Robust Decoding of Speech Line Spectral Frequencies over Packet Networks

by

Paul Rondeau

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

Master of Science

Department of Electrical and Computer Engineering
Faculty of Engineering
University of Manitoba
Winnipeg

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

Robust Decoding of Speech Line Spectral Frequencies over Packet Networks

BY

Paul Rondeau

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Master of Science

Abstract

The problem of transmitting line spectral frequencies (LSF) generated by the Federal Standard 1016 CELP speech encoder over a packet-loss network is considered. Multiple description (MD) coding techniques are investigated, which reduce the spectral distortion (SD) in speech due to packet losses while using the same transmission rate as the standard CELP encoder. We focus on exploiting the residual redundancy of the encoder output to estimate lost packets at the receiver, by using hidden Markov modeling of the encoder output and estimation based on the forward-backward algorithm. In particular, the problem of optimizing index assignments for Markov decoders is addressed. Experimental results are presented which compare the proposed techniques with other known techniques, such as linear estimation and Gaussian mixture modeling. It is demonstrated that the proposed Markov technique averages 2.18 dB of SD when one description is lost, compared to interpolation of odd-even split LSFs at 2.99 dB.

Acknowledgements

I thank my advisor, Dr. Pradeepa Yahampath, for his guidance in my work.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Description	Definition
ADPCM	Adaptive Differential Pulse Code Modulation	page 10
CELP	Code-Excited Linear Prediction	page 12
DPCM	Differential Pulse Code Modulation	page 8
FS-1016	Federal Standard 1016 CELP Encoder	page 12
GMM	Gaussian Mixture Model	page 61
IA	Index Assignment	page 43
LP	Linear Prediction	page 10
LSF	Line Spectral Frequency	page 15
MD	Multiple Description	page 33
MDIA	Multiple Description Index Assignment	page 42
PCM	Pulse Code Modulation	page 8
SD	Spectral Distortion	page 96
VLP	Vector Linear Prediction	page 53

Chapter 1

Introduction

Speech communication networks, such as the public telephone network, are ubiquitous. For example, in 2005 there were over 2.1 billion mobile phone subscribers worldwide [1]. Most such networks, such as the public switched telephone network (PSTN) [2] and mobile phone networks [3], use digital communications. Before speech can be transmitted digitally, it must first be converted into a digital representation. This process begins by sampling the speech signal, which records the signal's value at discrete times. Next, a speech encoder is used to form a digital representation of the samples. The digital representation is transmitted to a decoder, which reconstructs an approximation of the original speech signal. This process is illustrated in Figure 1.1.

Many techniques for encoding speech have been developed [4]. The

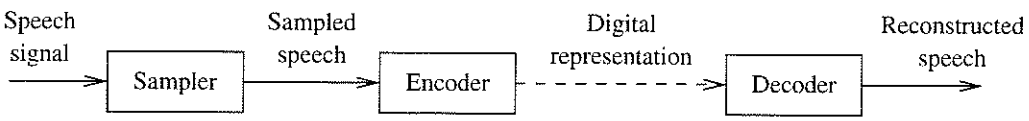

Figure 1.1: Block diagram of a digital speech coding system.

speech encoder used in the PSTN samples a speech signal 8000 times per second (8 kHz), and encodes each sample independently using 8 bits [2]. Thus the encoder transmits at a bit rate of 64 kbps (kilobits per second). Other speech encoders, which improve the bit rate or the speech quality, have been developed. Decreasing the bit rate is important, because it allows for more efficient use of limited communication resources. For example, adaptive differential PCM can encode speech at a bit rate of 32 kbps or less, by accounting for the time varying statistics of the speech signal. Hybrid encoders such as code excited linear prediction (CELP) [5] can lower the bit rate further, by using a parametric model of human speech production, and by considering a model of human perception when encoding the speech signal.

Voice over Internet protocol (VoIP) [6, 7] takes advantage of the Internet for speech communication. The Internet is a packet-based network. In a packet-based network, long messages are divided into smaller packets of data, which are transmitted separately. Such networks can exhibit different behaviour than a dedicated channel. For example, a packet may be lost or delayed before it can reach its destination. If a packet is lost, then the receiver will be missing the corresponding part of the sender's message. One approach for dealing with a lost packet is to ask the sender to retransmit the packet, which increases the communication delay (see the transmission control protocol (TCP) in, for example, [8]). The delay between two people having a conversation is an important factor in the usability of a speech system [7]. To help limit the delay, retransmission is not typically used in VoIP. Instead, packet loss concealment techniques have been developed which help to conceal the effect of such losses [9].

One method for reducing the impact of lost packets is multiple description (MD) coding [10]. In most communication systems, the encoder generates a single description of an input, which is sent to the decoder. In a MD coding system, the encoder generates more than one description for a single input. The descriptions should be transmitted so that when one description is lost, it is still possible to receive other descriptions. For example, the descriptions could be transmitted at different times, over different channels, or over different routes in a network. In addition, the descriptions should be designed so that the decoder can produce an acceptable reconstruction of the source using a subset of the descriptions. An example of such a technique is to transmit the odd and even samples in separate packets. If the packet holding the odd-numbered samples is lost, then it can be estimated from the even-numbered samples in the other packet, and vice versa [11].

In this thesis, our objective is to improve the quality of speech transmitted over a packet network when packet loss occurs. We restrict our investigation to the robust transmission of one speech coding parameter, line spectral frequencies (LSFs), as output by the Federal Standard 1016 CELP speech coder [12]. To this end, we encode the LSFs using multiple descriptions, which are designed to have the same bit rate as the LSF quantizers used by the standard CELP encoder. In addition, the system is designed so that it does not affect the speech coder's quality when no packet loss occurs. We examine the use of a MD index assignment (IA) [37] for quantized LSF vectors, as an alternative to odd-even splitting. It is known that there is dependence between the LSFs output by the FS-1016 encoder, and that this dependence can be exploited to im-

prove decoder performance [13]. In this thesis, we examine several MD encoders and decoders which exploit such dependence. In particular, we examine the use of a decoder based on hidden Markov models to decode descriptions generated using MDIAs, as an alternative to decoders based on estimating missing LSFs.

Contribution and Organization of the Thesis

This thesis examines the use of Markov model-based decoding for MD index assignment-encoded LSFs, which the author has not seen previously. In addition, we developed techniques for optimizing IA matrices and allocating bits between LSF descriptions for use with such decoders. These proposed techniques are experimentally compared with other known techniques based on linear estimation, Markov, and Gaussian mixture models, which are used to estimate missing LSFs.

This thesis is organized as follows. Chapter 2 describes some methods for speech coding, as well as the calculation and properties of LSFs. Chapter 3 describes some approaches used for MD encoding, and methods used previously to estimate missing LSFs. Chapter 4 describes a Markov-model based approach for decoding MDs, and proposes techniques for designing descriptions for use with such decoders. The performance of these systems is considered in terms of average distortion in Chapter 5. Finally, Chapter 6 presents the conclusions and describes related future work.

Chapter 2

Speech Coding

2.1 Introduction

This chapter briefly describes different speech encoding techniques. The chapter begins with procedures common to all of the speech encoders, such as sampling, filtering, and quantization. A description of these procedures may be found in [14]. This is followed by a description of three types of speech encoders. Waveform coders [2] attempt to encode the digitized speech waveform directly. Next is a parametric vocoder, which does not attempt to match the speech waveform. Instead, the coder uses a parametric model of human speech production, and speech is encoded by the model parameters. Finally, code-excited linear prediction (CELP) [5] is a hybrid technique, which uses a parametric model and also attempts to match the speech waveform. There are many resources which describe these and other speech coding techniques in greater detail, such as [15]. [16] describes several standard speech coders in detail. Papers which review speech coders include [4] and [7]. This chapter ends with a

description of the calculation and properties of line spectral frequencies (LSF), which are used to represent linear filter coefficients in some speech encoders. The robust transmission of LSFs is the focus of this thesis.

2.1.1 Sampling

Consider a speech signal $s(t)$, where the time t is a continuous variable. $s(t)$ may be captured using a microphone, for example. *Sampling* converts the continuous function $s(t)$ into a discrete function, which is required in order to process the signal digitally. We consider only periodic sampling, whereby the signal's value is recorded every T_s seconds. Here T_s is called the *sampling period*. The sampling process forms

$$s[n] = s(nT_s) \tag{2.1}$$

where n is the sample number, which takes integer values: $n \in \mathbb{Z}$. The *sampling frequency*, F_s, is the number of times per second that the signal is sampled. T_s and F_s are related by the equation

$$F_s = \frac{1}{T_s} \tag{2.2}$$

The sampling frequency is measured in Hertz (abbreviated as Hz), which is defined as one cycle per second. Typical values of F_s for speech are $F_s = 8$ kHz for narrowband coding over a telephone network, or $F_s = 16$ kHz for higher-quality wideband speech [16].

2.1.2 Aliasing

Two signals which are different in continuous time t can be indistinguishable after sampling into discrete time n. This phenomenon is called *aliasing*.

Consider a system with sampling frequency F_s, and assume that the input to this system is two sinusoids, with frequencies f_1 and f_2. If

$$|f_1 - nF_s| = |f_2 - mF_s| \tag{2.3}$$

for any $n, m \in \mathbb{Z}$, then the sinusoids will be indistinguishable after sampling. To avoid aliasing, we must ensure that

$$F_s > 2 \cdot F_{\text{max}} \tag{2.4}$$

where F_{max} is the highest frequency component in the input signal $s(t)$. The frequency components of a signal can be determined using the Fourier transform, which transforms a signal from the time domain – a function of time t, to the frequency domain – a function of frequency f:

$$S(f) = \int_{-\infty}^{\infty} s(t) e^{-j2\pi ft} dt \tag{2.5}$$

where $j = \sqrt{-1}$.

2.1.3 Filtering

The system has no control over the input signal, so the frequency components of $s(t)$ are unknown in advance. To avoid the aliasing problem, the input signal is generally pre-filtered using a low-pass filter whose cut-off frequency is less than $F_s/2$. For example, the FS-1016 speech encoder considered in this thesis uses $F_s = 8$ kHz, and hence a low-pass filter with a -3 dB cutoff at 3800 Hz [12].

2.2 Types of Speech Coders

2.2.1 Waveform Coders

The first type of speech coders considered here, waveform coders, attempt to follow the sampled input waveform exactly. These encoders include pulse code modulation (PCM) and differential PCM (DPCM). Waveform coding is described in detail in [2].

Pulse Code Modulation

A description of PCM, as well as sampling, quantization, and reconstruction, is given in [14]. A PCM encoder uses a codebook $\mathcal{C}$ of values to approximate, or quantize, a sample. The codebook has $M = |\mathcal{C}|$ entries, identified as $\mathcal{C}[1], \mathcal{C}[2], \ldots, \mathcal{C}[M]$. The numbers $1, 2, \ldots, M$ are codebook indices associated with the codebook entries. A distortion function $d(s[n], \mathcal{C}[i])$ is used to measure the distortion between the original sampled value $s[n]$ and a codebook entry $\mathcal{C}[i]$. Many PCM coders use the squared error as the distortion function:

$$d(a, b) = (a - b)^2 \tag{2.6}$$

For each waveform sample $s[n]$, the quantizer finds the codebook entry $\mathcal{C}[i]$ which gives the lowest distortion:

$$d(s[n], \mathcal{C}[i]) \leq d(s[n], \mathcal{C}[j]), \text{ for all } j \tag{2.7}$$

The codebook index i is transmitted to a decoder, which has a copy of the encoder's codebook. To estimate the original sample's value, $s[n]$, the decoder outputs $\hat{x}[n]$:

$$\hat{x}[n] = \mathcal{C}[i] \tag{2.8}$$

The set of points which are quantized to $C[i]$ form the ith *quantizer cell.* The value of the codebook entry $C[i]$ is the ith *reproduction level.*

The simplest codebook design is a uniform quantizer. In a uniform quantizer, the quantizer cell boundaries are equally spaced, and the reproduction level of each cell is the cell's midpoint [17]. Consider a uniform quantizer's codebook with $M = |C|$ entries, designed for an input signal with a range $x_{\min} \leq x \leq x_{\max}$. Such a quantizer has M codebook entries evenly spaced within the range of the input:

$$C[i] = \left(i - \frac{1}{2}\right) \frac{x_{\max} - x_{\min}}{M} + x_{\min} \tag{2.9}$$

for $i = 1 \ldots M$.

The uniform quantizer is optimal when using the squared error measure of (2.6) with a uniformly distributed signal, that is, when all waveform amplitudes between $x_{\min}$ and $x_{\max}$ are equally likely. However, in speech signals small amplitudes are more common than large amplitudes [18], so the probability distribution of speech signals is not uniform. In addition, if the quantizer cells are uniform, the quantizer error will be relatively greater for small signals than for large signals. To improve the perceived quality, telephone systems use logarithmic quantizers, which have small quantizer cells for small amplitudes, and larger quantizer cells for larger amplitudes [2].

An approach for designing a quantizer for a signal with an arbitrary probability distribution and distortion measure is the Lloyd algorithm [17]. This algorithm designs a codebook iteratively using training data, by alternating between quantizing the training data and finding the optimal codebook entry for the quantized data. More information on this

algorithm and other topics in quantization may be found in [17].

Differential PCM

In a speech signal, there is usually correlation between adjacent samples. Because of this correlation, it is possible to estimate future values of the signal by using its past values. Let $P()$ be a prediction function which performs this estimation. We define an error signal $e[n]$, which is the difference between the predictor's output and the true value of the signal:

$$e[n] = x[n] - P(x[1], x[2], \ldots, x[n-1]) \tag{2.10}$$

A commonly used form of $P()$ is a linear predictor, which estimates a future sample using a linear combination of L past values of the signal:

$$\tilde{x}[n] = \sum_{i=1}^{L} a_i x[n-i] \tag{2.11}$$

a_i is a vector of prediction coefficients, which weigh the past values of the input signal to form the prediction. A method for designing a may be found in §2.3.1, and [19] is an extensive review of linear prediction (LP) techniques.

If the predictor is good, then the variance of the error signal e will be lower than the variance of the original signal. The error signal can then be quantized with less distortion than the original signal at the same bit rate. The decoder uses a predictor on its past values to estimate the future values as well, and adds the received error signal to its prediction, reversing the encoder's process [17].

Adaptive differential PCM (ADPCM) is an improvement compared to DPCM. This type of encoder varies the predictor, the quantizer, or both, to better suit the input signal's time varying characteristics [2].

Figure 2.1: Excited LP model of speech production.

2.2.2 Linear Prediction Vocoder

The remainder of this chapter describes speech coders which model the human speech production in two parts. A time-varying linear filter is used to model the frequency spectrum shaping due to the speaker's vocal tract. The input to this filter is referred to as an excitation signal, which models the effect of the speaker's breath and vocal chords (see for example [4]). Such a system is illustrated in Figure 2.1. The robust encoding and decoding of these filter parameters for transmission over channels with packet loss is the focus of this thesis. A procedure for calculating and encoding these parameters is described in Chapter 2.3.

An LP-based vocoder (see, for example, [20]) uses a model for human speech production. Rather than attempt to describe the original waveform exactly, the system determines model parameters corresponding to the waveform. By transmitting only model parameters, the vocoder encodes speech at a lower bit rate than PCM, but sounds less natural and can be less intelligible [21]. A vocoder assumes that the speech signal has uniform statistics over small intervals of time. The speech signal is divided into contiguous blocks of samples, called *frames*, corresponding to a small interval of time. The encoder determines the model parameters for each such frame, assuming that the entire waveform in a frame can be reasonably modeled by the same set of parameters. Each frame is classified as either *voiced* or *unvoiced*. Voiced speech is driven by a

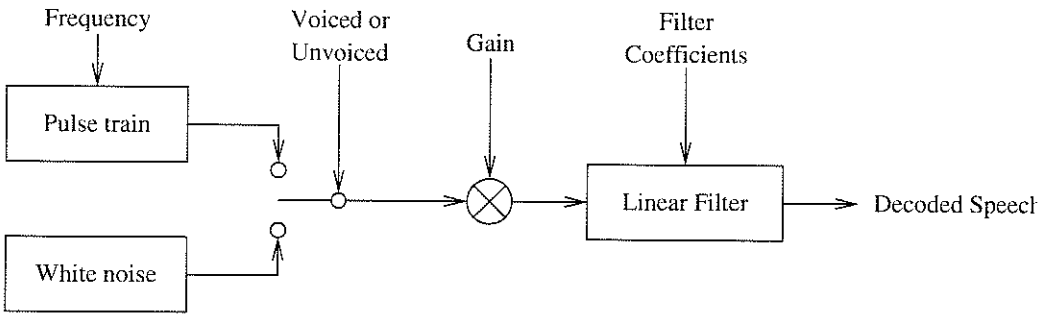

Figure 2.2: System diagram of a LP-based vocoder.

speaker's vocal chords at some pitch, which is estimated by the encoder. The voice's pitch determines the frequency of a pulse train which is used to simulate the vibration of the vocal chords. Unvoiced speech is driven by the speaker's breath alone, and is modeled by a random signal. A gain parameter is used to model the amplitude of the signal. This model is illustrated in Figure 2.2. The vocoder described in [20] operates at 2.4 kbps.

An LP vocoder uses an incomplete model of human speech production. For example, some speech sounds such as transitions between voiced and unvoiced segments cannot be strictly classified as voiced or unvoiced [4]. The encoder described in the next section does not attempt to perform such classification.

2.2.3 Code-Excited Linear Prediction

Code-excited linear prediction (CELP) [5], like the LP vocoder, uses a linear filter to model the vocal tract, which is driven by an excitation signal. Unlike the LP vocoder, CELP uses an excitation signal selected from a codebook, and attempts to match the input waveform. The system considered in this thesis is the Federal Standard 1016 (FS-1016) CELP [12]

speech coder, although other CELP systems operate similarly. This encoder operates at 4.8 kbps. What follows is a brief description of the operation of a CELP encoder. For more detail, see for example [5], [12], and [16].

FS-1016 samples the speech signal at $F_s = 8$ kHz. The sampled signal is divided into frames of 240 samples, and each frame is divided into four subframes of 60 samples. For each frame, the encoder determines filter coefficients which approximate the envelope of the voice's frequency spectrum for this frame. This filter is called the *short term predictor*. The calculation and encoding of the filter coefficients is described in more detail in §2.3.

The excitation signal in FS-1016 is derived from an *adaptive codebook* and a *stochastic codebook* [12]. The excitation signal is determined for each subframe, and is the same length as the subframe, or 60 samples. The adaptive codebook is used to model the periodicity of the excitation signal. It is used to find a sequence of 60 past samples of the excitation signal which best matches the current subframe. The codebook entries are measured in terms of the lag from the start of the current subframe, to the start of a sequence of 60 past values. Lag in fractions of a sample are possible through interpolation. The stochastic codebook is used to model the remainder of the excitation waveform. This codebook has 512 entries, which are derived from ternary-quantized Gaussian-distributed random numbers. In addition to the codebook entries, the encoder also determines and transmits a gain value for the adaptive and stochastic excitation signals. A system diagram of a CELP decoder is presented in Figure 2.3.

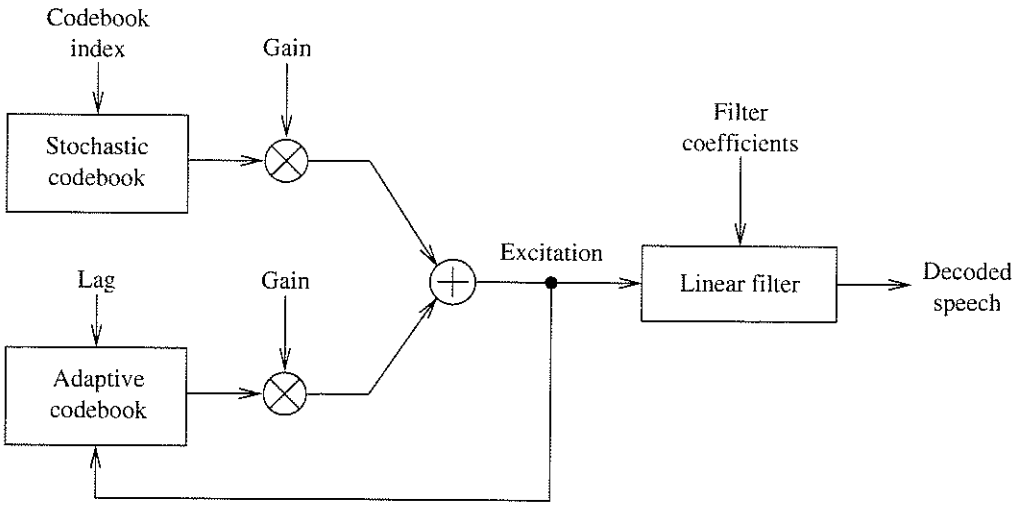

Figure 2.3: System diagram of a CELP-based decoder.

The search for the adaptive and stochastic codebook entries uses *analysis by synthesis*. The process begins by searching for an adaptive codebook entry. The encoder passes prospective codewords through the short term filter, which synthesizes waveforms similar to how a decoder operates – this is the synthesis operation. For each codeword, the encoder evaluates the error between the original quantized speech signal and the synthesized waveform, and the codeword which produces the lowest error is chosen. The search for a stochastic codebook entry is similar. Now the excitation signal is the sum of the selected adaptive codebook entry and the prospective stochastic codebook entry. Again, the encoder searches for the codeword which minimizes the error between the synthesized signal and the original signal.

The encoder chooses an excitation signal based on an error measure. The error calculation begins with finding the arithmetic difference between the original signal and the synthesized signal. Rather than using the difference directly, the encoder passes the difference signal through a

perceptual weighing filter, which takes into consideration the perceptual importance of waveform components. In FS-1016, this perceptual filter is derived from the short term predictor, and is intended to shape the quantization noise. Frequencies with greater energy in the spectral envelope are given relatively less weight by the perceptual filter than frequencies with less energy. Thus greater error is allowed in high-energy parts of the spectrum, where the error can be masked by the signal energy [16].

2.3 Line Spectral Frequencies

The previous section described the use of a linear filter to model the effect of the speaker's vocal tract. Line spectral frequencies (LSFs) are used in several speech encoders, such as FS-1016 [12] and GSM-AMR [22], to represent the coefficients of such linear filters. In this chapter we examine the calculation of LSFs, some of their properties, and how they are quantized in the FS-1016 voice coder. Finally we consider the dependence between quantized LSFs.

2.3.1 Preparing the Input Signal

The FS-1016 encoder begins with a speech signal, which is appropriately low-pass filtered, and sampled at 8 kHz. The standard states that the filter must have a 3 dB attenuation at 3.8 kHz or higher, and at least 18 dB attenuation at 4 kHz. The encoder operates on non-overlapping frames of 30 ms, or 240 samples. A 30 ms Hamming window [23] is applied to each frame. An example of a speech waveform before and after applying a Hamming window is presented in Figure 2.4.

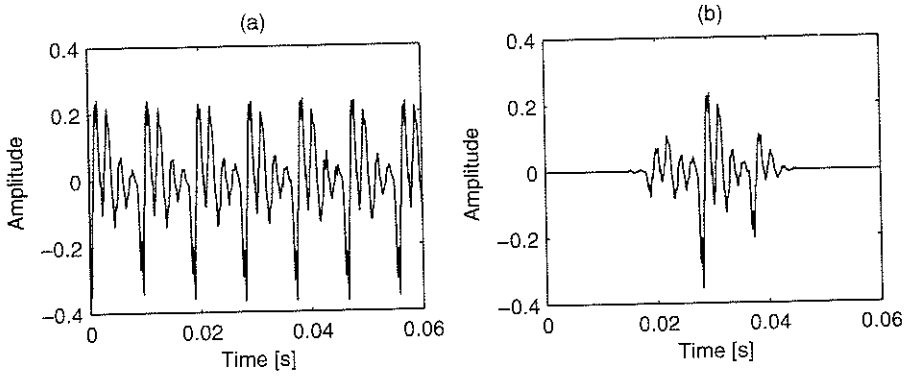

Figure 2.4: A sampled speech waveform (a) before and (b) after applying a Hamming window.

The encoder approximates the spectral envelope for a speech frame by using a 10th order all-pole linear filter:

$$\tilde{x}[n] = -\sum_{i=1}^{10} a_i \tilde{x}[n-i] + s[n] \tag{2.12}$$

x is the sampled speech signal, $x[n]$ is the nth sample of the speech signal, $\tilde{x}$ is a linear prediction of the speech signal, $s[n]$ is the excitation signal, and a is a vector of prediction coefficients which must be determined.

The encoder uses autocorrelation analysis [19] to calculate the filter coefficients. Consider a prediction error signal e, which is the difference between the signal's true value x and the linear approximation $\tilde{x}$:

$$e[n] = x[n] - \tilde{x}[n] \tag{2.13}$$

The encoder's objective is to minimize the mean squared prediction error

over the N samples in a frame. Let E denote the mean-squared error:

$$E = \frac{1}{N}\sum_{n=1}^{N}(e[n])^2 \tag{2.14}$$

$$= \frac{1}{N}\sum_{n=1}^{N}(x[n] - \tilde{x}[n])^2 \tag{2.15}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(x[n] + \sum_{i=1}^{10} a_i x[n-i]\right)^2 \tag{2.16}$$

For the sake of compactness, let $x_n = x[n]$ in the following matrix. (2.16) can be expanded and expressed in matrix form:

$$\frac{1}{N}\sum_{n=1}^{N}\begin{bmatrix} x_{n-1}^2 & x_{n-1}x_{n-2} & \cdots & x_{n-1}x_{n-10} \\ x_{n-1}x_{n-2} & x_{n-2}^2 & \cdots & x_{n-2}x_{n-10} \\ \vdots & & & \vdots \\ x_{n-1}x_{n-10} & x_{n-2}x_{n-10} & \cdots & x_{n-10}^2 \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{10} \end{bmatrix} = -\begin{bmatrix} x_n x_{n-1} \\ x_n x_{n-2} \\ \vdots \\ x_n x_{n-10} \end{bmatrix} \tag{2.17}$$

Let $r_x(i)$ denote the discrete autocorrelation of x with lag i:

$$r_x(i) = \frac{1}{N}\sum_{n=1}^{N} x[n]x[n-i] \tag{2.18}$$

(2.17) may be expressed in terms of (2.18):

$$\frac{1}{N}\sum_{n=1}^{N}\begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(9) \\ r_x(1) & r_x(0) & \cdots & r_x(8) \\ \vdots & & & \vdots \\ r_x(9) & r_x(8) & \cdots & r_x(0) \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{10} \end{bmatrix} = -\begin{bmatrix} r_x(1) \\ r_x(2) \\ \vdots \\ r_x(10) \end{bmatrix} \tag{2.19}$$

We can calculate the autocorrelation values on either side of (2.19), and then solve for the unknown $\boldsymbol{a}$ vector. The solution for $\boldsymbol{a}$ is a vector of prediction coefficients.

A time series $x[n]$ can be represented in the z-domain using the transformation [23]:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \tag{2.20}$$

where z is complex-valued, having a real part x and an imaginary part y:

$$z = x + jy \tag{2.21}$$

and $j = \sqrt{-1}$. Alternatively, z can be represented in polar form:

$$z = r\left(cos(2\pi f) + jsin(2\pi f)\right) = re^{j2\pi f/F_s} \tag{2.22}$$

where r is the magnitude of z, $r = \sqrt{x^2 + y^2}$; f is the frequency of z in Hertz; and F_s is the sampling frequency. The angle from the positive real axis corresponds to the frequency. The positive real line corresponds to 0 Hz, the positive imaginary axis corresponds to $F_s/4$ Hz, and the negative real line corresponds to $F_s/2$ Hz.

The 10th order filter with coefficients $\boldsymbol{a}$, determined using (2.19), can be expressed in the z domain as:

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{10} z^{-10} \tag{2.23}$$

The linear filter system of (2.12) can be expressed in the z-domain as:

$$X(z) = H(z)S(z) \tag{2.24}$$

where the linear filter's impulse response, $H(z)$, is given by:

$$H(z) = \frac{1}{A(z)} \tag{2.25}$$

This system is illustrated in Figure 2.5.

A pole-zero plot is a common graphical representation of a function in the z-domain. A pole is a root of the denominator of the system function,

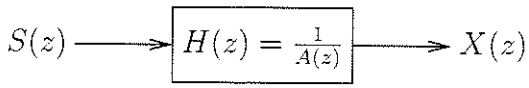

Figure 2.5: Block diagram of linear filter in the z-domain.

indicated by an 'x' in the plot. A zero is a root of the numerator of the system function, indicated by an 'o' in the plot. It is customary to plot a circle to indicate the unit circle, where $|z| = 1$. The unit circle is important in this thesis for two reasons. First, the value of $H(z)$, evaluated along the unit circle $|z| = 1$ corresponds to the filter's frequency response:

$$H(f) = H(e^{j2\pi f/F_s}) \tag{2.26}$$

Second, it indicates the stability of the function $H(z)$. Let $H(z)$ be a causal system, that is, its output depends only on the past and present values of its input signal, and not on future values of the input. If $H(z)$ is stable, then all of the poles of $H(z)$ will be inside the unit circle. [23], for example, has more detail on these topics.

After LP analysis, the encoder performs bandwidth expansion, which scales the poles of the linear filter toward the origin of the z-axis by a linear factor γ:

$$A'(z) = A(z/\gamma) \tag{2.27}$$

This corresponds to the following operation on the filter coefficients:

$$a_i' = a_i\gamma^i \tag{2.28}$$

where $\gamma = 0.994$. Bandwidth expansion decreases the magnitude of the peaks in the filter's frequency response, which can improve filter stability, and reduce unnatural chirps or oscillations in the decoded speech

[16]. The $A'(z)$ given by (2.27) is used to calculate the LSFs, using the procedure described in §2.3.2.

