AN ABSTRACT OF THE DISSERTATION OF

Patrick U. Volz for the degree of Doctor of Philosophy in Electrical and Computer Engineering presented on May 7, 1999. Title: Efficient Decision Feedback Receiver Design for Cellular CDMA Spread Spectrum Communications.

Abstract approved: _____

Mario E. Magaña

This dissertation presents the continued study of a non-iterative decision feedback (DF) receiver/decoder design and its application to cellular communications or wireless local loop systems based on the IS-95(-A) standard of the Telecommunications Industry Association and also Personal Communications Services systems based on the American National Standards Institute standard J-STD-008-1996, which use code-division multiple access (CDMA) spread spectrum technology. Specifically, the DF decoder presented herein can be used in the uplink of these systems, which simultaneously uses a concatenation of convolutional coding, interleaving, and orthogonal Walsh modulation.

The main contributions of this dissertation are the demonstration that the DF concept works well in multipath fading environments, the design of a new time-efficient decoding algorithm, and a new interleaver design.

Initially, the performance of the DF decoder is assessed in unfaded as well as Rayleigh fading multipath

propagation in additive white Gaussian noise interference. Simulation results using coherent and noncoherent detection are presented for both independent Rayleigh fading and Rayleigh fading with a commonly used Doppler spectrum. The results show improved performance compared to conventional non-DF receivers using the same decoding metric. This is a prerequisite for application of the DF decoder in an actual mobile communications environment.

The effectiveness of the initial DF decoder design, as it is applied to IS-95 based systems, is studied. It is found that the effectiveness of the DF decoder is determined by the decoding delay of the convolutional decoder and the interleaver specification. Based on these findings, two methodologies to improve the effectiveness of the DF decoder are investigated. First, the average decoding delay is reduced using sub-optimal convolutional decoding. Second, the combination of a new block interleaver design and the DF decoder is considered. Simulation results of average decoding delay, bit error rate and frame error rate are presented for coherent and noncoherent detection of unfaded and Rayleigh fading multipath signals. It is shown that both approaches result in better system performance, which can further improve the quality of service and/or capacity of an IS-95 based system.

Finally, a simplified analysis of the DF decoder performance is presented.

Efficient Decision Feedback Receiver Design for Cellular CDMA Spread Spectrum Communications

by

Patrick U. Volz

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented May 7, 1999
Commencement June 1999

Doctor of Philosophy dissertation of Patrick U. Volz presented on May 7, 1999

APPROVED:

Major Professor, representing Electrical and Computer Engineering

Chair of Department of Electrical and Computer Engineering

Redacted for privacy

Dean of Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Redacted for privacy

Patrick U. Volz, Author

Acknowledgement

I would like to thank everybody who assisted and supported me during the preparation of this work. Especially, I would like to thank my wife, my parents, and my advisor for their support, patience, and advice. Finally, I thank all the new friends we found in Corvallis for helping to make this a truly memorable time of our lives.

Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

List of Figures

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Tables

List of Appendices

List of Abbreviations

A/D	Analog-to-Digital Converter
AMPS	Advanced Mobile Phone System
ANSI	American National Standards Institute
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CDMA	Code-Division Multiple Access
CLT	Central Limit Theorem
CRC	Cyclic Redundancy Check/Code
DF	Decision Feedback
DFD	Decision Feedback Decoding
DS	Direct-Sequence
DS-SS	Direct-Sequence Spread Spectrum
E_b/N_0	Bit-energy-to-interference-density ratio
FDMA	Frequency-Division Multiple Access
FD/CDMA	Hybrid FDMA and CDMA system
FD/TDMA	Hybrid FDMA and TDMA system
FER	Frame Error Rate
FFT	Fast Fourier Transform
FH	Frequency-Hopping, Frequency-Hop
FHT	Fast Hadamard Transform
FH-SS	Frequency-Hopping Spread Spectrum
FM	Frequency Modulation

List of Abbreviations (Continued)

GSM	Global System for Mobile Communications (originally Groupe Spécial Mobile)
IS	Interim Standard
LFSR	Linear Feedback Shift-Register
MAI	Multiple Access Interference
MSK	Minimum-Shift Keying
O-QPSK	Offset Quadrature Phase-Shift Keying
PCS	Personal Communications Services
QPSK	Quadrature Phase-Shift Keying
pdf	Probability Density Function
PN	Pseudorandom Noise
S/I	Signal-to-Interference Ratio
SOVA	Soft-Output Viterbi Algorithm
SS	Spread Spectrum
TDMA	Time-Division Multiple Access
TIA	Telecommunications Industry Association
USDC	U.S. Digital Cellular

Dedicated to my wife
Melanie

EFFICIENT DECISION FEEDBACK RECEIVER DESIGN FOR CELLULAR CDMA SPREAD SPECTRUM COMMUNICATIONS

Chapter 1. Introduction

This dissertation presents the continued study of a non-iterative decision feedback (DF) receiver/decoder design and its application to cellular communications or wireless local loop (WLL) systems based on the IS-95(-A) standard of the Telecommunications Industry Association (TIA). It is also applicable to Personal Communications Services (PCS) systems based on the American National Standards Institute (ANSI) standard J-STD-008-1996. These systems use code-division multiple access (CDMA) spread spectrum technology and have been and are currently deployed in many countries [1].

The DF decoder used in the receiver presented herein can be efficiently used in the uplink of these systems. The characteristic that sets it apart from other decoders is that it simultaneously utilizes the concatenation of convolutional coding, interleaving, and orthogonal Walsh modulation specified in the IS-95 standard.

