0.5$ will get subtracted from the one with $\gamma_2 = 0.8$. The magnitude response of $H_{\mathrm{S0}}(z)$ with such settings is shown in Fig. 3.2 as the top curve. Using H_{S0} rather than a simple LPC spectrum does reduce the

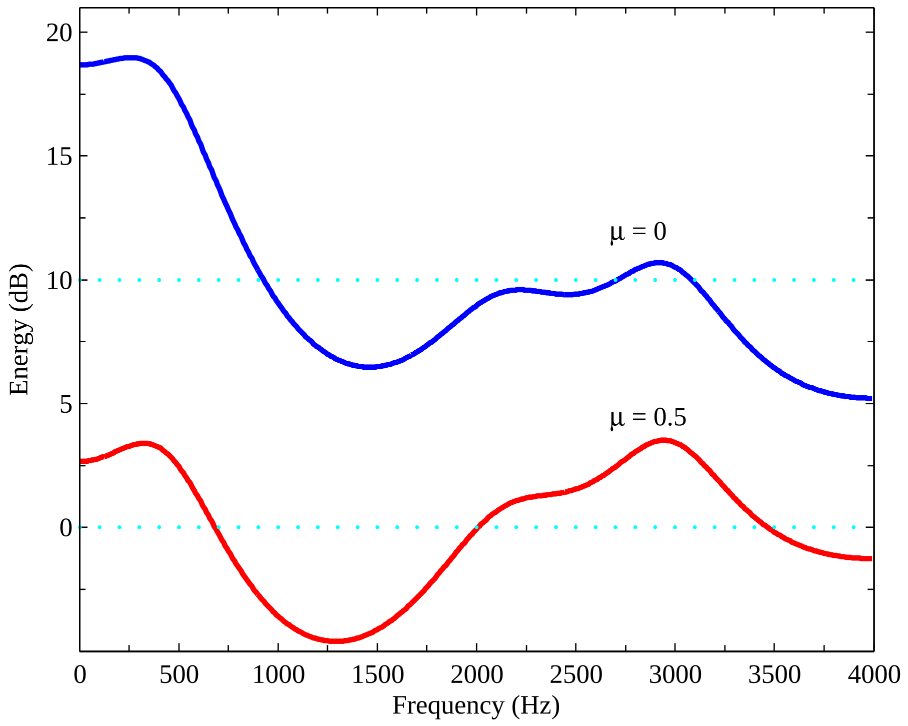


Fig. 3.2 Two short-term filters with $\mu = 0$ and $\mu = 0.5$. For clarity, the curves have been offset from each other by 10 dB

muffling effect quite a bit. However, some muffling can still be felt in the enhanced speech. We see from the top curve in Fig. 3.2 that there is still a spectral tilt. By adding H_{S1} in cascade to H_{S0}, Chen and Gersho further reduced the tilt to nearly no tilt at all. H_{S1} is usually referred to as the *tilt compensation factor*. The parameter μ in the first-order filter H_{S1} was set to 0.5 in the example. The resulting magnitude response of the overall short-term postfilter is shown as the lower curve in Fig. 3.2.

In later variations of the conventional postfilter [13][15], it was noted that an adaptation of the parameter μ further improves the performance of the formant postfilter. The adaptation consists of making μ dependant on the first reflection coefficient k_1. For example, μ can be define as $\mu = 0.5k_1$. The first reflection coefficient is computed as $k_1 = \frac{r[1]}{r[0]}$ where

$r[\tau]$ is the autocorrelation with lag τ. For a voiced speech frame, adjacent samples are highly correlated. Therefore, for such a frame $r[1] \approx r[0]$ and so $k_1 \approx 1$. On the other hand, the correlation of adjacent samples is small for an unvoiced frame. The magnitude of k_1 is small in this case. Using this adaptation, the tilt compensation is is greater for voiced frames than it is for unvoiced ones. This makes sense to do because a voiced frame spectrum typically has a steeper fall in high frequencies than an unvoiced frame.

3.2.2 Long-Term Filter

In [13], Chen and Gersho propose a long-term postfilter that is based on the pitch predictor. A one-tap pitch predictor with transfer function $(1 - gz^{-p})$ is used. Here, g is the pitch predictor coefficient and p is the pitch period in terms of number of samples. This results in a pitch synthesis filter with transfer function $\frac{1}{1-gz^{-p}}$.S All p poles have the same magnitude and they are located at uniformly spaced phase angles $(0, 2\pi/p, 4\pi/p, \ldots, (p-1)\pi/p)$. These phase angles correspond to the frequencies of the pitch harmonics. The proposed all-pole long-term postfilter is derived from the pitch synthesis filter as $\frac{1}{1-\lambda z^{-p}}$ with $0 \leq \lambda < 1$. We will see how λ is determined below. Yatsuzuka *et al.* used such an all-pole filter as their long-term postfilter in [10]. The magnitude response of an all-pole pitch postfilter is shown in Fig. 3.3 along with the pole-zero plot. Here $\lambda = 0.5$ and $p = 30$.

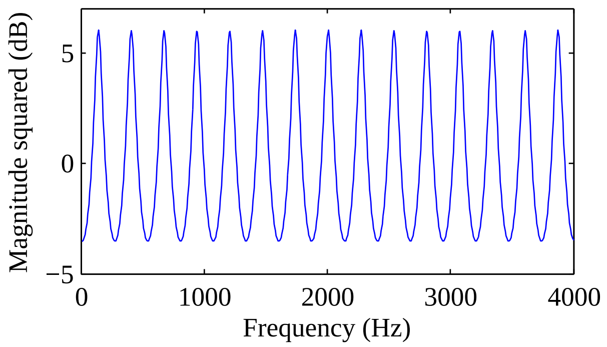


Fig. 3.3 All-pole long-term postfilter $H_L(z) = \frac{1}{1-\lambda z^{-p}}$ with $\lambda = 0.5$ and $p = 30$.

For additional control over the long-term postfilter, Chen and Gersho added as many zeros as there are poles to the all-pole filter. The zeros are specifically used to control the attenuation of the regions between the pitch harmonics. Thus, the zeros are places at uniformly spaced phase angles $(\pi/p, 3\pi/p, \ldots, (2p-1)\pi/p)$. A polynomial that satisfies this requirement is $1+\gamma z^{-p}$, with $\gamma > 0$. The overall zero-pole long-term postfilter transfer function is given by:

$$H_L(z) = \frac{1+\gamma z^{-p}}{1-\lambda z^{-p}}, \tag{3.7}$$

We will explain how the value of γ is chosen below. The magnitude response of such an zero-pole long-term postfilter is shown in Fig. 3.4 along with the pole-zero plot. Here $\lambda = 0.25$, $\gamma = 0.25$ and $p = 30$.

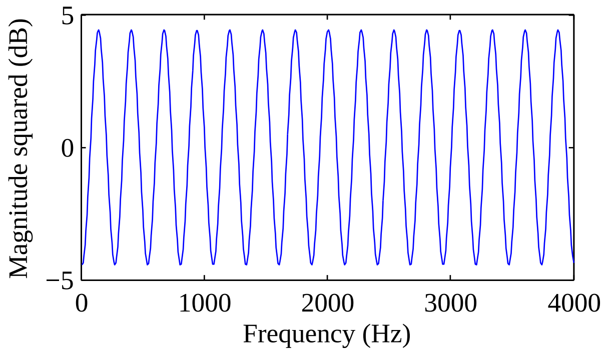


Fig. 3.4 Zero-pole long-term postfilter $H_L(z) = G_L\frac{1+\gamma z^{-p}}{1-\lambda z^{-p}}$ with $\lambda = 0.25$, $\gamma = 0.25$ and $p = 30$.

The parameters λ and γ are determined based on whether or not the frame under analysis is voiced or not. An indicator that can be used to determine the voicing property of the frame is the pitch predictor coefficient g. Its value is close to 1 when the frame is voiced and close to 0 when it is not.

The conventional postfilter consists of the combination of the short-term postfilter and the long-term postfilter. Fig. 3.5 shows the overall structure of the filter and its transfer function is given by:

$$H(z) = G\frac{1+\gamma z^{-p}}{1-\lambda z^{-p}}\frac{A(z/\gamma_1)}{A(z/\gamma_2)}(1-\mu z^{-1}). \tag{3.8}$$

The postfilter proposed by Chen and Gersho reduces the perceived coding noise greatly. It does so without making the enhanced speech sound muffled. Since its proposal, it has been widely used. Many systems made slight variations to the conventional postfilter to

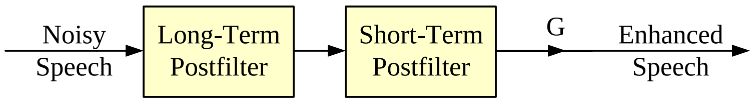


Fig. 3.5 Conventional Postfilter Structure

better suit their needs. For example, a postfilter was proposed in ITU-T G.723.1 [16]. This postfilter has both a pitch postfilter section and a formant postfilter section. The available pitch information and LP parameters are used to adaptively generate the postfilter.

3.3 Hybrid Postfilter/ Mixing methods

In Chapter 2, we've reviewed typical techniques used to remove acoustic background noise. In this chapter, we've reviewed the typical method used to remove coding noise in parametric systems. As previously stated, the techniques used to remove these two kinds of noise are generally different. However, sometimes, techniques usually used to remove one kind of noise are used to remove the other.

In [17], Grancharov *et al.* proposed an algorithm that attenuates both the acoustic background noise and the coding noise using a modified version of the conventional postfilter. Their version of the conventional postfilter uses only a gain and the short-term section. And although in the conventional system the emphasis parameters γ_1 and γ_2 are usually fix, they adapt their values according to noise statistics. They call their postfilter a *noise-dependent postfilter*.

In this thesis, we look at the reverse situation. Specifically, we look at a postfilter that attenuates coding noise while using typical background noise attenuation techniques. Such a postfilter was proposed in [18] for the G.711.1 speech coder. This coder is an extension of the legacy G.711 coder [3]. In the next chapter, we give an overview of the G.711.1 speech coder. One of the major modification done in the extended coder is coding noise shaping. Understanding how the noise is shaped is important in designing a filter that attenuates so we will continue our discussion in the next chapter by explaining the shaping procedure. Finally, we will look at the postfilter proposed in the standard.

Chapter 4

G.711 Quantizer

There exists many kinds of quantizers but one needs to select the most appropriate for a given application. Some popular ones are the simple uniform quantizer, the pdf-optimized quantizer and the logarithmic quantizer. For speech signals, the uniform and pdf-optimized quantizers are not adequate SNR-wise. These two quantizers are very sensitive to changes of the signal variance but the variance of speech signals varies a lot with time. On the other hand, a logarithmic quantizer SNR does not depend too much on the signal variance. The logarithmic quantizer is therefore a better selection for speech signals.

ITU-T G.711 pulse code modulation (PCM) of voice frequencies is a very popular narrow-band high-bitrate coder. It was standardized in 1972 by ITU-T. We will also refer to the ITU-T G.711 as the *legacy G.*711. The input and output signals of the coder are sampled at 8000 Hz. Each sample is encoded with 8 bits. As a result, the bitrate of the legacy G.711 coder is 64 kbit/s (8000 samples/sec $\times$ 8 bits/sample). Two encoding laws are supported by the legacy G.711. They are A-law and μ-law. These laws are logarithmic companding laws: the quantization step size changes depending on the input signal amplitude. Consequently, for speech signals the quantization error is smaller on average in this system compared to one that uses a quantizer with a fixed step size. The legacy G.711 was specifically designed for telephony-band signals (300–3400Hz).

4.1 Logarithmic Quantization

If one knows the probability distribution function (PDF) of the input signal, one can design a quantizer that will generate a better SNR than the simple uniform quantizer.

The resulting quantizer is nonuniform: the quantization intervals are smaller where the probability of the signal energy is the high and they are bigger where the probability of the signal energy is small. A model that achieves such a nonuniform quantization is one that consists of a compressor function $C(x)$ and a uniform quantizer at the encoder and then a dequantizer and an expander function at the decoder to recover the signal. The effect of applying the compressor on the input signal is that it renders its PDF uniform within its dynamic range. Jayant and Noll have shown in [19] that when the PDF $p(x)$ of the input is smooth, the quantization noise variance is given by:

$$\sigma_q^2 \approx \frac{x_{\max}^2}{3 \cdot 2^{2b}} \int_{-x_{\max}}^{x_{\max}} \frac{p(x)}{|C'(x)|^2} dx \tag{4.1}$$

where $C'(x)$ represents the derivative of $C(x)$.

One can find the companding function $C(x)$ that minimizes σ_q^2. The resulting SNR is maximized in this case but it still depends on the variance of the signal. Such a quantizer is not too appropriate for speech. One can also find a companding function which leads to a constant SNR over a broad range of signal variance values. As stated earlier, such quantizers better suit speech signal applications. Two popular examples of these quantizers are the logarithmic quantizers A-law and μ-law which we describe in the following section.

4.2 A-law and μ-law Quantizers

The compression function for the A-law compander is given by:

$$C(x) = \begin{cases} \dfrac{A|x|/x_{\max}}{1 + \ln A} \operatorname{sgn} x & 0 \le \frac{|x|}{x_{\max}} < 1/A, \\ x_{\max} \dfrac{1 + \ln A|x|/x_{\max}}{1 + \ln A} \operatorname{sgn} x & 1/A \le \frac{|x|}{x_{\max}} \le 1. \end{cases} \tag{4.2}$$

The compression function has a linear portion for small signals and a logarithmic portion for signals whose norms are greater than $x_{\max}/A$.

The compression function for the μ-law compander is given by:

$$C(x) = x_{\max} \frac{\ln(1 + \mu|x|/x_{\max})}{\ln(1 + \mu)} \operatorname{sgn} x \tag{4.3}$$

We can notice that the μ-law companding function is linear for small signals since $\ln(1+ax) \approx ax$. It is logarithmic for large signal values. When $\mu|x| \gg x_{max}$, Eq. (4.3) becomes:

$$C(x) = x_{\max} \frac{\ln(\mu|x|/x_{\max})}{\ln(1+\mu)} \operatorname{sgn} x \tag{4.4}$$

In the ITU-T standard, $A = 87.56$ and $\mu = 255$.

4.3 A-law and μ-law Approximations

In the standard, the compression functions are not directly used when coding with A-law or μ-law. Rather, piecewise linear approximations to the functions are used.

An A-law or μ-law quantizer encodes a 16-bit sample with 8 bits [3] as follows:

S	E2	E1	E0	M3	M2	M1	M0
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

More specifically, the legacy G.711 encoders are symmetric with 8 positive segments and 8 negative segments. The sign of the sample is stored at bit 7, often called the sign bit. The segment index is stored in the three exponent bits from bit 6 to bit 4 in the code. Each segment is associated to a 16 level uniform quantizer. Each level of the latter is stored from bit 3 to bit 0. This portion of the code is the mantissa.