2.3.2 Calculating Line Spectral Frequencies

The line spectral frequencies are calculated directly from the coefficients of $A(z)$. The process described here is also given in [24]. From the filter coefficients, we form an 11th order filter with symmetric coefficients:

$$P(z) = A(z) + z^{-(m+1)}A(z^{-1}) \tag{2.29}$$

$$= 1 + (a_1 + a_m)z^{-1} + (a_2 + a_{m-1})z^{-2} + \cdots + (a_m + a_1)z^{-m} + z^{-(m+1)} \tag{2.30}$$

and an 11th order filter with anti-symmetric coefficients:

$$Q(z) = A(z) - z^{-(m+1)}A(z^{-1}) \tag{2.31}$$

$$= 1 + (a_1 - a_m)z^{-1} + (a_2 - a_{m-1})z^{-2} + \cdots + (a_m - a_1)z^{-m} - z^{-(m+1)} \tag{2.32}$$

The original filter $A(z)$ can be recovered from $P(z)$ and $Q(z)$:

$$A(z) = \frac{1}{2}\left[P(z) + Q(z)\right] \tag{2.33}$$

An important property of $P(z)$ and $Q(z)$ is that their roots are on the unit circle. The roots of $P(z)$ and $Q(z)$ correspond to the line spectral frequencies. The roots are referred to by frequency because they lie on the unit circle, so they can be identified by frequency alone. The roots of $P(z)$ and $Q(z)$ take the form

$$z = e^{j2\pi f} \tag{2.34}$$

where f is a line spectral frequency, normalized into the range $0 \leq f < 1$. The frequency in Hertz can be found by calculating fF_s.

The linear filter $A(z)$ has 10 poles. It is symmetric about the real axis, because the input signal is real valued. Because of this symmetry, five complex values are needed to fully describe $A(z)$. The polynomials $P(z)$ and $Q(z)$ have 11 poles each, for a total of 22 complex values. The poles are on the unit circle, so they can each be described by one real value, for a total of 22 real values. Two of these poles are constant, so they are not encoded: $P(z)$ always has a pole at $z = -1$, and $Q(z)$ always has a pole at $z = +1$. This leaves 20 values. The input signal is real-valued, so the poles are symmetric about the real axis, leaving 10 real values to encode.

A graphical example of LSFs, and some of the steps involved in their calculation, is presented in Figure 2.6. Consider the windowed speech waveform in Figure 2.6 (a). The energy density spectrum of this waveform is presented in Figure 2.6 (b). A 10th order all-pole linear filter, as in (2.12), which approximates this waveform is determined using (2.19). The poles of the resulting filter are presented in a pole-zero plot in Figure 2.6 (c). The frequency response of this filter is presented in Figure 2.6 (d), with the frequency of the poles indicated by dotted lines. We see that the frequency of the poles corresponds to the peaks in the frequency response, and that the frequency response of the filter in Figure 2.6 (d) approximates the shape of the original signal's spectrum in Figure 2.6 (b). Next we determine the polynomials $P(z)$ and $Q(z)$, using (2.29) and (2.31), respectively. The roots of these polynomials are plotted as poles in the pole-zero plot in Figure 2.6 (e). The roots of these polynomials correspond to the LSFs. The resulting LSFs are shown as dotted lines over the linear filter's frequency response in Figure 2.6 (f).

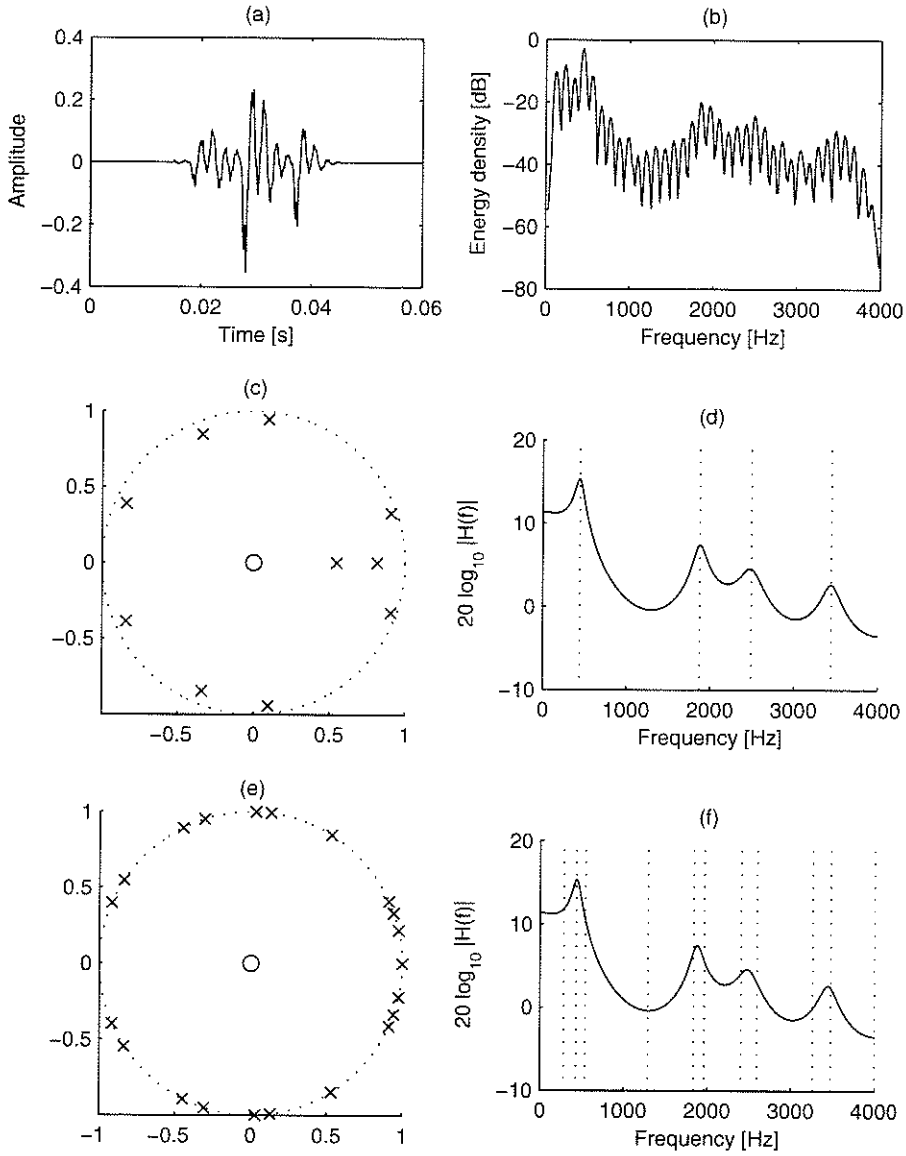

Figure 2.6: A graphical depiction of LSFs, and the steps involved in their calculation. (a) is a speech waveform, after filtering, sampling, and applying a Hamming window. (b) is the energy density of the speech waveform. (c) is a pole-zero plot of a linear filter which approximates (a). (d) is the frequency response of this linear filter, with the frequency of filter poles indicated by vertical lines. (e) shows the location of the roots of the polynomials $P(z)$ and $Q(z)$, which are used to determine the LSFs. (f) shows the resulting LSFs as dotted lines, superimposed over the frequency response of the linear filter.

2.3.3 Properties of LSFs

In [25], Itakura describes three important properties of LSFs. First, if a pole of $A(z)$ is close to the unit circle, then the corresponding poles of $P(z)$ and $Q(z)$ will be close to the original pole. Poles of $A(z)$ close to the unit circle correspond to regions where the frequency response is greatest. A similar property is described by Paliwal and Atal in [26], who state that a change in one LSF tends to affect the frequency response in the vicinity of the changed LSF. Combined with the first property, this means that the frequency of LSFs is related to the filter's frequency response in an intuitive manner. This characteristic is useful for quantizing LSFs according to perceptual concerns or error weighting, as described in §2.3.7.

The second property is that the poles of $P(z)$ and $Q(z)$ alternate (that is, a pole in $P(z)$ is adjacent to two poles in $Q(z)$ and vice versa), and they are ordered on the frequency axis, so the $l+1$th LSF is greater than the lth LSF. This property allows us to distinguish between the roots of $P(z)$ and the roots of $Q(z)$, given only a list of LSFs.

The final property is that the LSF representation simplifies the mathematics of locating poles, from finding complex poles to finding half the number of real poles.

2.3.4 Quantization

The FS-1016 CELP voice coder uses a scalar quantizer for each LSF. Let C_l denote the codebook used to quantize the lth LSF. The number of bits allocated for each codebook is listed in Table 2.1. Let $C_l[i]$ denote the ith entry of the codebook for LSF l. Let $\boldsymbol{X}[n]$ denote the LSFs from the

	LSF									
	1	2	3	4	5	6	7	8	9	10
Bit allocation	3	4	4	4	4	3	3	3	3	3
Quantization levels	8	16	16	16	16	8	8	8	8	8

Table 2.1: Bit allocation of LSF quantizers in FS-1016 CELP speech coder [12].

nth frame of a speech signal, and let $\boldsymbol{X}_l[n]$ denote the lth LSF in this frame. When quantizing an LSF $\boldsymbol{X}_l[n]$, the quantizer searches for the codebook entry $i_{\min}$ which minimizes the distortion $d()$ between the LSF being quantized and the codebook entries:

$$d\left(\boldsymbol{X}_l[n], \mathcal{C}_l[i_{\min}]\right) \leq d\left(\boldsymbol{X}_l[n], \mathcal{C}_l[i]\right) \tag{2.35}$$

for all i in the codebook. The FS-1016 LSF quantizer uses the squared-error distortion function:

$$d(a, b) = (a - b)^2 \tag{2.36}$$

Let $\boldsymbol{U}_l[n] = i_{\min}$ denote the quantizer index selected for LSF l in frame n. The encoder transmits the codebook index $i_{\min}$ to the decoder. The decoder reconstructs the transmitted LSF as $\hat{\boldsymbol{X}}_l[n] = \mathcal{C}_l[i_{\min}]$.

Quantizing the LSFs as described above could result in cases where the original LSFs are in order ($\boldsymbol{X}_l[n] < \boldsymbol{X}_{l+1}[n]$), but the selected codebook indices produce out-of-order LSFs at the decoder ($\hat{\boldsymbol{X}}_l[n] > \hat{\boldsymbol{X}}_{l+1}[n]$). To avoid such cases, the quantizer increments and decrements quantizer indices as necessary. The selected codebook index may no longer minimize the distortion function, but the ordering property will be preserved.

2.3.5 Redundancy in Quantized LSFs

Optimal source coding eliminates any redundancy in the transmitted bit-stream [27]. However, practical source encoders are not optimal. For example, the encoding procedure used by FS-1016 leaves redundancy between the quantized LSFs [13]. This section examines this redundancy, to consider its usefulness for estimating missing LSFs. We will begin by estimating the mean and variance of LSFs. Next we will consider the mean squared quantization error of the FS-1016 encoder. Finally we consider estimating a missing LSF by using a received quantizer index for another LSF. The calculations in this section use the training data set described in Appendix A. Other studies of LSF statistics and the dependence between LSFs include [24], which presents histograms of LSF distributions, and the difference $\boldsymbol{X}_l - \boldsymbol{X}_{l-1}$. [13] examines the three most significant bits of FS-1016 quantized LSFs, and finds the dependence between them based on entropy and the number of redundant bits.

First we consider the statistics of the training set, as a basis of comparison for subsequent work. We begin by calculating the mean value of the lth LSF in the training set, using N frames of training data:

$$\mathrm{E}\{\boldsymbol{X}_l\} \approx \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{X}_l[n] \tag{2.37}$$

The estimated mean value of each LSF is listed in Table 2.2. Next we find the variance of each LSF in the training data set, that is:

$$\mathrm{Var}(\boldsymbol{X}_l) = \mathrm{E}\left\{(\boldsymbol{X}_l - \mathrm{E}\{\boldsymbol{X}_l\})^2\right\} \tag{2.38}$$

Using N frames of training data, we estimate:

$$\mathrm{Var}(\boldsymbol{X}_l) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{X}_l[i] - \frac{1}{N}\sum_{j=1}^{N}\boldsymbol{X}_l[j]\right)^2 \tag{2.39}$$

LSF	Mean $\mathrm{E}\{\boldsymbol{X}_l\}$	Variance $\mathrm{E}\left\{(\boldsymbol{X}_l - \mathrm{E}\{\boldsymbol{X}_l\})^2\right\}$	MSQE $\mathrm{E}\left\{\left(\boldsymbol{X}_l - \hat{\boldsymbol{X}}_l\right)^2\right\}$
1	396.02	10 192.1	1 073.38
2	567.39	20 193.6	347.21
3	873.59	36 210.9	612.48
4	1 264.77	51 067.9	820.30
5	1 629.48	75 341.7	693.80
6	1 972.40	66 990.4	2 868.01
7	2 390.57	55 360.0	3 436.36
8	2 721.51	47 584.4	2 465.37
9	3 183.05	31 985.8	1 220.52
10	3 486.83	17 483.4	1 042.50

Table 2.2: Mean, variance and mean-squared quantization error (MSQE) of each LSF from our training set.

The estimated variance of each LSF is listed in Table 2.2.

Next we consider the mean squared error of the FS-1016 LSF quantizers. The mean squared quantization error (MSQE) is the average squared difference between the original value $\boldsymbol{X}_l[i]$ and the quantizer's reproduction value $\hat{\boldsymbol{X}}_l[i]$. This is estimated using N frames of training data:

$$\mathrm{E}\left\{(\boldsymbol{X}_l - \hat{\boldsymbol{X}}_l)^2\right\} \approx \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{X}_l[i] - \hat{\boldsymbol{X}}_l[i])^2 \tag{2.40}$$

The estimated MSQE for each LSF is listed in Table 2.2.

Finally we consider estimating the value of an LSF by using the quantizer index of another LSF. This configuration is intended to simulate the behaviour of a decoder, which uses the received quantizer index for one LSF to estimate an LSF which was lost over the communication channel. We will measure the performance of this estimator using the mean squared error. The values in Table 2.2 suggest a reasonable range of squared error values to expect. The variance, in the third column, is

the squared error of an estimator which does no better than using the mean value of a missing LSF. This is approximately the greatest error we should expect. The quantizer error, in the fourth column, is the squared error of an estimator which does as well as the LSF's quantizer. This is the smallest squared error we should expect.

To estimate $\boldsymbol{X}_l[n]$, the estimator uses a single received quantizer index, namely, the index for LSF l' in frame $n + \Delta$. Let $\tilde{\boldsymbol{X}}$ denote the estimated value of $\boldsymbol{X}$. The estimator uses a minimum mean squared error estimate of $\boldsymbol{X}$ [17]:

$$\tilde{\boldsymbol{X}}_l[n] \triangleq \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{U}_{l'}[n+\Delta]\right\} \tag{2.41}$$

The conditional expectation on the right-hand side of (2.41) is estimated by averaging over the N speech frames of the training set. We assume that the expected value is independent of n, so it is a function of l, l', and Δ alone. (2.41) is estimated by calculating:

$$\mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{U}_{l'}[n+\Delta] = i\right\} \approx \frac{\sum_{k=\max(1,1-\Delta)}^{\min(N,N-\Delta)} \boldsymbol{X}_l[k]\,\mathrm{I}\left(\boldsymbol{U}_l[k+\Delta] = i\right)}{\sum_{k=\max(1,1-\Delta)}^{\min(N,N-\Delta)} \mathrm{I}\left(\boldsymbol{U}_l[k+\Delta] = i\right)} \tag{2.42}$$

for $i = 1, \ldots, |\mathcal{C}_l|$. The summation limits are over all valid frames in the training set. The function $\mathrm{I}\,()$ used in (2.42) is defined as:

$$\mathrm{I}\,(l) \triangleq \begin{cases} 1 & \text{if } l \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{2.43}$$

so

$$\mathrm{I}\,(a = b) \triangleq \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases} \tag{2.44}$$

We evaluate the performance of the estimator designed in (2.42) by using the testing data set, which is described in Appendix A. We evaluate

mean squared error between the estimator's output and the LSF's true value, over all valid frames in the N frames of testing data:

$$\mathrm{mse}(l, l', \Delta) \approx \frac{1}{N - |\Delta|} \sum_{n=\max(1,1-\Delta)}^{\min(N,N-\Delta)} (\boldsymbol{X}_l[n] - \mathrm{E}\{\boldsymbol{X}_l \mid \boldsymbol{U}_{l'}(n+\Delta)\})^2 \quad (2.45)$$

For each estimated LSF $l = 1, \ldots, 10$, we will vary the estimator's input LSF over $l' = 1, \ldots, 10$, and we vary the frame offset Δ over the range $\Delta = -3, \ldots, 3$. The results are presented graphically with a grey scale representing the relative variance in Figure 2.7. This figure, and the underlying data, indicate the following:

- The best estimator for LSF $\boldsymbol{X}_l[n]$ is always $\boldsymbol{U}_l[n]$, as we would expect.

- Five LSFs (1, 2, 3, 6 and 8) have the lowest estimator error from an LSF in the same frame. The remaining five LSFs (4, 5, 7, 9 and 10) have the lowest estimator error from an adjacent frame. This suggests that dependence both within the same frame and between adjacent frames is important.

- We define the four-connected neighbours of LSF $\boldsymbol{X}_l[n]$ as LSFs $\boldsymbol{X}_{l-1}[n]$, $\boldsymbol{X}_{l+1}[n]$, $\boldsymbol{X}_l[n-1]$, and $\boldsymbol{X}_l[n+1]$. The second, third and fourth lowest average error is always from one of the four-connected neighbours of the lost LSF. This suggests that decoders should account for the dependence between the four-connected neighbours of an LSF.

- All LSFs have at least one neighbour for which the estimator error is less than 0.6 of the LSF's variance.

- Consider dividing the LSFs of each frame into pairs, $(\boldsymbol{X}_l[n], \boldsymbol{X}_{l+1}[n])$ for l odd. Thus LSFs 1 and 2 form a pair, 3 and 4 form a pair, and so forth. When considering LSFs in the same frame, $\Delta = 0$, most LSFs (1, 5–10) have the lowest estimator error from the other LSF in their pair. The remaining three LSFs (2,3,4) have the lowest estimator error from their other neighbour. This result, as well as the success of vector quantizers such as those mentioned in the next section, encourages us to consider exploiting the dependence in such pairs of LSFs.

A similar plot is presented in [28], which depicts the dependence within a frame using correlation coefficients.

2.3.6 Other LSF Quantizers

Some of the residual redundancy observed above could be eliminated by using more efficient coding schemes. In scalar quantization, each codeword represents one source sample, such as one LSF. In vector quantization (VQ) [17], one codeword represents multiple source samples, such as all ten LSFs in a frame. The vector codewords allow the quantizer to account for the dependence between LSFs in the same frame, and allow for more efficiently shaped quantizer cells. Quantizing all ten LSFs at once requires a prohibitively large codebook, so variations of VQ have been developed to make its use feasible. For example, [26] investigates two such variations. In split VQ, the ten dimensional vector of LSFs is split into two or more smaller, more manageable sub-vectors. In multi-stage VQ, the ten dimensional LSF vector is quantized in multiple stages. The

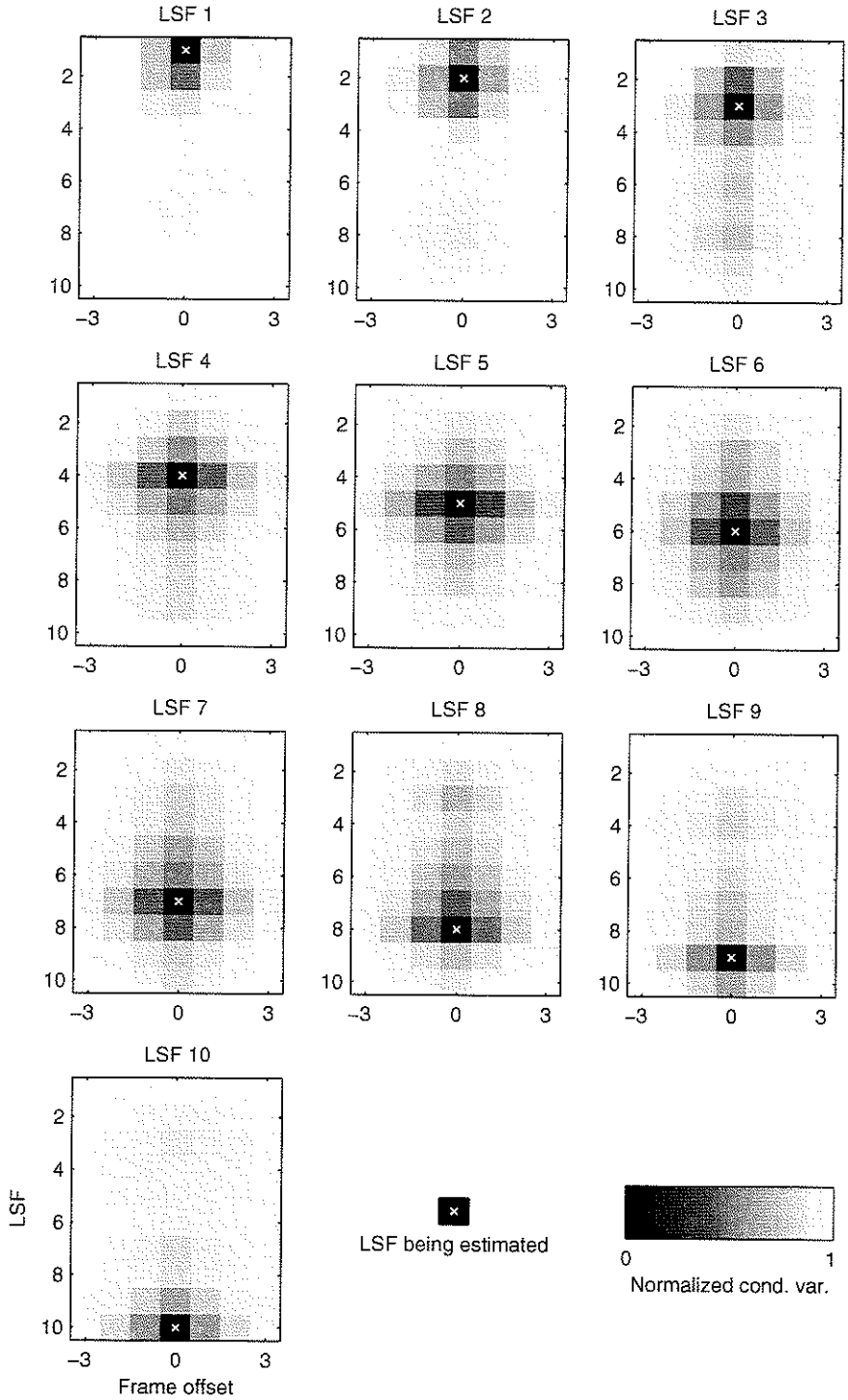

Figure 2.7: Error (MSE) in estimating LSFs using nearby quantized LSFs. In each plot, the LSF marked 'x' is the LSF being estimated. The surrounding squares represent the LSFs used to estimate the LSF. The rows are the ten LSFs in each frame. The columns represent the number of frames between the LSF being estimated, and the LSF being used to estimate it. Darker squares indicate a more accurate (lower error) estimate.

first stage uses a relatively coarse quantizer, and the second stage quantizes the difference between the source vector and the vector selected in the first stage. Predictive VQ [17] can take advantage of dependence within a frame, using VQ, as well as dependence between frames, by predicting the next frame using past frames. [29] examines a system which uses a switched linear predictor between frames, and quantizes the residual. The linear predictor is chosen by finding the best match for the frame being quantized.

2.3.7 Other Distortion Measures for LSFs

This thesis uses the squared error measure (2.36) throughout. However, other distortion measures have been designed to improve the performance of vector quantizers for LSFs.

A weighted Euclidean error measure is described in [26]. In this approach, the squared error is weighted by two factors:

$$d(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \sum_{l=1}^{10} (c_l w_l (\boldsymbol{X}_l - \hat{\boldsymbol{T}}_l)^2) \tag{2.46}$$

The fixed weight c_i accounts for a person's decreased sensitivity to frequency changes at higher frequencies. It is given the values $c_l = 1$ for $l = 1, \ldots, 8$, $c_9 = 0.8$, and $c_{10} = 0.4$. The variable weight $w_i = [P(f_i)]^r$ depends on the power spectrum's amplitude at the LSF's frequency, $P(f_i)$. This weight gives more importance to LSFs which form the spectral peaks, where a small change in frequency can affect a large change in the spectrum. The constant r is used to tune the relative weights.

The distortion measure in GSM-AMR [22] uses a variable weight similar to (2.46). Here the variable weight for LSF l is based on a piecewise

continuous function of the difference $\boldsymbol{X}_{l+1} - \boldsymbol{X}_{l-1}$. If the difference between the neighbouring LSFs is small, the LSF is given more weight.

Chapter 3

Multiple Description Coding of Line Spectral Frequencies

3.1 Introduction

In multiple description (MD) [10] coding, two or more descriptions are generated for an input. These descriptions are transmitted such that the loss of all the generated descriptions is less likely than losing the one description in a comparable single-description system. To accomplish this, the descriptions may be transmitted at different times, over different paths, at different frequencies, or using any other kind of diversity. The descriptions are designed so that receiving a subset of them is useful. That is, decoding a subset of the transmitted descriptions should give the receiver an acceptable reproduction of the source. Ideally, receiving more descriptions should allow the receiver to output a more accurate representation of the source.

This chapter introduces the system configuration and notation we use

in this thesis. Next, several known methods for generating multiple descriptions are reviewed, with an emphasis on methods which have been proposed for use with speech and audio coding. The methods we examine for use with LSFs are based on odd-even splitting, which is described in §3.4.1, and MD index assignment [37], which is described in §3.4.2. Later we review known methods for estimating missing LSFs, which we apply to estimate the missing odd or even LSFs which have been transmitted using odd-even splitting. Finally we review the operation of a memoryless MD index assignment decoder. We will evaluate and compare the performance of these methods in Chapter 5.

3.2 Rate-Distortion Perspective

In this section, we briefly consider the relationship between the average bit rate R and the distortion D of a coding system. We begin by considering the rate-distortion function in a single description system. This is the relationship between the rate R and the lowest possible distortion D of a quantizer at the rate R (or vice versa). A brief description of rate-distortion functions for single description and MD systems may be found in [10]. Assume that the source being quantized is memoryless, and the distortion measure is the mean-squared error. If the rate $R = 0$, the best a receiver can do is output the expected value of the source. The distortion of a system at this rate is the variance of the source, σ^2. As the rate R increases, the source can be described increasingly well. As the rate $R \to \infty$, the distortion decreases, $D \to 0$. The exact relationship between these extremes depends on the source.

In a MD system, the rate-distortion relationship becomes more complicated. In a two-description system, the total rate R is divided between two descriptions, with rates R_1 and R_2 respectively, such that $R = R_1 + R_2$. The decoder which operates when both descriptions are received is the *central decoder*, and the decoders which operate on a single received description are the *side decoders*. For each pair of rates, we have a set of possible distortions, D_0 for the central decoder, D_1 for the first side decoder, and D_2 for the second side decoder. Let us consider some extreme cases in a MD rate-distortion function. Let R_{SD} and D_{SD} denote the rate and distortion of a single description encoder, respectively. If $R_1 = R_{\mathrm{SD}}$, then we have a single description system, where $D_0 = D_1 = D_{\mathrm{SD}}$. Perhaps the simplest MD encoding scheme is repetition, for which $R = 2R_{\mathrm{SD}}$, $R_1 = R_2 = R_{\mathrm{SD}}$. Here all three decoders have the same distortion: $D_0 = D_1 = D_2 = D_{\mathrm{SD}}$. Repetition appears to use the rate poorly. The central decoder has a rate of $2R_{\mathrm{SD}}$, but achieves the same distortion as a system with half the rate. Most MD systems attempt to improve upon this performance. In [30], Gamal and Cover describe the achievable rates for a given distortion in a MD system.

In this thesis, we focus on the case where the central quantizer is fixed at the same R_{SD} used by the FS-1016 speech coder. We fix the rate of the MD encoder at $R_1 + R_2 = R_{\mathrm{SD}}$, and we maintain the same distortion as the FS-1016 speech coder. Within these constraints, we design MD encoders and decoders in an attempt to improve the side distortion, D_1 and D_2.

3.3 System Configuration and Notation

Now let us consider the system and notation we use in this thesis. A block diagram of the system is presented in Figure 3.1. The speaker's speech signal has amplitude $s(t)$ at continuous time t. $s(t)$ is filtered, sampled at 8 kHz, and quantized to form $s[k]$ for discrete time k. The FS-1016 CELP encoder [12] partitions $s[k]$ into non-overlapping *frames* of 240 successive samples each, which correspond to 30 ms of $s(t)$, and each frame is split into four 7.5 ms *subframes*. Each frame is modelled using a short-term prediction filter, which approximates the frame's frequency spectrum, and an excitation waveform, which drives the short-term prediction filter. The short-term prediction filter is updated every frame. The excitation is encoded using a long-term predictor, an excitation codebook entry, and the gain, which are all updated every subframe. Further detail on the FS-1016 encoder may be found in Chapter 2, and the references therein.