The main contributions of this dissertation are the following. First, the demonstration that the DF concept works well in multipath fading environments. Second, the design of a new time-efficient decoding algorithm. Finally, a new block interleaver design.

1.1. Cellular Communications

In the 1960s and 1970s, Bell Laboratories, among many others, developed the cellular concept and the technology required to implement a cellular telephone system. The first cellular system was build in Japan by Nippon Telephone and Telegraph and deployed in 1979.

The cellular concept [2] makes it possible to provide wireless telephone service to a large population using a limited amount of frequency spectrum. Depending on the chosen modulation scheme, a total fixed number of communications channels is available in the system. The Advanced Mobile Phone System (AMPS) is the first cellular system introduced in the United States in 1983. It uses frequency modulation (FM) and a channel bandwidth of 30 kHz.

The service area is divided into small regions called cells. The cells are served by base stations, which are either located in the center of the cell (center-excited) or on the cell boundary (edge-excited). The communications link between a base station and the mobile users is referred to as the forward link or downlink. Similarly, mobile-to-base communications is referred to as the reverse link or uplink.

By systematically assigning a subset of the available channels to each base station, the available frequency spectrum is reused throughout the coverage area. This is called frequency reuse. A collection of adjacent cells that use all of the available channels is called a cluster. Cells that use the same set of channels are called cochannel cells. To minimize the interference between cochannel cells, the transmit power of the base stations is set just large enough to cover each cell.

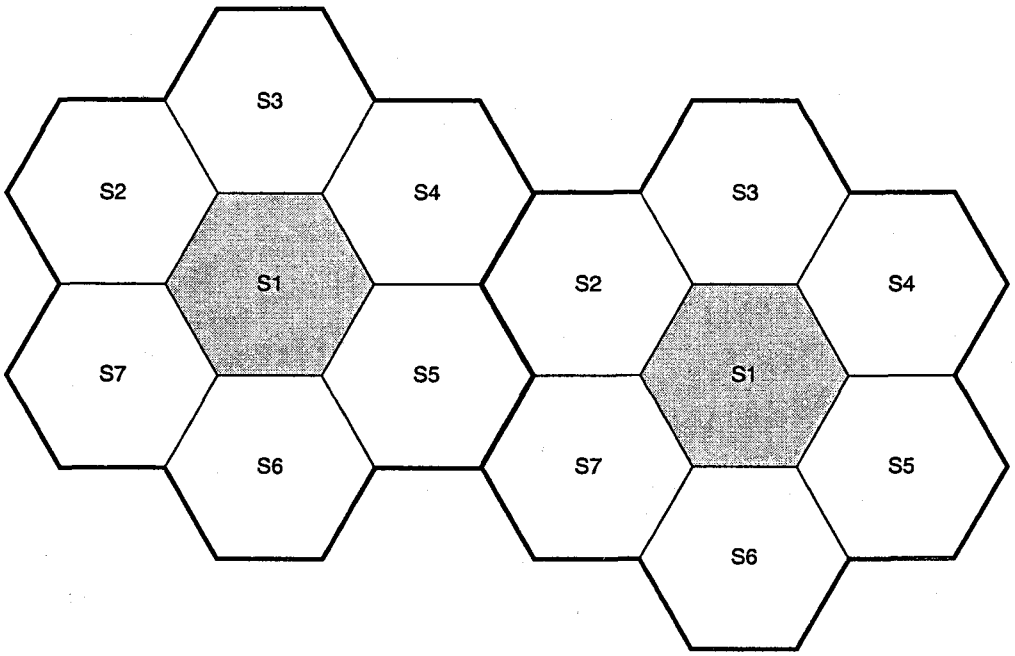


Figure 1.1: Cellular frequency reuse concept for a frequency reuse factor of 7.

The propagation path loss provides isolation between cochannel cells, but cochannel interference still exists.

Figure 1.1 shows two clusters of size $N = 7$, illustrating a frequency reuse factor[1] of 7. The hexagonal shape has been adopted to model the radio coverage of a cell for several reasons. It approximates the omni-directional free-space radiation pattern. At the same time, it allows for system analysis and can be used to cover an area without overlap.

As the mobile users travel through the coverage area, it becomes necessary to change channels and/or base stations during an ongoing call. This process is called a *handoff* and is a critical element of the cellular system.

[1] Sometimes the inverse of the cluster size, i.e., $1/N$ (1/7 in this case) is referred to as the frequency reuse factor.

The capacity of the cellular system, i.e., the total number of subscribers that can be serviced, is of great importance from a business standpoint. The use of trunking theory [3] makes it possible to provide service to a much higher number of subscribers per cell than there are physical channels. The number of physical channels is chosen to achieve a certain likelihood of blocked (unsuccessful) calls during peak periods. This is accomplished by using statistical models of subscriber behavior.

Ultimately, the number of simultaneous telephone calls that can be supported within a cell determines the capacity of the cellular system. The amount of available channels and the system interference limit this number.

The major source of interference is the cochannel interference. For reliable communications, a certain signal-to-interference ratio (S/I) is required at the receiver. For example, AMPS requires a S/I of 18 dB, which implies a minimum frequency reuse factor of 7 [4]. With a fixed cluster size, cell capacity becomes a function of the total number of available channels, i.e., the amount of allocated frequency spectrum.