4.4 A-law Properties and μ-law properties

In this thesis, we focus on A-law. In this section, we will explore some of the properties of A-law.

The compression function for the A-law compander is given in Eq. (4.2). Using it we can derive the SNR as a function of the load factor. The load factor is defined as $\Gamma = \sigma_x/x_{\max}$. This factor shows how well the signal uses its dynamic range. For small signals (uniform portion):

$$\mathrm{SNR}^A_{\mathrm{unif}} = 3 \cdot 2^{2b} \Big(\frac{A}{1+\ln A}\Big)^2 \frac{\sigma_x^2}{x_{\max}^2}. \tag{4.5}$$

In dB, when $b = 8$ bits, the SNR for the uniform portion is given by:

$$\mathrm{SNR}^A_{\mathrm{unif}} \approx 77.02 + 20 \log \big(\frac{\sigma_x}{x_{\max}}\big) \quad [\mathrm{dB}].$$

In terms of the load factor, we get:

$$\mathrm{SNR}^A_{\mathrm{unif}} \approx 77.02 - 20 \log \Gamma \quad [\mathrm{dB}]. \tag{4.6}$$

For large signals (logarithmic portion of the companding function), we get:

$$\mathrm{SNR}^A_{\log} = 3 \cdot 2^{2b} \big(\frac{1}{1 + \ln A}\big)^2. \tag{4.7}$$

In dB, when $b = 8$ bits, the SNR for the logarithmic portion is given by:

$$\mathrm{SNR}^A_{\log} \approx 38.16 \quad [\mathrm{dB}]. \tag{4.8}$$

The transition between the two portions can be obtain by equating equations Eq. (4.5) and Eq. (4.7) and solving for Γ. We will refer to this as the transition threshold Γ_{th}. We get:

$$\begin{aligned}
\mathrm{SNR}^A_{\mathrm{unif}} &= \mathrm{SNR}^A_{\log}, \\
10 \log \Big(3 \cdot 2^{2b} \big(\frac{A}{1 + \ln A}\big)^2 \frac{\sigma_x^2}{x_{\max}^2}\Big) &= 10 \log \Big(3 \cdot 2^{2b} \big(\frac{1}{1 + \ln A}\big)^2\Big), \\
10 \log \big(\frac{\sigma_x}{x_{\max}}\big)^2 &= 10 \log \big(\frac{1}{1 + \ln A}\big)^2 - 10 \log \big(\frac{A}{1 + \ln A}\big)^2, \\
-20 \log \big(\frac{\sigma_X}{x_{\max}}\big) &= 10 \log \big(\frac{A}{1 + \ln A}\big)^2 - 10 \log \big(\frac{1}{1 + \ln A}\big)^2, \\
20 \log \Gamma_{\mathrm{th}} &= 10 \log \big(\frac{A}{1 + \ln A}\big)^2 - 10 \log \big(\frac{1}{1 + \ln A}\big)^2.
\end{aligned}$$

We can see that the value of Γ is independent of b at the intersection of the two functions. We get:

$$\Gamma^A_{\mathrm{th}} \approx 38.86 [\mathrm{dB}]. \tag{4.9}$$

In Fig. 4.1 we plotted the SNR as a function of the load factor Γ for A-law.

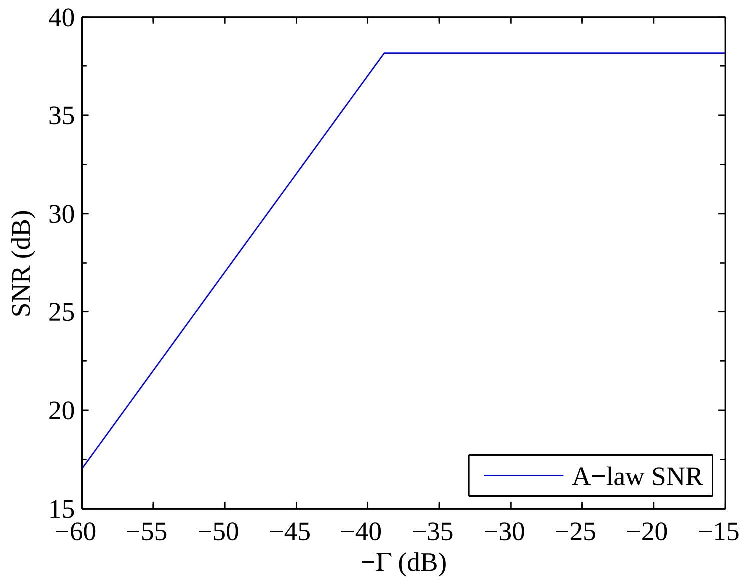


Fig. 4.1 SNR vs. load-factor Γ for A-law

μ-law properties can also be obtained. Following similar approach as the one taken above for A-law, we can get the SNR for μ-law. For small signals, we get:

$$\text{SNR}^{\mu}_{\text{unif}} = 3 \cdot 2^{2b} \Big(\frac{\mu}{\ln{(1+\mu)}}\Big)^2 \frac{\sigma_x^2}{x_{\max}^2}. \tag{4.10}$$

In dB, when $b = 8$ bits, the SNR for the uniform portion in terms of the load factor is given by:

$$\text{SNR}^{\mu}_{\text{unif}} \approx 86.19 - 20 \log \Gamma \quad [\text{dB}]. \tag{4.11}$$

For large signals, we get:

$$\text{SNR}^{\mu}_{\log} = 3 \cdot 2^{2b} \Big(\frac{1}{\ln{(1+\mu)}}\Big)^2. \tag{4.12}$$

For $\mu = 255$ and for $b = 8$ as in the standard, this gives:

$$\text{SNR}^{\mu}_{\log} \approx 38.06 \quad [\text{dB}]. \tag{4.13}$$

We get:

$$\Gamma^{\mu}_{\text{th}} \approx 48.13[\text{dB}]. \tag{4.14}$$

In Fig. 4.2 we plotted the SNR as a function of the load factor Γ for μ-law.

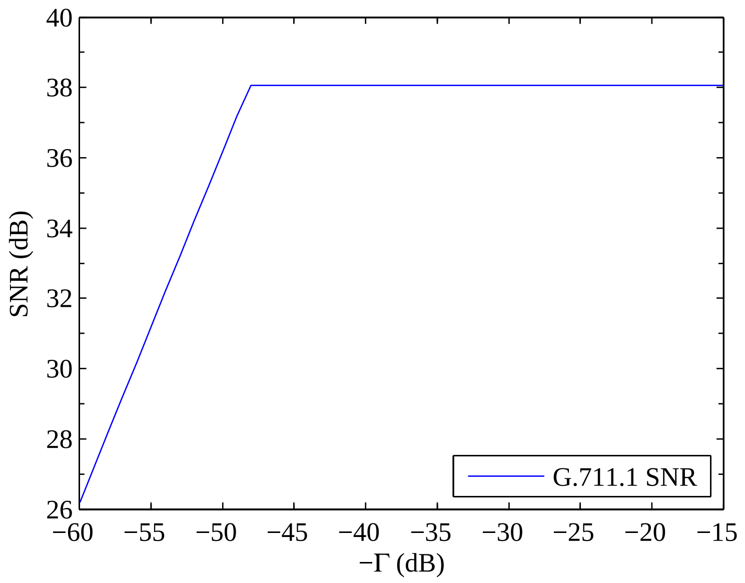


Fig. 4.2 SNR vs. load-factor Γ for μ-law

The SNRs for A-law and μ-law are very similar for large signals. We will see in Chapter 5 that the postfilter proposed in the G.711.1 standard only uses the A-law properties to estimate the quantization noise (even if the signal was coded using μ-law).

Chapter 5

ITU-T G.711.1

ITU-T G.711.1 [2] is a multirate wideband extension for the well-known ITU-T G.711 [3] pulse code modulation of voice frequencies. The extended system was approved in March 2008. G.711.1 is interoperable with G.711 at 64 kbit/s. It can also operate at 80 or 96 kbit/s.

The quantization noise is essentially white in the legacy G.711 coder. The postfilter proposed in the standard is only applied to signals encoded by the legacy G.711 encoder and decoded by the wideband G.711.1 decoder. This postfilter is therefore designed based on the assumption that the noise is white. We will look into applying a postfilter on signals that were encoded with G.711.1 as well. In this case however, the noise is spectrally shaped and therefore, it is not white.

In this chapter, we will start by giving an overview of G.711.1 speech coder in Section 5.1. We will then look into how the noise shaping is done in this wideband extension of the legacy G.711 coder as this information will be important for the design of a postfilter applied to signals encoded with G.711.1. Finally, we will explore the postfilter proposed in the G.711.1 standard.

5.1 Overview of the G.711.1 speech coder

G.711.1 is a multirate coder that supports 8 and 16 kHz signals. The system operates at 64, 80 or 96 kbit/s. It is fully interoperable with the legacy G.711 coder when it operates at 64 kbit/s. Specifically, a signal that is encoded with the legacy narrowband G.711 can be decoded by G.711.1 and vice-versa as shown in Fig. 5.1.

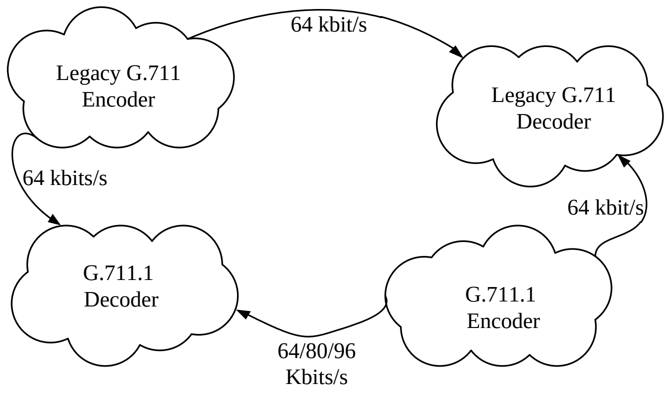


Fig. 5.1 Interoperability of G.711 and G.711.1

G.711.1 coder has four modes of operation. They are defined as R1, R2a, R2b and R3.

- In R1 mode, the bitrate of the coder is 64 kbit/s. Specifically, the input and output signals are sampled at 8 kHz. The coder generates a core bit-stream (Layer 0) I_{L0} that is compatible with legacy G.711. An important feature in this mode that makes it different from the legacy G.711 coder is the quantization noise shaping. This core layer is used in all 4 modes of the codec.

- In R2a, an additional lowerband enhancement layer (Layer 1) I_{L1} is produced at 16 kbit/s on top I_{L0} of making the overall bitrate of the coder 80 kbit/s.

- In R2b, an additional higherband enhancement layer (Layer 2) I_{L2} is produced at 16 kbit/s on top I_{L0} of making the overall bitrate of the coder 80 kbit/s.

- In R3, all three layers (I_{L0}, I_{L1} and I_{L2}) are present making the overall bitrate of the coder 96 kbit/s.

The four modes are summarized in Table 5.1.

5.1.1 The G.711.1 encoder

The input of the G.711.1 encoder can be sampled at a frequency of 16 kHz (wideband signal input) or at a frequency of 8 kHz (narrowband signal input). The encoder processes the signal on a frame-by-frame basis. Each frame is 5 ms long. Therefore, for a wideband signal the frame length is 80 samples and for a narrowband signal, the frame length is 40 samples.

Table 5.1 Modes of operations of G.711.1

Mode	Sampling Frequency kHz	Core Layer 64 kbit/s	Lowerband Enhancement Layer 16 kbit/s	Higherband Enhancement Layer 16 kbit/s	Overall Bitrate kbit/s
R1	8	✓	-	-	64
R2a	8	✓	✓	-	80
R2b	16	✓	-	✓	80
R3	16	✓	✓	✓	96

If the input is a wideband signal, it goes through a Quadrature Mirror Filter (QMF) analysis. The QMF splits the wideband signal into two narrowband signals: a lowband signal and a highband signal. This is done so that the two bands can be coded independently and thus, increase the coding performance. The QMF analysis is shown in Fig. 5.2. The

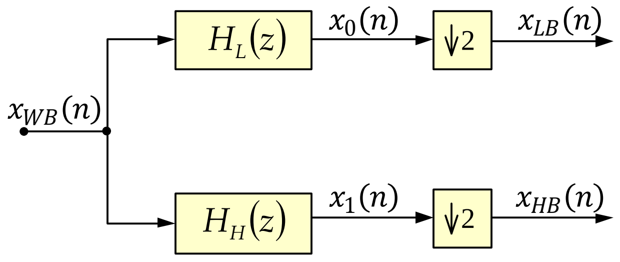


Fig. 5.2 QMF Analysis

8-kHz lowerband signal is obtained by filtering the 16-kHz input signal with a 32-tap linear phase low-pass filter h_L^{qmfA}. This filtering gives the 16-kHz signal $x_0(n)$ which only contains the lowerband portion of the input signal. Downsampling is then applied to $x_0(n)$ to obtain $x_{\text{LB}}(n)$. The latter contains the lowerband portion of the input signal just as $x_0(n)$ does but its sampling rate is 8 kHz. Similarly, the 8-kHz higherband signal $x_{\text{HB}}(n)$ is obtained by first filtering the 16-kHz input signal with a 32-tap linear phase high-pass filter h_H^{qmfA} to obtain $x_1(n)$ and and then by downsampling $x_1(n)$. This high-pass filter is related to h_L^{qmfA} by the following:

$$h_H^{\text{qmfA}}(n) = (-1)^n h_L^{\text{qmfA}}. \tag{5.1}$$

The lowerband signal $x_{\mathrm{LB}}(n)$ is then fed to the lowerband encoder and the higherband signal $x_{\mathrm{HB}}(n)$ is fed to the higherband encoder as shown in Fig. 5.3.

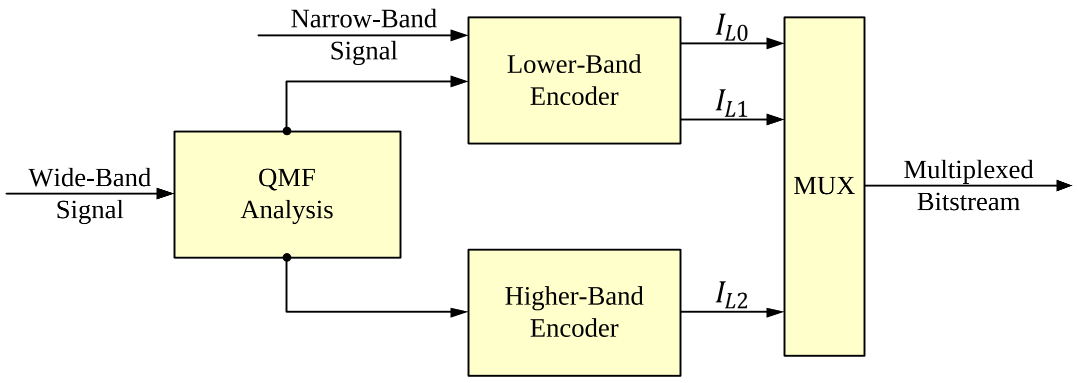


Fig. 5.3 G.711.1 high-level encoder diagram

Lowerband Encoder

The input to the lowerband encoder is $x_{\mathrm{LB}}(n)$. It was obtained either after a QMF analysis if the original input was sampled at 16 kHz or directly if the input signal to the codec was sampled at 8 kHz. The signal $x_{\mathrm{LB}}(n)$ is a narrowband signal.