The short-term prediction filter for the nth frame is converted to a vector of ten LSFs (see §2.3). We denote this vector by $\boldsymbol{X}[n]$, where the index $[n]$ denotes the speech frame, $n = 1, \ldots, N$. N is the total number of frames. To index the ten LSFs in a frame, we will use a subscript l: $\boldsymbol{X}_l[n]$ for $l = 1, \ldots, 10$. If we exclude the index n, we are referring to the values of the specified LSF, across all N speech frames:

$$\boldsymbol{X}_l \triangleq \{\boldsymbol{X}_l[n], n = 1 \ldots N\} \tag{3.1}$$

The FS-1016 speech encoder uses ten scalar quantizers, one for each LSF, to quantize $\boldsymbol{X}[n]$. These are the *central quantizers* of the system. A scalar quantizer Q_l uses codebook $\mathcal{C}_l$ to quantize $\boldsymbol{X}_l$, following the

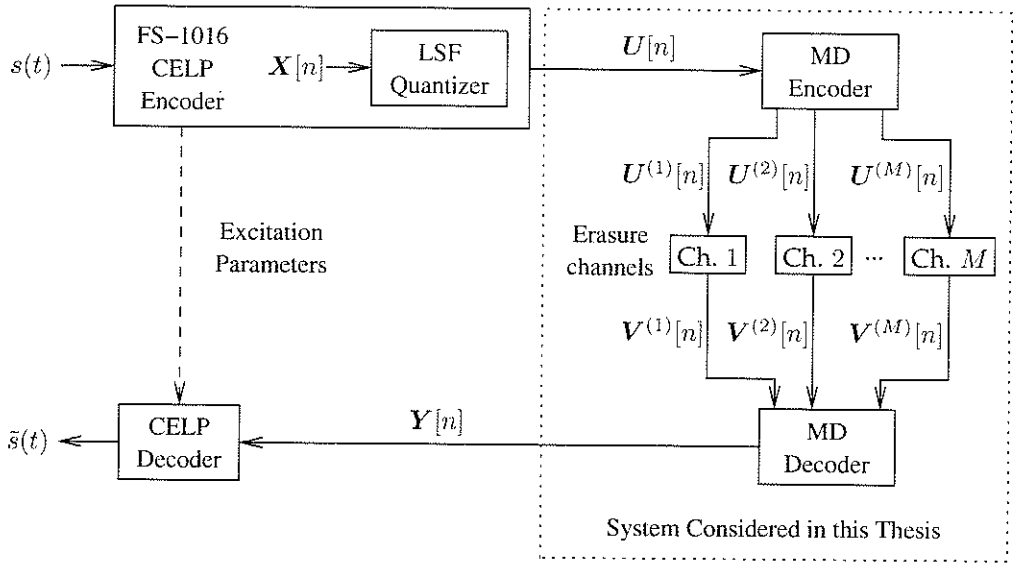

Figure 3.1: System block diagram.

procedure described in §2.3.4. The codebook index selected to quantize $\boldsymbol{X}_l[n]$ is $\boldsymbol{U}_l[n] = Q_l(\boldsymbol{X}_l[n])$. The codebook entry selected for $\boldsymbol{X}_l[n]$ is $\hat{\boldsymbol{X}}_l[n] \triangleq \mathcal{C}_l[\boldsymbol{U}_l[n]]$. This is also called the quantizer's *reproduction value* for $\boldsymbol{X}_l[n]$. We recall from §2.3.4 that the encoder selects codebook indices such that the quantized LSFs are monotonically increasing:

$$\hat{\boldsymbol{X}}_1[n] < \hat{\boldsymbol{X}}_2[n] < \cdots < \hat{\boldsymbol{X}}_{10}[n] \tag{3.2}$$

Next, we consider the MD components of the system, which are the focus of this thesis. To restrict the scope of our investigation, we encode the LSFs only, and we assume that other CELP parameters are transmitted without loss. A MD encoder generates M descriptions of $\boldsymbol{U}_l[n]$, which we denote by $\boldsymbol{U}_l^{(1)}[n], \ldots, \boldsymbol{U}_l^{(M)}[n]$. We design the MD encoder such that the decoder will perform the same as FS-1016 when no packet loss occurs. We will describe several approaches for generating these descriptions later in this chapter. For each frame, we form M *packets*. The mth packet holds the mth description of every LSF in the frame. Let $\boldsymbol{U}^{(m)}[n]$

denote the mth packet formed for frame n:

$$\boldsymbol{U}^{(m)}[n] \triangleq \left\{ \boldsymbol{U}_l^{(m)}[n], \text{for } l = 1, \ldots 10 \right\} \tag{3.3}$$

The packets are transmitted over an erasure channel to the MD decoder. Due to the erasure channel, a packet is either correctly received, or else it is *erased* and lost completely. Let $\boldsymbol{V}_l^{(m)}[n]$ denote the received version of $\boldsymbol{U}_l^{(m)}[n]$. We indicate that a description was erased by setting it to $\varnothing$:

$$\boldsymbol{V}_l^{(m)}[n] = \begin{cases} \varnothing & \text{if the channel erases } \boldsymbol{U}^{(m)}[n] \\ \boldsymbol{U}_l^{(m)}[n] & \text{otherwise} \end{cases} \tag{3.4}$$

for $m = 1 \ldots M$. If we exclude the superscript of $\boldsymbol{V}_l^{(m)}[n]$, we are referring to the collection of all descriptions received for the lth LSF in the nth frame:

$$\boldsymbol{V}_l[n] \triangleq \left\{ \boldsymbol{V}_l^{(m)}[n], m = 1 \ldots M \right\} \tag{3.5}$$

Using $\boldsymbol{V}_l[n]$, the MD decoder will form an estimate $\boldsymbol{Y}_l[n]$ of the original LSFs, $\boldsymbol{X}_l[n]$. We identify individual values in $\boldsymbol{Y}$ as we do with $\boldsymbol{X}$, thus $\boldsymbol{Y}_l[n]$ denotes the decoder's output for the lth LSF in frame n. We will consider several different procedures for forming $\boldsymbol{Y}$ in this chapter. The CELP decoder uses this $\boldsymbol{Y}$ along with the excitation parameters to reconstruct the original speech signal.

Before passing $\boldsymbol{Y}$ to the CELP decoder, we ensure that the LSFs are in increasing order:

$$\boldsymbol{Y}_1[n] < \boldsymbol{Y}_2[n] < \cdots < \boldsymbol{Y}_{10}[n] \tag{3.6}$$

We exchange the position of LSFs if necessary. In addition, we ensure a minimum spacing of 20 Hz between LSFs, $\boldsymbol{Y}_{l+1}[n] - \boldsymbol{Y}_l[n] \geq 20$ Hz, as in [40]. If two adjacent LSFs are too close, there will be a large peak in the

corresponding filter's frequency response. In extreme cases, a decoder can output two exactly equal LSFs, $\boldsymbol{Y}_l[n] = \boldsymbol{Y}_{l+1}[n]$, which gives the power spectrum an infinite amplitude at the corresponding frequency.

All of the decoders in this chapter use statistical properties of LSFs to conceal losses. Ideally, a decoder would base its decisions on the true statistical distribution of LSFs. However, we do not know the true statistical properties, so we estimate them using a *training data* set. The procedure we used to generate our training data set is described in Appendix A. We denote the training data using a dot over the associated variable. For example, $\dot{\boldsymbol{X}}$ denotes the LSFs of the training data before they are quantized, and $\dot{\boldsymbol{U}}$ denotes the quantizer indices generated for the training data. Let $\dot{N}$ denote the number of frames of training data.

3.4 Multiple Description Techniques

3.4.1 Odd-Even Splitting

The first and simplest procedure we consider for generating multiple descriptions is odd-even splitting. In the first type of MD coder we consider, the system begins with a traditional single-description coder. Let $\boldsymbol{X} = \{a, b, c, d\}$ be the output of a single description encoder. Assume that the MD encoder forms two descriptions, and let these descriptions be denoted by π_1 and π_2. One way these descriptions could be formed is:

$$\pi_1 = \{a, c\} \tag{3.7}$$

$$\pi_2 = \{b, d\} \tag{3.8}$$

We begin with two systems which have been proposed for use with waveform coders. In [10], Goyal describes the origins of MD coding at Bell Labs. The system described therein transmits sampled voice data over two physical channels. One channel carries the even-numbered samples, while the other carries the odd-numbered samples. In [11], Jayant and Christensen describe a MD system for transmitting PCM and DPCM data over a lossy packet network. The authors consider speech sampled at 8 kHz. The even samples are sent in one packet, and the odd samples are sent in another. The receiver estimates missing packets by estimating the missing samples from the adjacent samples in the corresponding frame. A predictor with fixed prediction coefficients is described, as well as an adaptive predictor, in which suitable prediction coefficients are determined and transmitted for each packet. The adaptive predictor gives considerably better performance. Alternatively, the authors suggest oversampling the speech signal; with oversampling, the fixed predictors can achieve similar results to the adaptive predictors without oversampling.

Next we review methods which have been proposed for use with LSFs. In [31], Lin and Wah describe a MD coding system for CELP over a packet network. For the LSF data, their approach uses an odd-even splitting technique similar to [11]. Rather than divide waveform samples, though, their approach divides the output LSFs for the analysis frames into odd and even frames, which are transmitted in separate packets. To recover a lost packet, the decoder interpolates between the adjacent frames, which are received in the other packet. To generate multiple descriptions of the excitation signal, the encoder determines the excitation

codewords for two 120 sample subframes, instead of four 60 sample subframes in FS-1016, using the same number of bits per subframe. Both excitation codewords are transmitted in both packets. [41] describes a more general approach for distributing correlated information between frames, which is used to divide split-VQ LSF vectors among packets. In [32], Anandakumar et al. describe a MD coding scheme for CELP over packet networks. Their approach divides CELP's parameters into redundant and non-redundant parts. The non-redundant part is transmitted once. The redundant part is transmitted twice: once, with the non-redundant data for the same speech frame, and again, piggy-backed on a future frame. Several methods for dividing the parameters are considered. The LSFs, the adaptive codebook index, and adaptive codebook gain are always part of the redundant data. The fixed codebook gain, and fixed codebook pulses are divided or duplicated between the redundant and non-redundant parts, in a trade-off between the bit rate and the side distortion.

To generate such descriptions for the system examined in this thesis, we place the odd-numbered LSFs from the nth speech frame in one packet, $\boldsymbol{U}^{(1)}[n]$, and the even-numbered LSFs in another packet, $\boldsymbol{U}^{(2)}[n]$. For convenience, we will sometimes refer to these packets as $\boldsymbol{U}^{(o)}[n]$ or $\boldsymbol{U}^{(e)}[n]$, respectively, where the superscripts denote the odd or even LSFs:

$$\boldsymbol{U}_l^{(1)}[n] = \boldsymbol{U}_l^{(o)}[n] \triangleq \begin{cases} \boldsymbol{U}_l[n] & \text{for } l \text{ odd} \\ \varnothing & \text{for } l \text{ even} \end{cases} \tag{3.9}$$

$$\boldsymbol{U}_l^{(2)}[n] = \boldsymbol{U}_l^{(e)}[n] \triangleq \begin{cases} \varnothing & \text{for } l \text{ odd} \\ \boldsymbol{U}_l[n] & \text{for } l \text{ even} \end{cases} \tag{3.10}$$

where $\varnothing$ indicates that this description does not hold the corresponding LSF. Thus our two descriptions are:

$$\boldsymbol{U}^{(1)}[n] = \boldsymbol{U}_l^{(o)}[n] = (\boldsymbol{U}_1[n], \varnothing, \boldsymbol{U}_3[n], \varnothing, \boldsymbol{U}_5[n], \ldots, \boldsymbol{U}_9[n], \varnothing) \tag{3.11}$$

$$\boldsymbol{U}^{(2)}[n] = \boldsymbol{U}_l^{(e)}[n] = (\varnothing, \boldsymbol{U}_2[n], \varnothing, \boldsymbol{U}_4[n], \varnothing, \boldsymbol{U}_6[n], \ldots, \boldsymbol{U}_{10}[n]) \tag{3.12}$$

Odd-even splitting is sensible when the odd descriptions can be estimated from the even descriptions, and vice-versa. We saw in §2.3.5 that there is strong dependence between the adjacent LSFs $\boldsymbol{X}_l[n]$ and $\boldsymbol{X}_{l+1}[n]$, and between $\boldsymbol{X}_l[n]$ and $\boldsymbol{X}_l[n+1]$, so we expect that odd-even separation of LSFs will work well.

3.4.2 Multiple Description Index Assignment

A MD index assignment (MDIA) is a mapping from a central quantizer index, i, to M multiple description indices, $i^{(1)}, \ldots, i^{(M)}$. In MD index assignment (MDIA) [37], a quantizer's output index is mapped into two or more indices, which are the multiple descriptions for the input. Here we will consider the two description case, as in [37], for a system with two channels. A scalar quantizer generates a single index i for each input sample, using the codebook $\mathcal{C}$. From i, a MD encoder generates two indices, $i^{(1)}$ and $i^{(2)}$. These descriptions are transmitted over channel 1 and channel 2, respectively. The receiver uses three different decoders. The first decoder is used when only the first description is received, and uses $i^{(1)}$ as an index into codebook $\mathcal{C}^{(1)}$. Similar to the first decoder, the second decoder is used when only the second description is received, and uses $i^{(2)}$ as an index into the codebook $\mathcal{C}^{(2)}$. The third and final decoder, called the central decoder, is used when both descriptions are received. The

central decoder combines the two MD indices, and uses the pair $(i^{(1)}, i^{(2)})$ as an index into the original codebook $\mathcal{C}$.

Previously in speech applications, MDSQ has been used for simulated speech waveforms [42], and MD vector quantization has been applied to predictively quantized LSF vectors [43].

Vaishampayan [37] presents a matrix form for visualizing a two-description index assignment (IA). An example of this matrix-based representation is presented in Figure 3.2. The numbers inside the grid form the IA matrix, which we denote by $\mathcal{I}$.

The numbers inside the grid form the IA matrix, which we denote $\mathcal{I}$. The matrix is two-dimensional, so this IA is used to generate two descriptions. The numbers inside the matrix correspond to the indices i which may be output by the central quantizer. The numbers outside the matrix, along the top and left edges, are the MD indices, $i^{(1)}$ and $i^{(2)}$. $i^{(1)}$ is associated with the rows of $\mathcal{I}$, and $i^{(2)}$ is associated with the columns. Let us consider an example. Let the central quantizer's output be $i = 5$. The MD encoder would output $i^{(1)} = 2$ as the first description, because 5 is in the second row of $\mathcal{I}$, and $i^{(2)} = 3$ as the second description, because 5 is in the third column of $\mathcal{I}$. Now assume that the decoder receives the pair $(i^{(1)}, i^{(2)}) = (4, 3)$ over channels 1 and 2. By referring to the 4th row and 3rd column of $\mathcal{I}$, the decoder can determine that the central quantizer's output was 9. Similarly, if the decoder receives only 3 over channel 1 (channel 2 is lost), then the decoder can determine that the central quantizer index must have been one of 6, 7, or 8. The empty cells of the IA matrix are unused pairs of side decoder indices. Unused decoder indices add redundancy to the output, which can improve the

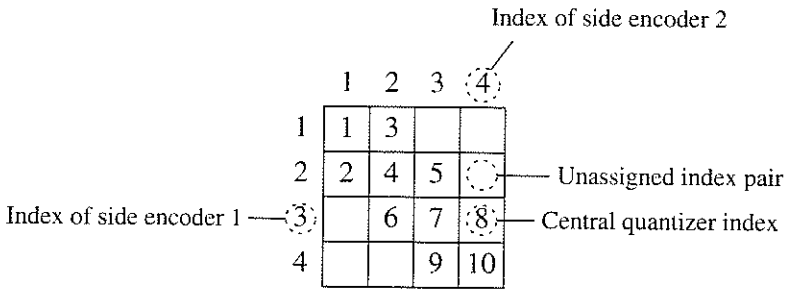

Figure 3.2: An index assignment matrix.

performance of the side decoders.

In the same paper, Vaishampayan also describes the design of an optimal decoder. First we consider designing the central decoder. Assume the input to the central quantizer is x, drawn from the random variable $\boldsymbol{X}$. Let $q_1(x) = i^{(1)}$ denote the MD encoder's output for the first description, and let $q_2(x) = i^{(2)}$ denote the MD encoder's output for the second description. Assume that the decoder receives both indices, $(i^{(1)}, i^{(2)})$. For a central decoder, the pair of received indices specifies the central quantizer index. By evaluating the input distribution $\boldsymbol{X}$ over all points x, we can find the regions of $\boldsymbol{X}$ for which $q_1(x) = i^{(1)}$ and $q_2(x) = i^{(2)}$, that is, all the possible input points which would result in the observed descriptions. We fidn the point y which minimises the expected distortion between the point y and the possible input points.

$$y^{(0)}(i,j) = \underset{y}{\operatorname{argmin}}\, \mathrm{E}\left\{d(\boldsymbol{X}, y) | q^{(1)}(\boldsymbol{X}) = i, q^{(2)}(\boldsymbol{X}) = j\right\} \tag{3.13}$$

where argmin_y is the value of the argument y which minimises the expression.

For a side decoder, the received index allows us to determine a set of central quantizer indices which could have resulted in the observed MD index. Similar to the case of the central decoder, we wish to find a point

y which minimises the distortion between the point y and the points of the input which could have resulted in the observed output index. For a distortion measure d, the optimal decoders are:

$$y^{(1)}(i) = \underset{y}{\operatorname{argmin}}\, E\left\{d(X,y)|q^{(1)}(X) = i\right\} \tag{3.14}$$

for the first side decoder, and

$$y^{(2)}(i) = \underset{y}{\operatorname{argmin}}\, E\left\{d(X,y)|q^{(2)}(X) = i\right\} \tag{3.15}$$

for the second side decoder.

For the squared-error distortion measure, $d(a,b) = (a-b)^2$, the optimal decoder outputs can be determined using the expected value operator. The optimal decoders simplify to:

$$y^{(1)}(i) = \mathrm{E}\left\{\boldsymbol{X}|q^{(1)}(\boldsymbol{X}) = i\right\} \tag{3.16}$$

for the first side decoder, and

$$y^{(2)}(i) = \mathrm{E}\left\{\boldsymbol{X}|q^{(2)}(\boldsymbol{X}) = i\right\} \tag{3.17}$$

for the second side decoder, and finally

$$y^{(0)}(i,j) = \mathrm{E}\left\{\boldsymbol{X}|q^{(1)}(\boldsymbol{X}) = i, q^{(2)}(\boldsymbol{X}) = j\right\} \tag{3.18}$$

for the central decoder.

Finally, Vaishampayan also describes an algorithm for designing the encoders and decoders used in MD scalar quantizers. In this work, we use a fixed central quantizer, so we examine only the design of the MDIA.

Compared to odd-even splitting, MDIA can generate more general descriptions, which may improve system performance. Let b denote the number of bits allocated for the central quantizer, and let b_m be the bit

allocation for the mth description. b_m may be any number of bits in the range $0 \leq b_m \leq b$. Odd-even separation can be considered an extreme case of MDIA, in which an LSF is alloted zero bits for one of its descriptions, and its full rate for the other. Consequently, any decoder which can decode MDIA indices can also decode odd-even separated LSFs.

The system proposed in this thesis generates two descriptions of $\boldsymbol{U}_l[n]$, denoted $\boldsymbol{U}_l^{(1)}[n]$ and $\boldsymbol{U}_l^{(2)}[n]$, using an IA matrix. Let $\mathcal{I}_l$ denote the IA matrix used for the lth LSF. Assume that the central quantizer output index is $i = \boldsymbol{U}_l[n]$. To encode this index, the MD encoder finds the position of i in the IA matrix, $\mathcal{I}_l$. The position of i along each of the M dimensions of $\mathcal{I}_l$ are the M multiple descriptions of i. In our system, $M = 2$, so the first description is the row of $\mathcal{I}_l$ which holds i, and the second description is the column which holds i:

$$\mathcal{I}_l[\boldsymbol{U}_l^{(1)}[n], \boldsymbol{U}_l^{(2)}[n]] = \boldsymbol{U}_l[n] \tag{3.19}$$

Let $\mathcal{I}_l^{(m)}(i)$ denote the mth description of i. Using this notation, the descriptions output by the MDIA encoder can also be expressed as:

$$\boldsymbol{U}_l^{(1)}[n] = \mathcal{I}_l^{(1)}(\boldsymbol{U}_l[n]) \tag{3.20}$$

$$\boldsymbol{U}_l^{(2)}[n] = \mathcal{I}_l^{(2)}(\boldsymbol{U}_l[n]) \tag{3.21}$$

The descriptions of all LSFs in frame n are combined to form two packets, $\boldsymbol{U}^{(1)}[n]$ and $\boldsymbol{U}^{(2)}[n]$, as in (3.3). We will describe procedures for designing MDIAs later, in §4.5.

3.4.3 Diverse Encoders

In the second type of MD coding scheme we consider, the MD coder uses the output of two different speech encoders to form the descrip-

tions. In [33], Hardman et al. describe an approach for MD coding over a packet network using a combination of an ADPCM encoder and a LPC encoder. Consider the nth speech segment. The ADPCM encoder's output for this segment is placed in packet n, along with the LPC encoder's output for segment $n-1$ (or $n-2$, if the encoder is intended for a high packet loss rate). When no losses occur, the decoder uses only the ADPCM representation. If the nth packet is lost, then the decoder uses the LPC representation of this segment, which is stored in the next packet. This decoder depends on the use of a decoding buffer, so that time spent waiting for the LPC encoded representation of the speech segment is imperceptible.

In [34], Lee describes different MD coding approaches for PCM, ADPCM and CELP-coded speech over a packet network. The techniques described combine the output of two similar encoders to improve the decoder's performance when both descriptions are received. For PCM, the proposed system uses a system with identical codebooks for each channel, and considers the mid-point between codebook entries as possible reproduction values for the central decoder. If the input is closest to a mid-point, one MD quantizer outputs the lower value closest to the mid-point, and the other quantizer outputs the next higher value. For ADPCM, one MD encoder quantizes as usual, and the other chooses a codebook entry which results in a quantizer error with the opposite polarity of error compared to the first MD encoder. For CELP, the author describes a system in which the two MD encoders are forced to choose different excitation codebook entries.

In [35], Zhong and Juang describe an approach for MD-coded speech

over erasure channels, using diverse encoders. Their approach begins with the design of a suitable side encoder, which meets the system's requirements, and attempts to combine the resulting side encoders in a beneficial manner. For waveform coding, the authors suggest using a different vector quantizer for each MD channel, or using the same VQ with different time offsets. The authors also suggest encoding both the original signal minus the previous encoder's quantizer error, so that the quantizer error is decreased if all descriptions are received. This final encoder performs better than the others described by the authors.

In [36], Singh and Ortega describe one approach for MD coding for a predictive coder over an erasure channel. Their approach uses unbalanced MD encoders, such that $R_1 > R_2$. The system is designed so that $D_0 = D_1$, that is, the first description alone produces the highest quality output attainable with the system. The two encoders operate with independent DPCM encoders, which encode the same output. When a description is lost, the decoder uses the other description for that sample to determine which quantizer levels could have been output by the lost description's quantizer for that sample. By looking ahead a number of samples, the decoder finds the most likely path through the predictive quantizer outputs.

3.4.4 Correlating Transform

In a MD system based on correlating transforms [38], two or more input variables are combined using a correlating transform to form two or more output variables. The outputs are quantized at a higher rate than the original inputs. The added correlation makes it possible to estimate the

input variables using a subset of the output variables.

In [39], Arean et al. describe a MD system for the Bell Labs' perceptual audio coder (PAC) over a packet network. The authors apply a correlating transform to the transform coefficients which are output by the PAC. This correlating transform maps pairs of transform coefficients into pairs of descriptions. The correlating transform has a parameter α which can be used to trade-off between the bit rate and the added redundancy. The authors found that simple interpolation works well for low packet loss probabilities, but the correlating transform works much better for a 50 % packet loss probability.

3.5 Optimal Decoding

An optimal decoder will minimize the expected distortion between the encoder's input, $\boldsymbol{X}_l[n]$, and the decoder's output, $\boldsymbol{Y}_l[n]$. The true value of $\boldsymbol{X}_l[n]$ is unknown to the decoder, so the decoder's output is based on its observation, $(\boldsymbol{V}[1], \ldots, \boldsymbol{V}[N])$, for N received frames. Using a distortion measure $d()$, an optimal decoder will output:

$$\boldsymbol{Y}_l[n] = \underset{y}{\operatorname{argmin}} \{d(y, \boldsymbol{X}) \mid \boldsymbol{V}[1], \ldots, \boldsymbol{V}[N]\} \tag{3.22}$$

Where $\operatorname{argmin}_y \{f(y)\}$ is the value y which minimizes the function $f()$. In this thesis, we use a squared-error distortion measure:

$$d(a, b) = (a - b)^2 \tag{3.23}$$

We use the squared error because it is the distortion measure used in the FS-1016 encoder for LSFs, and because it is easy to work with. For the

squared-error distortion measure, the solution for (3.22) is given by the conditional expected value of $\boldsymbol{X}_l$ [17]:

$$\boldsymbol{Y}_l[n] = \underset{y}{\operatorname{argmin}} \, \mathrm{E}\left\{(\boldsymbol{X}_l[n] - y)^2 \mid \boldsymbol{V}[1], \ldots, \boldsymbol{V}[N]\right\} \tag{3.24}$$

$$= \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{V}[1], \boldsymbol{V}[2], \ldots, \boldsymbol{V}[n], \boldsymbol{V}[n+1], \ldots, \boldsymbol{V}[N]\right\} \tag{3.25}$$

It is impractical to implement this decoder directly [44]. In FS-1016, a single vector $\boldsymbol{U}[n]$ is quantized using 34 bits. An estimate of $\boldsymbol{X}_l[n]$ based on even a single received frame, $\boldsymbol{V}[n]$, would require a lookup table with $2^{34} = 17\,179\,869\,184$ entries. An estimate based on N frames would require a codebook with $2^{34 \cdot N}$ entries. Because this decoder is impractical, we must consider ways to decrease the problem size. In the following sections, we will consider decoders which approximate the optimal decoder. The decoders are presented approximately in order of increasing complexity.

3.6 Estimation of Missing LSFs

In this section, we describe techniques which have been used previously to estimate missing LSFs based on other received LSFs. We implement these techniques at the receiver, in order to estimate LSFs which were lost over the packet network. In particular, we consider the problem of estimating odd or even numbered LSFs which were transmitted using odd-even splitting (§3.4.1) and subsequently lost.

3.6.1 Memoryless Decoders

We call a decoder *memoryless* if its output $\boldsymbol{Y}_l[n]$ depends only on its observation for this LSF, $\boldsymbol{V}_l[n]$, and on a priori information which is stored in the decoder. A memoryless decoder does not take advantage of the correlation between LSFs. Such a decoder approximates (3.25) by using:

$$\boldsymbol{Y}_l[n] = \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{V}_l[n]\right\} \tag{3.26}$$

We saw in Chapter 2.3 that there is in fact dependence between adjacent LSFs. In later sections, we will consider decoders which take advantage of such dependence. We consider memoryless decoders as a baseline on decoder performance. Through comparison with later decoders, this baseline will allow us to determine the importance of exploiting the dependence between LSFs when decoding.

Memoryless MD decoders have been used in speech applications previously. A memoryless MDIA decoder is used to decode multiple descriptions of a simulated speech waveform in [42]. In [41], a memoryless decoder is used to estimate missing split-VQ LSFs over a packet loss network, and to decode over a bit-error channel.

We will consider two known techniques for memoryless decoding. The first uses the mean value of $\boldsymbol{X}_l$ to recover missing LSFs. The second, described in §3.7, implements the minimum squared error MDIA decoder described by Vaishampayan in [37] and summarized here in §3.4.2. We compare these two decoders to examine the importance of using index assignments other than odd-even splitting.

Mean Value

The mean value decoder, which we call L-0, can only use a description $\boldsymbol{V}_l^{(m)}[n]$ if $\boldsymbol{V}_l^{(m)}[n] = \boldsymbol{U}_l[n]$. In other words, a description must describe $\boldsymbol{U}_l[n]$ exactly, or else it cannot be used. Odd-even separation, described in §3.4.1, generates descriptions which fit this requirement. With odd-even separation, this decoder receives two descriptions for each frame. However, for any one LSF in a frame, this is a single-description decoder. This decoder serves as a basis of comparison, to determine whether using multiple descriptions for an LSF can improve performance.

If the lth LSF in a frame is correctly received, this decoder uses the codebook C_l to find the LSF's value. If the LSF is missing, the average value of LSF l is used in pace of the missing LSF:

$$\boldsymbol{Y}_l[n] = \begin{cases} C_l\left[\boldsymbol{U}_l[n]\right] & \text{if } \boldsymbol{V}_l^{(m)}[n] = \boldsymbol{U}_l[n] \text{ for any } m \in \{1, \ldots, M\} \\ \mathrm{E}\left\{\boldsymbol{X}_l\right\} & \text{otherwise} \end{cases} \tag{3.27}$$

The LSF's average value is estimated by calculating its mean value in the training data set:

$$\mathrm{E}\left\{\boldsymbol{X}_l\right\} \approx \frac{1}{\dot{N}} \sum_{n=1}^{\dot{N}} \dot{\boldsymbol{X}}_l[n] \tag{3.28}$$

The estimated average values are presented in Table 2.2 on page 26.