Given a certain capacity per cell, the overall system capacity can be increased in a number of ways. The cell size can be reduced so that more clusters are used to cover a certain area. This can be done for congested areas and is referred to as cell splitting. Directional antennas, as opposed to omni-directional antennas, can be used to reduce cochannel interference. However, the capacity improvements of these measures have diminishing returns. Sectoring reduces the number of available channels per sector. The reduction of cell size and sectoring result in a higher number of required handoffs.

Both a reduction of available channels and a higher number of handoffs limit capacity.

The number of cellular subscribers has been growing annually by about 40% since the mid-1980s. Despite all optimization efforts, the AMPS system reached capacity within several years after its introduction. The same is true for other first-generation cellular systems worldwide. Additional spectrum was allocated to accommodate some of that growth. Eventually, more bandwidth efficient second-generation systems were needed.

In 1991, the U.S. Digital Cellular (USDC) system was introduced in the United States as standard IS-54 [5]. At the same time, Europe introduced the GSM (Global System for Mobile Communications) standard [6]. These systems use digital modulation and a combination of frequency-division multiple access and time-division multiple access (FD/TDMA). Compared to first-generation systems, the capacity is increased. For example, the USDC system transmits 3 calls in one 30 kHz AMPS channel, increasing the capacity by a factor of 3 [5].

In 1993, another second-generation digital system has been introduced in the U.S. as interim standard IS-95. It uses code-division multiple access (CDMA), a spread spectrum technique.

1.2. Spread Spectrum Communications

The origins of spread spectrum (SS) communications go back to the 1920s [7]. Development of practical SS systems did not start until the 1950s and was mainly for military applications. In recent years, commercial application of SS systems has been increasing. Previously classified

results were made available and are being used in non-hostile environments using new and cost-effective technology [8][9][10][11][12]. The IS-95 cellular system that is considered in this dissertation is an example.

An SS system is defined as a system that transmits an information signal using a radio-frequency bandwidth significantly greater than the information bandwidth. In addition, the bandwidth of the information signal is spread using a code, which is independent of the information. Reception of an SS signal is achieved with a receiver-generated, synchronized copy of the code used at the transmitter.

SS signals are characterized by noise-like appearance and low power spectral density, which makes them ideal for covert transmissions. They are also very robust against jamming, interference, multipath propagation, and frequency-selective fading.

Since the spreading code has to be available at both the transmitter and the receiver, it cannot be a purely random signal. In practice, so called pseudorandom noise (PN) sequences are used. They are completely deterministic and easily generated using shift-registers. Maximal length sequences (m-sequences) that are generated by an M-stage linear feedback shift-register (LFSR) have a period of 2^M-1. These sequences approximate the random properties of binary random sequences very well. Also, long sequence periods can be implemented relatively easy. Acquisition and tracking of the spreading sequence timing at the receiver is a fundamental task for implementing an SS system.

The two main methods used to spread the spectrum of a signal are direct-sequence (DS) and frequency hopping (FH).

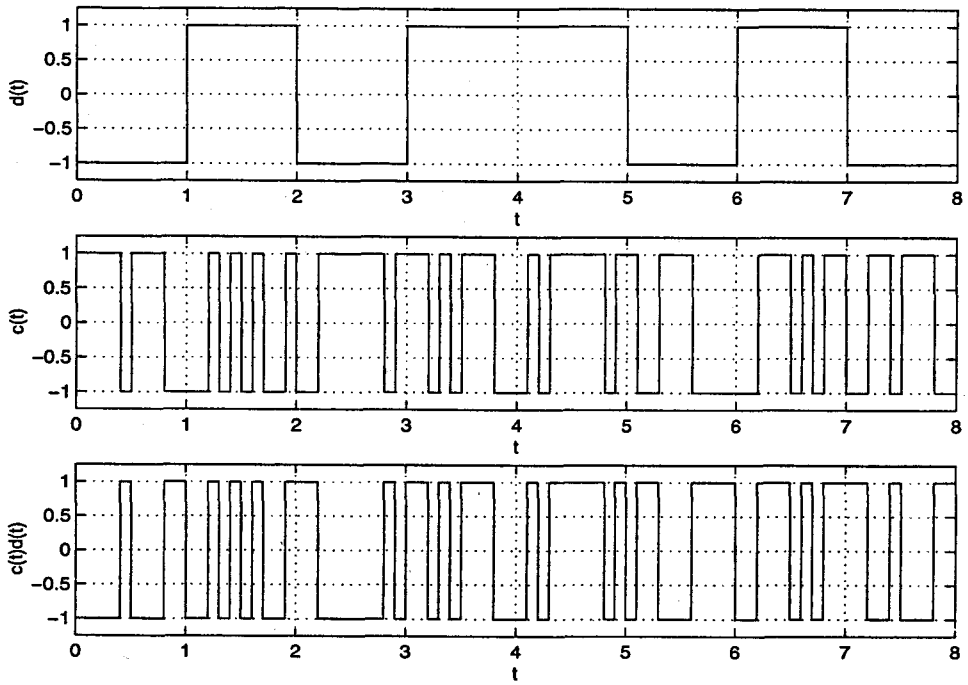


Figure 1.2: Time waveforms for generating a DS-SS signal. From top to bottom: data waveform $d(t)$, spreading code $c(t)$, SS signal $d(t)\cdot c(t)$.

For DS-SS, a data modulated signal is modulated a second time by a wideband spreading waveform. The spreading waveform is generated by a PN sequence.

To illustrate the effects of DS-SS consider the following example. A binary data signal $d(t)$ is multiplied with a higher rate spreading signal $c(t)$ to create a DS-SS signal $d(t)\cdot c(t)$ as shown in Figure 1.2. The duration of the data symbols T_d was set to unity. Also, it is assumed that both the data and the spreading sequence are binary random sequences. The individual symbols of $c(t)$ are called chips. T_c is the chip time or chip duration. Notice that the spread signal $d(t)\cdot c(t)$ appears random, just like the spreading waveform $c(t)$.