When the codec operates in R1 mode, the lowerband encoder produces the bitstream I_{L0} at a 64 kbit/s bitrate. When the codec operates in either R2a or R3 mode, the lowerband encoder produces two bitstreams, I_{L0} and I_{L1} at a 64 kbit/s bitrate and a 16 kbit/s bitrate respectively. The coder here is thought of as operating in a layered-based system where the bitstream I_{L0} is obtained from Layer 0 and the bitstream I_{L1} is obtained from Layer 1. Since I_{L0} is used in all operating modes of the G.711.1 codec, Layer 0 is also referred to as the *Core Layer*. Layer 1 is also called the *Lowerband Enhancement Layer*.

The core layer encoder consists of an embedded A-law or μ-law PCM encoder with noise feedback to perceptually shape the coding noise. We will see how this is done in Section 5.2.1. Signals with very low energy are encoded with a *dead-zone quantizer* [2]. Specifically, these low energy samples will be decoded as 0 samples. During the generation of I_{L0}, a refinement signal $x_{\mathrm{LBref}}(n)$ and the exponent signal $x_{\mathrm{LBexp}}(n)$ are computed. The lower band enhancement layer uses them to generate I_{L1}. In Appendix B, we give a detailed explanation on the Layer 1 computation.

Higherband Encoder

The input to the higherband encoder is $x_{\mathrm{HB}}(n)$. This signal is the higherband portion of the signal obtained after the QMF analysis of the wideband input signal. $x_{\mathrm{HB}}(n)$ is transformed into its frequency domain representation $X_{\mathrm{HB}}(k)$ through the Modified Discrete Cosine Transform (MDCT). The frequency components $X_{\mathrm{HB}}(k)$ are then quantized using an interleaved Conjugate-Structured Vector Quantization (CS-VQ) at a bitrate of 16 kbit/s. The codebook vectors indices are then multiplexed to form the bitstream I_{L2}. This bitstream is also referred to as the *Higherband Enhancement Layer*.

5.1.2 The G.711.1 decoder

The decoder is also organized in three layers. The multiplexed bitstream is first de-multiplexed to regenerate the three bitstreams I_{L0}, I_{L1} and I_{L2}. I_{L0} and I_{L1} are both fed into the lowerband decoder and I_{L2} is fed to the higherband decoder. I_{L0} and I_{L1} are decoded independently in their respective layers and they are then combine to give the decoded lowerband signal. I_{L2} is decoded into the higherband signal. Depending on the operation mode of the speech coder, either a narrowband decoded signal is produced or a wideband signal is produced. In the narrowband case, the output is obtained directly from the lowerband decoded signal. In the wideband case, the output is obtained after combining the lowerband decoded signal and the higherband decoded signal by using a QMF synthesis system. The high-level decoder is shown in Fig. 5.4. More details on the functions of the different blocks of the decoder are given below.

Lowerband Decoder

The inputs to the lowerband decoder are the de-multiplexed bitstreams I_{L0} and I_{L1}. The layer 0 bitstream is fed to the core-layer decoder to be decoded and the Layer 1 bitstream is fed to the lowerband enhancement layer decoder to be decoded. The core-layer decoder is a simple de-quantizer just as in the legacy G.711 codec discussed in Chapter 4. The decoded signal is generated according to the companding law that was used to encoded it. During this decoding stage, the exponent and the sign are recorded for each sample. The lowerband enhancement layer decoder uses them to decode I_{L1}.

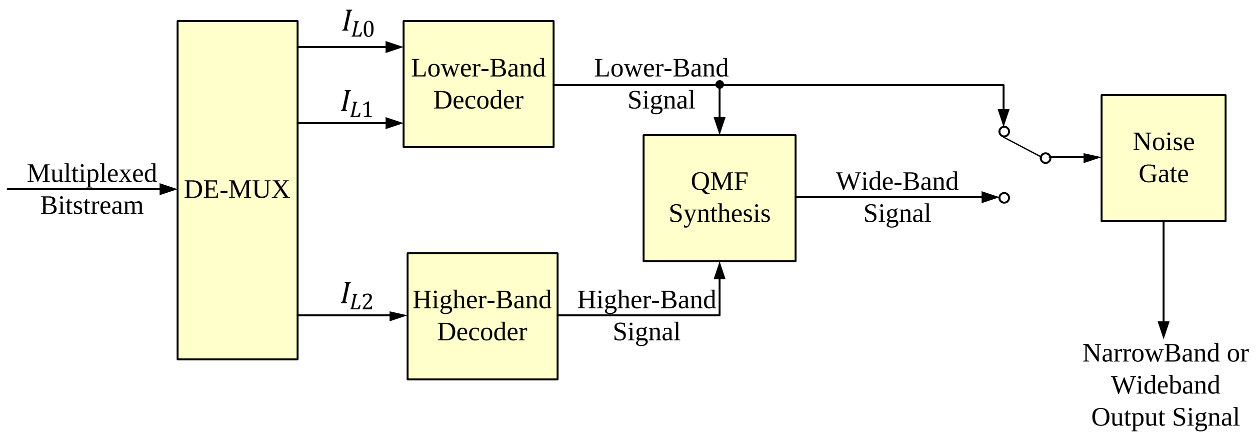


Fig. 5.4 G.711.1 high-level decoder diagram

Higherband Decoder

The vectors are decoded from I_{L2} and the MDCT coefficients are reconstructed. These coefficients are transformed back to time domain through the inverse MDCT to reconstruct the higher-band signal.

Noise-Gate

Before the final decoded signal is output, it goes through a noise-gate. The role of the latter is to reduce low-level background noise. It is particularly useful to reduce the noise when only a small number of samples in a frame were not set to zero by the dead-zone quantizer. It further improves the perceptual quality of the speech.

5.2 Noise Shaping in G.711.1

5.2.1 Noise-shaping at the encoder

The core-layer encoder codes each speech sample in a frame using a companding law (either A-law or μ-law) to produce the I_{L0} bitstream. The legacy G.711 uses the same laws to encode speech samples. However, the bitstream generated by the legacy codec is different from the one generated by the core-layer encoder in G.711.1.

In G.711, the quantization noise spectrum is flat and the telephony bandwidth (300–3400 Hz) is used. Perceptually, a flat coding noise is not optimal. For a typical speech

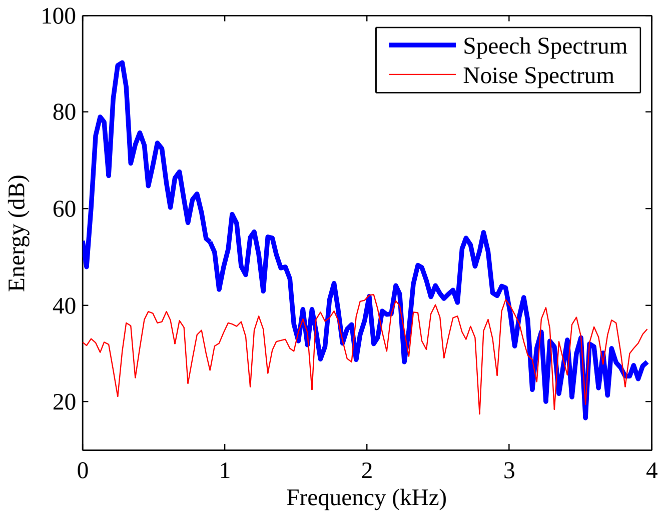


Fig. 5.5 Signal and quantization noise spectrum example of a 160-sample long frame. The signal is sampled at 8 kHz. The signal was encoded by the legacy G.711 encoder using A-law.

frame, the noise is audible at high frequencies because that is where the noise energy is typically higher than that of the speech. An example is shown in Fig. 5.5. We have plotted the speech spectrum in bold and the noise spectrum with a thin line. The speech signal here is sampled at 8 kHz. We sectioned the speech into blocks of 160 samples [1]. We then encoded and decoded each block with A-law. We computed the quantization error signal for each block as the difference between the input and the decoded blocks. We finally applied a Fast Fourier Transform (FFT) of length 256 to the each block for both the input signal and the error signal.

In G.711.1, for optimal wideband quality, the lowerband signal uses the range 50–4000 Hz. Therefore, a flat noise in G.711.1 would be more easily perceived and more annoying than in G.711. In order to remedy this problem, G.711.1 uses noise spectral shaping around the quantizer. The effect of noise shaping applied on the same block shown in Fig. 5.5 appears in Fig. 5.6. The mode of operation of the codec is R1 here. Using the local decoded value, we computed the error signal. We see that the noise is shaped to follow the spectral envelope of the speech. The auditory masking property of the human ear makes the noise less audible at high frequencies.

[1]Note that G.711.1 operates on frames of 40 samples. We use 160 samples for the plot to increase the frequency resolution

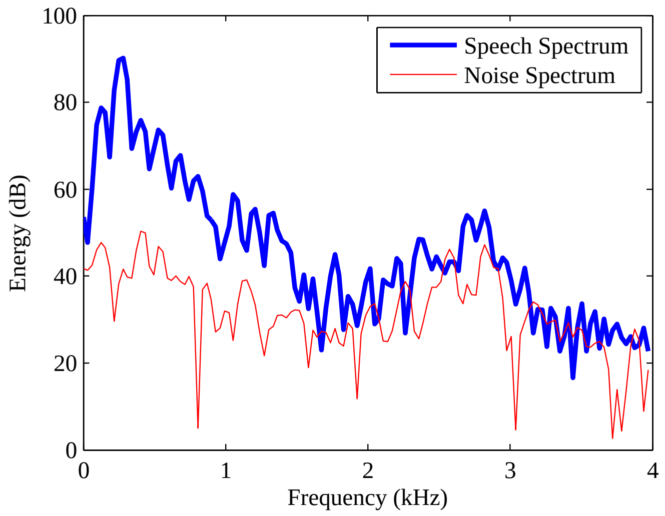


Fig. 5.6 Signal and quantization noise spectrum example of a 160-sample long frame. The signal is sampled at 8 kHz. The signal was encoded by the G.711.1 encoder using A-law in R1 mode.

Figure 5.7 shows a diagram of the core-layer encoder. The noise-shaping is done around the G.711 quantizer which really consists of either an A-law or a μ-law quantizer. This method of noise spectral shaping was proposed in [20] by Atal and Schroeder but it was used on a non-predictive coder. It is an efficient implementation of the recursive form of the noise shaping filter.

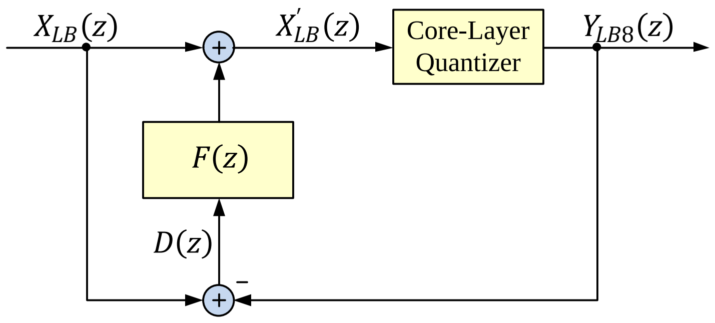


Fig. 5.7 Noise Shaping in G.711.1

The input signal to the encoder is $X_{\mathrm{LB}}(z)$. The input signal to the quantizer is $X'_{\mathrm{LB}}(z)$. The locally decoded output is $Y_{\mathrm{LB8}}(z)$. The perceptual filter used to shape the noise is $F(z)$. We can then write:

$$Y_{\mathrm{LB8}}(z) = X'_{\mathrm{LB}}(z) + Q_8(z), \tag{5.2}$$

where $Q_8(z)$ is the companding law's quantization noise with flat spectrum. The input to the quantizer is:

$$X'_{\mathrm{LB}}(z) = X_{\mathrm{LB}}(z) + (X_{\mathrm{LB}} - Y_{\mathrm{LB8}}(z))\, F(z). \tag{5.3}$$

By substituting X'_{LB} in Eq. (5.2), we get:

$$Y_{\mathrm{LB8}}(z) = X_{\mathrm{LB}} + (X_{\mathrm{LB}} - Y_{\mathrm{LB8}}(z))\, F(z) + Q_8(z). \tag{5.4}$$

Rearranging, we get:

$$Y_{\mathrm{LB8}}(z) = X_{\mathrm{LB}} + \frac{Q_8(z)}{1 + F(z)}. \tag{5.5}$$

We see from Eq. (5.5) that the decoded signal Y_{LB8} is simply equal to the input signal X_{LB} and the quantization noise shaped by the filter $\frac{1}{1+F(z)}$. The effect of noise shaping is shown in Fig. 5.6. We now see that the quantization noise is higher in lower frequencies compared to the flat noise case. However, the noise remains inaudible at these frequencies due to masking by the speech spectrum. At higher frequencies, the coding noise is now lower compared to the flat noise case. It is lower than the speech level and therefore, it get masked as well. As a result, the perceived noise is much less after noise shaping.

The perceptual filter $F(z)$ is estimated so that the quantization noise is shaped in a perceptually relevant way. The psychoacoustic model followed in G.711.1 for the computation of this filter is based on the noise weighing filter of the Adaptive Multirate Wideband (AMR-WB) standard speech codec [18]. The transfer function of the perceptual filter is given by:

$$F(z) = A(z/\gamma_1) - 1. \tag{5.6}$$

Here, $A(z)$ is a linear prediction (LP) transfer function and γ_1 is a factor which controls the perceptual weighting. γ_1 is real and bounded: $0 < \gamma_1 \leq 1$. The advantage of the noise weighing filter employed in the AMR-WB is that the linear prediction filter $A(z)$ is computed based on a pre-emphasized signal. This pre-emphasis is done on the signal to control the tilt of the noise shaping filter. Note that the ideas here are similar to the ones discussed for parametric postfiltering in Chapter 3.