3.6.2 Linear Decoders

In this section, we consider decoders which use a linear equation to model the dependence between LSFs. Unlike the memoryless decoders in the previous section, linear decoders can exploit the correlation between LSFs to conceal losses. The simplest linear decoders conceal losses

by repeating the previous frame, or by interpolating between adjacent frames. More sophisticated decoders use vector linear prediction (VLP) [45, 17] to conceal losses, by using a linear combination of all the LSFs in adjacent frames.

Some of the notation and procedure we use is common to all of the linear decoders. All assume that the multiple descriptions were generated using odd-even separation of LSFs (§3.4.1). Let $\boldsymbol{W}_l[n]$ denote the quantizer's reproduction value for $\boldsymbol{V}_l[n]$. We let $\boldsymbol{W}_l[n] = \varnothing$ if the decoder did not receive its value:

$$\boldsymbol{W}_l[n] \triangleq \begin{cases} \mathcal{C}_l[\boldsymbol{U}_l[n]] & \text{if } \boldsymbol{V}_l^{(m)}[n] = \boldsymbol{U}_l[n] \text{ for some } m \\ \varnothing & \text{otherwise} \end{cases} \tag{3.29}$$

The linear predictors work best on variables with zero mean value [17], so we subtract the LSFs' mean values before decoding. We denote a variable after subtracting its mean value by using a cup $\breve{}$ over the variable. For example, $\breve{\boldsymbol{X}}_l[n] = \boldsymbol{X}_l[n] - \mathrm{E}\{X_l\}$ is the encoder's output after we subtract its mean value. We let $\breve{\boldsymbol{W}}$ denote the received LSFs after their mean value has been subtracted:

$$\breve{\boldsymbol{W}}_l[n] \triangleq \begin{cases} \varnothing & \text{if } \boldsymbol{W}_l[n] = \varnothing \\ \boldsymbol{W} - \mathrm{E}\{\boldsymbol{X}_l\} & \text{otherwise} \end{cases} \tag{3.30}$$

where we estimate $\mathrm{E}\{\boldsymbol{X}_l\}$ as in (3.28). The linear decoders use $\breve{\boldsymbol{W}}$ to calculate $\breve{\boldsymbol{Y}}$, which is an estimate of $\breve{\boldsymbol{X}}$. Finally we add the LSFs' mean values to form $\boldsymbol{Y}_l[n] = \breve{\boldsymbol{Y}}_l[n] + \mathrm{E}\{\boldsymbol{X}_l\}$, which is passed to the CELP decoder. A summary of this notation is presented in Figure 3.3.

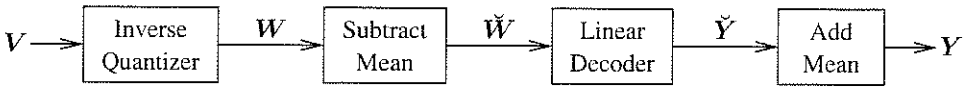

Figure 3.3: Notation used with linear decoders.

Linear Prediction using Past LSFs

This decoder estimates a missing LSF $\boldsymbol{Y}_l[n]$ using a linear combination of L past frames:

$$\check{\boldsymbol{Y}}_l[n] = \begin{cases} \sum_{i=1}^{L} \boldsymbol{a}_l[i]\check{\boldsymbol{Y}}_l[n-i] & \text{if } \check{\boldsymbol{W}}_l[n] = \varnothing \\ \check{\boldsymbol{W}}_l[n] & \text{otherwise} \end{cases} \tag{3.31}$$

Here we denote the missing LSF by $\boldsymbol{M}_l[n] = \check{\boldsymbol{X}}_l[n]$, and the LSFs used to estimate it are

$$\boldsymbol{P}_l[n] = \left(\check{\boldsymbol{X}}_l[n-1], \check{\boldsymbol{X}}_l[n-2], \ldots, \check{\boldsymbol{X}}_l[n-L]\right)^t \tag{3.32}$$

A special case of this decoder is *repetition*, for which $L = 1$, and $\boldsymbol{a}_l[0] = 1$ for all l. Repetition repeats the most recent correctly received LSF to conceal losses. Other decoders use different fixed constants for each LSF. For example, GSM-AMR [46] uses $\boldsymbol{a}_l[0] = 0.9$ for all l, so that missing LSFs will approach their mean value, instead of simply repeating the past frame. Similarly, the loss concealment in EIA-96-C [47] uses $\boldsymbol{a}_l[0] < 1$, and is designed so that the LSFs tend toward equally spaced values, which corresponds to a flat frequency response.

For the decoder we call L-a, the coefficients $\boldsymbol{a}_l[i]$ are designed to minimise the mean squared prediction error (3.23), using the same approach described for linear prediction analysis of the speech signal in §2.3.1. As in [17], we calculate the prediction coefficients using:

$$\boldsymbol{a}_l = \mathrm{E}\left\{\boldsymbol{M}_l\boldsymbol{P}_l^t\right\} R_{P_l}^{-1} \tag{3.33}$$

Decoder	Missing, $\boldsymbol{M}_l[n]$	Present, $\boldsymbol{P}_l[n]$
L-a	$\breve{\boldsymbol{W}}_l[n]$	$\breve{\boldsymbol{W}}_l[n-1]$
L-b	$\breve{\boldsymbol{W}}_l[n]$	$\breve{\boldsymbol{W}}_{l-1}[n], \breve{\boldsymbol{W}}_{l+1}[n]$
L-ab	$\breve{\boldsymbol{W}}_l[n]$	$\breve{\boldsymbol{W}}_l[n-1], \breve{\boldsymbol{W}}_{l-1}[n], \breve{\boldsymbol{W}}_{l+1}[n]$
L-ac	$\breve{\boldsymbol{W}}_l[n]$	$\breve{\boldsymbol{W}}_l[n-1], \breve{\boldsymbol{W}}_l[n+1]$
L-abc	$\breve{\boldsymbol{W}}_l[n]$	$\breve{\boldsymbol{W}}_l[n-1], \breve{\boldsymbol{W}}_{l-1}[n], \breve{\boldsymbol{W}}_{l+1}[n], \breve{\boldsymbol{W}}_l[n+1]$

Table 3.1: Decoders based on LP from neighbouring LSFs.

where R is the discrete correlation function, and the superscript $^{-1}$ denotes matrix inversion. The correlation matrices are estimated using our training data:

$$\mathrm{E}\left\{\boldsymbol{M}_l\boldsymbol{P}_l^t\right\} \approx \frac{1}{\dot{N}-L+1}\sum_{n=L+1}^{\dot{N}} \dot{\boldsymbol{M}}_l[n]\left(\dot{\boldsymbol{P}}_l[n]\right)^t \tag{3.34}$$

$$R_{\boldsymbol{P}_l} \approx \frac{1}{\dot{N}-L+1}\sum_{n=L+1}^{\dot{N}} \dot{\boldsymbol{P}}_l[n]\left(\dot{\boldsymbol{P}}_l[n]\right)^t \tag{3.35}$$

Linear Prediction using Adjacent LSFs

Now we consider a decoder which uses an LSF's four nearest neighbours to estimate its value. A missing LSF $\breve{\boldsymbol{Y}}_l[n]$ is estimated by using:

$$\breve{\boldsymbol{Y}}_l[n] = a_l\breve{\boldsymbol{Y}}_l[n-1] + b_l^{(-)}\breve{\boldsymbol{W}}_{l-1}[n] + b_l^{(+)}\breve{\boldsymbol{W}}_{l+1}[n] + c_l\breve{\boldsymbol{W}}_l[n+1] \tag{3.36}$$

We use these specific four neighbours because we saw in §2.3.5 that they usually have the strongest correlation with the missing LSF.

A special case of this decoder is *interpolation*, for which $a_l = c_l = 0.5$ and $b_l^{(-)} = b_l^{(+)} = 0$ for all l. This interpolates between the last correctly received LSF and one future LSF to conceal errors. [48] examines interpolation between received frames, compared to scalar prediction of missing LSFs. Interpolation between frames is commonly used for LSFs, for ex-

ample, it is used to estimate the LSFs for subframes in the FS-1016 [12] and GSM-AMR [22] speech coders.

In addition to interpolation, we consider using prediction coefficients a_l, $b_l^{(-)}$, $b_l^{(+)}$, and c_l designed to minimize the squared prediction error. These coefficients depend on the LSF l being estimated. Variations on this decoder eliminate some of these prediction terms. For example, a decoder without an added delay frame drops the c_l term:

$$\breve{\boldsymbol{Y}}_l[n] = a_l \breve{\boldsymbol{Y}}_l[n-1] + b_l^{(-)} \breve{\boldsymbol{Y}}_{l-1}[n] + b_l^{(+)} \breve{\boldsymbol{Y}}_{l+1}[n] \tag{3.37}$$

The prediction coefficients must be re-designed for each variation. We will refer to these decoders using the prefix "L-", followed by the letters corresponding to the prediction coefficients used by the decoder. For example, a predictor which uses $a_l \breve{\boldsymbol{Y}}_l[n-1]$, $b_l^{(-)} \breve{\boldsymbol{W}}_{l-1}[n]$ and $b_l^{(+)} \breve{\boldsymbol{W}}_{l+1}[n]$ will be called L-ab. All such decoders we consider are listed in Table 3.1, and are illustrated in Figure 3.4. The *missing* column lists the missing LSF which the decoder estimates, and the *present* column lists the LSFs used to estimate the missing LSF.

A decoder which uses a_l, $b_l^{(-)}$, $b_l^{(+)}$, and c_l (which we call L-abc) is appropriate if all of the adjacent frames are received, but it would depend on undefined values if an adjacent frame is damaged. To avoid this, the decoders are designed to use other predictors when adjacent LSFs are lost. The L-abc decoder depends on four decoders: L-a, L-ab, L-ac, and L-abc. The decoder is selected based on which adjacent LSFs are present (or correctly received) as listed in Table 3.1. The other linear decoders similarly depend on simpler decoders when adjacent LSFs are lost. To simplify the implementation, the decoders use their estimate for $\breve{\boldsymbol{X}}_l[n-1]$,

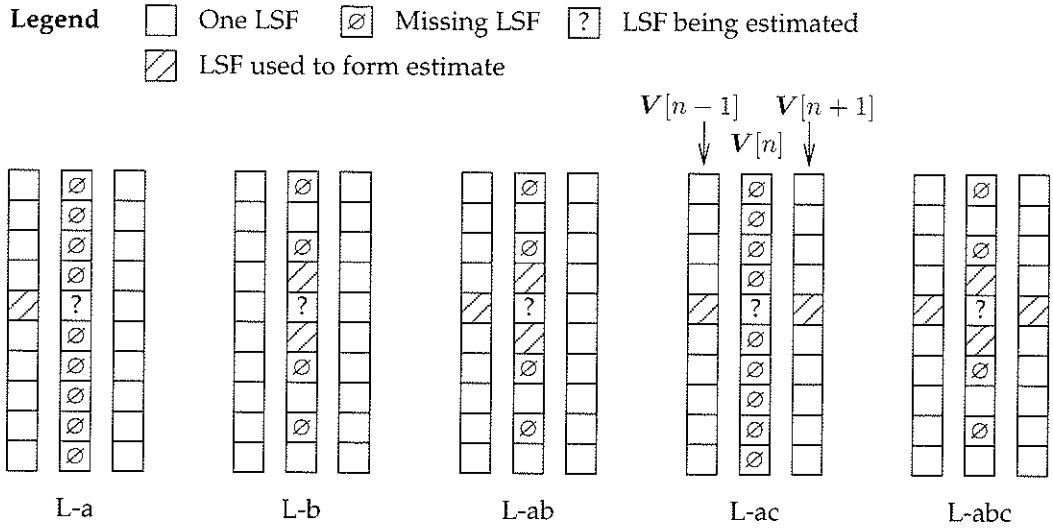

Figure 3.4: Decoders based on LP.

$\breve{\boldsymbol{Y}}_l[n-1]$, even if it was not correctly received. Thus a vector for frame $n-1$ is always available to estimate frame n, but the decoder's error propagates forward until a frame is correctly received.

Now we consider the procedure for designing predictor coefficients. The predictors are described in terms of the missing and present LSFs, as listed in Table 3.1. Consider the decoder L-abc as an example. LSFs are excluded if they are undefined, for example, LSF $\breve{\boldsymbol{W}}_{l-1}[n]$ would not be used to estimate $\breve{\boldsymbol{W}}_l[n]$ for $l=1$. Here we have:

$$\boldsymbol{M}_l[n] = \breve{\boldsymbol{X}}_l[n] \tag{3.38}$$

$$\boldsymbol{P}_l[n] = \begin{cases} \left(\breve{\boldsymbol{X}}_l[n-1], \breve{\boldsymbol{X}}_{l+1}[n], \breve{\boldsymbol{X}}_l[n+1]\right)^t & \text{for } l=1 \\ \left(\breve{\boldsymbol{X}}_l[n-1], \breve{\boldsymbol{X}}_{l-1}[n], \breve{\boldsymbol{X}}_{l+1}[n], \breve{\boldsymbol{X}}_l[n+1]\right)^t & \text{for } l=2,\ldots,9 \\ \left(\breve{\boldsymbol{X}}_l[n-1], \breve{\boldsymbol{X}}_{l-1}[n], \breve{\boldsymbol{X}}_l[n+1]\right)^t & \text{for } l=10 \end{cases} \tag{3.39}$$

We calculate [17]:

$$\boldsymbol{A}_l = \mathrm{E}\left\{\boldsymbol{M}\boldsymbol{P}^t\right\} R_{P_l}^{-1} \tag{3.40}$$

in which $\mathrm{E}\{\boldsymbol{M}\boldsymbol{P}^t\}$ is calculated as in (3.34), and $R_{\boldsymbol{P}_l}$ is calculated as in (3.35). Finally we extract the individual prediction coefficients from $\boldsymbol{A}_l$. For $2 \leq l \leq 9$:

$$(a_l, b_l^{(-)}, b_l^{(+)}, c_l) = \boldsymbol{A}_l \tag{3.41}$$

For $l = 1$ we remove the $b_l^{(-)}$ term, and for $l = 10$, we remove the $b_l^{(+)}$ term.

Vector Linear Prediction using Adjacent Frames

Now we consider decoders based on VLP, which use all the received LSFs in adjacent frames to estimate a missing LSF. The VLP-based decoders we use can be described in general as:

$$\check{\boldsymbol{Y}}_l[n] = \boldsymbol{a}_l\check{\boldsymbol{Y}}[n-1] + \boldsymbol{b}_l\check{\boldsymbol{Y}}[n] + \boldsymbol{c}_l\check{\boldsymbol{Y}}[n+1] \tag{3.42}$$

The vectors $\boldsymbol{a}_l$, $\boldsymbol{b}_l$, and $\boldsymbol{c}_l$ are designed to minimise the squared prediction error. They must be designed separately for each missing LSF l.

The decoders described in the previous section used only the LSFs immediately adjacent to a missing LSF when decoding. As we saw in §2.3.5, such neighbours tend to have the strongest correlation with the missing LSF. By comparing a VLP-based decoder to the decoders in §3.6.2, we will be able to find the importance of the less strongly correlated LSFs in a linear decoder.

In [49], VLP is used to predict an frame of LSFs using past frames, and in [50], VLP is used to estimate missing LSFs using multiple past frames, or from past and future frames. [28] uses VLP to estimate missing LSFs by using received LSFs in the same frame. In this section, we combine the

Decoder	Missing, $\boldsymbol{M}[n]$	Present, $\boldsymbol{P}[n]$
VLP-a	$\breve{\boldsymbol{W}}[n]$	$\breve{\boldsymbol{W}}[n-1]$
VLP-b	$\breve{\boldsymbol{W}}^{(o)}[n]$	$\breve{\boldsymbol{W}}^{(e)}[n]$
	$\breve{\boldsymbol{W}}^{(e)}[n]$	$\breve{\boldsymbol{W}}^{(o)}[n]$
VLP-ab	$\breve{\boldsymbol{W}}^{(o)}[n]$	$\breve{\boldsymbol{W}}[n-1], \breve{\boldsymbol{W}}^{(e)}[n]$
	$\breve{\boldsymbol{W}}^{(e)}[n]$	$\breve{\boldsymbol{W}}[n-1], \breve{\boldsymbol{W}}^{(o)}[n]$
VLP-ac	$\breve{\boldsymbol{W}}[n]$	$\breve{\boldsymbol{W}}[n-1], \breve{\boldsymbol{W}}[n+1]$
VLP-abc	$\breve{\boldsymbol{W}}^{(o)}[n]$	$\breve{\boldsymbol{W}}[n-1], \breve{\boldsymbol{W}}^{(e)}[n], \breve{\boldsymbol{W}}[n+1]$
	$\breve{\boldsymbol{W}}^{(e)}[n]$	$\breve{\boldsymbol{W}}[n-1], \breve{\boldsymbol{W}}^{(o)}[n], \breve{\boldsymbol{W}}[n+1]$

Table 3.2: Decoders based on VLP.

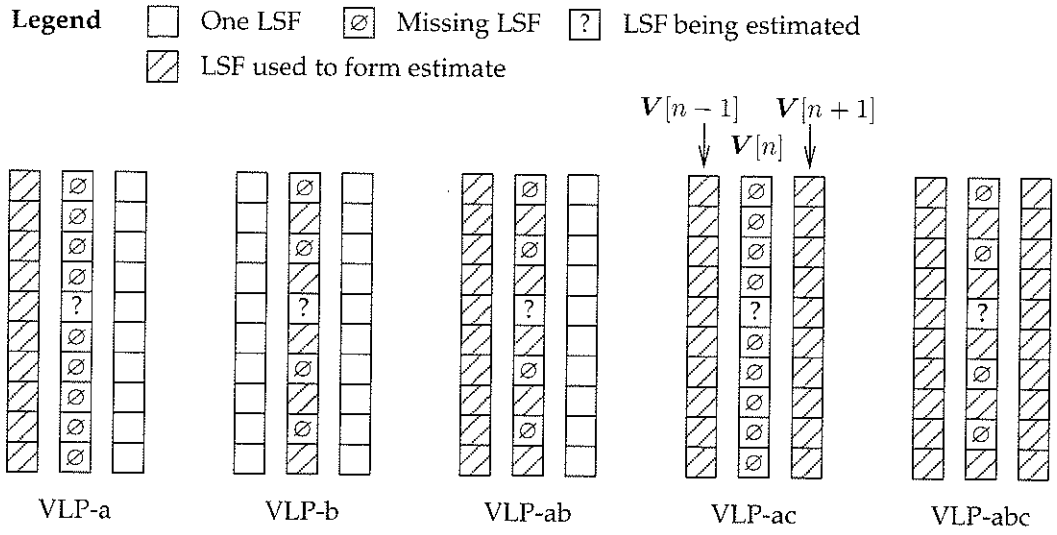

Figure 3.5: Decoders based on VLP.

two approaches to estimate a missing LSF in frame f using a combination of the received LSFs in frames $f-1$, f, and $f+1$.

We will refer to VLP-based decoders using the prefix "VLP-", followed by the letters corresponding to which prediction coefficients are used. For example, a predictor which uses vectors $\boldsymbol{a}\breve{\boldsymbol{Y}}[n-1]$ and $\boldsymbol{b}\breve{\boldsymbol{Y}}[n]$ and does not use $\boldsymbol{c}$ will be called VLP-ab. These decoders are summarized in Table 3.2, and illustrated in Figure 3.5. Similar to the previous section, the *missing* column corresponds to the vector of missing LSFs

which the decoder estimates, and the *present* column corresponds to the LSFs used to estimate the missing LSFs.

Let us consider the decoder VLP-ab, missing the odd LSFs $\check{\boldsymbol{X}}^{(o)}[n]$, as an example. Here we have:

$$\boldsymbol{M}[n] = \check{\boldsymbol{X}}^{(o)}[n] \tag{3.43}$$

$$\boldsymbol{P}[n] = \left((\check{\boldsymbol{X}}[n-1])^t, (\check{\boldsymbol{X}}^{(e)}[n])^t\right)^t \tag{3.44}$$

where the present vectors are combined into one column vector. We find the prediction coefficient matrix $\boldsymbol{A}$ as in (3.40). The prediction coefficient vectors for use in (3.42) can be extracted from $\boldsymbol{A}$:

$$\begin{pmatrix} (\boldsymbol{a}_1)_{1\times 10} & (\boldsymbol{b}_1)_{1\times 5} \\ (\boldsymbol{a}_3)_{1\times 10} & (\boldsymbol{b}_3)_{1\times 5} \\ \vdots & \vdots \\ (\boldsymbol{a}_9)_{1\times 10} & (\boldsymbol{b}_9)_{1\times 5} \end{pmatrix} = \boldsymbol{A}_{5\times 15} \tag{3.45}$$

As in §3.6.2, the VLP decoders use their estimate for $\check{\boldsymbol{X}}_l[n-1]$, $\check{\boldsymbol{Y}}_l[n-1]$, even if it was not correctly received. Thus a vector for frame $n-1$ is always available to estimate frame n, but the decoder's error propagates forward until a frame is correctly received.

A decoder which uses $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ (which we call VLP-abc) is appropriate if all of the adjacent frames are received, but it would depend on undefined values if an adjacent frame is damaged. The decoder must use other predictors to handle random losses. The decoder VLP-abc uses eight different vector predictors, depending on the combination of received descriptions. The eight predictors are $\boldsymbol{b}$ for missing odd or even LSFs, $\boldsymbol{ab}$ for $\boldsymbol{b}$ missing odd or even LSFs, $\boldsymbol{abc}$ for $\boldsymbol{b}$ missing odd or even LSFs, $\boldsymbol{ac}$, and $\boldsymbol{a}$. Table 3.3 lists the scenarios in which each VLP is used.

Received Descriptions $\boldsymbol{V}[n]$	$\boldsymbol{V}[n+1]$	Use Predictor...
$^{(o)}$ or $^{(e)}$	All	VLP-abc
$^{(o)}$ or $^{(e)}$	$^{(o)}$ or $^{(e)}$	VLP-abc, after filling $\boldsymbol{W}[n+1]$ using VLP-b
$^{(o)}$ or $^{(e)}$	None	VLP-ab
None	All	VLP-ac
None	$^{(o)}$ or $^{(e)}$	VLP-ac, after filling $\boldsymbol{W}[n+1]$ using VLP-b
None	None	VLP-a

Table 3.3: VLP decoding scenarios.

The other decoders in this section similarly use simpler decoders depending on which combination of descriptions is received.

3.6.3 Gaussian Mixture Model-Based Decoder

The decoder described in this section, which we call decoder NL-5-b, can use all of the received LSFs in a frame to estimate a lost LSF. Through comparison with the VLP-b decoder described in §3.6.2, this decoder will allow us to determine whether a non-linear model of the dependence within one LSF vector can improve system performance. This decoder is designed to operate on odd-even separated LSFs (§3.4.1), and its output is based on:

$$\boldsymbol{Y}^{(o)}[n] = \mathrm{E}\left\{\boldsymbol{X}^{(o)}[n] \mid \boldsymbol{X}^{(e)}[n]\right\} \tag{3.46}$$

when only the even LSFs are received, or

$$\boldsymbol{Y}^{(e)}[n] = \mathrm{E}\left\{\boldsymbol{X}^{(e)}[n] \mid \boldsymbol{X}^{(o)}[n]\right\} \tag{3.47}$$

when only the odd LSFs are received. If both descriptions are lost, then the decoder outputs the mean value as in §3.6.1. The decoder described in this section is based on a Gaussian mixture model (GMM) [53], which

is a technique for modelling probability distributions. The approach we use is from [28], which uses a GMM to model missing split vectors of LSFs.

A Gaussian (or normal) distribution of an L dimensional random variable $\boldsymbol{X}$ is given by [53]:

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.48}$$

$$= \frac{1}{(2\pi)^{L/2}} |\boldsymbol{\Sigma}|^{1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{3.49}$$

where $\boldsymbol{x}$ is the point at which the probability is evaluated, $\boldsymbol{\Sigma}$ is the covariance matrix of $\boldsymbol{X}$, and $\boldsymbol{\mu}$ is the mean value of $\boldsymbol{X}$. In this thesis, $L = 10$, corresponding to the number of LSFs.

A GMM combines two or more Gaussians to model a more complicated distribution. Let c denote the number of Gaussians in the model. The ith Gaussian has its own mean vector, $\boldsymbol{\mu}_i$, and its own covariance matrix, $\boldsymbol{\Sigma}_i$. The Gaussians are combined by adding them together, after weighing each Gaussian by a probability $\Pr(\omega_i)$. The GMM distribution is given by:

$$p(\boldsymbol{x}) = \sum_{i=1}^{c} \Pr(\omega_i) \frac{1}{(2\pi)^{L/2}} |\boldsymbol{\Sigma}_i|^{1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right) \tag{3.50}$$

In order to apply the GMM, we must first determine the model parameters, $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $\Pr(\omega_i)$. We use the expectation maximization (EM) algorithm to estimate these parameters, as described in [53].

Once we have determined the model parameters, the GMM can be used to estimate the missing LSFs. The LSFs are divided into odd and even descriptions, denoted by $\boldsymbol{X}^{(o)}$ and $\boldsymbol{X}^{(e)}$, respectively. Assume that one of the descriptions for frame n is missing. Let $\boldsymbol{X}^{(p)}[n]$ denote the

present description, and let $\boldsymbol{X}^{(\imath n)}[n]$ denote the missing description. For example, if $\boldsymbol{X}^{(o)}$ alone is received, then $\boldsymbol{X}^{(p)}[n] = \boldsymbol{X}^{(o)}[n]$, and $\boldsymbol{X}^{(m)}[n] = \boldsymbol{X}^{(e)}[n]$. The missing LSFs are estimated using:

$$\boldsymbol{Y}^{(m)}[n] = \mathrm{E}\left\{\boldsymbol{X}^{(m)}[n] \mid \hat{\boldsymbol{X}}^{(p)}[n]\right\} \tag{3.51}$$

We find this expected value by using the procedure described in [28]. We begin by rearranging the mean vectors and covariance matrices according to the missing and present LSFs:

$$\boldsymbol{\mu}_i' = \begin{pmatrix} \boldsymbol{\mu}_i^{(m)} \\ \boldsymbol{\mu}_i^{(p)} \end{pmatrix} \tag{3.52}$$

$$\boldsymbol{\Sigma}_i' = \begin{pmatrix} \boldsymbol{\Sigma}_i^{(m,m)} & \boldsymbol{\Sigma}_i^{(m,p)} \\ \boldsymbol{\Sigma}_i^{(p,m)} & \boldsymbol{\Sigma}_i^{(p,p)} \end{pmatrix} \tag{3.53}$$

Next we calculate the conditional mean vectors, conditioned on the value of the received LSFs:

$$\boldsymbol{\mu}_i^{(m|p)} = \boldsymbol{\mu}_i^{(m)} + \boldsymbol{\Sigma}_i^{(m,p)}(\boldsymbol{\Sigma}_i^{(p,p)})^{-1}(\boldsymbol{X}^{(p)}[n] - \boldsymbol{\mu}_i^{(p)}) \tag{3.54}$$

We calculate the posterior cluster probabilities, as defined in [28]:

$$\Pr(\omega_i)^{(m|p)} = \frac{\Pr(\omega_i)\,\mathcal{N}(\boldsymbol{X}^{(p)}[n], \boldsymbol{\mu}_i^{(p)}, \boldsymbol{\Sigma}_i^{(p)})}{\sum_{j=1}^{c}\Pr(\omega_j)\,\mathcal{N}(\boldsymbol{X}^{(p)}[n], \boldsymbol{\mu}_j^{(p)}, \boldsymbol{\Sigma}_j^{(p)})} \tag{3.55}$$

Finally, (3.54) and (3.55) are combined to estimate the missing LSFs:

$$\boldsymbol{Y}^{(m)}[n] = \sum_{i=1}^{c}\Pr(\omega_i)^{(m|p)}\,\boldsymbol{\mu}_i^{(m|p)} \tag{3.56}$$

3.7 Memoryless Multiple Description Decoder

Now we will consider a decoder which can take advantage of multiple descriptions for an LSF. We will call this decoder NL-0. This decoder

implements a minimum squared-error decoder for MD scalar quantizers, as described in [37]. Similar to decoder L-0, this decoder assumes that the current LSF is independent of all other received LSFs. Again, the decoder's output is based on:

$$\boldsymbol{Y}_l[n] = \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{V}_l[n]\right\} \tag{3.57}$$

We will begin by defining a function which indicates whether a central quantizer index could lead to the observed descriptions, $\boldsymbol{V}_l[n]$. Afterwards, we will express (3.57) in terms of such a function. We describe this decoder's behaviour in this manner as an introduction to the Markov-based decoders in Chapter 4.