If the spreading sequence is known, the SS signal can be multiplied with a time-synchronized replica of the code and the original data signal is recovered. This operation is referred to as despreading of the signal.

The data signal $d(t)$ with rate $R_d = 1/T_d$ bits/s has the power spectrum

$$S_d(f) = T_d\left(\frac{\sin(\pi f T_d)}{\pi f T_d}\right)^2 .$$

The spreading signal $c(t)$, having a higher rate $R_c = 1/T_c$, and the DS-SS signal $d(t)\cdot c(t)$ both have the power spectrum

$$S_c(f) = T_c\left(\frac{\sin(\pi f T_c)}{\pi f T_c}\right)^2 .$$

The effect of the signal spreading on the power spectrum of the data signal is shown in Figure 1.3 for a spreading rate $R_c = 10\cdot R_d$ chips/s or equivalently $T_c = T_d/10$. The frequency spectrum is expanded proportionally to the ratio of the chip rate R_c and the data rate R_d. The power spectral density is reduced by the same amount.

This ratio is often defined as the *processing gain* of the SS system. More generally, the processing gain is a measure of the performance improvement obtained by using SS compared to when SS is not used. This is assuming that everything else remains the unchanged.

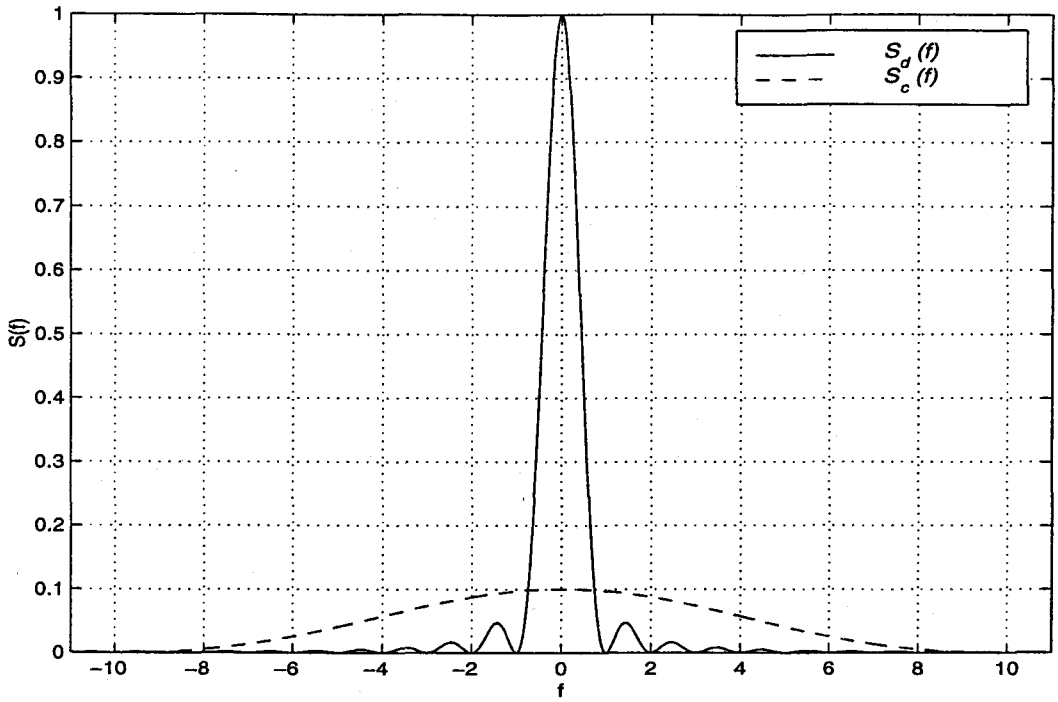


Figure 1.3: Power spectra of data signal $d(t)$ and spreading waveform $c(t)$.

During the despreading of the signal, any narrowband interference that is present in the signal bandwidth will be spread just like the data signal at the transmitter. The interference that remains in the bandwidth of the recovered data signal is reduced by an amount proportional of the processing gain.

DS-SS can be used in conjunction with any form of phase modulation like quadrature phase-shift keying (QPSK) or minimum-shift keying (MSK). In theory, analog modulation is also possible, but this is not used in practice.

Multiple copies of a DS-SS signal, time-offset from one another by more than a chip time T_c, are approximately uncorrelated. This fact lead to the development of the RAKE diversity receiver concept, which was first introduced in 1958 by R. Price and P. E. Green [13]. In a

fading multipath channel, a RAKE receiver has the ability to demodulate and combine several strong multipath signals. This effectively increases the S/I ratio of the received signal.

A frequency-hop (FH) SS system transmits a signal using a set of available carrier frequencies, which are selected based on a PN sequence. If the carrier frequency changes at a rate higher than the data rate, the system is fast FH, else it is slow FH. Compared to a DS-SS system the bandwidth of the transmitted signal remains instantaneously narrowband. On average however, the signal bandwidth is expanded and the power spectral density is reduced as before. In this case, the degradation due to narrowband interference is reduced because it is avoided by the system. An advantage of FH is that it does not require a contiguous band of frequency spectrum. For high data rates, fast FH is difficult due to implementation issues of the carrier frequency synthesizer. But even slow FH can provide performance improvement in systems where the S/I on any available carrier can be high. For example, slow FH is implemented in the GSM standard [6].