For each frame, the LP filter is computed based on past decoded signal of two frame length (10 ms) in G.711.1. We denote this past signal as $x_{\text{past}}(n)$ with $n = -80, ..., -1$ where the range $-80, ..., -1$ represent the past 80 samples i.e. the past two frames. The signal $x_{\text{past}}(n)$ is pre-emphasized with a first-order filter $P(z) = 1 - \beta z^{-1}$ where β is computed based on the zero-crossing rate of the signal. The pre-emphasized signal is denoted $x_{\text{past}}^{\text{emp}}(n)$. The LP analysis filter $A(z)$ is computed based on $x_{\text{past}}^{\text{emp}}(n)$. The order of the LP filter is 4:

$$A(z) = 1 + \sum_{k=1}^{4} a_k z^{-k}, \tag{5.7}$$

and after applying the weighing factor and using Eq. (5.6), we get the perceptual filter $F(z)$:

$$F(z) = \sum_{k=1}^{4} \gamma_1^k a_k z^{-k}. \tag{5.8}$$

In most cases, γ_1 is set to the constant value of 0.92. There are two situations however where the G.711.1 further attenuates the perceptual weighting filter. The first attenuation of the perceptual filter occurs for signals with very low energy. In such a situation, the noise might not be masked even with the noise shaping. A power mismatch of the decoded signal can happen and the noise feedback loop may become saturated. The attenuation of the perceptual filter prevents this saturation. The second attenuation of the perceptual filter occurs for signals with energy concentrated at high frequencies. This case is rare but if it occurs, the noise feedback could become unstable. The attenuation prevents this problem.

One other feature of the G.711.1 codec to further improve the quality of the signal is the presence of a *dead-zone* quantizer. This quantizer replaces the log-PCM quantizer when the following two conditions are met:

- the energy of the frame is very low,
- the sample level is within a certain low region.

The dead-zone quantizer enlarges the zero-output interval of the quantizer when compared to the μ-law and A-law quantizer. More specifically, if portions of the signal satisfying the two conditions were directly processed by the log-PCM quantizer, they would sound like background noise when decoded. The dead-zone quantizer removes this perceptual effect by nullifying the signal when the two conditions are met.

In summary, the core-layer encoder of G.711.1 uses the G.711 quantizer as a basis to encode the lowerband signal. This is what makes the two systems interoperable. The presence of a noise shaper around the embedded quantizer in G.711.1 is the first reason for a difference in the bitstream outputted by the two codecs. Indeed, the signals encoded by the quantizers in the each codec are different due to this added noise shaper in the newer system. Furthermore, the additional processing done in G.711.1 to improve the perceptual quality (perceptual filter attenuation and dead-zone quantizer) also contribute to the difference of the bitstreams.

5.2.2 Noise-Shaping at the decoder

The refinement signal consists of 2 bits per sample *on average*. These refinement bits are used to increase the quality of the core-layer decoded signal. We have also seen that the noise shaping applied at the encoder only takes the quantization noise of the core-layer into account. As a matter of fact, it cannot include the refinement signal as it does not know whether or not the lower band enhancement layer will be used at the decoder side or not. Therefore, for noise shaping to happen in both layers, additional processing has to be done *at the decoder* for Layer 1.

We will now explore what kind of processing has to be done on the decoded refinement signal so that proper shaping is applied to it as well. We will start by assuming the situation shown in Fig. 5.8 where Y_{LB10} is the output of a 10-bit quantizer i.e. a G.711 quantizer with 6 bits allocated for the mantissa. Y_{LB2} represents the two least significant bits of the mantissa of Y_{LB10} and Y_{LB8} represents the 8 most significant bits of Y_{LB10}. Therefore, Y_{LB8} is exactly the same as the one in Fig. 5.7. We see that in Fig. 5.8, the noise feedback is still computed based on Y_{LB8}. Therefore, Eq. (5.3) still holds here. We can write:

$$Y_{\text{LB10}}(z) = X'_{\text{LB}}(z) + Q_{10}(z), \tag{5.9}$$

where $Q_{10}(z)$ is the quantization noise gotten from the 10-bit quantizer. It is different from

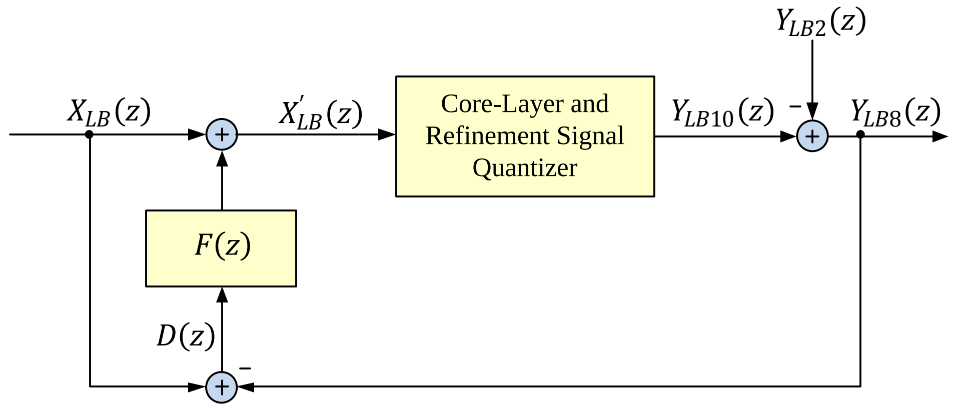


Fig. 5.8 Noise Shaping in G.711.1 if we include the lowerband enhancement layer

$Q_8(z)$. We also have:

$$Y_{\mathrm{LB10}}(z) = Y_{\mathrm{LB8}}(z) + Y_{\mathrm{LB2}}(z). \tag{5.10}$$

Substituting for $X'_{\mathrm{LB}}(z)$ in Eq. (5.3), we get:

$$Y_{\mathrm{LB10}}(z) - Q_{10}(z) = X_{\mathrm{LB}}(z) + (X_{\mathrm{LB}} - Y_{\mathrm{LB8}}(z))\,F(z). \tag{5.11}$$

Then, using Eq. (5.10) and re-arranging, we get:

$$Y_{\mathrm{LB10}}(z) = X_{\mathrm{LB}} + \frac{Q_{10}(z)}{1+F(z)} + Y_{\mathrm{LB2}}(z)\frac{F(z)}{1+F(z)}. \tag{5.12}$$

Equation (5.12) is the decoded signal we get if no processing is done to the decoded reference signal. What we would really want to have for proper shaping in both layers is just the first two terms of Eq. (5.12). Let $Y_D(z)$ denote this desired signal:

$$\begin{aligned}
Y_D(z) &= Y_{\mathrm{LB10}}(z) - Y_{\mathrm{LB2}}(z)\frac{F(z)}{1+F(z)}, \\
&= Y_{\mathrm{LB8}}(z) + Y_{\mathrm{LB2}}(z) - Y_{\mathrm{LB2}}(z)\frac{F(z)}{1+F(z)}, \\
&= Y_{\mathrm{LB8}}(z) + Y_{\mathrm{LB2}}(z)\big(1 - \frac{F(z)}{1+F(z)}\big).
\end{aligned}$$

This gives us,

$$Y_D(z) = Y_{\mathrm{LB8}}(z) + Y_{\mathrm{LB2}}(z)\frac{1}{1+F(z)}. \tag{5.13}$$

Therefore, the decoded reference code $Y_{\mathrm{LB2}}(z)$ needs to be filtered by $\frac{1}{1+F(z)}$ before it gets added to the decoded core-layer signal $Y_{\mathrm{LB8}}(z)$. $F(z)$ is estimated from $Y_{\mathrm{LB8}}(z)$.

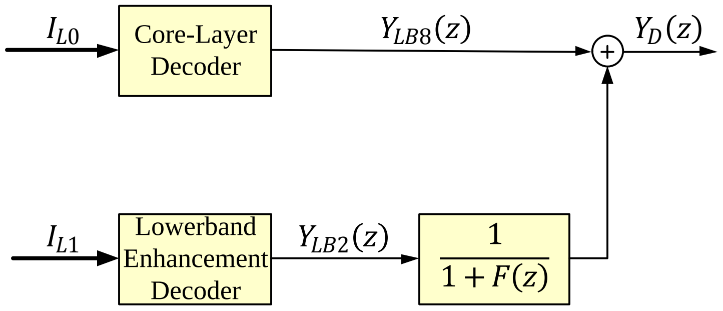


Fig. 5.9 Lowerband Decoder

5.3 Post-Filtering in G711.1

The quantization noise generated by a legacy G.711 codec is easily audible in high frequencies of the lower band. G.711.1 partly solved the problem by implementing noise shaping around the quantizer. As a result, a signal encoded with a G.711.1 encoder and then decoded by a legacy G.711 decoder will have a better quality then a signal encoded and decoded by a legacy G.711 codec. However, when we consider the other interoperable situation, i.e. a signal encoded by the G.711 encoder and then decoded by G.711.1 decoder, the quantization noise is not shaped and it can be heard. A solution to produce a better quality output is to implement a postfilter to reduce the quantization noise. The G.711.1 standard implements an optional postfilter described in Appendix I of [2].

The input of the postfilter is the decoded signal. For each frame, the postfilter estimates the quantization noise. It then estimates a Wiener filter that it applies on the frame to reduce the quantization noise. We notice that the conventional parametric postfilter methods as explained in Section 3.2 are not applied. Rather, this method is closer to the one used by speech enhancers to reduce acoustic background noise. The difference though, is that we estimate quantization noise rather than background noise. Therefore, rather than using unvoiced frames (used to estimate background noise), we use the properties of A-law to estimate the quantization noise for each frame. The proposed postfilter only uses the properties of A-law assuming that the noise estimate obtained by this method is also valid for signals that were encoded by μ-law.

In Section 4.4, we explored some of the properties of A-law. We will see how these properties are used to estimate the quantization noise for each frame below. We will then explain how the Wiener filter is estimated. Finally, we will explain how the filtering is carried out and the enhanced speech is obtained.

5.3.1 Quantization Noise Estimation

The quantization noise is well-known in its intensity and in its spectral shape for both A and μ-law. It is essentially white so it is completely defined by its variance. As we can see from Fig. 4.1, with a good estimate of the load factor, the SNR can be determined. Using the SNR and the estimated signal variance, one can obtain an estimate of the noise variance.

The first step consists of estimating the variance of the original signal. We will use "$\hat{}$" to denote estimated values. Let $x(n)$ be the original clean signal and $y(n)$ be the decoded signal. The output signal $y(n)$ is corrupted by quantization noise $q(n)$ and we have $y(n) = x(n) + q(n)$. In A-law and μ-law, the quantization noise level is usually lower than the signal. Therefore, the variance of the original signal can be estimated directly from the variance of the decoded signal: $\hat{\sigma}_x^2 \approx \hat{\sigma}_y^2$.

The variance of the decoded signal is estimated in time domain on a frame by frame basis. For a frame of length L, the estimation is given by:

$$\hat{\sigma}_x^2 = \frac{1}{L} \sum_{n=0}^{L-1} y^2(n). \tag{5.14}$$

In G.711.1, $L = 40$ samples. This corresponds to a frame length of 5 ms. The load factor is then estimated as:

$$\hat{\Gamma}^2 = \frac{x_{\max}^2}{\hat{\sigma}_x^2}, \tag{5.15}$$

Using this estimate and Fig. 4.1, the portion of the quantizer that was used to code the signal can easily be determined. When the estimated load factor $\hat{\Gamma}^2$ is greater than Γ^2_{th}, it can be concluded that the uniform part of the quantizer was used. This means that the SNR in this case is $\mathrm{SNR}_{\mathrm{unif}}$ and this leads to a constant noise variance:

$$\mathrm{SNR}_{\mathrm{unif}} = 3 \cdot 2^{2b} (\frac{A}{1 + \ln A})^2 \frac{\sigma_x^2}{x^2_{\mathrm{max}}},$$
$$\frac{\sigma_x^2}{\sigma_q^2} = 3 \cdot 2^{2b} (\frac{A}{1 + \ln A})^2 \frac{\sigma_x^2}{x^2_{\mathrm{max}}}.$$

Therefore, a constant variance is estimated in this case as:

$$\hat{\sigma}_q^2 = \frac{1}{3 \cdot 2^{2b} (\frac{A}{1 + \ln A})^2} x^2_{\mathrm{max}}. \tag{5.16}$$

When the signal energy becomes comparable to that of the quantization error, the approximation $\hat{\sigma}_x^2 \approx \hat{\sigma}_y^2$ is no longer valid. In such cases, the postfilter forces the estimated $\hat{\sigma}_q^2$ to be 15dB lower than the signal's variance.

When the estimated load factor $\hat{\Gamma}^2$ is smaller than Γ^2_{th}, it can be concluded that the logarithmic part of the quantizer was used. This means that the SNR in this case is $\mathrm{SNR}_{\mathrm{log}}$ and this leads the following noise variance:

$$\mathrm{SNR}_{\mathrm{log}} = 3 \cdot 2^{2b} (\frac{1}{1 + \ln A})^2,$$
$$\frac{\sigma_x^2}{\sigma_q^2} = 3 \cdot 2^{2b} (\frac{1}{1 + \ln A})^2.$$

Therefore,

$$\hat{\sigma}_q^2 = \frac{\hat{\sigma}_x^2}{3 \cdot 2^{2b} (\frac{1}{1 + \ln A})^2}. \tag{5.17}$$

Unfortunately, two mistakes were made in the standard and ted reference code accompanying the standard in implementing this noise estimator. We explain the mistakes in Appendix A.

5.3.2 Wiener Filter Estimation

The filter used to reduce the coding noise here is based on a Wiener filter. This filter is computed based on the estimated noise and the two-step noise reduction (TS-NR) method described in Chapter 2. The computation steps are summarized below.

The input frame is first windowed and then transformed into frequency domain:

$$y_w(n) = w(n) \cdot y(n) \tag{5.18}$$

$$Y_w(k) = \mathrm{FT}\{y_w(n)\} \tag{5.19}$$

The length of a frame being 40 samples here, the window and the DFT lengths are 64. The signal PSD is estimated using the decision-directed algorithm with $\beta = 0.98$:

$$|\hat{X}(p,k)|^2 = \beta |\hat{X}(p-1,k)|^2 + (1-\beta)\max(0, |Y(p,k)|^2 - |\hat{B}(p,k)|^2). \tag{5.20}$$

The first spectral gain is then computed as:

$$G_{\mathrm{DD}}(p,k) = \frac{|\hat{X}(p,k)|^2}{|\hat{B}(p,k)|^2 + |\hat{X}(p,k)|^2}. \tag{5.21}$$

Using this first gain, the first step enhancement is computed as:

$$|\hat{X}_{\mathrm{DD}}(p,k)|^2 = |G_{\mathrm{DD}}(p,k)|^2 |Y(p,k)|^2 \tag{5.22}$$

The enhancing Wiener filter is finally computed as:

$$G_{\mathrm{TS-NR}}(p,k) = \frac{|\hat{X}_{\mathrm{DD}}(p,k)|^2}{|\hat{B}(p,k)|^2 + |\hat{X}_{\mathrm{DD}}(p,k)|^2}. \tag{5.23}$$

After the filter is obtained in the frequency domain, it is transformed back to the time domain. It is then truncated to 33 samples and smoothed through windowing before it is applied to the frame. There is an additional distortion control system that is applied to make sure that the postfilter does not distort the signal too much.