Using the IA matrix, $\mathcal{I}_l$, the decoder can determine which central quantizer indices may have resulted in the observed descriptions, $\boldsymbol{V}_l[n]$. We use a function $b_l^{(m)}(\boldsymbol{U}_l[n], \boldsymbol{V}_l^{(m)}[n])$ to indicate whether it is possible for the decoder to receive the description $\boldsymbol{V}_l^{(m)}[n]$ if $\boldsymbol{U}_l[n]$ is the central quantizer index. We let $b^{(m)}() = 1$ if the description is possible, and $b^{(m)}() = 0$ if it is not. If the description is lost, $\boldsymbol{V}_l^{(m)}[n] = \varnothing$, then any central quantizer index is possible, so $b_l^{(m)}(i, \varnothing) = 1$ for all i. If the description is received, then $b()$ depends on the IA matrix:

$$b_l^{(m)}(i,j) \triangleq \begin{cases} 1 & \text{if } \mathcal{I}_l^{(m)}(i) = j \text{ or } j = \varnothing \\ 0 & \text{otherwise} \end{cases} \tag{3.58}$$

Next we consider combining the output of $b^{(m)}()$ from all descriptions to form $b()$. If $b^{(m)}(i, \boldsymbol{V}_l^{(m)}[n]) = 0$ for any m, then i is not a possible central quantizer index: $\boldsymbol{U}_l[n] \neq i$. Conversely, if $b_l^{(m)}(i, \boldsymbol{V}_l^{(m)}[n]) = 1$ for all m, then i may be the true quantizer index. To combine the output of $b^{(m)}(i, \boldsymbol{V}_l^{(m)}[n])$ for all m, we wish to find the intersection of the

possible central quantizer indices i indicated by each description. This corresponds to taking the product of $b^{(m)}()$ over all m:

$$b_l(i, \boldsymbol{V}_l[n]) \triangleq \prod_{m=1}^{M} b_l^{(m)}(i, \boldsymbol{V}_l^{(m)}[n]) \tag{3.59}$$

For convenience, we also define a vector-valued function $\boldsymbol{b}()$, which combines the value of $b_l(i, \boldsymbol{V}_l[n])$ for all i into a vector:

$$\boldsymbol{b}_l(\boldsymbol{V}_l[n]) \triangleq [b_l(1, \boldsymbol{V}_l[n]), b_l(2, \boldsymbol{V}_l[n]), \ldots, b_l(|\mathcal{C}_l|, \boldsymbol{V}_l[n])] \tag{3.60}$$

Let us consider an example of calculating $b()$. Again, consider the IA matrix in Figure 3.2. First we examine the case where both descriptions are lost, so $\boldsymbol{V}_l^{(1)}[n] = \boldsymbol{V}_l^{(2)}[n] = \varnothing$. Here the central quantizer may have output any index, so $b_l(i, \boldsymbol{V}_l[n]) = 1$ for all i:

$$\boldsymbol{b}_l(\boldsymbol{V}_l[n]) = \boldsymbol{b}_l^{(1)}(\boldsymbol{V}_l^{(1)}[n]) = \boldsymbol{b}_l^{(2)}(\boldsymbol{V}_l^{(2)}[n]) = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1] \tag{3.61}$$

Next we assume that the decoder receives the first description $\boldsymbol{V}_l^{(1)}[n] = 3$, and loses the second description as before, $\boldsymbol{V}_l^{(2)}[n] = \varnothing$. The third row of $\mathcal{I}$ corresponds to $\boldsymbol{U}_l[n] = 6$, 7, or 8, so:

$$\boldsymbol{b}_l^{(1)}(\boldsymbol{V}_l^{(1)}[n]) = [0, 0, 0, 0, 0, 1, 1, 1, 0, 0] \tag{3.62a}$$

$$\boldsymbol{b}_l^{(2)}(\boldsymbol{V}_l^{(2)}[n]) = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1] \tag{3.62b}$$

$$\boldsymbol{b}_l(\boldsymbol{V}_l[n]) = [0, 0, 0, 0, 0, 1, 1, 1, 0, 0] \tag{3.62c}$$

Here $\boldsymbol{b}_l(\boldsymbol{V}_l[n]) = 1$ in positions 6, 7, and 8, so the central quantizer must have output one of $\boldsymbol{U}_l[n] = 6$, 7, or 8.

Finally we assume that the decoder receive both $\boldsymbol{V}_l^{(1)}[n] = 3$, and $\boldsymbol{V}_l^{(2)}[n] = 2$. The second column of $\mathcal{I}$ corresponds to $\boldsymbol{U}_l[n] = 5$, 7, or

9. Here we have:

$$\boldsymbol{b}_l^{(1)}(\boldsymbol{V}_l^{(1)}[n]) = [0,0,0,0,0,1,1,1,0,0] \tag{3.63a}$$

$$\boldsymbol{b}_l^{(2)}(\boldsymbol{V}_l^{(2)}[n]) = [0,0,0,0,1,0,1,0,1,0] \tag{3.63b}$$

$$\boldsymbol{b}_l(\boldsymbol{V}_l[n]) = [0,0,0,0,0,0,1,0,0,0] \tag{3.63c}$$

The only 1 in $\boldsymbol{b}_l(\boldsymbol{V}_l[n])$ is in position 7. This corresponds to $\boldsymbol{U}_l[n] = 7$, so $\boldsymbol{U}_l[n] = 7$ must have been output by the central quantizer.

We will see shortly that the decoder output depends on $\Pr(\boldsymbol{V}_l[n] \mid \boldsymbol{U}_l[n] = i)$ which is directly related to the function $b()$:

$$\Pr(\boldsymbol{V}_l[n] \mid \boldsymbol{U}_l[n] = i) = c(\boldsymbol{V}_l[n]) b_l(\boldsymbol{U}_l[n] = i, \boldsymbol{V}_l[n]) \tag{3.64}$$

The value of c depends on the channel loss probability, and the number of descriptions received for $\boldsymbol{U}_l[n]$. In all the cases where we need (3.64), the $c()$ is common to both the numerator and denominator, so it will be simplified out of all further equations. Because c is never used, we are not concerned about its exact value.

Now we consider the decoder, which generates an estimate $\boldsymbol{Y}_l[n]$ of the transmitted LSF $\boldsymbol{X}_l[n]$, based on the observed descriptions $\boldsymbol{V}_l[n]$. Let $\chi = \{\boldsymbol{X}_l \mid \boldsymbol{X}_l[n] \rightarrow \boldsymbol{V}_l[n]\}$ be the set of possible values in $\boldsymbol{X}_l$ which could produce the observation $\boldsymbol{V}_l[n]$. An optimal decoder should find a y which minimizes the expected distortion between y and the points in χ:

$$\boldsymbol{Y}_l[n] = \underset{y}{\operatorname{argmin}} \, \mathrm{E}\{d(\boldsymbol{X}_l, y) \mid \boldsymbol{X}_l \rightarrow \boldsymbol{V}_l[n]\} \tag{3.65}$$

For a mean squared distortion measure (3.23), this is given by

$$\boldsymbol{Y}_l[n] = \mathrm{E}\{\boldsymbol{X}_l[n] \mid \boldsymbol{V}_l[n]\} \tag{3.66}$$

We expand (3.66) in terms of $\boldsymbol{U}_l[n]$ to obtain:

$$\boldsymbol{Y}_l[n] = \sum_{i=1}^{|\mathcal{C}_l|} \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{U}_l[n] = i\right\} \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[n]\right) \tag{3.67}$$

Consider the first half of (3.67). If the quantizer's codebook is optimal for the source, then [17]:

$$\mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{U}_l[n] = i\right\} = \mathcal{C}_l[i] \tag{3.68}$$

Next we consider the second half of (3.67). By Bayes' rule:

$$\Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[n]\right) = \frac{\Pr\left(\boldsymbol{V}_l[n] \mid \boldsymbol{U}_l[n] = i\right) \Pr\left(\boldsymbol{U}_l[n] = i\right)}{\Pr\left(\boldsymbol{V}_l[n]\right)} \tag{3.69}$$

$$= \frac{\Pr\left(\boldsymbol{V}_l[n] \mid \boldsymbol{U}_l[n] = i\right) \Pr\left(\boldsymbol{U}_l[n] = i\right)}{\sum_{k=1}^{|\mathcal{C}_l|} \Pr\left(\boldsymbol{V}_l[n] \mid \boldsymbol{U}_l[n] = k\right) \Pr\left(\boldsymbol{U}_l[n] = k\right)} \tag{3.70}$$

Now we consider simplifying this expression. We assume that the probability distribution of $\boldsymbol{U}_l$ is independent of n, so that $\Pr\left(\boldsymbol{U}_l[n] = i\right) = \Pr\left(\boldsymbol{U}_l = i\right)$. Also, we substitute $b_l(i, \boldsymbol{V}_l[n])$ for $\Pr\left(\boldsymbol{V}_l[n] \mid \boldsymbol{U}_l[n] = i\right)$, as described in (3.64):

$$\Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[n]\right) = \frac{\Pr\left(\boldsymbol{U}_l = i\right) b_l(i, \boldsymbol{V}_l[n])}{\sum_{k=1}^{|\mathcal{C}_l|} \Pr\left(\boldsymbol{U}_l = k\right) b_l(k, \boldsymbol{V}_l[n])} \tag{3.71}$$

Finally, we substitute (3.68) and (3.71) into (3.67), to obtain:

$$\boldsymbol{Y}_l[n] = \frac{\sum_{i=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \Pr\left(\boldsymbol{U}_l = i\right) b_l(i, \boldsymbol{V}_l[n])}{\sum_{i=1}^{|\mathcal{C}_l|} \Pr\left(\boldsymbol{U}_l = i\right) b_l(i, \boldsymbol{V}_l[n])} \tag{3.72}$$

which is the equation used by this decoder to evaluate $\boldsymbol{Y}_l[n]$.

Equation (3.72) depends on $\Pr\left(\boldsymbol{U}_l = i\right)$, the a priori probability that the encoder outputs index i. We estimate this using our training data:

$$\Pr\left(\boldsymbol{U}_l = i\right) \approx \frac{1}{\dot{N}} \sum_{n=1}^{\dot{N}} \mathrm{I}\left(\dot{\boldsymbol{U}}_l[n] = i\right) \tag{3.73}$$

where $I()$ is the unit indicator function, as defined in (2.43).

Chapter 4

Markov Model-Based Techniques

4.1 Introduction

In the previous chapter, we examined decoders which either exploit the dependence between LSFs, or can operate on the output of an MDIA encoder. In this chapter, we examine decoders which are capable of both. Later in this chapter, we will introduce an optimization procedure for designing IA matrices for such decoders. Throughout this chapter we will continue to use the system configuration and notation introduced in §3.3 and §3.7.

The MD decoders described in this chapter use a non-linear model of the dependence between LSFs. Specifically, this dependence is modelled using the transition probability from the quantizer index in one LSF to another. For example, a decoder's output may be based on the transition probability from quantizer level i in frame $n-1$, to quantizer level j in

frame n: $\Pr(\boldsymbol{U}_l[n] = j \mid \boldsymbol{U}_l[n-1] = i)$. While the linear decoders in the previous section use just one parameter for each LSF used to estimate a missing LSF, the decoders in this chapter require at least one parameter per quantizer level of an LSF. This allows these decoders to model more elaborate dependence between LSFs, but the decoders are more complex and have more parameters to design. Unlike linear decoders, these decoders can take advantage of multiple descriptions for an LSF, such as the descriptions generated using MDIA described in §3.4.2.

The decoders in this chapter are divided into *inter-frame* decoders, which use LSFs in adjacent frames to estimate a damaged LSF, and *intra-frame* decoders, which use LSFs in the same frame to estimate a damaged LSF. These decoders model the dependence between the quantized LSFs, $\boldsymbol{U}_l$, as a discrete-time Markov chain (see, for example, [51]). In a Markov process, the transition probability to the next state depends only on the current state, and not on the past. Assume that the output of the central quantizer for the lth LSF, $\boldsymbol{U}_l[n]$, follows a Markov chain. Then:

$$\Pr(\boldsymbol{U}_l[n] = i \mid \boldsymbol{U}_l[1], \ldots, \boldsymbol{U}_l[n-1]) = \Pr(\boldsymbol{U}_l[n] = i \mid \boldsymbol{U}_l[n-1]) \qquad (4.1)$$

The decoders in this section model the LSFs, encoder, and channel using a *hidden Markov model* [52]. The quantizer indices are the possible states of the system. For the lth LSF, the possible states are $\boldsymbol{U}_l \in \{1 \ldots |\mathcal{C}_l|\}$. The output of the central quantizer, $\boldsymbol{U}_l[n]$, is the true state of the system. The decoder observes the system's state through the received multiple descriptions, $\boldsymbol{V}_l[n]$. When no descriptions are lost, the decoder observes the true state of the system, $\boldsymbol{U}_l[n]$. If a description is lost, the true state of the system is hidden by the effect of channel losses. In this

case, the decoder uses the received descriptions to estimate the true state.

Decoders based on transition probabilities have been used previously for speech applications. In [13], intra- and inter-frame transition probabilities are used to help decode CELP-encoded LSFs with bit errors. In [41], intra- and inter-frame transition probabilities are used to decode split-VQ encoded LSFs, with either bit errors or erasures. A decoder for MD-encoded waveforms based on the forward-backward algorithm is presented in [42]. The decoders described in this chapter combine the intra- and inter-frame decoding of [13] and [41], with the MD decoding of [42], to decode MD-encoded LSFs.

4.2 Inter-Frame Decoding

The inter-frame decoders base their output on:

$$\boldsymbol{Y}_l[n] = \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n], \ldots, \boldsymbol{V}_l[N]\right\} \tag{4.2}$$

while assuming that $\boldsymbol{U}_l[n]$ follows a Markov chain. The Nth frame is the last received frame used by the decoder to help estimate the damaged frame n. Unlike the optimal decoder of §3.5, only the lth LSF is examined when making a decoding decision for $\boldsymbol{X}_l$. Our decoding approach is based on the recursive side decoder for multiple descriptions described in [42].

Using Past Frames Only

Assume that we are decoding the lth LSF in the nth frame. The decoder described in this section, which we call NL-a, assumes that the LSF being decoded depends on the descriptions received for the LSF $\boldsymbol{V}_l[n]$, and the

lth LSF in past frames, $\boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n-1]$. Thus this decoder's output is based on:

$$\boldsymbol{Y}_l[n] = \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n]\right\} \tag{4.3}$$

The operation of this decoder is based on the transition probability between system states, specifically, the transition between central quantizer indices. We define a state transition probability matrix, $\boldsymbol{A}_l[i,j]$ for LSF l:

$$\boldsymbol{A}_l[i,j] \triangleq \Pr\left(\boldsymbol{U}_l[n] = j \mid \boldsymbol{U}_l[n-1] = i\right) \tag{4.4}$$

This is the probability that the central encoder for LSF l outputs index j for frame n, given that it output index i in the previous frame, $n-1$. The calculation of $\boldsymbol{A}$ is described later in this section, in (4.15).

Now we consider the operation of the decoder, as in [42]. As in (3.67), we expand (4.3) in terms of the encoder's output, $\boldsymbol{U}_l[n]$:

$$\boldsymbol{Y}_l[n] = \sum_{i=1}^{|\mathcal{C}_l|} \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{U}_l[n] = i\right\} \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n]\right) \tag{4.5}$$

Using the relationship for optimal quantizers in (3.68), the conditional expected value of $\boldsymbol{X}_l[n]$ can be expressed in terms of the quantizer's codebook:

$$\boldsymbol{Y}_l[n] = \sum_{i=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n]\right) \tag{4.6}$$

This is an exact relationship for an optimal codebook, or an approximation otherwise.

As described in [42], the solution to $\Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n]\right)$ in (4.6) can be found using the forward part of the *forward-backward algorithm* for hidden Markov models [52]. The forward-backward algorithm uses a variable $\boldsymbol{\alpha}$ to find the joint probability of the encoded quantizer

index, and the received descriptions:

$$\boldsymbol{\alpha}_{l,n}[i] \triangleq \Pr\left(\boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n], \boldsymbol{U}_l[n] = i\right) \tag{4.7}$$

Now we consider the calculation of $\boldsymbol{\alpha}$. Using procedure described in §3.7, we find the possible central quantizer outputs, $\boldsymbol{b}_l$. $\boldsymbol{b}_l[i,j]$ is 1 if $\boldsymbol{U}_l = i$ could produce the observation $\boldsymbol{V}_l = j$, and is 0 otherwise. We initialise $\boldsymbol{\alpha}_{l,n}$ for the first frame, $n = 1$:

$$\boldsymbol{\alpha}_{l,1}[i] = \Pr\left(\boldsymbol{U}_l[1] = i\right) \Pr\left(\boldsymbol{V}_l[1] \mid \boldsymbol{U}_l[1] = i\right) \tag{4.8}$$

$$= \Pr\left(\boldsymbol{U}_l = i\right) cb_l(\boldsymbol{U}_l[1] = i, \boldsymbol{V}_l[1]) \tag{4.9}$$

For subsequent frames, $n > 1$, $\boldsymbol{\alpha}_{l,n}$ is defined in terms of its past values, using the transition probabilities and the new observations:

$$\boldsymbol{\alpha}_{l,n}[j] = \boldsymbol{b}_l(\boldsymbol{U}_l[n] = j, \boldsymbol{V}_l[n]) \sum_{i=1}^{|\mathcal{C}_l|} \boldsymbol{\alpha}_{l,n-1}[i] \boldsymbol{A}_l[i,j] \tag{4.10}$$

Now we consider calculating the decoder's output, as in (4.6), in terms of $\boldsymbol{\alpha}$. By using the definition of $\boldsymbol{\alpha}$ in (4.7):

$$\boldsymbol{Y}_l[n] = \sum_{i=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n]\right) \tag{4.11}$$

$$= \sum_{i=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \frac{\Pr\left(\boldsymbol{U}_l[n] = i, \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n]\right)}{\sum_{j=1}^{|\mathcal{C}_l|} \Pr\left(\boldsymbol{U}_l[n] = j, \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n]\right)} \tag{4.12}$$

$$= \frac{\sum_{i=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \boldsymbol{\alpha}_{l,n}[i]}{\sum_{j=1}^{|\mathcal{C}_l|} \boldsymbol{\alpha}_{l,n}[j]} \tag{4.13}$$

Equation (4.13) is used by the decoder NL-a to generate its output.

The decoder we just described depends on the inter-frame transition probabilities, $\boldsymbol{A}$. These are estimated using a histogram-based approach over the training data. Consider the lth LSF, $\boldsymbol{U}_l$. The transition count for this LSF, $\boldsymbol{T}_l$, is a $(|\mathcal{C}_l| \times |\mathcal{C}_l|)$ matrix. $\boldsymbol{T}_l[i,j]$ is the number of transitions from

quantizer level i in frame n to quantizer level j in frame $n+1$ observed in the training data:

$$\boldsymbol{T}_l[i,j] = \sum_{n=1}^{\dot{N}-1} \mathrm{I}\left(\dot{\boldsymbol{U}}_l[n] = i\right) \cdot \mathrm{I}\left(\dot{\boldsymbol{U}}_l[n+1] = j\right) \tag{4.14}$$

for $l = 1 \ldots 10$. $\dot{N}$ is the number of training data frames, and $\mathrm{I}()$ is the logical indicator function, defined in (2.43). The transition probabilities, $\boldsymbol{A}_l$, are estimated using the transition counts, $\boldsymbol{T}_l$:

$$\begin{aligned} \boldsymbol{A}_l[i,j] &\triangleq \Pr\left(\boldsymbol{U}_l[n] = j \mid \boldsymbol{U}_l[n-1] = i\right) && (4.15)\\ &= \frac{\Pr\left(\boldsymbol{U}_l[n] = j, \boldsymbol{U}_l[n-1] = i\right)}{\sum_{k=1}^{|\mathcal{C}_l|} \Pr\left(\boldsymbol{U}_l[n] = k, \boldsymbol{U}_l[n-1] = i\right)} && (4.16)\\ &\approx \frac{\boldsymbol{T}_l[i,j]}{\sum_{k=1}^{|\mathcal{C}_l|} \boldsymbol{T}_l[i,k]} && (4.17) \end{aligned}$$

4.2.1 Using Past Frames and One Future Frame

Decoder NL-a, described in the previous section, conceals a loss in frame n by using frames $\leq n$. In this section, we consider decoder NL-ac, which uses future frames as well:

$$\boldsymbol{Y}_l[n] = \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n], \ldots, \boldsymbol{V}_l[N]\right\} \tag{4.18}$$

where $N > n$ is the last frame used by the decoder to help estimate the damaged frame n. Once again, we assume that the quantized LSFs $\boldsymbol{U}_l[n]$ follow a Markov chain, and the form of this decoder is based on [42]. Using future frames when decoding may improve performance. However, waiting to receive future frames increases the decoding delay, which is an important factor in perceived system quality [7].

As before, we express the decoder's output (4.18) in terms of $\boldsymbol{U}_l[n]$:

$$\boldsymbol{Y}_l[n] = \sum_{i=1}^{|\mathcal{C}_l|} \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{U}_l[n] = i\right\} \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n], \ldots, \boldsymbol{V}_l[N]\right) \tag{4.19}$$

As in (3.68), $\mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{U}_l[n] = i\right\}$ can be expressed in terms of the quantizer's codebook:

$$\boldsymbol{Y}_l[n] \approx \sum_{i=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n], \ldots, \boldsymbol{V}_l[N]\right) \tag{4.20}$$

This is an approximation for a sub-optimal codebook.

The past frames are taken into account by the forward variable, $\boldsymbol{\alpha}$, which is calculated the same as in decoder NL-a. Whereas the forward decoder considers the transition from past observations to the current frame n, we now consider the transition from the state in frame n to the observations in frame $n+1$. As in the forward-backward algorithm in [52], this is accounted for by the backwards variable, which is denoted by $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}_{l,n}[i] \triangleq \Pr\left(\boldsymbol{V}_l[n+1], \ldots, \boldsymbol{V}_l[N] \mid \boldsymbol{U}_l[n] = i\right) \tag{4.21}$$

In our implementation, the decoder uses only a single future frame, to limit the delay added by the decoder. Thus we modify (4.21) so that $N = (n+1)$:

$$\boldsymbol{\beta}_{l,n}[i] = \Pr\left(\boldsymbol{V}_l[n+1] \mid \boldsymbol{U}_l[n] = i\right) \tag{4.22}$$

The calculation of $\boldsymbol{\beta}$ follows the backwards part of the forward-backward algorithm [52]. This calculation uses the same transition matrix $\boldsymbol{A}_l$ as the forward decoder in (4.4):

$$\boldsymbol{\beta}_{l,n}[i] = \sum_{j=1}^{|\mathcal{C}_l|} \boldsymbol{A}_l[i,j] b_l(\boldsymbol{U}_l[n+1] = j, \boldsymbol{V}_l[n+1]) \tag{4.23}$$

The forward probabilities, α, and the backwards probabilities, β, are combined to estimate the probability that $\boldsymbol{U}_l[n] = i$, based on the full sequence of received descriptions. This is denoted by γ:

$$\gamma_{l,n}[i] \triangleq \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[N]\right) \tag{4.24}$$

$$= \frac{\alpha_{l,n}[i]\beta_{l,n}[i]}{\sum_{j=1}^{|\mathcal{C}_l|} \alpha_{l,n}[j]\beta_{l,n}[j]} \tag{4.25}$$

for all i.

Similar to the forward decoder, once we have calculated γ, the decoder calculates and outputs $\boldsymbol{Y}$, which is the estimated value of $\boldsymbol{X}$:

$$\boldsymbol{Y}_l[n] = \sum_{i=1}^{|\mathcal{C}_l|} C_l[i] \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[n], \ldots, \boldsymbol{V}_l[N]\right) \tag{4.26}$$

$$= \sum_{i=1}^{|\mathcal{C}_l|} C_l[i]\gamma_{l,n}[i] \tag{4.27}$$

4.3 Decoding Based on a Single Frame

In the previous section, we considered decoding based on the dependence between frames. Now we consider decoding based on a the dependence within a single frame. Unlike the optimal decoder of §3.5, only the nth frame is examined when estimating $\boldsymbol{X}_l[n]$:

$$\boldsymbol{Y}_l[n] = \mathrm{E}\left\{\boldsymbol{X}_l[n] \mid \boldsymbol{V}_1[n], \ldots, \boldsymbol{V}_{10}[n]\right\} \tag{4.28}$$

The decoder described in this section, which we call decoder NL-b, assumes that the LSFs in a frame are independent of all other received frames. Its decoding procedure is similar to decoder NL-ac. We model the dependence between LSFs using a Markov chain, as in the inter-frame decoder. However, in this decoder, the transition probabilities are considered within a frame, rather than between frames. Here the transition

probability matrix is based on:

$$\boldsymbol{A}'_l[i,j] \triangleq \Pr\left(\boldsymbol{U}_{l+1}[f] = j \mid \boldsymbol{U}_l[f] = i\right) \tag{4.29}$$

$\boldsymbol{A}'_l$ is a $(|\mathcal{C}_l| \times |\mathcal{C}_{l+1}|)$ array of transition probabilities. We will describe how to calculate $\boldsymbol{A}'$ at the end of this section.

To estimate the lost LSF, we use a procedure similar to the forward-backward algorithm, as described in [52] and as applied in decoder NL-ac and [42]. Similar to $\boldsymbol{\alpha}$ in NL-ac, $\boldsymbol{\alpha}'$ accounts for the transitions from lower-numbered LSFs to higher numbered ones:

$$\boldsymbol{\alpha}'_{l,n}[i] \triangleq \Pr\left(\boldsymbol{V}_1[n], \ldots, \boldsymbol{V}_{10}[n], \boldsymbol{U}_l[n] = i\right) \tag{4.30}$$

And $\boldsymbol{\beta}'$ accounts for the transitions to higher LSFs:

$$\boldsymbol{\beta}'_{l,n}[i] \triangleq \Pr\left(\boldsymbol{V}_{l+1}[n], \ldots, \boldsymbol{V}_{10}[n] \mid \boldsymbol{U}_l[n] = i\right) \tag{4.31}$$

$\boldsymbol{\alpha}'$ and $\boldsymbol{\beta}'$ are calculated similar to before. First we consider the forward procedure. For the first LSF, $l = 1$:

$$\boldsymbol{\alpha}'_{1,n}[i] = \Pr\left(\boldsymbol{U}_1 = i\right) cb_1(\boldsymbol{U}_1[n] = i, \boldsymbol{V}_1[n]) \tag{4.32}$$

And for the subsequent LSFs, $l > 1$:

$$\boldsymbol{\alpha}'_{l,n}[j] = \boldsymbol{b}_l(\boldsymbol{U}_l[n] = j, \boldsymbol{V}_l[n]) \sum_{i=1}^{|\mathcal{C}_l|} \boldsymbol{\alpha}'_{l-1,n}[i] \boldsymbol{A}'_l[i,j] \tag{4.33}$$

Next we consider the backward procedure. In the inter-frame decoder, we used only a single future LSF when decoding, to avoid adding excessive delays. Here the full vector of LSFs, $l = 1, \ldots, 10$, is available at once, so we can apply the backwards procedure over the full vector without increasing the transmission delay. The backwards procedure begins by initializing $\boldsymbol{\beta}'$ for the final LSF, $l = 10$:

$$\boldsymbol{\beta}'_{10,n}[i] = 1 \tag{4.34}$$

for $i = 1, \ldots, |\mathcal{C}_{10}|$. For the earlier LSFs, $l < 10$:

$$\boldsymbol{\beta}'_{l,n}[i] = \sum_{j=1}^{|\mathcal{C}_{l+1}|} \boldsymbol{A}'_l[i,j] b_l(\boldsymbol{U}_{l+1}[n] = j, \boldsymbol{V}_{l+1}[n]) \boldsymbol{\beta}'_{l+1,n}[j], \text{ for } i = 1, \ldots, |\mathcal{C}_l| \tag{4.35}$$

We combine the forward and backward probabilities to form $\boldsymbol{\gamma}'$:

$$\boldsymbol{\gamma}'_{l,n}[i] = \frac{\boldsymbol{\alpha}_{l,n}[i] \boldsymbol{\beta}_{l,n}[i]}{\sum_k \boldsymbol{\alpha}_{l,n}[k] \boldsymbol{\beta}_{l,n}[k]} \tag{4.36}$$

Finally we calculate the decoder's output, by finding the expected value of the input based on the calculated probability $\boldsymbol{\gamma}'$, as in (4.27):

$$\boldsymbol{Y}_l[n] = \sum_{i=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \boldsymbol{\gamma}'_{l,n}[i] \tag{4.37}$$

We estimate the transition probability matrices, $\boldsymbol{A}'_l$, using a histogram-based approach, similar to the approach we used to estimate $\boldsymbol{A}_l$. The quantizers for the corresponding LSFs have $|\mathcal{C}_l|$ and $|\mathcal{C}_{l+1}|$ indices, respectively. We calculate and store a $|\mathcal{C}_l| \times |\mathcal{C}_{l+1}|$ transition probability matrix for each pair of adjacent LSFs in a frame. For the $N = 10$ LSFs in a frame, we must calculate 9 transition probability matrices. We calculate these matrices by counting the number of coincident indices in our training data. We denote the $|\mathcal{C}_l| \times |\mathcal{C}_{l+1}|$ matrix of transition counts from the lth to the $(l+1)$th LSF as $\boldsymbol{T}'_l$. Using $\dot{N}$ frames of training data, we calculate:

$$\boldsymbol{T}'_l[i,j] = \sum_{f=1}^{\dot{N}} \mathrm{I}\left(\dot{\boldsymbol{U}}_l[f] = i\right) \cdot \mathrm{I}\left(\dot{\boldsymbol{U}}_{l+1}[f] = j\right) \tag{4.38}$$

Using the transition counts, $\boldsymbol{T}'_l$, from (4.38), we calculate the required conditional transition probabilities:

$$\boldsymbol{A}'_l[i,j] \triangleq \Pr\left(\boldsymbol{U}_{l+1}[f] = j \mid \boldsymbol{U}_l[f] = i\right) \tag{4.39}$$

$$\approx \frac{\boldsymbol{T}'_l(i,j)}{\sum_{k=1}^{|\mathcal{C}_{l+1}|} \boldsymbol{T}'_l(i,k)} \tag{4.40}$$

4.4 Combining Decoders

The non-linear decoders we have described so far use either the inter-frame dependence between LSFs, or the intra-frame dependence to estimate a damaged LSF. We would like to examine whether taking advantage of both types of dependence together can improve decoder performance. The approach we consider is to combine the output of two decoders, one of which uses inter-frame dependence and another which uses intra-frame dependence. In particular, we consider combining:

- Inter-frame decoder NL-a (§4.2) or NL-ac (§4.2.1) with intra-frame decoder NL-b (§4.3). We call the resulting decoders NL-ab and NL-abc, respectively.