The performance of both DS-SS and FH-SS systems is degraded in the presence of broadband interference. Therefore some form of channel coding and interleaving is implemented in most SS systems. This becomes especially important when DS-SS is used to share a wideband channel among many users, i.e., in a CDMA system.

Many textbooks and articles about SS have been published. A good tutorial of SS can be found in [14]. For an even more detailed introduction into the theory of SS communications the reader is referred to [15].

1.3. Code-Division Multiple Access

Any system where a number of users share a given frequency spectrum simultaneously has to implement some form of multiple access technique. Shannon's channel capacity theorem limits the maximum amount of information that can be transmitted error-free in additive white Gaussian noise (AWGN) interference using a given bandwidth [16]. Therefore, theoretically, the maximum number of users that can share the available bandwidth in AWGN is independent of the used multiple access method [17]. In practical systems, the differences are due to implementation issues and the effects of the mobile radio channel. Traditionally most systems use frequency-division multiple access (FDMA), time-division multiple access (TDMA) or combinations thereof.

In a system using FDMA, the available bandwidth is divided into narrowband channels. When requested, a channel is exclusively assigned to a user. Provided that there is an appropriate separation between channels (guard bands, channel allocation strategy), the interference between users is minimized. FDMA has to be used for analog systems.

If the system is digital, TDMA can be implemented. A channel is time-shared between several users by assigning each user a different time slot. Such a system depends on synchronization between users. Guard times between time slots simplify this task in practical applications.

Code-division multiple access (CDMA) is a multiple access scheme based on DS-SS. Many users simultaneously transmit on a wideband channel. Instead of being separated in frequency or time, they are separated by assigning each user a different spreading code. The signal of the user of

interest is recovered by despreading the received signal with the user's spreading code. The other users' signals are approximately uncorrelated with the signal of the desired user and add to the interference. In fact, this multiple access interference (MAI) is the major source of interference in a CDMA system and limits the system capacity.

These three different multiple access schemes are illustrated in Figure 1.4.

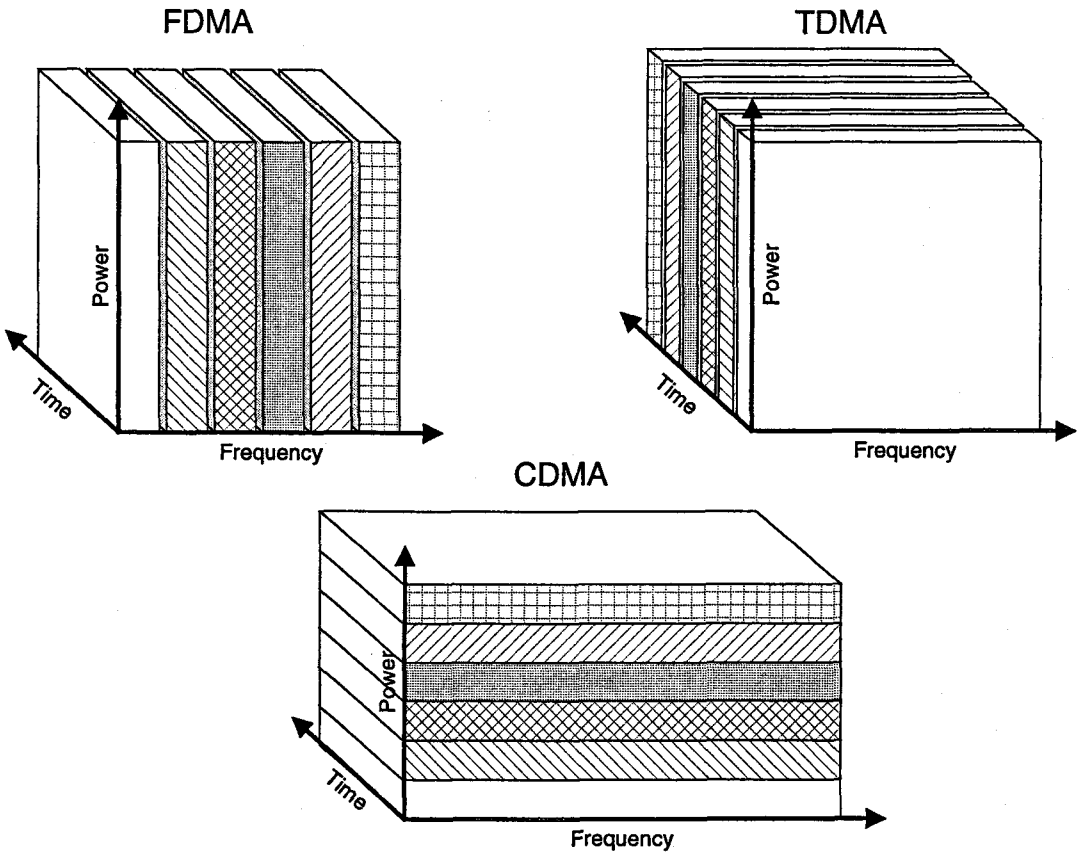


Figure 1.4: Multiple access schemes: Frequency-division multiple access (FDMA), time-division multiple access (TDMA), and code-division multiple access (CDMA).

CDMA has several unique features that make it attractive for mobile communications [18]. Guard times within the allocated channel resource, which are required for TDMA, are not necessary. The use of different spreading codes for each user inherently provides a level of privacy. Using a RAKE receiver, an SS system has the ability to mitigate the deteriorating effects of the mobile environment, which is characterized by multipath propagation and fading [19].