Chapter 6

Improved Noise Estimator

G.711.1 is a high bit-rate non-parametric coder. As such, the parameters needed by conventional postfilters are not available at the decoder end as they typically are for low bit-rate parametric coders. Designing and implementing a conventional postfilter for G.711.1 would require the computations of these parameters. The complexity would increase. One of the main goals of the wideband extension system for the legacy G.711 was to keep the complexity as low as possible. The postfilter proposed in G.711.1 is a low-complexity postfilter. As we have seen in Chapter 5, it estimates the noise using properties of A-law. It then uses the acoustic background noise attenuation methods to reduce the perceptual effect of the coding noise. Thus, no parameters need to be computed and the complexity remains relatively low.

The postfilter design is based on a windowed signal. Therefore, its design should account for the window effect. However, this consideration is omitted in G.711.1 and the postfilter's performance is thus reduced. The postfilter in G.711.1 is designed only to enhance signals that were encoded by the legacy coder, i.e. signals where the noise was not shaped. We will however note that even when the noise is shaped, some of the coding noise can still be heard.

In this chapter, we propose a more accurate noise estimator. The noise estimate is then passed on to the TS-NR algorithm for the generation of the quantization noise filter. We also account for the window applied to the signal before it is transformed into its frequency version. We finally propose a low complexity postfilter for signals that were encoded by G.711.1.

6.1 Improved noise estimator

In this section, we estimate the quantization noise present in a signal encoded by the legacy G.711 system. Rather than estimating the noise from the overall input frame to the postfilter as it is done in the filter proposed in the G.711.1 standard (Section 5.3), we estimate it from each sample in the frame. This results in a better noise estimate.

The postfilter handles the decoded signal frame-by-frame. The noise associated to each sample can be estimated as follows. From a decoded sample, one can easily determine the segment in which it was coded. Each segment is associated with a uniform quantizer. The quantization noise resulting from the uniform quantizer is uniformly distributed on its dynamic range. Therefore, it is easy to obtain the quantization noise distribution for each decoded sample.

For A-law, each segment corresponds to a uniform coder. For segment $i_s[0-7]$, the step size is:

$$\Delta(i_s) = \begin{cases} 1 & i_s = 0, 1, \\ 2^{i_s - 1} & i_s > 1. \end{cases} \tag{6.1}$$

And for μ-law, for segment $i_s[0-7]$, the step size is:

$$\Delta(i_s) = \begin{cases} 1 & i_s = 0, \\ 2^{i_s} & i_s > 0. \end{cases} \tag{6.2}$$

For each segment, this leads to a noise variance of:

$$\sigma_q^2(i_s) = \frac{\Delta^2(i_s)}{12}. \tag{6.3}$$

For dynamic signals like speech, it is reasonable to assume that noise is independent on a sample-to-sample basis [19]. Therefore, for a frame of length L, the noise variance of a signal encoded by the legacy G.711 can simply be estimated as:

$$\hat{\sigma}_q^2 = \frac{1}{L} \sum_{n=0}^{L-1} \sigma_q^2\big(i_s(n)\big), \tag{6.4}$$

where $\sigma_q^2\left(i_s(n)\right)$ is the variance from Eq. (6.3) for each sample n in the frame.

An advantage of this method over the one proposed in the standard is that it does not rely on the load factor. The fact that the load factor threshold values for A-law and μ-law are different could lead to a reduced performance for the method in the standard as only the A-law parameters are used. In our method, the law that was used to code the signal is all we need to provide an accurate noise estimate.

6.2 Windowing Effect

The estimation of the postfilter response is performed in the frequency domain. The decoded signal is windowed in time domain and then it is transformed into its frequency form as shown below:

$$y_w(n) = w(n) \cdot y(n), \tag{6.5}$$

$$Y_w(k) = \text{FT}\{y_w(n)\}, \tag{6.6}$$

where $y(n)$ is the decoded signal, $y_w(n)$ is the widowed decoded signal, $w(n)$ is the applied window and FT$\{\cdot\}$ is the Fourier Transform. The model assumed here is the additive model. As we have previously seen, in such a model, $y(n) = x(n)+q(n)$ where $x(n)$ denote the clean signal and let $q(n)$ denote the quantization noise. The window thus affects both the clean signal portion and the quantization noise. We should therefore consider its effect while generating the postfilter that will be applied to $y(n)$. This was omitted in the postfilter proposed in G.711.1.

In frequency domain, the postfilter gain $G(k)$ is computed by the two-step noise reduction method. This gain calculation is based on the SNR:

$$\text{SNR} = \frac{|Y_w(k)|^2}{|\hat{N}(k)|^2}, \tag{6.7}$$

where$|\hat{N}(k)|^2$ is the estimated Power Spectral Density (PSD) of the noise. Since the windowed decoded signal is used here, windowing effects must be accounted for in the noise estimate.

Assume that we have obtained an estimate for the variance of the noise $\hat{\sigma}_q^2$ through one of the methods discussed previously. This estimate is for the “unwindowed” signal.

Parseval's theorem states that for any signal $s(n)$ with DFT $S(k)$, we have:

$$\sum_{n=0}^{L-1} |s(n)|^2 = \frac{1}{L} \sum_{k=0}^{L-1} |S(k)|^2. \tag{6.8}$$

Therefore:

$$\sum_{n=0}^{L-1} \mathrm{E}\big(|s(n)|^2\big) = \frac{1}{L} \sum_{k=0}^{L-1} \mathrm{E}\big(|S(k)|^2\big). \tag{6.9}$$

If $s(n)$ is white, then $S(k)$ is constant across all frequencies. Therefore:

$$\sum_{k=0}^{L-1} \mathrm{E}\big(|S(k)|^2\big) = L|S(k)|^2, \tag{6.10}$$

and,

$$\begin{aligned} \sum_{n=0}^{L-1} \mathrm{E}\big(|s(n)|^2\big) &= \frac{1}{L} L|S(k)|^2 \\ &= |S(k)|^2. \end{aligned} \tag{6.11}$$

Therefore, since the quantization noise is assumed white here, we have:

$$|Q(k)|^2 = \sum_{n=0}^{L-1} \mathrm{E}\big(|q(n)|^2\big). \tag{6.12}$$

The estimated quantization noise is given by:

$$\begin{aligned}|\widehat{Q}(k)|^2 &= \sum_{m=0}^{L-1} \hat{\sigma}_q^2, \\ &= \sum_{m=0}^{L-1} \Big(\frac{1}{L} \sum_{n=0}^{L-1} \sigma_q^2\big(i_s(n)\big)\Big), \\ &= L\frac{1}{L} \sum_{n=0}^{L-1} \sigma_q^2\big(i_s(n)\big), \\ &= \sum_{n=0}^{L-1} \sigma_q^2\big(i_s(n)\big).\end{aligned} \tag{6.13}$$

Let $q_w(n)$ be the windowed version of $q(n)$ i.e. $q_w(n) = w(n) \cdot q(n)$. We then get:

$$\mathrm{E}\{q_w^2(n)\} = \mathrm{E}\big(w^2(n)q^2(n)\big). \tag{6.14}$$

Therefore,

$$\sum \mathrm{E}(q_w^2(n)) = \mathrm{E}\big(\sum_{n=0}^{L-1} w^2(n) \cdot q^2(n)\big). \tag{6.15}$$

Since q_w is a windowed version of q, it is also white. So, from Eq. (6.13) and Eq. (6.15), we have:

$$|Q_w(k)|^2 = \mathrm{E}\big(\sum_{n=0}^{L-1} w^2(n) \cdot q^2(n)\big). \tag{6.16}$$

For $\mathrm{E}\big(q^2(n)\big)$, we can use the estimated $\hat{\sigma}_q^2$ by either method discussed previously. As a result, a good estimate for the PSD of the windowed noise is:

$$|\hat{Q}_w(k)|^2 = \sum_{n=0}^{L-1} w^2(n) \frac{\Delta^2\big(i_s(n)\big)}{12}. \tag{6.17}$$

6.3 Complexity

In the method proposed in G.711.1, one has to compute the energy of the decoded signal has shown in Eq. (5.14). This operation takes L multiplications and $L-1$ additions. Having that value, one can immediately get the estimated $\hat{\sigma}_q^2$. If the window was considered, one could re-write Eq. (6.16) as:

$$|Q_w(k)|^2 = \sum_{n=0}^{L-1} w^2(n)\hat{\sigma}_q^2. \tag{6.18}$$

The energy of the window could be pre-computed and stored. Therefore, only one extra multiplication is needed to complete the noise PSD computation.

In our method, one needs to compute the noise PSD as shown in Eq. (6.17). The noise variances associated to each segment can all be pre-computed and stored in a table. Thus, the computation of our estimate takes L multiplications and $L-1$ additions. Note that in this case, the window is multiplied with the nose variance associated to each segment sample-to-sample.

Our proposed noise estimation procedure therefore has the same complexity than the method proposed in the standard while being more accurate.

6.4 Shaped noise Estimation

At this high bitrate, the noise shaping is quite efficient. For most cases, the shaped noise spectrum is completely below that of the speech. Thus, in most cases, the noise is masked by the speech. However, for quiet talkers, a problem similar to the one encountered in low bitrate coders occurs. The noise level is high relative to the signal in those cases. Therefore, under these circumstances, the noise shaping only partially works: there are frequency bins where the noise level will still exceed the signal level and will be audible. An example of this situation is shown in Fig. 5.6. A postfilter will help increase the perceptual quality of the speech when such a situation presents itself.

The noise estimation process is a little different than the one proposed in Section 6.1 as the noise is not flat here. With Eq. (5.5), we've seen that the decoded lower-band signal in R2a and R3 modes is equal to the sum of the clean signal and the quantization noise shaped by the perceptual filter. The quantization noise that is shaped is the same one we would have obtained without noise shaping, i.e., the same as the one produced in the legacy G.711 speech coder. Thus, as before, we can assume that the unshaped quantization noise is white. The unshaped quantization noise can therefore be estimated with the improved noise estimator proposed in Section 6.1. Windowing should still be taken into account.

Since the perceptual filter is computed based on the decoded signal, it can estimated at the decoder side just as it is done when a refinement signal is present in G.711.1. The shaped noise is then simply estimated by filtering the estimated quantization noise with the estimated filter as:

$$|\hat{Q}_{\mathrm{SN}}(k)|^2 = |\hat{Q}_w(k)|^2 \left|\frac{1}{1+F(k)}\right|^2. \tag{6.19}$$

6.5 Simulations And Discussion

6.5.1 Estimate Accuracy Tests

We conducted a series of experiments to verify the accuracy of our noise estimator. The setup of the first experiment we ran is as follows. We encoded and decoded a speech signal (8kHz sampling frequency) on a frame-by-frame basis in MATLAB using an A-law quantizer. Each frame had a length of 64 samples. We computed the true error of each block as:

$$\sigma_q^2 = \frac{1}{L}\sum_{n=0}^{L-1}(x(n)-y(n))^2 \tag{6.20}$$

Note that no window is applied in this first experiment. We then implemented three different noise estimates: the noise estimation proposed in the G.711.1 standard. The "incorrect" implementation of the estimator in the reference code (see Appendix A) and the noise estimation we proposed in Section 6.1. Figure 6.1 shows the noise estimates relative to the true noise as a function of the signal energy. Here, a speech file of 24 seconds was encoded and decoded with A-law. The signal was segmented into blocks of 40 samples. For the purpose of plotting, we gathered statistics of the signal energy for each block. These values are assigned to 0.2–dB bins. For each bin of signal values, we calculated the average noise variance. The true noise is shown in bold. The correct noise estimate proposed in G.711.1 is shown with a thin line. The reference code estimate is shown with a line with

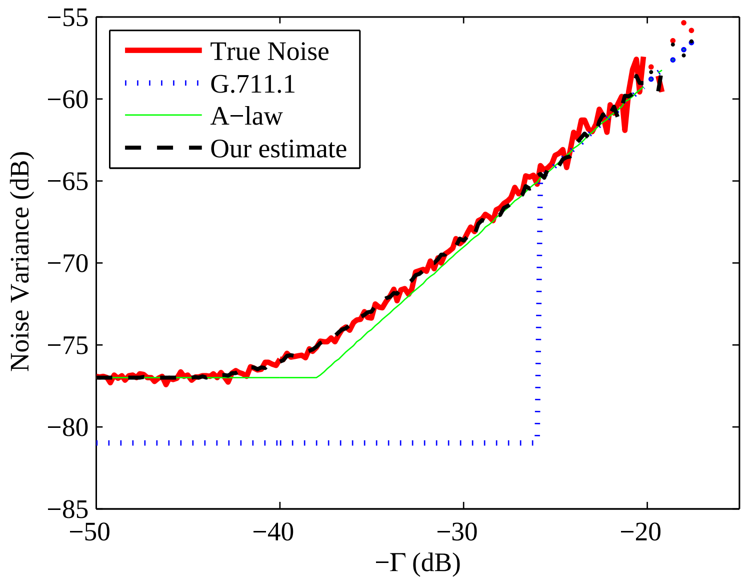


Fig. 6.1 Comparison of the different noise estimation methods when no window is applied. A speech file of 24 seconds was encoded and decoded with A-law. The signal was segmented into blocks of 40 samples. Statistics of the signal energy were gathered for each block and they were assigned to 0.2–dB bins. For each bin of signal values, the average noise variance was computed for each method.

small dots. Finally, the estimate from our proposed method is shown with a line with large dots. We can see that the noise estimate that we get with our method is more accurate than the one proposed by the G.711.1.