- Paired inter-frame decoder NL-2-a or NL-2-ac with paired intra-frame decoder NL-2-b (§4.6). We call the resulting decoders NL-2-ab and NL-2-abc, respectively.

Techniques for similar problems have been documented for use over bit-error and lossy channels. [54] proposes a HMM-based decoder for images, in which the pixels are modeled using a two-dimensional mesh. For LSFs, [41] proposes a decoder based on the forward-backward algorithm, similar to the one used in this thesis. It uses the equivalent of a forward recursion to account for past frames, and three backward recursions to account for the future frames and for higher and lower LSFs in the same frame. The approach used in this thesis allows for the use of other intra-frame decoders which can estimate quantizer output levels, such as the decoder based on GMMs.

Each inter-frame decoder forms an estimate of

$$\Pr\left(\boldsymbol{U}_l[n] \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[N]\right) \tag{4.41}$$

where N is the last frame used by the decoder to help estimate the nth frame, and each intra-frame decoder forms an estimate of

$$\Pr\left(\boldsymbol{U}_l[n] \mid \boldsymbol{V}[n]\right) \tag{4.42}$$

We will combine the estimate from the inter-frame decoder with the estimate from the intra-frame decoder to estimate:

$$\Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}[n], \ldots, \boldsymbol{V}_l[N]\right) \tag{4.43}$$

For the sake of compactness, let $\mathcal{X}$ denote $\boldsymbol{U}_l[n]$, $\mathcal{Y}$ denote $\boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[N]$, and $\mathcal{Z}$ denote $\boldsymbol{V}[n]$. We re-write (4.43) in terms of $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, to obtain $\Pr\left(\mathcal{X} = i \mid \mathcal{Y}, \mathcal{Z}\right)$. To simplify the problem, we assume that the inter-frame dependence ($\mathcal{Y}$) and the intra-frame dependence ($\mathcal{Z}$) are statistically independent given $\mathcal{X}$. Then:

$$\Pr\left(\mathcal{X} = i \mid \mathcal{Y}, \mathcal{Z}\right) = \frac{\Pr\left(\mathcal{X} = i \mid \mathcal{Y}\right) \Pr\left(\mathcal{X} = i \mid \mathcal{Z}\right)}{\Pr\left(\mathcal{X} = i\right)} \tag{4.44}$$

We denote $\Pr\left(\boldsymbol{U}_l[n] = i \mid \mathcal{Y}, \mathcal{Z}\right)$ by $\boldsymbol{\delta}_{l,n}[i]$, and we replace $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ with their original meanings above to obtain:

$$\boldsymbol{\delta}_{l,n}[i] \triangleq \frac{\Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[N]\right) \Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}[n]\right)}{\Pr\left(\boldsymbol{U}_l[n] = i\right)} \tag{4.45}$$

where $\Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}_l[1], \ldots, \boldsymbol{V}_l[N]\right)$ is obtained from the inter-frame decoder, $\Pr\left(\boldsymbol{U}_l[n] = i \mid \boldsymbol{V}[n]\right)$ is obtained from the intra-frame decoder, and $\Pr\left(\boldsymbol{U}_l[n] = i\right)$ is estimated as in (3.73). We use $\boldsymbol{\delta}_{l,n}[i]$ to calculate the decoder's output, similar to (4.27):

$$\boldsymbol{Y}_l[n] = \frac{\sum_{i=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \boldsymbol{\delta}_{l,n}[i]}{\sum_{i=1}^{|\mathcal{C}_l|} \boldsymbol{\delta}_{l,n}[i]} \tag{4.46}$$

4.5 Index Assignment Design

In this section, we consider the problem of designing IA matrices. A procedure for generating multiple descriptions using an IA matrix is described in §3.4.2, and decoders which operate on such descriptions were described in §3.7, §4.2, and §4.3. Our objective in this section is to design descriptions which minimize the distortion between the encoder's input and the decoder's output.

We do not consider methods for generating an IA matrix directly. Instead, we define a cost function $c(\mathcal{I}_l)$ for an IA matrix $\mathcal{I}_l$. The cost function measures the fitness of an IA, where a lower-cost IA is better. Given a cost function and a bit allocation, the simulated annealing algorithm [55] (described in Chapter B) is used to search for a low-cost IA matrix. The procedure we use to optimize an individual IA matrix is based on [56], which describes and evaluates the use of simulated annealing to design MDIAs.

In this thesis, the central decoder is fixed as the LSF quantizers used in FS-1016, so the cost function considers only the fitness when decoding a single description. Assume that the central quantizer index is $\boldsymbol{U}_l[n] = i$. Then the expected encoder input is $\mathrm{E}\{\boldsymbol{X}_l[n] \mid \boldsymbol{U}_l[n] = i\}$, which we approximate as $\mathcal{C}_l[i]$, as in (3.68). Let $o_l^{(m)}[i]$ denote the expected output of the decoder, given that the true index is $\boldsymbol{U}_l = i$, and only the mth description is received:

$$o_l^{(m)}[i] = \mathrm{E}\left\{\boldsymbol{Y}_l[n] \mid \boldsymbol{U}_l[n] = i, \boldsymbol{V}_l[n] = U_l^{(m)}[n]\right\} \tag{4.47}$$

This function depends on the type of IA being designed, and will be defined later. Then the squared error between the expected encoder input

and the expected decoder output is $(\mathcal{C}_l[i] - o_l^{(m)}[i])^2$ for the ith central quantizer index. To find the mean squared error for description m, we sum the expected squared error for each i, weighed by $\Pr(\boldsymbol{U}_l[n] = i)$. We assume that both descriptions are equally likely to be lost, so both descriptions are given equal weight in the cost function. Thus we form the cost function by adding the mean squared error over both descriptions:

$$c(\mathcal{I}_l) = \sum_{m=1}^{2} \sum_{i=1}^{|\mathcal{C}_l|} \Pr(\boldsymbol{U}_l = i) (\mathcal{C}_l[i] - o_l^{(m)}[i])^2 \tag{4.48}$$

4.5.1 Bit Allocation Optimization

In addition to optimizing individual IA matrices, we also consider the problem of allocating bits between the two descriptions for each LSF. As with the procedure used to optimize for the individual IA matrices, the procedure for optimizing the bit allocation uses a search algorithm with a cost function.

We begin by introducing the notation and optimization constraints we use for the bit allocations. Let b_l denote the number of bits used by the FS-1016 speech encoder's quantizer for the lth LSF, as listed in Table 2.1. The total number of bits used by the standard encoder for all LSFs is $\sum_{l=1}^{10} b_l = 34$. Further, let $b_l^{(1)}$ and $b_l^{(2)}$ denote the number of bits allocated for the first and second descriptions of LSF l, respectively. The number of bits must be non-negative. The dimensions of the IA matrix for LSF l are $2^{b_l^{(1)}} \times 2^{b_l^{(2)}}$. We place two constraints on the bit allocations. First, the total number of bits allocated for both descriptions of LSF l is equal to the number of bits used by the standard encoder for this LSF:

$$b_l^{(1)} + b_l^{(2)} = b_l \tag{4.49}$$

And second, the total number of bits allocated for all LSFs in either description is half the number of total bits used by the standard encoder:

$$\sum_{l=1}^{10} b_l^{(1)} = \sum_{l=1}^{10} b_l^{(2)} = \left(\sum_{l=1}^{10} b_l\right)/2 = 17 \tag{4.50}$$

We use a taboo search [57] to explore possible bit allocations. The search space of this optimization is the IA matrix bit allocations, $b_l^{(1)}$ and $b_l^{(2)}$ for $l = 1 \ldots 10$, subject to the constraints listed above. For each examined bit allocation, the search procedure calls the IA optimization procedure. The IA optimization procedure finds a low-cost IA matrix for each LSF, with the matrix dimensions selected by the bit allocation search: $2^{b_l^{(1)}} \times 2^{b_l^{(2)}}$ for LSF l. These ten IA matrices, $\mathcal{I}_l$ for $l = 1 \ldots 10$, are passed back to the bit allocation search, along with the combined cost of all ten IA matrices:

$$\sum_{l=1}^{10} c(\mathcal{I}_l) \tag{4.51}$$

where the function $c(\mathcal{I}_l)$ is defined by the IA design procedure. This combined cost, determined by the IA optimization procedure, is used as the cost function of the bit allocation search. To summarize, the bit allocation optimization is responsible for determining the dimensions of the IA matrices, while the IA optimization is responsible for determining the arrangement of indices within the IA matrices, and the cost of these IA matrices. The bit allocation search procedure outputs the IA matrices with the lowest cost that it encounters during its search.

4.5.2 MDIA for a Memoryless Decoder

We begin by considering the design of an IA for the memoryless MD decoder described in §3.7. We call the resulting IA the "memoryless' IA.

This decoder uses no information aside from descriptions of the LSF being decoded, so the decoder considers only the a priori $\Pr(\boldsymbol{X}_l)$ when decoding LSF l. In this IA design and in all others considered in this work, we assume that the system uses two descriptions: $M = 2$. The approach used here is identical to the one used in [56], for the case assuming optimal quantizers. Subsequent IA designs described in this thesis use the same approach, but with a different cost function.

Let $i = \boldsymbol{U}_l[n]$ denote the true central quantizer index, and assume that the decoder receives description m: $\boldsymbol{U}_l^{(m)}[n] = \mathcal{I}_l^{(m)}(i)$. By using the IA matrix, $\mathcal{I}_l$, the decoder can determine a set of indices which may have been output by the central quantizer. The function $b_l^{(m)}(j, \mathcal{I}_l^{(m)}(i))$, defined in (3.58), outputs a 1 if the central quantizer index j could have produced the observed description $\mathcal{I}_l^{(m)}(i)$. As in decoder NL-0 in §3.7, the received description will be decoded as:

$$\boldsymbol{Y}_l[n] = \frac{\sum_{j=1}^{|\mathcal{C}_l|} \mathcal{C}_l[j] \Pr(\boldsymbol{U}_l = j)\, b_l^{(m)}(j, \mathcal{I}_l^{(m)}(i))}{\sum_{j=1}^{|\mathcal{C}_l|} \Pr(\boldsymbol{U}_l = j)\, b_l^{(m)}(j, \mathcal{I}_l^{(m)}(i))} \tag{4.52}$$

This corresponds exactly to the expected value of the decoder's output, $o_l^{(m)}[i]$, in (4.48):

$$o_l^{(m)}[i] = \frac{\sum_{j=1}^{|\mathcal{C}_l|} \mathcal{C}_l[j] \Pr(\boldsymbol{U}_l = j)\, b_l^{(m)}(j, \mathcal{I}_l^{(m)}(i))}{\sum_{j=1}^{|\mathcal{C}_l|} b_l^{(m)}(j, \mathcal{I}_l^{(m)}(i))} \tag{4.53}$$

By substituting (4.53) into (4.48), we form the complete cost function for the design of the memoryless IA. Simulated annealing, described in the next section, is used to search for good index assignments.

4.5.3 MDIA for an Inter-Frame Decoder

In this section, we develop a procedure for designing an IA matrix which is optimized for use with the Markov model-based inter-frame decoder

NL-a (§4.2). The design procedure is developed based on the operation of decoder NL-a, and we call the resulting IA design the "forward" IA.

The inter-frame IA design begins similar to the memoryless IA design. We consider estimating the lth LSF in frame n, when only the mth description is received this frame: $\boldsymbol{V}_l[n] = \boldsymbol{U}_l^{(m)}[n]$. Assume that the central encoder output is $\boldsymbol{U}_l[n] = i$. The output of decoder NL-a depends on the decoder's forward probability vector $\boldsymbol{\alpha}$ in the previous frame, $n-1$. To account for this, we assume that the previous frame was correctly received, and we consider a step backwards from frame n to frame $n-1$. We define a backwards transition probability matrix $\boldsymbol{B}_l$:

$$\boldsymbol{B}_l[i,j] \triangleq \Pr\left(\boldsymbol{U}_l[n-1] = j \mid \boldsymbol{U}_l[n] = i\right) \tag{4.54}$$

The procedure we used to calculate $\boldsymbol{B}_l$ is presented at the end of this section, in (4.62). $\boldsymbol{B}_l[i,j]$ can be used to obtain the normalized $\boldsymbol{\alpha}_{l,n-1}[j]$ that the decoder will encounter if $\boldsymbol{U}_l[n] = i$ and the previous frame was correctly received, averaged over n for all such frames:

$$\mathrm{E}\left\{ \frac{\boldsymbol{\alpha}_{l,n-1}[j]}{\sum_{k=1}^{|\mathcal{C}_l|} \boldsymbol{\alpha}_{l,n-1}[k]} \mid \boldsymbol{U}_l[n] = i \right\} = \boldsymbol{B}_l[i,j] \tag{4.55}$$

From frame $n-1$, we step forward one frame, as in the forward part of the forward-backward algorithm used by decoder NL-a (§4.2). The forward step is based on the forward transition probability matrix $\boldsymbol{A}_l$ defined in (4.4). We calculate the expected $\boldsymbol{\alpha}$ for frame n, assuming that the mth description is received for frame n. To account for the received description, we determine which central quantizer indices could have produced the received description $\boldsymbol{U}_l^{(m)}[n]$. Any central quantizer indices which could not have produced this description have their probability in $\boldsymbol{\alpha}$ set to zero. This corresponds to the behaviour of the function

$I(\mathcal{I}_l^{(m)}(i) = \mathcal{I}_l^{(m)}(k))$, which outputs 1 if i and k would both produce the same description, and 0 otherwise. Thus we calculate the expected $\boldsymbol{\alpha}$ for frame n:

$$\begin{aligned}
&\mathrm{E}\left\{\boldsymbol{\alpha}_{l,n}[k] \mid \boldsymbol{V}_l[n] = \varnothing, \boldsymbol{U}_l[n] = i\right\} \\
&\quad = \sum_{j=1}^{|\mathcal{C}_l|} I(\mathcal{I}_l^{(m)}(i) = \mathcal{I}_l^{(m)}(j)) \Pr\left(\boldsymbol{U}_l[n-1] = j \mid \boldsymbol{U}_l[n] = i\right) \\
&\qquad\qquad \cdot \Pr\left(\boldsymbol{U}_l[n] = k \mid \boldsymbol{U}_l[n-1] = j\right) \quad (4.56)
\end{aligned}$$

Using $\boldsymbol{A}_l$ and $\boldsymbol{B}_l$, (4.56) can be expressed as:

$$\begin{aligned}
&\mathrm{E}\left\{\boldsymbol{\alpha}_{l,n}[k] \mid \boldsymbol{V}_l[n] = \boldsymbol{U}_l^{(m)}, \boldsymbol{U}_l[n] = i\right\} \\
&\qquad = \sum_{j=1}^{|\mathcal{C}_l|} I(\mathcal{I}_l^{(m)}(i) = \mathcal{I}_l^{(m)}(j)) \boldsymbol{B}_l[i,j] \boldsymbol{A}_l[j,k] \quad (4.57)
\end{aligned}$$

We take the time average of (4.57) over all frames and normalize the probabilities to obtain:

$$\tilde{\boldsymbol{\alpha}}_l^{(m)}[i,k] \triangleq \frac{\sum_{j=1}^{|\mathcal{C}_l|} I(\mathcal{I}_l^{(m)}(i) = \mathcal{I}_l^{(m)}(k)) \boldsymbol{B}_l[i,j] \boldsymbol{A}_l[j,k]}{\sum_{r=1}^{|\mathcal{C}_l|} \sum_{j=1}^{|\mathcal{C}_l|} I(\mathcal{I}_l^{(m)}(i) = \mathcal{I}_l^{(m)}(k)) \boldsymbol{B}_l[i,j] \boldsymbol{A}_l[j,r]} \quad (4.58)$$

By using this estimated $\boldsymbol{\alpha}$, we estimate the decoder's output as in (4.13) to obtain:

$$o_l^{(m)}[i] = \frac{\sum_{k=1}^{|\mathcal{C}_l|} \mathcal{C}_l[i] \tilde{\boldsymbol{\alpha}}_l^{(m)}[i,k]}{\sum_{k=1}^{|\mathcal{C}_l|} \tilde{\boldsymbol{\alpha}}_l^{(m)}[i,k]} \quad (4.59)$$

which is the $o_l^{(m)}[i]$ we use in the cost function (4.48) to design the "forward" IA.

Finally let us consider the calculation of the backwards transition probability matrices $\boldsymbol{B}_l$, as defined in (4.54). These matrices are estimated

using the inter-frame transition counts, $\boldsymbol{T}_l$, as calculated in (4.14):

$$\boldsymbol{B}_l[i,j] \triangleq \Pr\left(\boldsymbol{U}_l[n-1]=j \mid \boldsymbol{U}_l[n]=i\right) \tag{4.60}$$

$$= \Pr\left(\boldsymbol{U}_l[n]=j \mid \boldsymbol{U}_l[n+1]=i\right) \tag{4.61}$$

$$= \frac{\Pr\left(\boldsymbol{U}_l[n]=j, \boldsymbol{U}_l[n+1]=i\right)}{\sum_{k=1}^{|\mathcal{C}_l|}\Pr\left(\boldsymbol{U}_l[n]=k, \boldsymbol{U}_l[n+1]=i\right)} \quad \approx \frac{\boldsymbol{T}_l[j,i]}{\sum_{k=1}^{|\mathcal{C}_l|}\boldsymbol{T}_l[k,i]} \tag{4.62}$$

4.5.4 MDIA for an Intra-Frame Decoder

In this section, we describe a procedure for optimizing an IA for an intra-frame decoder, such as NL-b (§4.3). We call the resulting IA matrix the "intra" IA.

Assume that we are decoding the lth LSF in frame n, and that the true encoder output is $\boldsymbol{U}_l[n] = i$. Assume that the decoder receives only the mth description for this frame, so $\boldsymbol{V}[n] = \boldsymbol{U}^{(m)}[n]$. Thus the true encoder output is not available at the decoder due to packet loss. We recall from §4.3 that the output of decoder NL-b is based directly on the variable $\gamma_{l,n}[k]$, which is an estimate of the probability that the encoder output was k, given the observed descriptions. Our optimization procedure is based on estimating the decoder's value of $\gamma_{l,n}[k]$, given that the true transmitted index is $\boldsymbol{U}_l[n] = i$, and only the mth description is received, averaged over n for all such frames. This estimate is denoted by $\tilde{\gamma}_l^{'(m)}[i,k]$:

$$\tilde{\gamma}_l^{'(m)}[i,k] \triangleq \mathrm{E}\left\{\gamma_{l,n}[k] \mid \boldsymbol{V}[n] = \boldsymbol{U}^{(m)}[n], \boldsymbol{U}_l[n]=i\right\} \tag{4.63}$$

We recall from (4.36) that the calculation of $\gamma_{l,n}[k]$ depends on $\alpha_{l,n}[i]$, which is from the forward step of the forward-backward algorithm, and $\beta_{l,n}[i]$, from the backward step. We divide the optimization procedure which follows into two parts, first estimating $\alpha_{l,n}[i]$ and then estimating

$\boldsymbol{\beta}_{l,n}[i]$. To simplify the problem, we assume that the intra-frame decoder begins at LSF $l-1$, and ends at LSF $l+1$. This means that the design procedure described here cannot take advantage of dependence between non-adjacent LSFs. We expect that a model which accounts for the other LSFs in a frame would improve performance, however, this is left as future work.

We begin by considering the forward step. For now, we will not take the MD index received for LSF l into account. Our objective is to estimate:

$$\tilde{\boldsymbol{\alpha}}_l^{'(m)}[i,k] = \mathrm{E}\left\{\boldsymbol{\alpha}_{l,n}[k] \mid \boldsymbol{V}[n] = \boldsymbol{U}^{(m)}[n], \boldsymbol{U}_l[n] = i\right\} \tag{4.64}$$

For LSF 1, this is given by the initialization procedure of decoder NL-b:

$$\tilde{\boldsymbol{\alpha}}_1^{'(m)}[i,k] = \Pr\left(\boldsymbol{U}_1 = k\right) \tag{4.65}$$

Now we consider LSFs > 1. Similar to the inter-frame IA design, we will first consider a step backwards to the previous LSF, $l-1$. Later, the decoder will perform a forward step from LSF $l-1$ back to LSF l. We consider the probability that $\boldsymbol{U}_{l-1}[n] = j$, which is given by:

$$\boldsymbol{B}_l'[i,j] \triangleq \Pr\left(\boldsymbol{U}_{l-1}[n] = j \mid \boldsymbol{U}_l[n] = i\right) \tag{4.66}$$

Now we will estimate the decoder's expected average $\boldsymbol{\alpha}'$ for LSF $l-1$. For the inter-frame decoder, we assumed that the previous frame was correctly received. In contrast, here we know that the decoder received only the mth description for all LSFs in the frame. Using the IA, we calculate the probability that the received MD index is $\boldsymbol{U}_{l-1}^{(m)}[n] = a$:

$$\Pr\left(\boldsymbol{U}_{l-1}^{(m)}[n] = a \mid \boldsymbol{U}_l[n] = i\right) = \sum_{x=1}^{|\mathcal{C}_{l-1}|} I(\mathcal{I}_{l-1}(x) = a)\boldsymbol{B}_l'[i,x] \tag{4.67}$$

We initialize the average $\boldsymbol{\alpha}'$ for LSF $l-1$ as:

$$\bar{\boldsymbol{\alpha}}_{l-1}[j] \triangleq \Pr\left(\boldsymbol{U}_{l-1} = j\right) \tag{4.68}$$

Now assume that the decoder received the description $\boldsymbol{U}_{l-1}^{(m)}[n] = a$ for LSF $l-1$. By considering the IA matrix, the decoder determines which central quantizer indices could have resulted in this description. For all other central quantizer indices j, the decoder sets $\boldsymbol{\alpha}_l[j] = 0$. Combining this knowledge with our estimate for $\bar{\boldsymbol{\alpha}}_{l-1}[j]$, we estimate that $\boldsymbol{\alpha}_{l-1}$ will average:

$$\mathrm{E}\left\{\boldsymbol{\alpha}_{l-1}[j] \mid \boldsymbol{U}_{l-1}^{(m)}[n] = a\right\} \approx \bar{\boldsymbol{\alpha}}_{l-1}[j] I(\mathcal{I}_{l-1}^{(m)}(j) = a) \tag{4.69}$$

We normalize this and define:

$$\bar{\boldsymbol{\alpha}}_{l-1}[j \mid a] \triangleq \frac{\bar{\boldsymbol{\alpha}}_{l-1}[j] I(\mathcal{I}_{l-1}^{(m)}(j) = a)}{\sum_{y=1}^{|\mathcal{C}_{l-1}|} \bar{\boldsymbol{\alpha}}_{l-1}[y] I(\mathcal{I}_{l-1}^{(m)}(y) = a)} \tag{4.70}$$

Now we form our approximation for the expected value of $\boldsymbol{\alpha}_{l-1}$ given i, considering all possible MD indices a. We combine the estimated probability that the received MD index is a, from (4.67), with the estimated $\boldsymbol{\alpha}_{l-1}$ given that the received MD index is a, from (4.70). We sum over all MD indices a to obtain:

$$\begin{aligned}\mathrm{E}\left\{\boldsymbol{\alpha}_{l-1}[j] \mid \boldsymbol{U}_l = i, \boldsymbol{V} = \boldsymbol{U}^{(m)}\right\} \\ \approx \sum_{a=1}^{|\mathcal{I}_{l-1}^{(m)}|} \bar{\boldsymbol{\alpha}}_{l-1}[j \mid a] \sum_{x=1}^{|\mathcal{C}_{l-1}|} I(\mathcal{I}_{l-1}(x) = a) \boldsymbol{B}_l'[i, x]\end{aligned} \tag{4.71}$$

From this expected $\boldsymbol{\alpha}$ for LSF $l-1$, the decoder will perform a forward step to LSF l, using the transition probability matrix $\boldsymbol{A}_{l-1}'$. This gives us our estimate of $\boldsymbol{\alpha}$ for LSF l:

$$\tilde{\boldsymbol{\alpha}}_l'^{(m)}[i, k] = \sum_{j=1}^{|\mathcal{C}_{l-1}|} \mathrm{E}\left\{\boldsymbol{\alpha}_{l-1}[j] \mid \boldsymbol{U}_l = i, \boldsymbol{V} = \boldsymbol{U}^{(m)}\right\} \boldsymbol{A}_{l-1}'[j, k] \tag{4.72}$$

in which we use the approximation of $\mathrm{E}\{\cdot\}$ given in (4.71).

Next, we consider the backwards step to estimate $\boldsymbol{\beta}$. Our procedure for the backwards step is largely similar to our procedure for the forwards step. Our objective is to estimate:

$$\tilde{\boldsymbol{\beta}}_l^{\prime(m)}[i,k] \approx \mathrm{E}\left\{\boldsymbol{\beta}_{l,n}[k] \mid \boldsymbol{V}[n] = \boldsymbol{U}^{(m)}[n], \boldsymbol{U}_l[n] = i\right\} \tag{4.73}$$

Based on the operation of decoder NL-b, for LSF 10 this is given by:

$$\tilde{\boldsymbol{\beta}}_{10}^{\prime(m)}[i,k] = 1 \tag{4.74}$$

Now we consider LSFs $l < 10$. From the lth LSF, we consider stepping forward to estimate the state probabilities in LSF $l+1$. The probability that $\boldsymbol{U}_{l+1}[n] = j$, given i, is:

$$\boldsymbol{A}_l'[i,j] \triangleq \Pr\left(\boldsymbol{U}_{l+1}[n] = j \mid \boldsymbol{U}_l[n] = i\right) \tag{4.75}$$

We recall that only the mth description has been received for each LSF in the frame, so the decoder has received only the MD index $\boldsymbol{U}_{l+1}^{(m)}[n]$ for LSF $l+1$. By considering the transition probabilities and the IA, we calculate the probability that the received MD index is $\boldsymbol{U}_{l+1}^{(m)}[n] = a$:

$$\Pr\left(\boldsymbol{U}_{l+1}^{(m)}[n] = a \mid \boldsymbol{U}_l[n] = i\right) = \sum_{x=1}^{|\mathcal{C}_{l+1}|} I(\mathcal{I}_{l+1}(x) = a)\boldsymbol{A}_l'[i,x] \tag{4.76}$$

We initialize $\boldsymbol{\beta}$ for LSF $l+1$ as:

$$\bar{\boldsymbol{\beta}}_{l+1}[j] = 1, \text{ for all } j \tag{4.77}$$

We proceed with the same reasoning we used for $\boldsymbol{\alpha}$ above to obtain an analogue of (4.70):

$$\bar{\boldsymbol{\beta}}_{l+1}[j \mid a] \triangleq \frac{\bar{\boldsymbol{\beta}}_{l+1}[j] I(\mathcal{I}_{l+1}^{(m)}(j) = a)}{\sum_{y=1}^{|\mathcal{C}_{l+1}|} \bar{\boldsymbol{\beta}}_{l+1}[y] I(\mathcal{I}_{l+1}^{(m)}(y) = a)} \tag{4.78}$$

Now assume that the received MD index for LSF $l+1$ is a, and consider the backward step performed by decoder NL-b as in (4.23). After normalizing, this would result in the following $\boldsymbol{\beta}_l$:

$$\boldsymbol{\beta}_l[k] = \frac{\sum_{j=1}^{|\mathcal{C}_{l+1}|} \boldsymbol{A}_l[k,j]\bar{\boldsymbol{\beta}}_{l+1}[j \mid a]}{\sum_{y=1}^{|\mathcal{C}_l|}\sum_{j=1}^{|\mathcal{C}_{l+1}|} \boldsymbol{A}'_l[y,j]\bar{\boldsymbol{\beta}}_{l+1}[j \mid a]} \tag{4.79}$$

This will occur with the probability $\Pr\left(\boldsymbol{U}_{l+1}^{(m)}[n] = a \mid \boldsymbol{U}_l[n] = i\right)$, given in (4.76). We combine (4.76) with (4.79) to obtain our estimate for $\boldsymbol{\beta}_l$:

$$\begin{aligned}&\tilde{\boldsymbol{\beta}}_l^{'(m)}[i,k] \\ &= \sum_{a=1}^{|\mathcal{I}_{l+1}^{(m)}|}\sum_{x=1}^{|\mathcal{C}_{l+1}|} I(\mathcal{I}_{l+1}(x)=a)\boldsymbol{A}'_l[i,x]\frac{\sum_{j=1}^{|\mathcal{C}_{l+1}|} \boldsymbol{A}_l[k,j]\bar{\boldsymbol{\beta}}_{l+1}[j \mid a]}{\sum_{y=1}^{|\mathcal{C}_l|}\sum_{j=1}^{|\mathcal{C}_{l+1}|} \boldsymbol{A}'_l[y,j]\bar{\boldsymbol{\beta}}_{l+1}[j \mid a]}\end{aligned} \tag{4.80}$$

Now we calculate $\tilde{\boldsymbol{\gamma}}_l^{'(m)}[i,k]$. We take the effect of the IA for LSF l into account using $I(\mathcal{I}_l^{(m)}(i) = \mathcal{I}_l^{(m)}(k))$, as we did in §4.5.3. We combine the expected average alpha in (4.72) with the expected average beta in (4.80) to form:

$$\tilde{\boldsymbol{\gamma}}_l^{'(m)}[i,k] = \frac{\tilde{\boldsymbol{\alpha}}_l^{'(m)}[i,k]\tilde{\boldsymbol{\beta}}_l^{'(m)}[i,k]I(\mathcal{I}_l^{(m)}(i) = \mathcal{I}_l^{(m)}(k))}{\sum_{j=1}^{|\mathcal{C}_l|}\tilde{\boldsymbol{\alpha}}_l^{'(m)}[i,j]\tilde{\boldsymbol{\beta}}_l^{'(m)}[i,j]I(\mathcal{I}_l^{(m)}(i) = \mathcal{I}_l^{(m)}(j))} \tag{4.81}$$

Finally, we estimate the decoder output as in (4.13), using $\tilde{\boldsymbol{\gamma}}_l^{'(m)}[i,k]$:

$$o_l^{(m)}[i] = \frac{\sum_{k=1}^{|\mathcal{C}_l|} \mathcal{C}_l[k]\tilde{\boldsymbol{\gamma}}_l^{'(m)}[i,k]}{\sum_{k=1}^{|\mathcal{C}_l|}\tilde{\boldsymbol{\gamma}}_l^{'(m)}[i,k]} \tag{4.82}$$

this definition of $o_l^{(m)}[i]$ is substituted into the cost function (4.48) when we are designing an optimized IA for an intra-frame decoder.