In a cellular CDMA system, the same frequency band can be reused in all cells resulting in a frequency reuse factor of 1. This *universal frequency reuse* is probably the most important feature of cellular CDMA. As a result, the capacity of such a system is strictly interference limited. Any reduction of system interference translates directly into an increase of system capacity. For example, the voice activity of users, which is approximately 35%-50% [20], can be exploited. The transmitter power can be reduced during periods of silence, thereby reducing the interference to the other users by as much as 65%. Another example is sectorization of cells, which also reduces the interference. If three 120° directional antennas are used, the interference is reduced by a factor of three. While sectorization is also used in non-CDMA systems to reduce cochannel interference, no degradation of the trunking efficiency results in a CDMA system.

Frequency management and allocation in a CDMA system is not necessary since all cells are using the same wideband channel. Different cells or sectors are separated by assignment of different spreading sequences. It is therefore relatively easy to add new cells to a system or

introduce sectorization to existing cells. Several other advantages of CDMA can be found in [17][21].

The capacity of a cellular CDMA system was studied in [22]. It is determined by the minimum bit-energy-to-interference-density (E_b/N_0) ratio[2] required by the receiver/decoder to achieve a certain bit error rate (BER) or frame error rate (FER). Once the system interference is minimized, capacity can be increased by new receiver designs and/or modifications to existing modulation schemes that result in a reduction of required E_b/N_0.

As already mentioned, the capacity-limiting factor of a cellular CDMA system is the MAI. This is sometimes referred to as self-jamming of the system.

On the one-to-many link between the base station and the mobile users synchronous transmission of the user signals is possible. In that case the system can be designed to achieve orthogonality between those signals. As a result, the MAI interference of the other users in the same cell can be removed at the receiver.

The many-to-one link between the mobile users and the base station is asynchronous and orthogonality between signals cannot be guaranteed. The signals of the other users in the same cell therefore contribute to the MAI. If all users transmit with the same power, then the signals of nearby users are received with much higher signal strength than the signals of users that are near the cell boundary. The interference caused by these strong signals can severely degrade the performance of the weak users. This is known as the *near-far problem* of a CDMA system.

If conventional single-user detectors are used, it is necessary to control the transmit power of the mobile

[2] N_0 includes all multiple access interference and background noise.

users so that they are received with the same signal strength at the base station. Furthermore, to maximize capacity the received signal strength should be no larger than necessary to achieve acceptable link performance. The implementation of effective power control is an important task in a CDMA system [23].

As already pointed out in [24], a single-user detection approach was chosen for the base station implementation of the IS-95 system. Receiver circuits for each user operate on the received signal independently from each other. In general, there are other methods for detection of these signals, e.g., multiuser detection and interference cancellation.

Multiuser detectors simultaneously detect the signals of all users. They also require knowledge of all users' spreading sequences and timings. However, this added complexity has advantages. The multiuser detectors can achieve near-far resistance, i.e., solve the near-far problem. They also have better performance than single-user detectors even with perfect power control.

The optimum multiuser detector for the asynchronous AWGN channel was derived by Verdú [25]. It consists of a bank of conventional single-user matched filters followed by a Viterbi algorithm. The complexity of this detector grows exponentially with the number of users. Therefore less complex asymptotically optimum and suboptimum K-user demodulation methods were proposed [26][27][28]. Multiuser detectors that are especially modified for Rician fading channels were also derived [29][30]. These receivers operate on the maximum likelihood principle in that they select the vector of user bits that was transmitted with the highest probability by calculating an appropriate metric. This metric depends on the channel model used and

assumptions about the transmission characteristics (synchronous, asynchronous, etc.).

Another way to solve the near-far problem caused by MAI is by interference cancellation. Interference cancellation usually consists of successive detection of the strongest users' signals, which are then removed from the received signal [31][32][33]. It can also be implemented so that the interference cancellation between users is done simultaneously [34].

Implementation complexity, which is caused by the IS-95 specification of the spreading sequences, and the modulation scheme make the application of these methods unfeasible in IS-95 based systems [24].

The advantages of CDMA led to the development of the IS-95 standard for a cellular CDMA system, which is described in the next section. Also, most of the standards proposed for third-generation (3G) systems incorporate some form of CDMA, generally using an even larger signal bandwidth than the IS-95 standard.

1.4. IS-95 Interim Standard

1.4.1. General Information

IS-95, an interim standard that was developed by Qualcomm, Inc. [35] and adopted by the Telecommunications Industry Association (TIA) in 1993 [36], represents the first commercial application of CDMA SS technology for digital mobile communications. Making use of several features that are unique to a CDMA system, initial estimates claimed an up to twenty-fold increase in system capacity over of the first generation analog AMPS system.

Implementation issues reduce this claim to about ten times the capacity of AMPS in the mobile environment [37], which still represents a considerable increase in system capacity.

A first revision of the standard, IS-95-A, was published in May of 1995. Worth mentioning here is the introduction of Rate Set 2, a second set of transmission rates in the physical layer definition of the standard. This allows the use of a 13 kbits/s vocoder with superior voice quality than the 8 kbits/s vocoder used with the original data rates (Rate Set 1), which remain available.

Also in 1995, CDMA was selected as a standard (ANSI J-STD-008-1996) for Personal Communications Services (PCS) [38]. This standard differs from IS-95-A only a little (frequency-plan, network issues).

The IS-95 standard covers the radio system parameters and call-processing procedures for dual-mode operation of mobile receivers/transmitters, i.e., mobiles that are capable of both analog and digital operation.