In the second experiment we ran, we applied a window as described in Section 6.2. We used the same window that is used in the G.711.1 standard. It is defined as:

$$w(n) = \begin{cases} \frac{1}{2} - \frac{1}{2}\cos(\frac{\pi n}{48}) & n = 0, \dots, 47, \\ \frac{1}{2} + \frac{1}{2}\cos(\frac{\pi(n-47)}{16}) & n = 48, \dots, 63. \end{cases} \tag{6.21}$$

The energy of this window is:

$$\sum_{n=0}^{L-1} w^2(n) \approx 13.6 \quad [\text{dB}]. \tag{6.22}$$

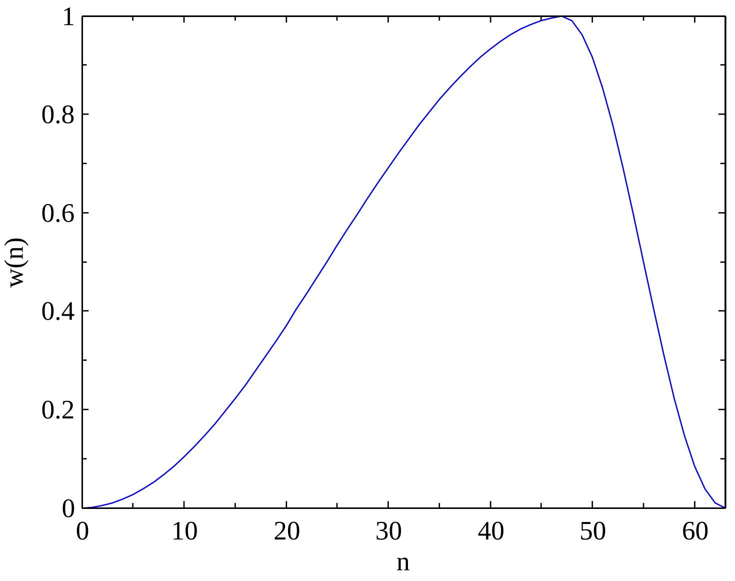


Fig. 6.2 Pre-window to postfilter computation in G.711.1: Window applied to the signal prior to taking computing its frequency response. The window length is 64. It is applied on 64-sample blocks which consists of the most recent 24 samples of the past frame and the current frame (40 samples)

We plotted the window in Fig. 6.2. The results of this second experiment are shown in Fig. 6.3. Similarly as the first experiment, a speech file was encoded and decoded with A-law. The windowed noise has less energy than the unwindowed one. This is expected as the window is tapered. As a matter of fact, from Eq. (6.12) and Eq. (6.16), it is clear that our estimate (with window considered) is:

$$\hat{\sigma}_{wq}^2 = \hat{\sigma}_q^2 \sum_{n=0}^{L-1} w^2(n) \frac{1}{L}, \tag{6.23}$$

which, in dB, is equivalent to:

$$\hat{\sigma}_{wq}^2 = 10 \log \hat{\sigma}_q^2 + 10 \log \sum_{n=0}^{L-1} w^2(n) - 10 \log L \quad \text{[dB]}. \tag{6.24}$$

Here, we took L = 64. Therefore,

$$\hat{\sigma}_{wq}^2 \approx 10 \log \hat{\sigma}_q^2 - 4.45 \quad \text{[dB]}. \tag{6.25}$$

Our estimate coincides well with the true windowed quantization error variance. This experiment confirms that the window should be taken into account. Otherwise, the noise would be overestimated. One small difference to note between the two experiments is the block size. In the first experiment, the block size is 40 samples just as in the proposed postfilter in G.711 when they perform the noise estimation. In the second experiment, the block size is 64 samples. It consists of the last 24 samples of the previous frame and the 40 samples of the current frame. As we explained earlier, the window affects the signal and the noise. Therefore, when the noise is estimated, we consider the entire windowed block rather than just the current frame portion. In Fig. 6.3, we also represented the estimate G.711.1 incorrect estimate. We see that though there estimate does not coincide as well with the true windowed noise, it estimates it better than it does the unwindowed one over part of the load factor range.

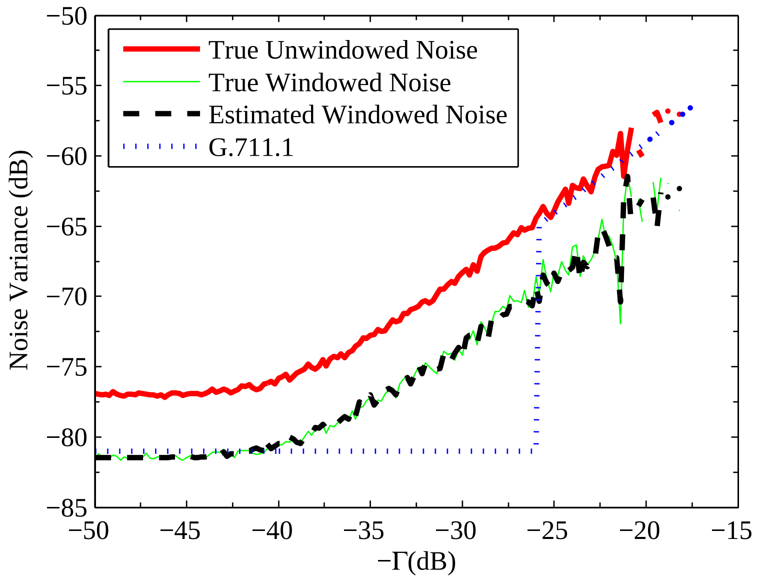


Fig. 6.3 Comparison of the different noise estimation methods when the window is applied. A speech file was encoded and decoded with A-law. The signal was segmented into blocks of 64 samples where the first 24 samples of each block corresponds to the last 24 samples of the previous frame. Statistics of the signal energy were gathered for each block and they were assigned to 0.2–dB bins. For each bin of signal values, the average noise variance was computed for each method.

The first two tests we performed show the accuracy of the noise estimate on average for all the frames in the speech signal. In order to verify the accuracy of the estimates on individual frames, we conducted a third experiment. Here, we encoded a speech signal using A-law. After decoding it, we computed its error signal. We then sectioned the original speech and the decoded speech into blocks of length of 64 samples. We applied the window shown in Fig. 6.2 to each block. After taking the Fast Fourier Transform (FFT) of length 64, we plotted the PSD of the windowed original signal (with a bold line), the PSD of the windowed quantization noise (with a thin line) and the PSD of our windowed noise estimate (with a dotted line) for subsequent blocks in Fig. 6.4. The figure clearly shows that the PSD of the windowed quantization noise is very well estimated by our noise estimator.

In order to test the proposed shaped noise estimate in Section 6.4, we encoded the same speech file with G.711.1 encoder. We modified it so that the frame length would be 160 samples as well. The quantization noise is shaped in this case. Figure 6.5 shows the results of this experiment. We plotted the PSD of the windowed original signal (with a bold line), the PSD of the windowed and shaped quantization noise (with a thin line) and the PSD of our windowed and shaped noise estimate (with a dotted line). We can see that our noise estimator quite accurately estimates the PSD of the windowed and shaped quantization noise.

As we can observe from the different tests, the noise estimation process we proposed in this chapter is very good. In the experiments we discuss below, we replace the noise estimator of the postfilter proposed in the G.711.1 by the one we propose in this Chapter.

6.5.2 G.711.1 Tests

We partially implemented the ITU-T G.711.1 in MATLAB to facilitate our research. The implementation was based on the ITU-T floating-point reference code [21]. We implemented the encoder and decoder for R1 mode. We implemented the G.711.1 reference code postfilter as well in MATLAB. We tested our implementation by using the test files provided with the reference code. The bit file generated by our encoder is identical to the one generated by the reference code. The output generated by our decoder (with or without the postfilter activated) is also the same as the one produced by the reference code.

We also implemented the legacy G.711 encoder in MATLAB. The output of this encoder is a bit file that can be read by the G.711.1 decoder.

In a final set of experiments, we replaced the noise estimator in G.711.1 postfilter with

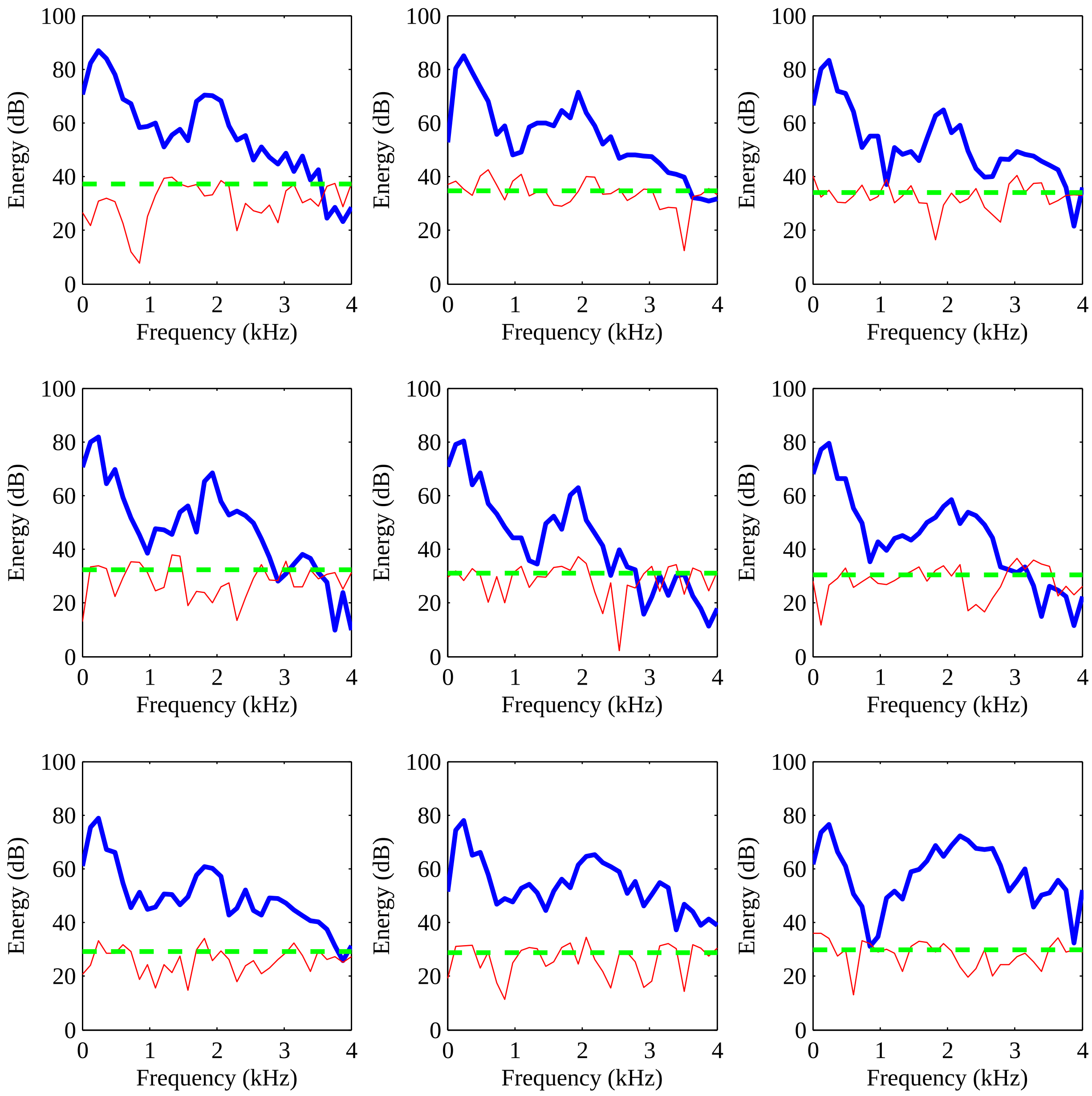


Fig. 6.4 Quantization noise estimation using the improved noise estimator described in Section 6.2. A-law was applied on block of 64 samples with 24 samples overlap. The PSD of the windowed speech spectrum (bold line), the PSD of the windowed quantization noise (thin line) and the PSD of the noise estimate (dashed line) are shown. Each block directly follows the previous one in time starting from the top-left corner to the bottom-right corner reading one row after another from left to right.

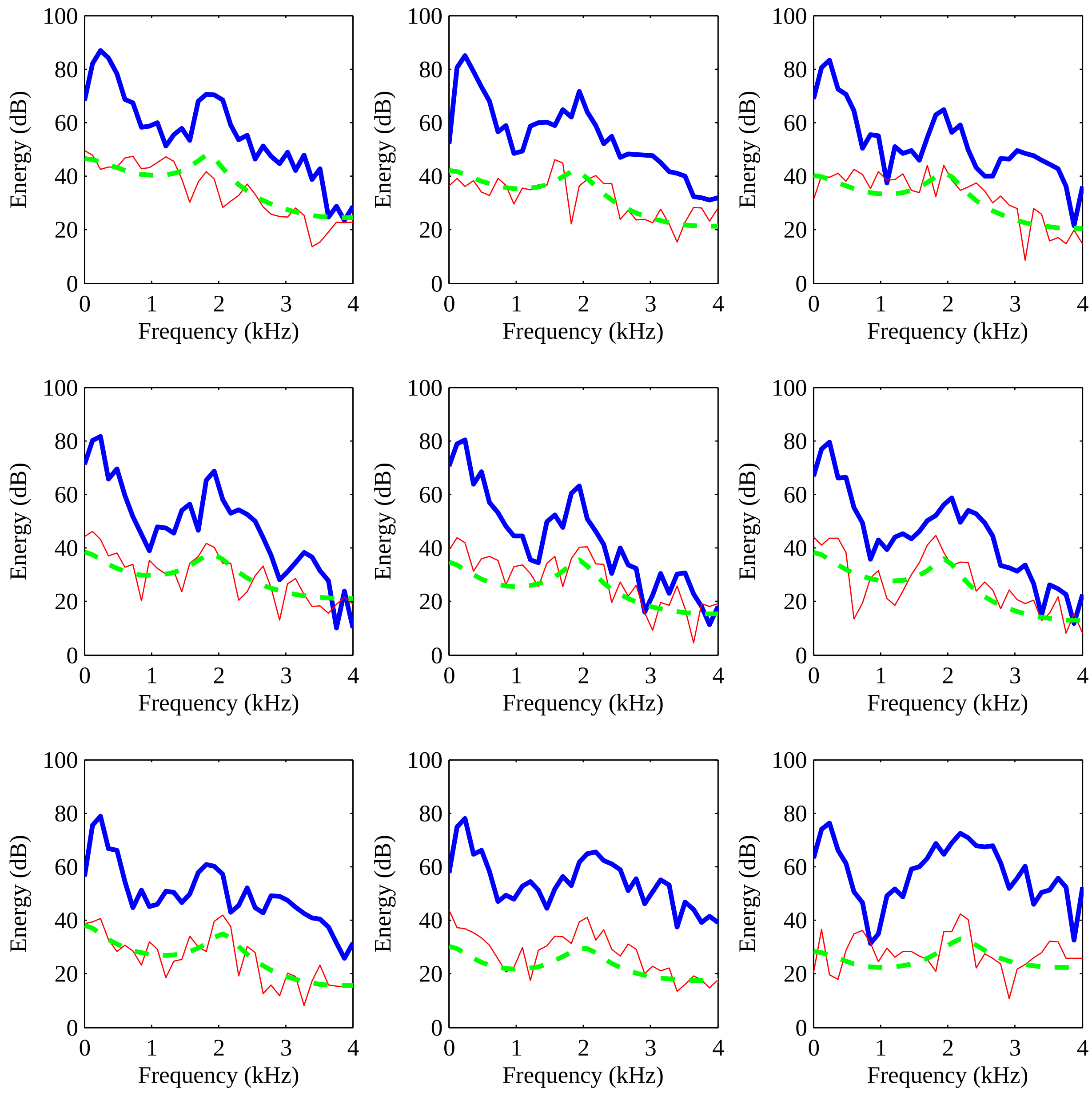


Fig. 6.5 Quantization noise estimation using the shaped-noise estimator described in Section 6.4. A-law was applied on block of 64 samples with 24 samples overlap. The PSD of the windowed speech spectrum (bold line), the PSD of the windowed quantization noise (thin line) and the PSD of the noise estimate (dashed line) are shown. Each block directly follows the previous one in time starting from the top-left corner to the bottom-right corner reading one row after another from left to right.

the different noise estimators we have seen in this chapter.