The cost function we use for intra-frame IAs (from (4.48) and (4.82)) depends on the IA of adjacent LSFs. To handle this, we optimize the IA for all LSFs at the same time. As with the other IA design procedures presented in this thesis, we use simulated annealing to search for low cost

IA matrices. We examine each LSF's IA matrix in turn, probabilistically accepting a small change. After making a small change to one IA matrix, we update the cost function for the adjacent LSFs, and then proceed to the next IA. This process is repeated until the stopping condition is reached.

4.5.5 MDIA for a Combined Intra- and Inter-Frame Decoder

Now we consider the design of an IA for a combined intra- and inter-frame decoder, such as NL-ab or NL-abc (§4.4). To form this cost function, we combine the estimate for α from the inter-frame cost function in §4.5.3, with the estimate for γ' from the intra-frame cost function in §4.5.4. These estimates are denoted by $\tilde{\alpha}_l^{(m)}[i,k]$ and $\tilde{\gamma}_l^{'(m)}[i,k]$, respectively. Similar to (4.45) in the combined intra- and inter-frame decoder, these are combined to form:

$$\tilde{\boldsymbol{\delta}}_l^{(m)}[i,k] \triangleq \frac{\tilde{\boldsymbol{\alpha}}_l^{(m)}[i,k]\tilde{\boldsymbol{\gamma}}_l^{'(m)}[i,k]}{\Pr\left(\boldsymbol{U}_l = k\right)} \tag{4.83}$$

Using the quantizer codebook and $\tilde{\boldsymbol{\delta}}_l^{(m)}[i,k]$, we estimate the decoder's output, similar to (3.72):

$$o_l^{(m)}[i] = \frac{\sum_{k=1}^{|\mathcal{C}_l|} \mathcal{C}_l[k]\tilde{\boldsymbol{\delta}}_l^{(m)}[i,k]}{\sum_{k=1}^{|\mathcal{C}_l|} \tilde{\boldsymbol{\delta}}_l^{(m)}[i,k]} \tag{4.84}$$

We use this $o_l^{(m)}[i]$ in the cost function (4.48) to design an IA for a combined intra- and inter-frame decoder.

4.6 Paired LSF Indices

All of the encoders and decoders we considered so far operate on a single LSF at a time. Now we consider extending the transition probability-

based decoders to operate on a pair of LSFs at a time. We consider pairs made up of two adjacent LSFs in the same frame. We use a function $f_l()$ to generate an index i for such a pair:

$$i = f_l\left(\boldsymbol{U}_l[n], \boldsymbol{U}_{l+1}[n]\right) \tag{4.85}$$

for $l = 1, 3, 5, 7, 9$. f is invertible, so we can recover the original single LSF indices from the paired index.

The LSFs are paired before generating the multiple descriptions. Thus the MD encoder forms multiple descriptions for a pair of LSFs, rather than for a single LSF. Similarly, the MD decoder operates on the pair of LSFs as well, and considers transitions based on pairs of LSFs. For LSFs l and $l+1$ in a paired LSF inter-frame decoder, the transition probabilities are based on:

$$\boldsymbol{A}_l[i,j] \approx \Pr\left(f(\boldsymbol{U}_l[n+1], \boldsymbol{U}_{l+1}[n+1]) = j \mid f(\boldsymbol{U}_l[n], \boldsymbol{U}_{l+1}[n]) = i\right) \tag{4.86}$$

and for an intra-frame decoder, the transition probabilities are based on:

$$\boldsymbol{A}'_l[i,j] \approx \Pr\left(f(\boldsymbol{U}_{l+2}[n], \boldsymbol{U}_{l+3}[n+1]) = j \mid f(\boldsymbol{U}_l[n], \boldsymbol{U}_{l+1}[n]) = i\right) \tag{4.87}$$

When designing decoders based on pairs of LSFs, our limited training data becomes a problem. In decoders based on a single LSF, the transition probability matrices, $\boldsymbol{A}_l$, have dimensions $|\mathcal{C}_l| \times |\mathcal{C}_l|$. The largest codebook we use has $|\mathcal{C}_l| = 16$, so the largest $\boldsymbol{A}_l$ has $16 \times 16 = 256$ elements. With 407 806 samples of training data, we average $407\,806/256 \approx 1\,593$ training samples per matrix element, which was sufficient to cover most of the transition probability matrices. Now consider $\boldsymbol{A}_l$ for pairs of LSFs. The paired index for the largest codebooks can take $16 \times 16 = 256$ different values, so the largest $\boldsymbol{A}_l$ has $256 \times 256 = 65\,536$ elements. Our training

data gives us an average of $407\,806/65\,536 \approx 6$ samples per matrix element, which is insufficient to cover the matrix. If we use this histogram directly in our decoder, some of the transitions in the testing set will be missing. In some instances, the decoder will output $\gamma = 0$, so the decoder must fall back to some other approach.

To help estimate the transition probability matrices for paired LSFs, we model the transitions probabilities using a GMM (§3.6.3). We do not use the GMM's output alone, because the GMM we designed gives poorer performance than the transition counts. Instead, we add the output of the resulting GMM to the transition count matrices to fill the missing values. Unfortunately, in addition to filling missing values, the output of the GMM also gives $\boldsymbol{A}_l$ non-zero values at invalid points, where the LSFs are out of order: $U_l \geq U_{l+1}$. To account for this, we scan through $\boldsymbol{A}_l$ and set such invalid points to zero.

The paired LSF decoders, which we call NL-2-a, NL-2-ac, and NL-2-b, are equivalent to the decoders NL-a (§4.2), NL-ac (§4.2.1), and NL-b (§4.3), but operate on pairs of LSFs rather than one LSF at a time. The output of decoders NL-2-ac and NL-2-b is γ. For the inter-frame decoder NL-2-ac:

$$\gamma_{l,n}[f_l(i,j)] = \Pr\left(\boldsymbol{U}_l[n] = i, \boldsymbol{U}_{l+1}[n] = j \mid \boldsymbol{V}_l[1], \boldsymbol{V}_{l+1}[1], \ldots, \boldsymbol{V}_l[N], \boldsymbol{V}_{l+1}[N]\right) \tag{4.88}$$

And for the intra-frame decoder NL-2-b:

$$\gamma'_{l,n}[f_l(i,j)] = \Pr\left(\boldsymbol{U}_l[n] = i, \boldsymbol{U}_{l+1}[n] = j \mid \boldsymbol{V}_1[n], \ldots, \boldsymbol{V}_{10}[n]\right) \tag{4.89}$$

For the inter-frame decoder NL-2-ac, the output is the expected value

of $\boldsymbol{X}_l[n]$ and $\boldsymbol{X}_{l+1}[n]$, calculated using a procedure similar to (4.27):

$$\boldsymbol{Y}_l[n] = \sum_{i=1}^{|\mathcal{C}_l|} \sum_{j=1}^{|\mathcal{C}_{l+1}|} \mathcal{C}_l[i]\gamma_{l,n}[f_l(i,j)] \tag{4.90}$$

$$\boldsymbol{Y}_{l+1}[n] = \sum_{i=1}^{|\mathcal{C}_l|} \sum_{j=1}^{|\mathcal{C}_{l+1}|} \mathcal{C}_{l+1}[j]\gamma_{l,n}[f_l(i,j)] \tag{4.91}$$

For an intra-frame decoder, we replace γ with γ' in (4.90) and (4.91). For the decoder NL-2-a, the output is based on a normalized $\alpha_{l,n}$ in place of γ.

4.6.1 MDIA Design for Paired LSF Indices

The IA design procedure for paired LSF decoders is similar to the procedures described for intra-frame decoders in §4.5.4, and for combined inter- and intra-frame decoders in §4.5.5. With paired LSFs, however, the design is based on the paired indices as described in §4.6 rather than indices which represent a single LSF. We do not attempt to optimize the bit allocation for paired LSFs, due to the computer time this would require.

Chapter 5

Experimental Results and Discussion

5.1 Introduction

In this chapter, we evaluate and compare the performance of the encoding and decoding techniques described in Chapter 3.6. We will begin by describing the evaluation procedure we use. The remainder of the chapter is dedicated to evaluating the encoding and decoding techniques. The decoders are grouped according to which neighbouring LSFs they use to estimate a missing description. The emphasis of our discussion is on which techniques make the best use of a particular set of neighbouring LSFs.

The performance is evaluated using a set of recorded speech signals. The speech signals are divided into two sets. A training data set is used to design the encoders and decoders, by providing an estimate of the statistics and dependence between LSFs. A separate testing data set is

used to evaluate the performance of the designed systems. To generate the data sets, we encode the TIMIT [58] recorded speech database using the FS-1016 CELP speech coder [59]. We use the training and testing data sets recommended in the TIMIT documentation. The procedure used to generate the data sets is described in further detail in Appendix A.

The FS-1016 speech encoder outputs excitation parameters, and LP coefficients which are encoded as LSFs. To simplify the scope of this work, the encoders and decoders operate only on LSFs, and assume that the excitation parameters are received error-free by the decoder.

To compare systems, we use a distortion measure which can objectively evaluate the system performance. Because the proposed systems operate only on LSFs, we would prefer to use a measure of the distortion in LSFs alone. We recall that LSFs are used to model the spectral envelope due to the speaker's vocal tract. The *log spectral distortion* (SD) [60] is a commonly used metric for evaluating the difference between two spectra in speech applications. In this thesis, the spectra we compare are the power spectrum before quantization, $P(f)$, and the power spectra output by the decoder, $\tilde{P}(f)$. These power spectra can be determined from the LSFs by determining the polynomials $P(z)$ and $Q(z)$ given that the LSFs are their roots, then using (2.33) to determine the filter's $A(z)$, and calculating [26]:

$$P(f) = \frac{1}{|A(e^{j2\pi f/F_s})|^2} \tag{5.1}$$

the calculation of $\tilde{P}(f)$ is similar, with $\tilde{A}$ in place of A. The log SD is given by the log of the root mean squared difference between these two power

spectra [26]:

$$\mathrm{SD} = \sqrt{\frac{1}{F_s}\int_0^{F_s}\left(10\log_{10}P(f) - 10\log_{10}\tilde{P}(f)\right)^2 df} \tag{5.2}$$

As in [26], we evaluate the SD over the range $f = 0 - -3$ kHz rather than over the full sampling frequency. This is because the higher frequencies outside this range are less perceptually important. Rather than evaluate the integral in (5.2) directly, we use a discrete approximation similar to [16]:

$$\mathrm{SD} \approx \sqrt{\frac{1}{M}\sum_{n=0}^{M-1}\left(10\log_{10}P(F_s n/N) - 10\log_{10}\tilde{P}(F_s n/N)\right)^2} \tag{5.3}$$

where N is the number of discrete points at which the SD is sampled, and $M \leq N$ is the number of points necessary to approximate the frequency range. We use $N = 240$, which corresponds to the number of samples in the speech segment. For the range $0 - -3$ kHz out of $F_s = 8$ kHz, $M = 3/8N$.

We are interested in some perspective on the range of values for SD. According to [26], the requirements for transparent or imperceptible quantization are: an average of 1 dB, less than 2 % of frames are in the range 2 – 4 dB, and no frames are > 4 dB. Following these criteria, the SD results in this thesis are reported in terms of the average SD, the percentage of frames in the range 2 – 4 dB, and the percentage of frames over 4 dB. The performance of the FS-1016 CELP encoder on the testing data set, without losses or bit errors, is presented in Table 5.1. We expect this distortion to increase when losses occur.

The FS-1016 encoder, as well as all other systems considered in this thesis, use 34 bits per frame to encode the LSFs for one frame. However,

	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	% Frames > 4 dB
Error-free transmission	1.56	15.81	0.27

Table 5.1: SD of the testing set.

Index Assignment	Description Number	Bits Allocated for LSF... 1	2	3	4	5	6	7	8	9	10
Odd-Even	1	3	0	4	0	4	0	3	0	3	0
	2	0	4	0	4	0	3	0	3	0	3
Memoryless	1	1	2	2	2	1	3	1	1	2	2
	2	2	2	2	2	3	0	2	2	1	1
Forward	1	1	1	3	2	3	2	1	1	1	2
	2	2	3	1	2	1	1	2	2	2	1
Intra	1	1	3	2	3	0	2	1	2	1	2
	2	2	1	2	1	4	1	2	1	2	1
Intra-Forward	1	1	3	2	2	1	2	1	2	1	2
	2	2	1	2	2	3	1	2	1	2	1
Paired (all)	1	4		4		3		3		3	
	2	3		4		4		3		3	

Table 5.2: Bit allocations for IAs.

for the MD encoders, the 34 bits are divided into two descriptions with 17 bits each. For the odd-even and paired index MDIA designs, the bit allocations were based directly on the bit allocation used by the FS-1016 encoder. For the other MDIA designs, the bit allocation was determined using the optimization procedure introduced in §4.5. The bit allocation for each LSF in our MDIAs is presented in Table 5.2. By design, all of the encoders and decoders have the same distortion as FS-1016 when no losses occur. We are interested in finding and emphasizing the difference between approaches. To this end, we measure and report the SD only for *damaged* frames, which are frames for which one or more description is

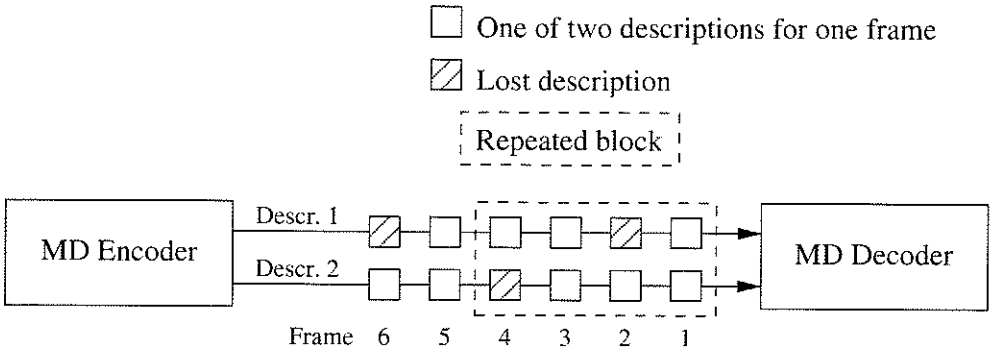

Figure 5.1: Isolated loss pattern.

lost.

We begin evaluating the encoding and decoding schemes by using the best case loss scenario, which we call an *isolated loss*. We define an isolated loss as the loss of one description, while all other descriptions used by the decoder to estimate the lost description are correctly received. A recurring pattern which meets this requirement for all the decoders considered here is: all descriptions are correctly received for frames $f = 1, 3, 5, \ldots$; only description 1 is received for frames $f = 2, 6, 10, \ldots$; and only description 2 is received for frames $f = 4, 8, 12, \ldots$. This loss pattern, which we call an isolated loss pattern, is illustrated in Figure 5.1.

The distortion during isolated losses approximates the distortion of damaged frames when losses are rare. This is because most packet losses are isolated when the loss probability is low and losses are independently distributed. Also, we will see that the relative performance of decoders under isolated losses tends to carry over to the performance at higher loss probabilities as well.

In addition to isolated losses, we also consider the performance of decoders subjected to uniformly distributed random packet losses. Let

P_L denote the packet loss probability. To simulate the packet loss distribution, we use a uniform random number generator, which generates a random number r in the range $0 \leq r < 1$ for each description of each frame. If $r < P_L$, the description is lost. At low P_L, most losses are isolated. As P_L increases, non-isolated losses become more common. When a non-isolated loss occurs, some of the information used by the decoder to estimate the damaged frame is not available. On average, the decoder has less information to estimate damaged frames, so the average distortion increases.

5.2 Memoryless Decoding

We begin by considering memoryless decoders, which use only the descriptions received for an LSF to estimate its value. We expect that decoders which exploit the redundancy between LSFs will improve upon such decoders' performance.

First, we investigate the usefulness of an optimized MDIA with a memoryless decoder. For this experiment, we use a memoryless decoder which we call NL-0 (§3.7). We test two multiple description schemes: odd-even splitting of LSFs (§3.4.1), and an optimized IA which we call "memoryless" (see §4.5.2 for the design procedure and Table 5.2 for the bit allocation). With odd-even splitting, NL-0 uses an LSF's mean value to estimate the missing LSFs. With an MDIA, NL-0 uses side codebooks to estimate the value of a damaged LSF. The performance of these systems is presented in Table 5.3. We find that the memoryless IA performs an average of 0.59 dB better than odd-even splitting at the same bit rate.

	Index Assignment	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	% Frames > 4 dB
NL-0	Odd-Even	5.15	26.74	71.54
	Memoryless	4.56	45.25	50.41
Repetition	Odd-Even	3.76	48.61	37.46

Table 5.3: Performance of memoryless MD decoding and repetition.

This result encourages us to consider the use of MDIA with more sophisticated decoders.

Next we test a decoder which uses odd-even splitting and estimates a missing LSF by simply repeating its value from the previous frame. We call this decoding scheme "repetition" (§3.6.2). The performance of this decoder is also presented in Table 5.3. Repetition performs an average of 0.80 dB better than NL-0 with a memoryless IA. This shows that the previous frame is a good predictor for the current frame. Below, we attempt to exploit such correlation to improve decoder performance and MDIA design.

5.3 Improving Repetition

Assume that the lth LSF in the nth frame is lost. The L-a decoder (§3.6.2) uses a linear combination of the lth LSF in the p previous frames to estimate the missing LSF. Here p is called the predictor order. Repetition is a special case of this linear decoder, in which the prediction order is 1 and the prediction coefficient is fixed at $a = 1$. The performance of decoder L-a is presented in Table 5.4. Using an optimized prediction coefficient gives L-a a small improvement in performance compared to repetition, decreasing the average distortion by 0.04 dB. This shows that repetition

	Predictor Order	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	% Frames > 4 dB
Repetition	1	3.76	48.61	37.46
L-a	1	3.72	53.19	36.37
	2	3.71	53.89	36.03
	3	3.71	53.87	36.02
Interpolation	—	2.99	52.95	20.72
L-ac	—	2.99	53.39	20.71

Table 5.4: Performance of decoders which improve on repetition.

is not the best possible predictor of future frames. Next we evaluate whether increasing the prediction order can improve performance further. The performance of L-a with $p = 2$ and 3 is also presented in Table 5.4. Here, using more than one past frame makes little difference in the distortion.

If a decoder uses frame $n+1$ to help decode the nth frame, we say that this decoder uses a *future frame*. If we can tolerate an increased decoding delay, and frame $n + 1$ is available, we can use interpolation (§3.6.2) between frames $n - 1$ and $n + 1$ to estimate a missing LSF in frame n. The performance of interpolation is also presented in Table 5.4. Interpolation decreases the average distortion by 0.77 dB compared to repetition. This demonstrates that using a future frame in addition to a past frame is a better predictor than using the past frame alone. Finally we consider improving interpolation by optimizing its prediction coefficients. The use of optimized prediction coefficients in L-ac (§3.6.2) makes little difference in the distortion compared to interpolation.

	Predictor Order	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	> 4 dB
L-a	1	3.72	53.19	36.37
	2	3.71	53.89	36.03
NL-a	1	3.68	53.12	35.43
	2	3.64	54.12	34.27

Table 5.5: Linear and non-linear prediction from past frames.

5.4 Non-Linear Decoders

In this section, we attempt to improve the performance of the decoders considered in the previous section, by using a non-linear model of the dependence between LSFs. We begin with a decoder which uses the transition probability from LSF l in frame $n-1$ to estimate a damaged LSF l in frame n. We call such a decoder NL-a (§4.2). Table 5.5 presents a comparison between this decoder and L-a, using odd-even splitting to generate the descriptions for both decoders. Using the transition probability instead of a linear model improves the distortion by 0.04 dB. This suggests that a linear decoder captures much, but not all, of the dependence between the same LSF in adjacent frames. As shown in Table 5.5, increasing the prediction order to 2 improves the performance further, by 0.08 dB compared with a linear predictor. Here the non-linear relationship between frames becomes more important in a second-order predictor, than in a first-order predictor. If the transition probability between LSFs was truly governed by a Markov chain, as we model them in this thesis, we would not expect to see an improvement from using two past frames instead of one.

Next we evaluate the NL-a decoder with other IA matrices. The re-

	Index Assignment	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	% Frames > 4 dB
L-a	Odd-Even	3.72	53.19	36.37
NL-a	Odd-Even	3.68	53.12	35.43
	Memoryless	3.57	59.02	30.28
	Forward	3.49	61.6	26.10
L-ac	Odd-Even	2.99	53.39	20.71
NL-ac	Odd-Even	3.00	53.50	20.86
	Memoryless	2.94	60.76	16.60
	Forward	2.84	63.03	13.23

Table 5.6: Using non-linear decoders to improve performance.

sulting performance is presented in Table 5.6. As we observed with NL-0, using an optimized IA improves performance compared to using odd-even splitting. Use of the memoryless IA improves the average distortion by an average of 0.11 dB compared with odd-even splitting. Using an IA matrix which is designed to work with this decoder, which we call the forward IA (§4.5.3), improves the performance further, by an average of 0.08 dB. This demonstrates that IA design can be improved by using the proposed technique, which exploits the dependence between frames and the behaviour of the decoder.

Finally we evaluate the use of a future frame to improve the performance of decoder NL-a. We call the modified decoder NL-ac (§4.2.1). If the nth frame is damaged, this decoder accounts for the transitions from frame $n-1$ to frame n, as well as the transitions from frame n to frame $n+1$. The performance of this decoder is presented in Table 5.6. When using odd-even splitting, NL-ac does not improve performance compared to the linear decoder L-ac. In fact, the distortion becomes slightly worse, partly because our assumption of optimal scalar quantizers (see (3.68)

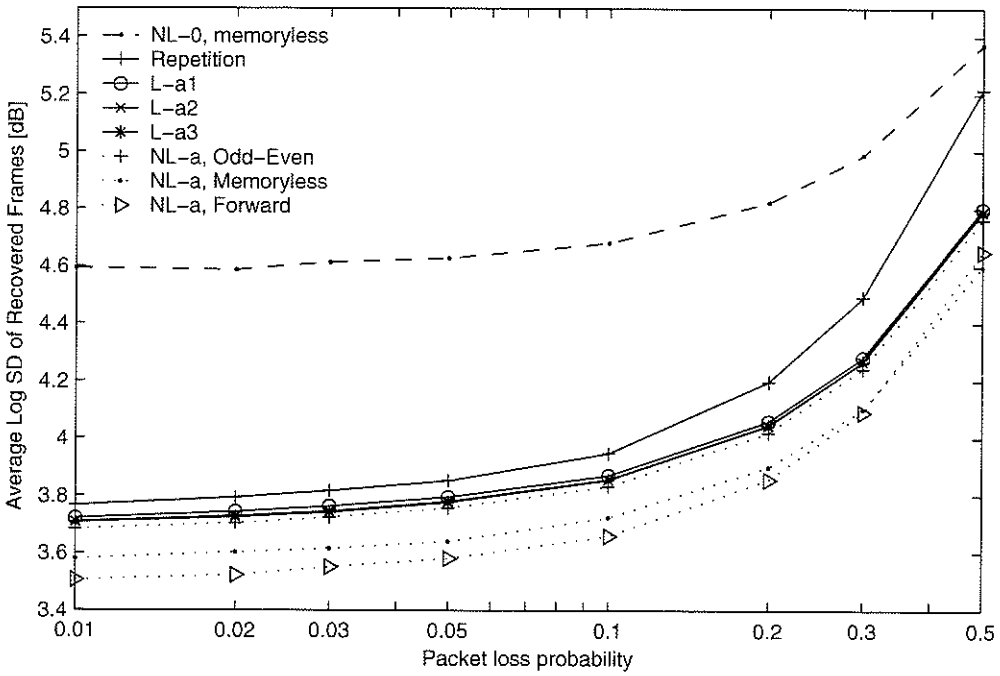

Figure 5.2: Performance of inter-frame decoders subjected to random losses.

and the discussion in §3.7) does not hold for this data set. If we use $\mathrm{E}\{\boldsymbol{X}_l \mid \boldsymbol{U}_l[n]\}$ in place of $C_l[\boldsymbol{U}_l[n]]$ in (4.13), the distortion drops slightly, to 2.98 dB. The use of an optimized IA improves performance compared to odd-even splitting. The memoryless IA improves the average distortion by 0.06 dB, and the forward IA improves the average distortion by another 0.10 dB.

5.4.1 Inter-Frame Decoders with Random Losses

In this section, we evaluate the performance of the decoders examined so far when they are subjected to randomly distributed packet losses. We consider the average distortion of damaged frames, as a function of packet loss probability. The distortion of correctly received frames is ex-

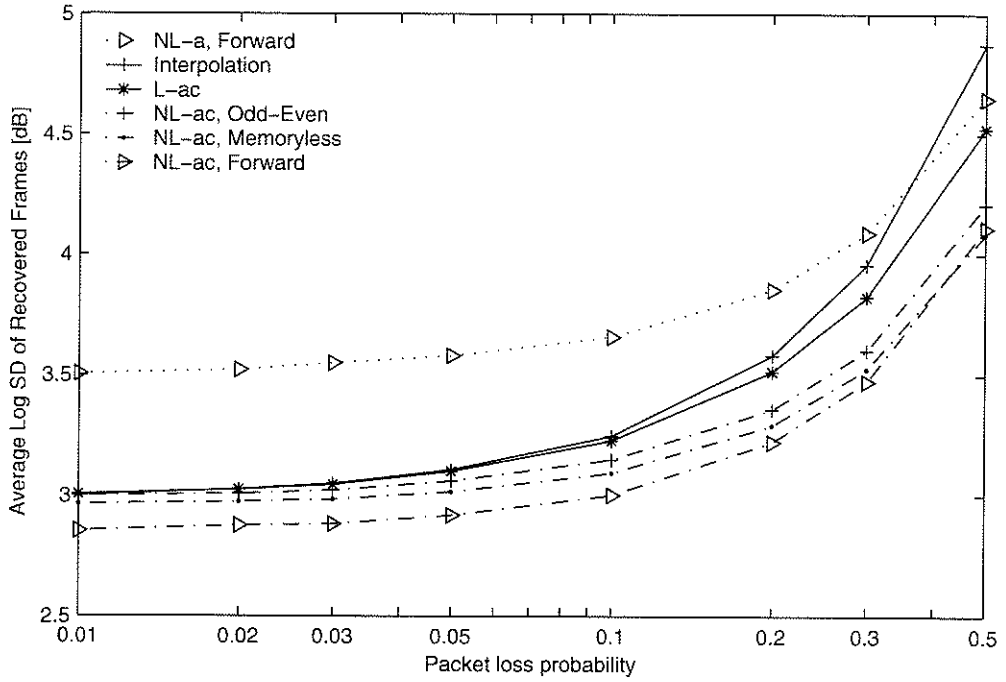

Figure 5.3: Performance of inter-frame decoders using one future frame, subjected to random losses.

cluded to emphasize the difference between decoders. The performance of several decoders which do not use a future frame is presented in Figure 5.2, along with the performance of the best memoryless decoder for comparison. The performance of several decoders which use a future frame are presented in Figure 5.3.

At low loss probabilities, most of the random losses are isolated, so the distortion with randomly distributed losses is approximately equal to the average isolated loss distortion we saw earlier in Tables 5.3, 5.4, and 5.6. As the loss probability increases, losses in adjacent frames and the loss of both descriptions for the same frame becomes increasingly common. Such losses give the decoders less residual information to estimate the damaged frames, so the average distortion increases.

At high loss probabilities, most decoders hold the same ranking as they did for isolated losses, and most decoders follow a similar increasing trend in distortion. We observe two notable exceptions. First, the memoryless IA begins to out-perform the forward IA for the NL-a and NL-ac decoders. This is because the forward IA is designed assuming that an adjacent frame is available, which becomes less likely at higher loss probabilities. Second, the interpolation and repetition coders perform relatively worse than other decoders as the loss probability increases. Most decoders either use descriptions which describe all the LSFs in a frame, or they slowly approach the LSF's mean value. In contrast, repetition and interpolation repeat a previous LSF exactly, which seems to put them at a disadvantage when the same LSF is lost in successive frames.