1.4.2. IS-95 Physical Layer

The physical layer for digital data transmission on the downlink and the uplink is specified in the standard.

The channel bandwidth is 1.25 MHz, which corresponds to 10% of the bandwidth available to cellular service providers in the United States. Since several of these channels are available in the allocated frequency spectrum, the IS-95 system is strictly speaking an FD/CDMA system. The spreading sequence chip rate used is 1.2288 Mchips/s. The maximum data transmission rate including

overhead is 9.6 kbits/s resulting in a system processing gain of 128.

Several channel types are used in the system. The downlink uses a pilot channel, a sync channel, paging channels, and forward traffic channels. The pilot channel is shared between all users of a common base station and allows for easy signal acquisition and coherent demodulation at the mobile station. The optional sync channel is used for initial time synchronization. A paging channel is used to transmit system overhead information and mobile station specific messages. The forward traffic channels are used for user and signaling data.

The uplink uses two different types of channels: access channels and uplink traffic channels. The access channels are used to initiate communication to a base station and also to respond to messages received on a downlink paging channel. The traffic channels are used for transmission of user and signaling information to the base station.

The modulation for all channel types is specified in the standard. Generally, data transmissions are grouped into frames with a duration of 20 milliseconds and have transmission rates of 9.6 kbits/s, 4.8 kbits/s, 2.4 kbits/s, and 1.2 kbits/s[3]. Depending on the channel type and rate, transmitted data is protected by different combinations of cyclic redundancy codes (CRC), convolutional codes, interleavers, and orthogonal modulation.

The system uses two types of pseudo-random spreading sequences. The long code spreading sequence is based on a 42-bit shift register and is mostly used for privacy in the downlink. In the uplink, it is used also for privacy

[3] This is Rate Set 1.

but most importantly to identify and separate user signals. An in-phase and a quadrature short code spreading sequence are also used. They are based on 15-bit shift-registers with different generator polynomials and have a period of 26.666… ms. Their purpose is to separate the in-phase and quadrature components of the transmitted signals and the signals transmitted by different base stations. The latter is achieved by using different sequence offsets for each base station. Due to their fast repetition rate, they are also used for signal timing acquisition.

The downlink traffic channel uses a CRC for the two higher data rates, i.e., 9.6 kbits/s and 4.8 kbits/s. Eight zero tail-bits are added to the data, which is encoded using a convolutional code with rate: $r = 1/2$ and constraint length: $K = 9$. Code symbol repetition is used for the lower data rates resulting in a constant rate of 19200 modulation symbols per second. The modulation symbols are interleaved using a block interleaver spanning the entire 20 ms frame. The interleaved modulation symbols are scrambled using a decimated version of the long code spreading sequence. For closed-loop power control of the received mobile signal power, an 800 Hz power control channel is multiplexed onto the scrambled modulation symbols. The base station measures the received uplink signal power and transmits a power control bit instructing the mobile station to either decrease or increase its transmitter power by a fixed amount. Finally, each modulation symbol is spread using one of 64 Walsh sequences at the chip rate, spread in quadrature using the short code sequences, baseband filtered, and transmitted using quadrature phase shift keying (QPSK). All user signals are transmitted synchronized. Since the Walsh sequences that are used for signal spreading are

orthogonal to each other, this effectively eliminates all multiple access interference (MAI) from active users of the same base station. The received interference at the mobile stations is therefore determined by the background noise, signals from neighboring base stations, and other interference.

The specified modulation for the uplink traffic channel is relevant to this dissertation and summarized with more details in the following subsection.

1.4.3. Uplink Traffic Channel Modulation

The uplink traffic channel modulation differs significantly from the downlink modulation. Fist, no pilot channel is available for signal acquisition and coherent demodulation. Therefore, the uplink was specifically designed for noncoherent demodulation at the base station. Signal acquisition is generally done in a noncoherent fashion but is more difficult without the pilot channel. Second, the signals of the mobile users arrive at the base station with constantly changing relative time delays since they are transmitted independently from each other and the mobile stations are moving. Thus, orthogonality between received signals cannot be achieved, resulting in MAI also from active users in the cell of interest. For these reasons, the uplink represents the more challenging communication link of the system. To compensate for this, a more powerful (complex) convolutional code is used in the uplink. At the same time, more complex receiver signal processing is possible at the base station without the constraints of a small, battery-operated device.

The uplink traffic channel also uses a CRC for the two higher data rates, i.e., 9.6 kbits/s and 4.8 kbits/s. Eight zero tail-bits are added to the data, which is encoded using a convolutional code with rate: $r = 1/3$ and constraint length: $K = 9$ (Section 1.4.3.1). Code symbol repetition is used for the lower data rates resulting in a constant rate of 28800 convolutionally coded bits per second. They are then interleaved using a block interleaver spanning the entire 20 ms frame (Section 1.4.3.2). To allow for noncoherent detection at the base station, the interleaved convolutionally coded bits are 'modulated' using 64-ary orthogonal Walsh modulation (Section 1.4.3.3). On the downlink, the repeated convolutionally coded bits are all transmitted and the symbol energy is varied so that each data bit is transmitted with the same bit energy. The uplink uses data burst randomizing. Each convolutionally coded bit is transmitted only once with energy $E_b/3$. An algorithm based on the long code spreading sequence and the frame data rate determines which of the multiple copies is transmitted. For the other symbols, the transmitter output stage is gated-off. Finally, four chips of the long code spreading sequence spread each Walsh code chip. The signal is then spread in quadrature using the short code sequences, baseband filtered, and transmitted using offset quadrature phase shift keying (O-QPSK). The use of O-QPSK allows more efficient power amplification of the transmitted signal, which is an important issue for mobile telephone handsets.