The test signal we used is the one provided with the G.711.1 reference code. The signal is a 57 seconds long speech file composed of 6 different speakers (3 females and 3 males). The original signal is sampled at 16 kHz. We obtained the narrowband 8 kHz signal after filtering and downsampling the 16 kHz signal with a G.712 [22] filter. The software tool library (STL) of G.191 was used for this purpose [23]. The resulting narrowband signal level is -26 dBov. This signal level was computed using the signal level definition in [23]. Rather then performing a subjective test as it was done in [24] using Absolute Category Rating (ACR), we conducted an objective test. We used the PESQ (Perceptual Evaluation of Speech Quality) methodology [25] to compute PESQ scores. This objective method predicts results that one would obtain based on subjective testing conditions. In subjective tests, the listener is asked to grade the sample from 1 (bad) to 5 (excellent). These score is known has the MOS (Mean Opinion Score). To produce scores that are similar to the MOS, the PESQ produces MOS-LQO (Mean Opinion Score - Listening Quality Objective). The range of the MOS-LQO is 1.0 (worst) to 4.5 (best).

For the first set of tests, the input signal was encoded with the legacy G.711 codec. We then decoded the bitstream using five different methods: the legacy G.711 decoder, the G.711.1 decoder, the G.711.1 decoder with the Reference Code “incorrect” G.711.1 postfilter, the G.711.1 decoder with the Fixed A-law Postfilter postfilter and finally the G.711.1 decoder with our proposed postfilter (taking windowing into account). To test for the case of quiet talkers, we also attenuated the signal by 20 dB. The PESQ results for this test are summarized in Table 6.1.

Table 6.1 PESQ Results with input signal encoded by legacy G.711. The noise is essentially white here.

Decoder Type	No Attenuation	20 dB Attenuation
Legacy	4.34	3.39
G.711.1	4.35	3.45
G.711.1 + incorrect PF	4.43	3.77
G.711.1 + A-law PF	4.44	3.77
G.711.1 + our PF	4.44	3.83

The results from this experiment show that the presence of the postfilter increases the quality of the decoded signal. For the case of the non-attenuated signal, the output signals from using all 3 postfilters have very similar scores. They have an increase of about 0.1 compared to the un-enhanced signal. For the attenuated case, the postfilter benefit measured by PESQ is larger. This is expected as the noise affects the attenuated signal more than it does the non-attenuated one. The increase of the score for the G.711.1 decoded signal and the postfiltered one is about 0.3 when the A-law method is used and about 0.4 when our method is used. We see that the fact that our noise estimate is better increases the performance of the postfilter.

Additionally to the objective PESQ tests, we conducted informal listening tests. These tests confirmed the results obtained through objective testing. We verified that the presence of a postfilter does increase the perceptual quality of the signal. For the non-attenuated cases, no major audible difference was noted when comparing the three different postfilter methods. For the attenuated cases, the "incorrect" postfilter and the A-law postfilter methods produced just about the same signal quality. Our proposed postfilter perceptually ranked best just as predicted by the PESQ test.

In [24], it was stated that the postfilter proposed in the standard to enhance a decoded signal that was encoded by the legacy G.711 can be applied to a signal that was encoded and decoded by G.711.1 in R1 mode because it is robust to noise that is not white. We therefore ran the following tests to compare the performance when we use the postfilters that use a flat noise assumption and a postfilter that uses the shaped noise estimator we proposed in Section 6.4.

The signal was encoded with G.711.1 in R1 mode. To test for the case of quiet talkers, we also attenuated the signal by 20 dB. The different decoded methods were: the G.711.1 decoder, the G.711.1 decoder with the Reference Code "incorrect" postfilter, the G.711.1 decoder with the Fixed A-law postfilter, the G.711.1 decoder with our proposed postfilter with the flat noise estimator (taking windowing into account) and finally the G.711.1 decoder with our proposed postfilter with the shaped noise estimator (taking windowing into account). The PESQ results are summarized in Table 6.2.

The results show once again that the use of the postfilter increases the quality of the signal. The "incorrect" postfilter, the A-law postfilter our our postfilter all assume that the noise is white. Their noise estimate here is not as good as it was for the tests summarized in Table 6.1 because the noise is shaped. This explains why the score does not increase as much as it did for the first set of tests.

A surprising result however is the one obtained when we use the postfilter with our

Table 6.2 PESQ Results with input signal encoded by G.711.1. The noise is shaped here.

Decoder Type	No Attenuation	20 dB Attenuation
G.711.1	4.36	3.42
G.711.1 + incorrect PF	4.37	3.56
G.711.1 + A-law PF	4.37	3.56
G.711.1 + our PF (flat)	4.37	3.56
G.711.1 + our PF (shaped)	4.37	3.56

shaped noise estimator. As we have seen before, this estimator estimate the shaped noise quite well. We expect the postfilter to increase the quality of the signal a little more than the other postfilters but it scores the same as the do. We discuss this further below.

Informal listening tests were also conducted for this test series. For both the non-attenuated and attenuated cases, no major audible difference was noted among the different methods.

6.5.3 Discussion and Summary

As the results show in Table 6.2, it seems that no benefit occurs from the usage of the better noise estimator for shaped noise in the postfilter. We can see however from Table 6.1 that the quality of the signal when encoded with the legacy G.711 encoder, decoded by the G.711.1 decoder and postfiltered with our improved noise estimator is better than the quality of the signal encoded and decoded by G.711.1 and our better shaped noise estimator. One should expect this situation to be reverse since in the second case, not only is the noise perceptually shaped, it is suppressed by the postfilter. The postfilter itself is designed under the assumption that the noise is white. Therefore, we can suspect that better results could be obtained if other sections of the postfilter are changed accounting for the fact that the noise is shaped.

As we have seen in Section 5.3.2, after the Wiener filter is estimated in the frequency domain using the TS-NR method, it is transformed back in time domain where it is truncated and smoothed through windowing. Figure 6.6 shows a speech spectrum, the noise spectrum and the noise estimate using our shaped noise estimator.

Figure 6.7 shows the same speech spectrum, the noise spectrum and the Wiener filter response of the estimated postfilter by the TS-NR method using our shaped noise estimator.

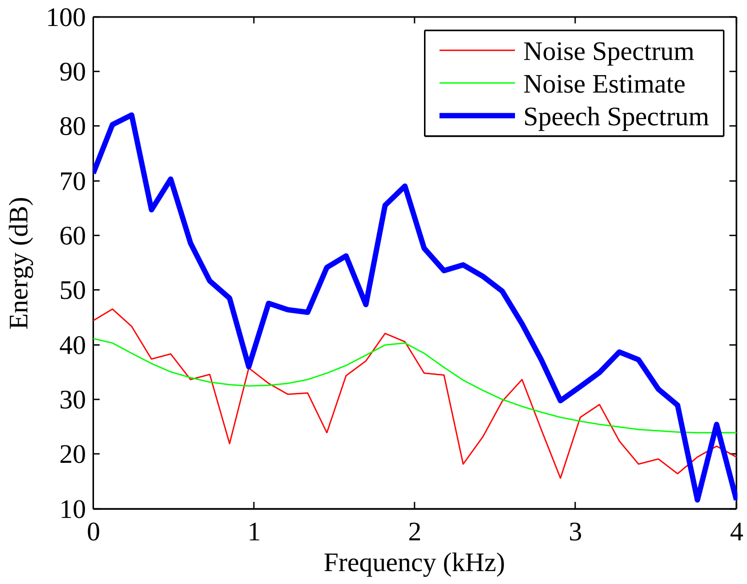


Fig. 6.6 Estimated shaped noise for a speech block example plotted along with the true noise spectrum and the speech spectrum

We see that as we would expect the gain of the filter is approximately 0 dB when the SNR is large and the filter gain falls when the SNR is small (around 3 kHz for example).

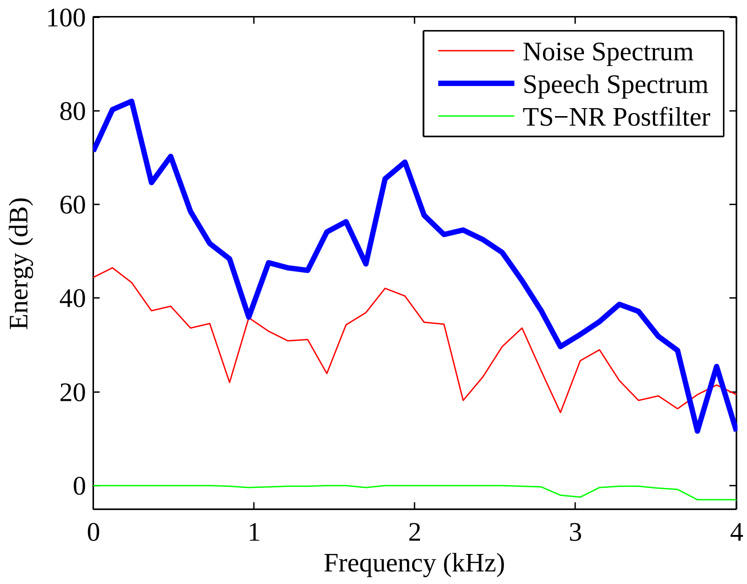


Fig. 6.7 TS-NR postfilter response example with the speech speech spectrum of the signal it applies to and the shaped noise spectrum.

Figure 6.8 shows the TS-NR filter at a larger scale. We can see the dips of the filter more clearly here. The figure also shows the response of the filter after its been taken into time domain, truncated and smoothed. The truncated filter is 33 samples long. To compare it with the TS-NR filter, we padded it with zeros to extend its length to 64 and took the FFT. We see that the general trend of the filter is preserved. However the dips are not as pronounced for the final filter. The consequence is that the filter might end up not being as useful as it should be for some of the frequency regions.

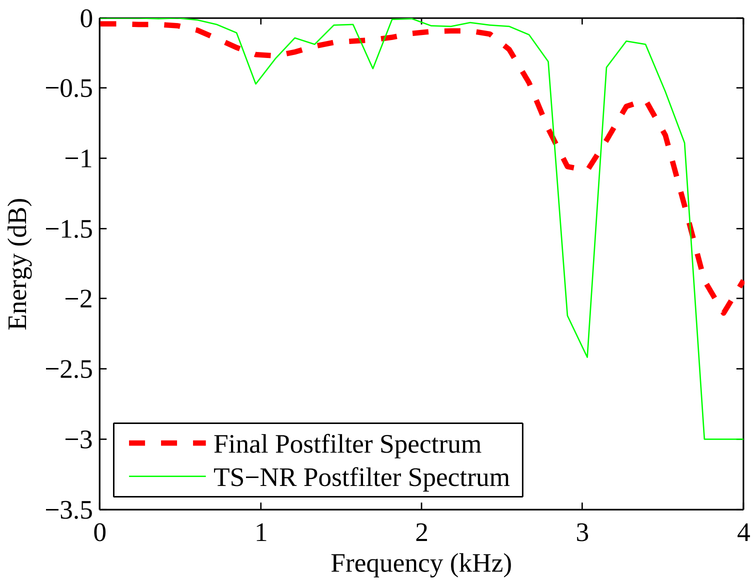


Fig. 6.8 Comparison of the TS-NR generated postfilter and the final generated postfilter responses. After the postfilter is generated frequency domain using the TS-NR method, it is then transformed back to time domain, truncated and smoothed to generate the final filter. Note the change of scale relative to the previous figure.

One other point to note is the fact that the shaping filter only has 4 coefficients. The shaped noise can therefore not follow the speech formants as efficiently as a longer filter would. This is visible for example in Fig. 6.6 where the third formant in the frame is missed. Also, recall from Section 5.2.1 that the shaping filter is computed based on the past 80 samples. Therefore, the filter better matches the two previous blocks of data rather than the current one. This filter is applied to the flat noise estimate of the current block. There is a delay that can occur before the shaped noise really matches the block but that delay is not too noticeable because the frames are only 5 ms each. This could however affect the performance of the postfilter in some cases.

We saw through the discussion of our results that the noise estimators proposed for the flat noise and the shaped noise are quite accurate. The better estimate of the flat noise estimate led to an improved performance of the postfilter. The quality of the enhanced signal when the noise is shaped is not as good as it could be even though a better noise estimator is used. We have seen that subsequent processing to obtain the time domain postfilter are likely to be the reason behind this fact.

Chapter 7

Conclusion and Future Research Direction

Postfilters have been used on decoded signals to enhance their perceptual quality. Though parametric postfilters are typically used for low-bitrate codec, they are not the natural choice for high bit-rate coders since the required parameters are generally not available in these cases.

The high bit-rate coder G.711.1 proposes a non-parametric postfilter. For each decoded frame, the frame energy is computed. Using the energy of the frame and some A-law properties, the quantization noise PSD is estimated. A postfilter is then generated using the TS-NR algorithm. The method used here is one that is typically used to reduce acoustic background noise.

In this thesis, we proposed a more accurate noise estimator. For each decoded frame, the noise energy is estimated on a sample by sample basis by using other A-law properties. We then average the noise energy over the frame. Our estimator also considers the effect of the window that is applied to the signal when it is transformed into its frequency counterpart before the postfilter is estimated. This innovative system has the same complexity as the one proposed in the standard.

The noise estimator in the postfilter proposed in G.711.1 was replaced by our proposed noise estimator for windowed flat noise. Test signals were encoded by the legacy G.711 codec. We then decoded the signals using five different methods:

- the legacy decoder,

- the G.711.1 decoder,
- the G.711.1 decoder with the postfilter proposed in the standard,
- the G.711.1 decoder with the corrected postfilter proposed in the standard,
- the G.711.1 decoder with the postfilter using our proposed noise estimator.

The tests were carried without attenuation and with a 20 dB attenuation to model quiet talkers. Using the PESQ methodology, we carried objective tests to compare the quality of decoded signals. Informal subjective testing was also done. The perceptual quality of the enhanced signal obtained using the postfilter with our noise estimator was better than the one obtained with the original noise estimator.

We proposed an additional noise estimator for shaped noise. This noise is estimated by applying a noise shaping filter to the estimated flat shaped noise.

The noise estimator we proposed for shaped noise was also tested with the G.711.1 postfilter. Although we have verified that the noise is indeed better estimated using this postfilter, the results we obtained seem to indicate that no major difference occur when we use this better estimate: the perceptual quality obtained with our improved noise estimator is the same as the one gotten when the noise is assumed flat. As discussed in Section 6.5.3, parameters other than the noise estimator can affect the quality of the enhanced signal.

As a future research direction, we propose that one studies other parameters that might affect the postfiltering generation for shaped noise:

- Analyze the effect of the truncation and the smoothing that occurs after the filter is estimated with the TS-NR method.
- Explore the effects of the noise shaping filter used at the encoder. One could for example try to use the current frame and a past frame to generate the shaping filter rather than basing it completely on past signals. The usage of a longer filter (more coefficients) can also be investigated.