5.5 Intra-Frame Decoding

In this section, we consider decoding damaged LSFs by using the dependence between LSFs in the same frame. We begin by considering L-b (§3.6.2), an intra-frame linear decoder. If the lth LSF in the nth frame is missing, L-b estimates it by using a linear combination of LSFs $l-1$ and $l+1$ in frame n. We use odd-even splitting to generate multiple descriptions for L-b. We compare L-b to a memoryless decoder, NL-0, which we examined earlier. The results of this comparison are presented in Table 5.7. L-b performs an average of 1.35 dB better than NL-0. This shows that using the dependence between LSFs in a damaged frame can improve decoder performance.

Next we compare L-b to VLP-b (§3.6.2), again using odd-even split-

ting. VLP-b uses a linear combination of all five correctly received LSFs in a damaged frame to estimate the five missing LSFs. The performance of VLP-b is given in Table 5.7. We find that this decoder improves the average distortion by 0.05 dB compared to L-b. This shows that using all of the LSFs received for a frame can improve performance compared to using the adjacent LSFs alone.

Next we consider a non-linear decoder, NL-b (§4.3). If the lth LSF is damaged, NL-b estimates it using the transition probability from LSF $l-1$ to LSF l, and from LSF l to LSF $l+1$. Once again, we use odd-even splitting. As shown in Table 5.7, this decoder improves on the performance of L-b by an average of 0.04 dB, but its performance is slightly worse than VLP-b. The improvement relative to L-b shows that there is a non-linear relationship between adjacent LSFs, but the ability of VLP-b to use all five correctly received LSFs in the damaged frame gives a greater benefit.

Now we consider possible improvements to NL-b. First we evaluate the impact of different IA designs. The results of this evaluation are also presented in Table 5.7. The memoryless IA performs an average of 0.26 dB worse than odd-even splitting. This is the opposite of the relative performance with NL-0 in §5.2. With NL-0, a less accurate description of all ten LSFs from the memoryless IA was better than receiving five LSFs correctly from odd-even splitting. On the other hand, with NL-b, having the original quantized value of every second LSF is more useful, because it means we both have a more accurate representation of the received LSFs, and we can use the more accurate representation to estimate the missing LSFs. Using the intra-frame IA (§4.5.4), which is designed for an intra-frame decoder, gives a 0.08 dB improvement in average distortion

	Index Assignment	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	% Frames > 4 dB
NL-0	Memoryless	4.56	45.25	50.41
L-b	Odd-Even	3.21	61.31	24.19
VLP-b	Odd-Even	3.16	61.49	22.83
NL-b	Odd-Even	3.17	61.72	22.86
	Memoryless	3.43	61.89	27.11
	Intra	3.08	64.57	19.46
NL-2b	Odd-Even	3.13	62.04	21.68
	Paired Intra	2.84	65.87	12.67

Table 5.7: Performance of intra-frame decoders.

compared to VLP-b, and performs better than all intra-frame decoders we've examined so far.

Next we consider NL-2-b (§4.6), which operates similar to NL-b, but decodes based on pairs of LSFs. To estimate the pair of LSFs $(l, l+1)$, this decoder uses the transition probability from the LSFs $(l-2, l-1)$ to $(l, l+1)$, and from $(l, l+1)$ to $(l+2, l+3)$. The performance of this decoder is presented in Table 5.7, both for an odd-even IA, and for a paired LSF intra-frame IA which was designed for this decoder (§4.5.4). With an odd-even IA, this decoder performs 0.04 dB better than NL-b with the same IA. With the paired intra IA, this decoder's average distortion is 0.24 dB better than any intra-frame decoder we have considered so far.

Finally we consider a decoder which models the dependence between all ten LSFs in a frame, by using a Gaussian mixture model. We call this decoder NL-5b (§3.6.3). As with VLP-b, NL-5b works with odd-even separated LSFs, and uses all five received LSFs to estimate the missing LSFs. The performance of this decoder is presented in Table 5.8. The average performance improves as the number of Gaussian clusters increases, and

	Number of Gaussians	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	% Frames > 4 dB
VLP-b	—	3.16	61.49	22.83
NL-5b (GMM)	2	3.13	61.93	21.90
	4	3.09	61.3	21.13
	8	3.03	61.43	19.48
	16	3.00	61.28	18.70
	32	2.98	61.09	18.24
	64	2.97	60.87	18.01

Table 5.8: Using GMMs for intra-frame decoding.

[28] reports that this trend continues. At 64 clusters, NL-5b performs an average of 0.16 dB better than the previous best intra-frame decoder for odd-even separated LSFs. This shows that a non-linear decoder can benefit from using more than just its neighbours to estimate a missing LSF. However, by using a different MD encoding scheme, decoder NL-2-b decreases the average distortion by an average of 0.13 dB compared to NL-5b with 64 clusters.

5.6 Combined Intra- and Inter-Frame Decoding

In this section we consider decoders which use a combination of inter- and intra-frame redundancy. We begin by considering two linear decoders. Assume that the lth LSF in frame n is lost. Decoder L-ab (§3.6.2) uses a linear combination of the lth LSF in frame $n-1$, and LSFs $l-1$ and $l+1$ in frame n, to estimate the lost LSF. VLP-ab (§3.6.2) uses a linear combination of the entire frame $n-1$ and all correctly-received LSFs for frame n. The performance of these decoders is presented in Table 5.9.

	Index Assignment	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	% Frames > 4 dB
NL-a	Forward	3.52	60.56	27.32
NL-5b (GMM)	Odd-Even	2.98	61.08	18.25
L-ab	Odd-Even	2.91	68.94	13.10
VLP-ab	Odd-Even	2.71	65.70	9.91
NL-ab	Odd-Even	2.74	66.73	10.09
	Memoryless	2.89	65.06	14.22
	Forward	2.77	64.23	11.16
	Intra	2.69	65.44	9.71
	Intra-Forward	2.66	61.92	9.81
NL-2-ab	Paired Intra-F.	2.47	53.89	7.12

Table 5.9: Performance of combined intra- and inter-frame decoders, without using a future frame.

In addition to the combined decoders, we also present the performance of the best inter-frame decoder, NL-a, and the best intra-frame decoder, NL-5b. VLP-ab improves on L-ab by an average of 0.2 dB, making it the lowest distortion decoder we've seen so far. Not only is VLP-ab the best decoder we've seen so far, it is also computationally simpler than NL-a or NL-5b.

Next we consider the decoder NL-ab (§4.4). If the lth LSF in frame n is missing, NL-ab uses the transition probability from LSF $l-1$ to l and from LSF l to $l+1$ in frame n, as well as the transition probability from the lth LSF in frame $n-1$ to the damaged LSF. The performance of this decoder, with several IA designs, is also presented in Table 5.9. When using the odd-even, memoryless or inter-frame IA, NL-ab performs worse than VLP-ab. With the intra-forward IA (§4.5.5), decoder NL-ab is the best performing decoder we've seen so far, at an average of 2.66 dB.

We consider improving NL-ab by operating on pairs of LSFs, rather

	Index Assignment	Average Spectral Distortion [dB]	% Frames 2 – 4 dB	% Frames > 4 dB
NL-ac	Forward	2.86	62.69	13.76
NL-5b (GMM)	Odd-Even	2.98	61.08	18.25
L-abc	Odd-Even	2.61	66.32	7.47
VLP-abc	Odd-Even	2.41	59.75	4.90
NL-abc	Odd-Even	2.46	62.33	5.13
	Memoryless	2.54	60.53	7.84
	Forward	2.45	59.47	5.89
	Intra	2.42	60.19	5.26
	Intra-Forward	2.37	56.84	5.25
NL-2-abc	Paired Intra-F.	2.19	47.09	3.66

Table 5.10: Performance of combined intra- and inter-frame decoders, using a future frame.

than on one LSF at a time. This decoder uses a combined inter-intra frame paired IA (§4.6.1). In this system, the IA, and the intra- and inter-frame transition probabilities, are all based on pairs of LSFs. We call such a decoder NL-2-ab (§4.6). NL-2-ab performs better than NL-ab by an average of 0.19 dB. As we have found previously, there is a benefit from considering multiple LSFs simultaneously, both in decoding and in IA design.

Now we consider decoding using a combination of inter- and intra-frame redundancy, while using a future frame. The performance of such decoders is presented in Table 5.10. This table also presents the performance of the best inter-frame decoder, NL-ac, and the best intra-frame decoder, NL-5b. Once again, we find that all the combined decoders we consider outperform the best intra- or inter-frame decoder. All of these decoders outperform their counterpart which does not use a future frame by an average of at least 0.26 dB. Most of these decoders also out-perform decoder NL-2-ab, which is the best combined inter- and intra-frame de-

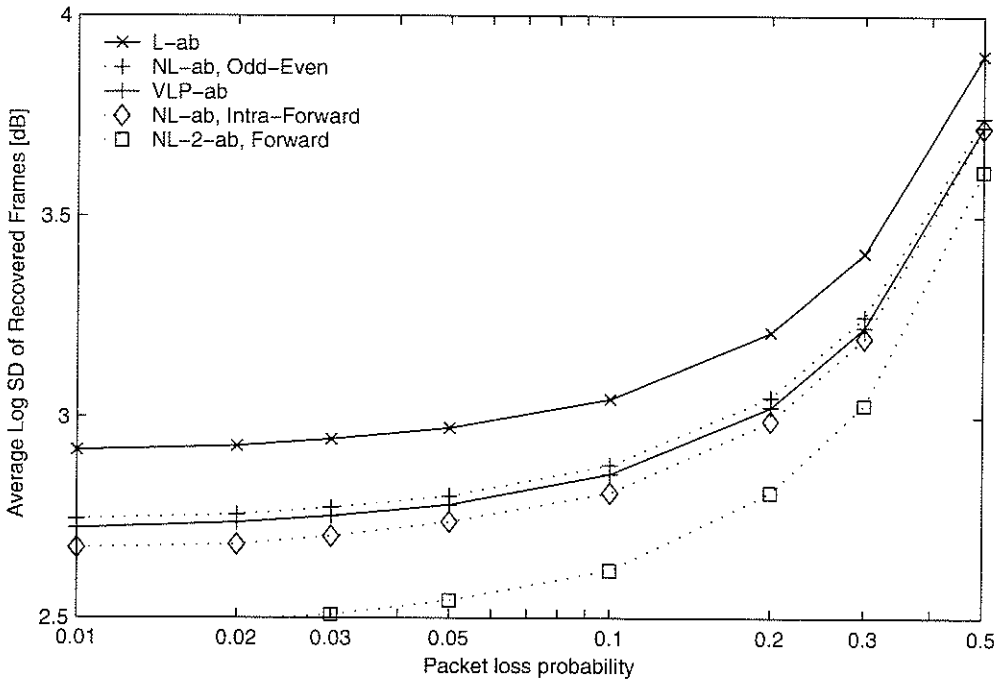

Figure 5.4: Performance of combined intra- and inter-frame decoders, without using a future frame, subjected to random losses.

coder which does not use a future frame. Once again we find that the average distortion can be improved by using a future frame, assuming a future frame is available.

The ranking of decoders here is nearly the same as for those which do not use a future frame. VLP-abc (§3.6.2) performs better than L-abc (§3.6.2). NL-2-abc (§4.6) improves on VLP-abc by an average of 0.23 dB, and has the lowest average distortion of all decoders considered in this thesis.

Finally we consider subjecting combined intra- and inter-frame decoders to randomly distributed packet losses. The performance of decoders which do not use a future frame is presented in Figure 5.4, and the performance of decoders which use a future frame is presented in

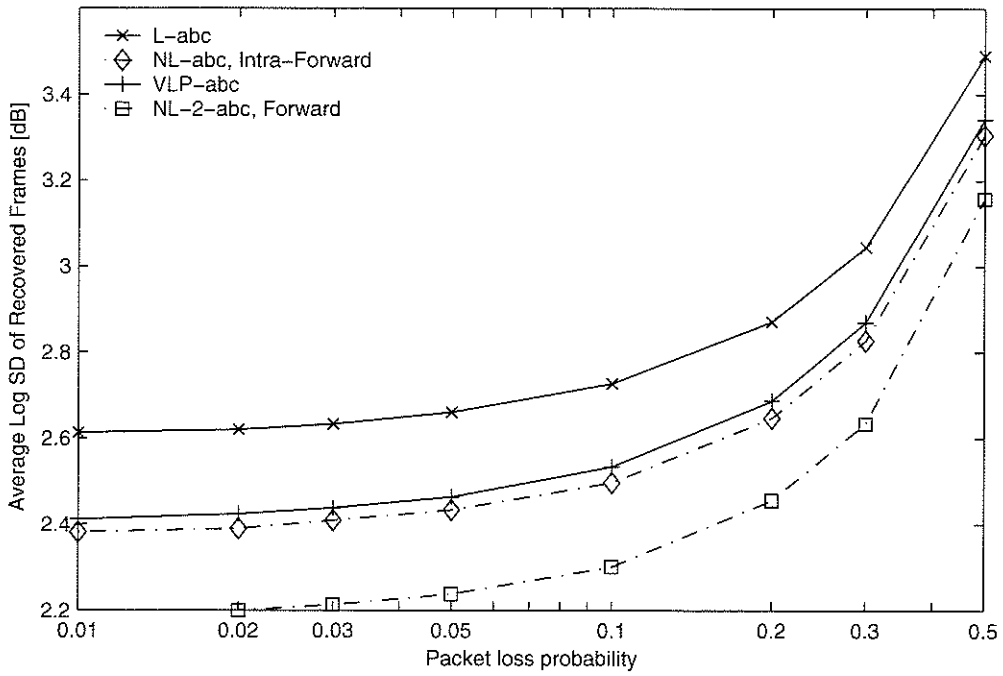

Figure 5.5: Performance of combined intra- and inter-frame decoders, using one future frame, subjected to random losses.

Figure 5.5. The paired LSF decoders, NL-2-ab and NL-2-abc, perform best at all loss probabilities. In most cases, the decoders have the same ranking in terms of average distortion at all loss probabilities. We note one exception, in decoder NL-ab. At low loss probabilities, the performance is better with the intra-forward IA. At high loss probabilities, the performance is better with odd-even splitting. Similar to the situation with inter-frame decoders, this appears to be due to the intra-forward IA's greater dependence on adjacent frames, which are more likely to be missing at higher loss probabilities.

Chapter 6

Summary and Conclusions

This thesis examined the use of MD encoding for LSFs, with applications to the FS-1016 speech coding standard. The descriptions were designed to match the bit rate of the quantized LSFs, and to match the performance of the FS-1016 speech coder when no losses occur. We considered two aspects of MD system design: a MD encoder which generates descriptions of the quantized LSFs, and a decoder which exploits dependence in the encoder's output to conceal losses.

Based on the experimental results in Chapter 5, we draw the following conclusions:

- The odd-even separation scheme is simple and performed well. Separated LSFs lend themselves well to decoding schemes which can model the dependence between many LSFs, such as linear decoders and GMM-based decoders.

- The proposed IA optimization techniques for Markov decoders can lower the average distortion compared to odd-even splitting. The

greatest gains from optimized IA design were observed with the memoryless decoder NL-0 and with the paired LSF IA.

- Paired LSF indices combined with the IA optimization techniques produced the lowest observed distortion in our experiments.

- We found that loss concealment approaches which take advantage of MDs were more effective than single description techniques, such as repetition and interpolation.

- As we would expect, decoders which take advantage of more received information exhibit better performance. For example, loss concealment approaches which use a future frame, or neighbouring LSFs in the same frame, are more effective than approaches which do not.

- We found that non-linear models, such as the Markov-based decoder, can improve loss concealment compared with linear models. However, linear models were more computationally efficient and usually required less training data than non-linear models with used the same number of input variables.

- The lowest distortion system encoded paired LSF indices using a MDIA which was optimized using the techniques proposed in this thesis. The system's decoder is based on the forward-backward algorithm for hidden Markov models, and accounted for the dependence between LSFs both within a frame and between frames.

6.1 Future Work

The performance evaluation in this work considered only spectral distortion, which does not necessarily indicate the perceived quality. To better evaluate the systems, a listening test should be performed, in which listeners judge the quality of the decoded speech.

This work considered the use of scalar quantizers only, for the sake of simplicity. As vector quantizers give much better performance for LSFs (see for example [26]), one may consider extending the work to VQ. With a change to VQ, perceptual distortion measures can be used (§2.3.7). We should examine the impact of such distortion measures on the system and optimization procedures.

The testing and optimization procedures considered here assumed that each description was placed in its own packet, and that the excitation parameters are always correctly received. Due to the overhead of packet headers, placing each description in its own packet increases the system's bit rate. One may consider other packetization approaches, for example, combining descriptions from two frames, n and $n+1$, into one packet. While considering more practical packetization methods, one may consider methods for generating MD of the excitation parameters as well.

This work used the same model of LSFs for all speakers. However, the distribution of LSFs is speaker-dependent, to the extent that LSFs have been used for speaker recognition [61]. One may consider methods for adapting the models used by the encoder and decoder for different speakers.

Bibliography

[1] Central Intelligence Agency. (2006, Dec.) World factbook. [Online]. Available: http://www.cia.gov/cia/publications/factbook/index.html

[2] N. S. Jayant and P. Noll, *Digial Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice-Hall, 1984.

[3] T. S. Rappaport, *Wireless Communications: Principles and Practice,* 2nd ed. Prentice Hall, 2002.

[4] A. S. Spanias, "Speech coding: a tutorial review," *Proc. IEEE,* vol. 82, no. 10, pp. 1541 – 1582, Oct. 1994.

[5] M. R. Schroeder and B. S. Atal, "Code-excited linear prediction (CELP): High-quality speech at very low bit rates," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '85),* vol. 10, Apr. 1985, pp. 937 – 940.

[6] B. Goode, "Voice over Internet protocol (VoIP)," *Proc. IEEE,* vol. 90, no. 9, pp. 1495 – 1517, Sept. 2002.

[7] J. D. Gibson, "Speech coding methods, standards, and applications," *IEEE Circuits and Systems Magazine,* vol. 5, no. 4, pp. 30– 49, 2005.

[8] W. Stallings, *Data & Computer Communications*, 6th ed. Prentice Hall, 2000.

[9] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, no. 5, pp. 40–48, Sept. 1998.

[10] V. K. Goyal, "Multiple description coding: Compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74–93, Sept. 2001.

[11] N. S. Jayant and S. W. Christensen, "Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure," *IEEE Transactions on Communications*, vol. 29, no. 2, pp. 101–109, Feb. 1981.

[12] J. P. Campbell Jr., T. E. Tremain, and V. C. Welch, "The Federal Standard 1016 4800 bps CELP voice coder," *Digital Signal Processing*, vol. 1, no. 3, pp. 145–155, July 1991.

[13] F. I. Alajaji, N. C. Phamdo, and T. E. Fuja, "Channel codes that exploit the residual redundancy in CELP-encoded speech," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 325–336, Sept. 1996.

[14] B. M. Oliver, J. R. Pierce, and C. E. Shannon, "The philosophy of PCM," *Proceedings of the IRE*, vol. 36, no. 11, pp. 1324 – 1331, Nov. 1948.

[15] T. P. Barnwell III, K. Nayebi, and C. H. Richardson, *Speech Coding: A Computer Laboratory Handbook*, ser. The Georgia Tech Signal Processing Series. John Wiley & Sons, Inc., 1996.

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

[17] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[18] J. Wilbur B. Davenport, "An experimental study of speech-wave probability distributions," *The Journal of the Acoustical Society of America*, vol. 24, no. 4, pp. 390–399, July 1952.

[19] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, no. 4, pp. 561 – 580, Apr. 1975.

[20] S. Maitra and C. R. Davis, "A speech digitizer at 2400 bits/s," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 27, no. 6, pp. 729 – 733, Dec. 1979.

[21] J. L. Flanagan, M. R. Schroeder, B. S. Atal, R. E. Crochiere, N. S. Jayant, and J. M. Tribolet, "Speech coding," *IEEE Trans. Commun.*, vol. 27, no. 4, pp. 710 – 737, Apr. 1979.

[22] E. T. S. Institute, *Digital cellular telecommunications system (Phase 2+); Adaptive Multi-Rate (AMR) speech transcoding (GSM 06.90 version 7.2.1 Release 1998).*

[23] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed. Prentice Hall, 1996.

[24] F. K. Soong and B.-H. Juang, "Line spectrum pair (LSP) and speech data compression," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '84)*, vol. 9, no. 1, pp. 37–40, Mar. 1984.

[25] F. Itakura, "Line spectrum representation of linear predictor coefficients of speech signals," in *The 89th Meeting of the Acoustical Society of America*, vol. 57, no. S1, 1975, p. S35.

[26] K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 1, pp. 3–14, Jan. 1993.

[27] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, July, October 1948. [Online]. Available: http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html

[28] R. Martin, C. Hoelper, and I. Wittke, "Estimation of missing LSF parameters using Gaussian mixture models," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001)*, 2001, pp. 729–732.

[29] M. Yong, G. Davidson, and A. Gersho, "Encoding of LPC spectral parameters using switched-adaptive interframe vector prediction," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '88)*, vol. 1, Apr. 1988, pp. 402 – 405.

[30] A. E. Gamal and T. M. Cover, "Achievable rates for multiple descriptions," *IEEE Trans. Inform. Theory*, vol. 28, no. 6, pp. 851 – 857, Nov. 1982.

[31] D. Lin and B. W. Wah, "LSP-based multiple-description coding for real-time low bit-rate voice transmissions," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, vol. 2, 2002, pp. 26–29.

[32] A. Anandakumar, A. McCree, and V. Viswanathan, "Efficient, CELP-based diversity schemes for VoIP," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2000)*, vol. 6, 2000, pp. 3682–3685.

[33] V. Hardman, M. A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the Internet," *Proceedings of INET, Oahu, Hawaii*, 1995.

[34] C.-C. Lee, "Diversity control among multiple coders: A simple approach to multiple descriptions," in *Proceedings of the IEEE Workshop on Speech Coding*, 2000, pp. 69–71.

[35] X. Zhong and B.-H. Juang, "Multiple description speech coding with diversities," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2002)*, vol. 1, 2002, pp. I–177 – I–180.

[36] R. Singh and A. Ortega, "Erasure recovery in predictive coding environments using multiple description coding," in *IEEE 3rd Workshop on Multimedia Signal Processing*, Sept. 1999, pp. 333–338.

[37] V. A. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 821–834, May 1993.

[38] Y. Wang, M. T. Orchard, V. Vaishampayan, and A. R. Reibman, "Multiple description coding using pairwise correlating transforms," *IEEE Trans. Image Processing*, vol. 10, no. 3, pp. 351–366, Mar. 2001.

[39] R. Arean, J. Kovacevic, and V. K. Goyal, "Multiple description perceptual audio coding with correlating transforms," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 2, pp. 140–145, Mar. 2000.

[40] J. Lindén, "Channel optimized predictive vector quantization," *IEEE Trans. Speech Audio Processing*, vol. 8, no. 4, pp. 370–384, July 2000.

[41] F. Lahouti and A. K. Khandani, "Soft reconstruction of speech in the presence of noise and packet loss," *IEEE Transactions on Audio, Speech and Language Processing*, to appear.

[42] P. Yahampath, "A recursive side-decoder for multiple description quantization," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2005)*, 2005.

[43] P. Yahampath and P. Rondeau, "Design of multiple description predictive vector quantizers," in *Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, vol. 4, 2006, pp. IV–209 – IV–212.

[44] D. J. Miller and M. Park, "A sequence-based approximate MMSE decoder for source coding over noisy channels using discrete hidden

Markov models," *IEEE Trans. Commun.*, vol. 46, no. 2, pp. 222 – 231, Feb. 1998.

[45] V. Cuperman and A. Gersho, "Vector predictive coding of speech at 16 kbits/s," *IEEE Trans. Commun.*, vol. 33, no. 7, pp. 685–696, July 1985.

[46] E. T. S. Institute, *Digital cellular telecommunications system (Phase 2+); Substitution and muting of lost frames for Adaptive Multi Rate (AMR) speech traffic channels (GSM 06.91 version 7.1.1 Release 1998).*

[47] 3rd Generation Partnership Project 2 (3GPP2), *Speech Service Option Standard for Wideband Spread Spectrum Systems*, ANSI/TIA/EIA-96-C.

[48] J. Wang and J. D. Gibson, "Parameter interpolation to enhance the frame erasure robustness of CELP coders in packet networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, vol. 2, May 2001, pp. 745–748.

[49] M. Yong, G. Davidson, and A. Gersho, "Encoding of LPC spectral parameters using switched-adaptive interframe vector prediction," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '88)*, Apr. 1988, pp. 402–405.

[50] Y. Agiomyrgiannakis and Y. Stylianou, "Coding with side information techniques for LSF reconstruction in voice over IP," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '05)*, 2005, pp. I–141–I–144.

[51] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed. McGraw-Hill, 1991.

[52] L. R. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *IEEE Acoustics, Speech and Signal Processing Magazine*, pp. 4–16, Jan. 1986.

[53] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2001.

[54] M. Park and D. J. Miller, "Improved image decoding over noisy channels using minimum mean-squared estimation and a markov mesh," *IEEE Trans. Image Processing*, vol. 8, no. 6, pp. 863 – 867, June 1999.

[55] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[56] P. Yahampath, "On index assignment and the design of multiple description quantizers," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, vol. 4, May 2004, pp. iv–597 – iv–600.

[57] F. Glover, E. Taillard, and D. de Werra, "Tabu search: An introduction," *Annals of Operations Research*, vol. 41, no. 1, pp. 1–28, Mar. 1993.

[58] National Institute of Standards and Technology, "The DARPA TIMIT acoustic-phonetic continuous speech corpus (TIMIT)," CD-ROM.

[59] J. Campbell *et al.*, "FS-1016 CELP 3.2a source code," Aug. 1993. [Online]. Available: ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/coding/celp_3.2a.tar.Z

[60] A. H. Gray, Jr. and J. D. Markel, "Distance measures for speech processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 24, no. 5, pp. 380–391, Oct. 1976.

[61] C.-S. Liu, W.-J. Wang, M.-T. Lin, and H.-C. Wang, "Study of line spectrum pair frequencies for speaker recognition," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '90)*, Apr. 1990, pp. 277 – 280.

Appendices

Appendix A

Training and Testing Data Set

We use DARPA's TIMIT corpus [58] for all the training and testing data in our work. The TIMIT corpus holds spoken sentences from 630 speakers in the United States. Ten sentences are recorded for each speaker, for a total of 6300 recorded sentences.

We use the entire `TRAIN` directory for training data unless stated otherwise, and the entire `TEST` directory for testing data.

We high-pass filter the `wav` files, using a 6th order IIR Butterworth filter, with -20 dB attenuation at 60 Hz. Next we low-pass filter the waveforms. The filter has -3 dB attenuation at 3800 Hz, and -20 dB attenuation at 4 kHz. The waveforms are resampled to 8 kHz.

To generate our testing and training data, we use a FS-1016 CELP coder, the CELP Voice Coder v3.2c [59]. We modify the program to save the LSF vectors both before and after quantization.

The resulting training and testing sets were scaled from normalized frequencies $(0 \ldots 1)$ to the sampling frequency $(0 \ldots 8$ kHz). The input files for the training and testing sets were concatenated to form one file

for the training data, and one file for the testing data. The result of these operations is a $407\,806 \times 10$ matrix of training data, and a $172\,061 \times 10$ matrix of testing data.

Appendix B

Simulated Annealing

Simulated annealing [55] is an algorithm for combinatorial optimization problems, which approximates the annealing process for materials. Annealing is intended to bring atoms into lower-energy configurations. The process begins by raising the material's temperature. This breaks the bonds between atoms, and releases them from their initial minima. Next the temperature is carefully lowered. As the temperature drops, less energy is available to put atoms into higher-energy configurations, so such configurations become less likely. In simulated annealing, the energy is the value of a cost function we are attempting to minimize, and the temperature is a variable which controls the permitted changes in the cost function.

In this work, based on [56], the objective is to find an IA matrix $\mathcal{I}$ which minimizes a cost function. We start with a high temperature T_0 and a randomly generated IA matrix. For each iteration, we perturb the IA by exchanging the position of two randomly-selected matrix elements. This perturbation may or may not lower the cost function (energy). The

perturbed system is probabilistically accepted as the new system, based on its energy E and the temperature of the system T. If the temperature is high, increases in energy are more likely to be accepted. Systems which decrease the energy are always accepted. After a predetermined number of new systems are accepted or rejected, the temperature is lowered by a factor α: $T_{\text{new}} = \alpha T$. The algorithm stops after the temperature reaches a predefined minimum, T_f. The algorithm is presented in Algorithm 1.

$\mathcal{I} \leftarrow \mathcal{I}_0$;
$T \leftarrow T_0$;
$E \leftarrow$ `GetCost(`$\mathcal{I}$`)` ;
while $T > T_f$ **do**
  AcceptCount $\leftarrow 0$;
  RejectCount $\leftarrow 0$;
  while *AcceptCount* $<$ *MaxAccept* **and** *RejectCount* $<$ *MaxReject*
  do
    $\mathcal{I}' \leftarrow$ `Perturb(`$\mathcal{I}$`)` ;
    $E' \leftarrow$ `GetCost(`$\mathcal{I}'$`)` ;
    $\Delta E \leftarrow (E' - E)$;
    Accept $\leftarrow (\Delta E < 0)$ **or** (`UniformRand()` $< \exp(-\Delta E/T)$) ;
    if Accept **then**
      $\mathcal{I} \leftarrow \mathcal{I}'$;
      $E \leftarrow E'$;
      AcceptCount $\leftarrow 1 +$ *AcceptCount* ;
    else
      RejectCount $\leftarrow 1 +$ *RejectCount* ;
    end
  end
  $T \leftarrow \alpha T$;
end
return $\mathcal{I}$;

Algorithm 1: Simulated annealing.