Appendix A

G.711.1 Noise Estimator

The compression function for the A-law compander is given by:

$$C(x) = \begin{cases} \dfrac{A|x|/x_{\max}}{1+\ln A}\operatorname{sgn} x & 0 \leq \frac{|x|}{x_{\max}} < 1/A, \\ x_{\max}\dfrac{1+\ln A|x|/x_{\max}}{1+\ln A}\operatorname{sgn} x & 1/A \leq \frac{|x|}{x_{\max}} \leq 1. \end{cases} \tag{A.1}$$

As discussed in Chapter 4, the SNR for A-law can be derived as a function of the load factor. The load factor is defined as $\Gamma = \sigma_x/x_{\max}$. This factor shows how well the signal uses its dynamic range. For small signals (uniform portion):

$$\mathrm{SNR}_{\mathrm{unif}} = 3 \cdot 2^{2b}\left(\frac{A}{1+\ln A}\right)^2 \frac{\sigma_x^2}{x_{\max}^2}. \tag{A.2}$$

For large signals (logarithmic portion of the companding function), we get:

$$\mathrm{SNR}_{\mathrm{log}} = 3 \cdot 2^{2b}\left(\frac{1}{1+\ln A}\right)^2. \tag{A.3}$$

As seen in Section 4.4, the value of the load factor where the transition between the logarithmic SNR and the uniform SNR occurs is:

$$\Gamma_{\mathrm{th}} \approx 38.86[\mathrm{dB}]. \tag{A.4}$$

The value of the load factor Γ_{th} at the transition is referred to as the transition threshold.

An unfortunate mistake in [24] has propagated to the G.711.1 [2] standard and to the reference code accompanying the standard [21]. The error is the omission of the square on the bracketed term in Eq. (A.2) i.e. the SNR formula in the uniform portion. Equating this version of the uniform SNR with the logarithmic portion SNR, the standard gets a load factor threshold value of:

$$\Gamma_{\text{th}}^{G.711.1} \approx 26.81[\text{dB}]. \tag{A.5}$$

The consequence of the omission of the square is an erroneous transition threshold value.

An additional error in the standard and the reference code is the presence of a factor of 40 (the frame length) in the computation of the SNR in the uniform portion. This creates a discontinuity of the SNR at the load factor threshold value.

Assume for simplicity that the signal was normalized so that $x_{\max} = 1$. We get $\sigma_x^2 = 2.0845 \cdot 10^{-3}$. Define E_{LB} such that:

$$\sigma_x^2 = \frac{1}{L} E_{\text{LB}}. \tag{A.6}$$

where L is the frame length. For $L = 40$, we get $E_{\text{LB}} = 8.3303 \cdot 10^{-2}$ just as stated in the standard. When $E_{\text{LB}} > 8.3303 \cdot 10^{-2}$, this is equivalent to the situation where $\hat{\Gamma}^2$ is smaller than Γ_{th}^2 and so the noise is estimated as:

$$\hat{\sigma}_q^2 = \frac{\hat{\sigma}_x^2}{3 \cdot 2^{2b} (\frac{1}{1 + \ln A})^2}. \tag{A.7}$$

In terms of E_{LB}, we have:

$$\hat{\sigma}_q^2 = \frac{1}{3 \cdot 2^{2b} \left(\frac{1}{1+\ln A}\right)^2} \frac{1}{L} E_{\text{LB}}, \tag{A.8}$$

$$L\hat{\sigma}_q^2 = \frac{1}{3 \cdot 2^{2b} \left(\frac{1}{1+\ln A}\right)^2} E_{\text{LB}}. \tag{A.9}$$

Let $N = L\hat{\sigma}_q^2$. We can write:

$$N = \frac{1}{3 \cdot 2^{2b} \left(\frac{1}{1+\ln A}\right)^2} \times E_{\text{LB}}. \tag{A.10}$$

At the transition ($E_{\mathrm{LB}} = 8.3303 \times 10^{-2}$) and with $A = 87.56$, we get $N = 1.2725 \times 10^{-5}$. However the standard uses $N_{\mathrm{STD}} = 3.1789 \times 10^{-7}$. We notice that $N_{\mathrm{STD}} = \frac{N}{L}$. This is what causes the discontinuity that we observed at the transition point in Fig. A.1.

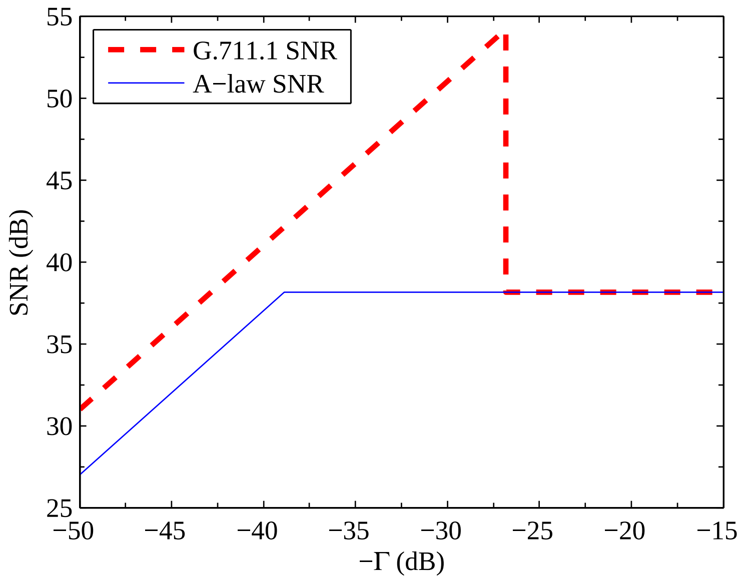


Fig. A.1 Comparison of G.711.1 SNR a the correct A-law SNR: the SNR in the standard specifications with a bold and dotted line and the correct version of the A-law SNR with a thin line. We observe the erroneous transition value and the discontinuity at the transition value for the G.711.1 SNR

Appendix B

Bit Allocation Algorithm For Refinement Signal in G.711.1

In G.711.1, the lowerband enhancement layer (Layer 1) is encoded at a bit-rate of 16 kbit/s. Since each frame is 5 ms (40 samples), this effectively means a bit budget of an average of 2 bits per sample (80 bits per frame) available to code the refinement signal. However, the refinement signal has up to 3-bit resolution per sample. For all samples to be fully coded in Layer 1, we would therefore need 120 bits per frame. Therefore G.711.1 uses an adaptive multiplexing in order to reduce the number of bits. This adaptive multiplexing dynamically allocates more bits to samples with larger exponent values as these samples have more quantization errors. As a result, the number of bits allocated to encode the refinement signal per sample varies from 0 to 3. The bit-allocation table generation algorithm is described in this Appendix.

B.1 Signal Exponent Map

Each sample in a frame has an exponent value which specifies the segment it belongs in the companding law. The exponent value is the factor that the the bit-allocation algorithms uses to determine the number of bits it will allocate to a sample to code its refinement signal. The higher the exponent value, the more bits a sample will receive from the algorithm.

The first step of the algorithm consists of an expansion of the exponent signal into an exponent map $M_{\text{exp}}(j,n)$ for a frame where $j, = 0, \cdots, 9$ and $n = 0, \cdots, 39$. $M_{\text{exp}}(j,n)$ is a matrix which stores the indices of samples that use a specific exponent index j. As the mapping is done for each sample in the frame, a counting vector is generated as well. This counting vector $N_{\text{exp}}(j)$ is of length 10 with $j, = 0, \cdots, 9$. It counts the number of samples that have the same exponent index j. The algorithm for the exponent mapping is summarized below:

Initialize all the elements in both the exponent map matrix and the counting vector to 0. Then, for each sample, perform 3 iterations $i = 0, 1, 2$:

1. Compute the exponent index j as: $j = x_{\text{LBexp}}(n) + i$
2. Update the exponent map matrix as: $M_{\text{exp}}(j, n) = n$
3. Update the counting vector as: $N_{\text{exp}}(j) = N_{\text{exp}}(j) + 1$

As we can see, each sample is associated to 3 exponent indices. These indices are adjacent to each other. Therefore, a particular sample index always appears 3 times in the exponent map matrix and 3 subsequent values are incremented in the counting vector for each sample.

B.2 Bit Allocation Table Generation

Once M_{exp} and N_{exp} have been filled, the bit allocation table is $B_A(n)$, where $n = 0, \cdots, 39$, is ready to be generated. $B_A(n)$ is really a vector of length 40. It stores the number of bits allocated (from 0 to 3)to each sample for the reference code. For a given frame, the bit allocation table is computed as follows:

1. Initialization:
   (a) Set all the elements in the bit allocation table $B_A(n)$ to 0
   (b) Set the remaining bit budget as $b^{[0]} = 80$
   (c) Set the exponent index $j = 9$
   (d) Set the iteration number $i = 0$
2. Get the number of bits that is going to be allocated to all samples with index j: $q = \min(b^{[i]}, N_{\text{exp}}(j))$

3. For each sample with exponent index j, increment the number of bits allocated: $B_A(n) = B_A(n) + 1$ for $n = M_{\text{exp}}(j, k)$, $k = 0, \cdots, q - 1$
4. Update the remaining bit budget as: $b^{[i+1]} = b^{[i]} - q$
5. Check if bit budget is empty, i.e. if $b^{[i+1]} = 0$
   (a) If bit budget is empty, then the algorithm is done
   (b) If the bit budget is not empty, decrement exponent index as $j = j - 1$, increment iteration as $i = i + 1$ and then go to Step 2

The bit allocation table computation clarifies what the exponent index is and how it is used to allocate bits to samples. A sample that is in segment 7, will have an exponent value of $x_{\text{LBexp}}(n) = 7$. The signal mapping will associate that sample to three exponent indices $j = 7, 8, 9$. For each of these exponent indices and as long as bits are available, the sample will receive 1 bit. Therefore, as long as bits are available, this sample will receive 3 bits for the refinement code.

A sample that is in segment 0 , will have an exponent value of $x_{\text{LBexp}}(n) = 0$. The signal mapping will associate that sample to three exponent indices $j = 0, 1, 2$. Just as in the previous case, for each of these exponent indices and as long as bits are available, the sample will receive 1 bit. So again, this sample can receive up to 3 bits to code its refinement signal. However, for low j's, it is likely that the remaining bit budget is very low or already empty. Therefore, the reference value of the sample in the first example is more likely to be coded with 3 bits than the one in the second example.

References

[1] P. Scalart and L. JV CNET Filho, "Speech Enhancement Based on A Priori Signal to Noise Estimation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP)*, vol. 2, pp. 629–632, May 1996.

[2] ITU-T, *Wideband Embedded Extension to G.711 Pulse Code Modulation.* Recommendation G.711.1, Mar. 2008.

[3] ITU-T, *Pulse Code Modulation (PCM) of Voice Frequencies.* Recommendation G.711, 1972.

[4] Y. Ephraim and D. Malah, "Speech Enhancement Using a Minimum-Mean Square Error Short-Time Spectral Amplitude Estimator," *IEEE Trans. Acoustics Speech, Signal Process.*, vol. 32, no. 6, pp. 1109–1121, Dec. 1984.

[5] O. Cappé, "Elimination of the Musical Noise Phenomenon with the Ephraim and Malah Noise Suppressor," *IEEE Trans. Speech, Audio Process.*, vol. 2, no. 2, pp. 345–349, Apr. 1994.

[6] C. Plapous, C. Marro, and P. Scalart, "Improved Signal-to-Noise Ratio Estimation for Speech Enhancement," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 14, no. 6, pp. 2098–2108, Nov. 2006.

[7] C. Plapous, C. Marro, L. Mauuary, P. Scalart, R. France Telecom, and F. Lannion, "A Two-Step Noise Reduction Technique," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP)*, vol. 1, pp. 289–92, Aug. 2004.

[8] J. Smith and J. Allen, "Variable Bandwidth Adaptive Delta Modulation," *Bell Syst. Tech. J.*, vol. 60, no. 5, pp. 719–737, May-June 1981.

[9] V. Ramamoorthy and N. Jayant, "Enhancement of ADPCM Speech by Adaptive Post-Filtering," *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 8, pp. 1465–1475, Oct. 1984.

[10] Y. Yatsuzuka, S. Iizuka, and T. Yamazaki, “A Variable Rate Coding by APC with Maximum Likelihood Quantization from 4.8 kbit/s to 16 kbit/s,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP)*, vol. 11, pp. 3071–3074, Apr. 1986.

[11] J. Chen, *Low-Bit-Rate Predicitve Coding of Speech Waveforms Based on Vector Quantization.* Ph.D. Thesis, University of California at Santa Barbara, Winter 1987.

[12] J. Chen and A. Gersho, *Vector Adaptive Predictive Coder for Speech and Audio.* US Patent 4,969,192, Nov. 1990.

[13] J. Chen and A. Gersho, “Adaptive Postfiltering For Quality Enhancement of Coded Speech,” *IEEE Trans. Speech, Audio Process.*, vol. 3, no. 1, pp. 59–71, Jan. 1995.

[14] D. O’shaughnessy, *Speech Communications: Human and Machine.* Universities press., 1999.

[15] W. Chu, *Speech Coding Algorithms: Foundation and Evolution of Standardized Coders.* Wiley-Interscience, 2003.

[16] ITU-T, *Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbit/s.* Recommendation G.723.1, May 2006.

[17] V. Grancharov, J. Samuelsson, and W. Kleijn, “Noise-Dependent Postfiltering,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP)*, vol. 1, pp. 457–460, Aug. 2004.

[18] ITU-T, *Wideband Coding of Speech at Around 16 kbits/s Using Adaptive Multi-Rate Wideband (AMR-WB).* Recommendation G.722.2, Jan. 2002.

[19] N. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice-Hall, 1990.

[20] B. Atal and M. Schroeder, “Predictive coding of speech signals and subjective error criteria,” *IEEE Trans. Acoustics Speech, Signal Process.*, vol. 27, no. 3, pp. 247 – 254, June 1979.

[21] ITU-T, *Wideband Embedded Extension to G.711 Pulse Code Modulation: New Annex A on a Reference Floating-Point Implementation for G.711.1 and Editorial Corrections to the Main Body Text.* Recommendation G.711.1 Annex A, Nov. 2008.

[22] ITU-T, *Transmission Performance Characteristics of PCM Channels.* Recommendation G.712, Nov. 2001.

[23] ITU-T, *ITU-T Software Tool Library 2000 User’s Manual.* Recommendation G.191, December 2001.

[24] J. Garcia, C. Marro, and B. Kosvesi, "A PCM Coding Noise Reduction for ITU-T G. 711.1," pp. 57–60, Sept. 2008.

[25] ITU-T, *Perceptual Evaluation of Speech Quality.* Recommendation P.862, Feb. 2001.