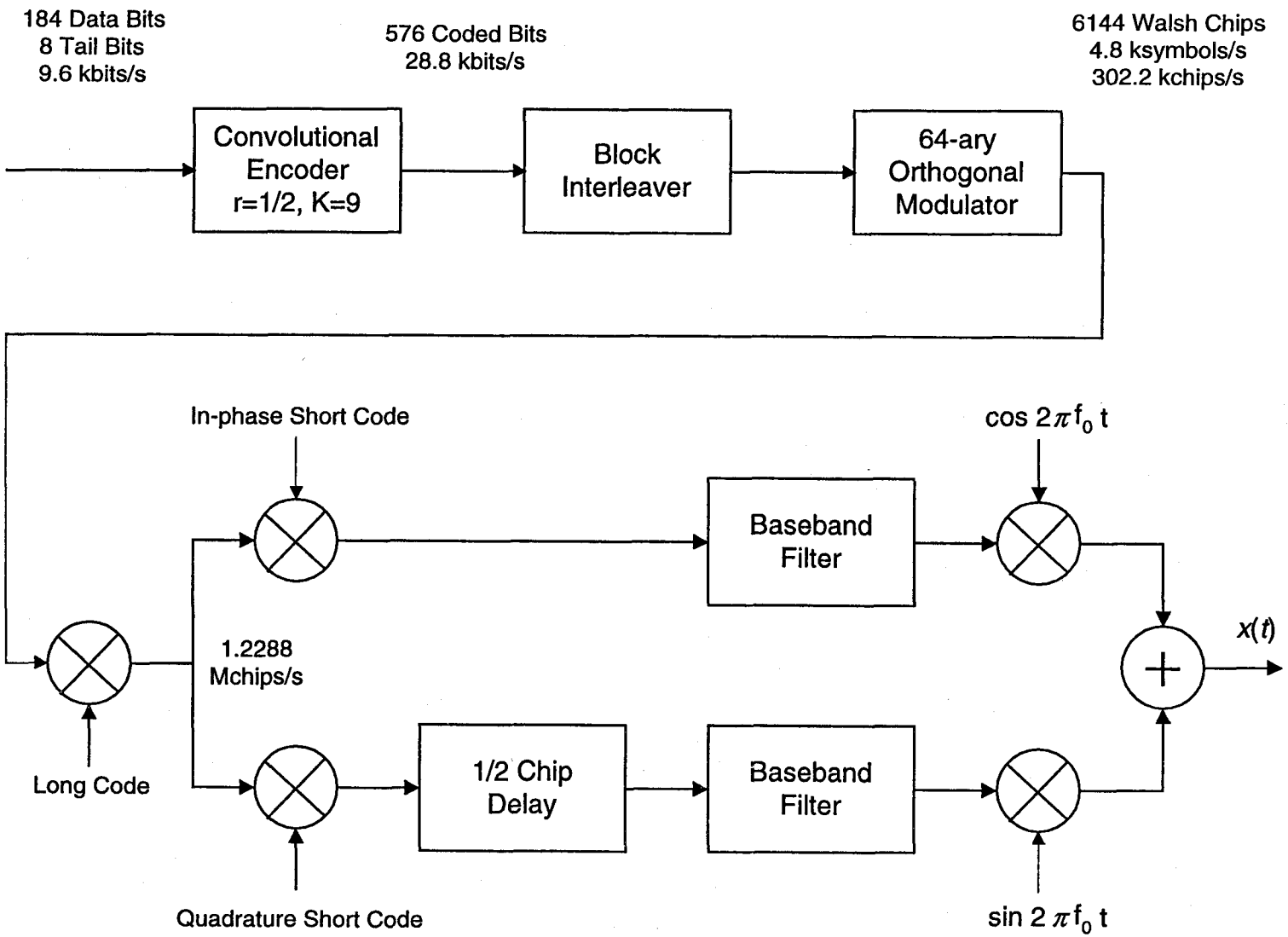


Figure 1.5: Simplified block diagram of the IS-95 9.6 kbits/s uplink traffic channel.

The highest data rate (9.6 kbits/s) of the uplink traffic channels is considered in this dissertation. Each frame consists of 192 data bits, of which 184 contain CRC protected data and 8 are encoder tail-bits that are set to zero so that the convolutional encoder returns to the zero state at the end of each frame. This effectively makes the convolutional code into a block code. The input to the Walsh modulator is a sequence of 576 interleaved convolutionally coded bits. Code symbol repetition as well as data burst randomizing do not apply to frames using this data rate. The corresponding simplified block diagram of the IS-95 uplink modulation is shown in Figure 1.5.

The following subsections give additional information about the convolutional code, the interleavers, and orthogonal Walsh modulation. This information is helpful for understanding the DF receiver/decoder design.

1.4.3.1. Uplink Convolutional Code

The protected data bits that are transmitted on the IS-95 uplink traffic channel are channel coded with a convolutional code that can be represented by the shift-register structure shown in Figure 1.6. This process adds substantial redundancy to the transmitted signal that can be used at the receiver to reduce the probability of transmission error.

The code rate r of this code is 1/3, meaning that three convolutionally coded bits (c_0, c_1, and c_3) are generated for each data bit. The output rate of the encoder is three times the input data rate. Each convolutionally coded bit is a parity check on the current input bit and the eight previous input bits.

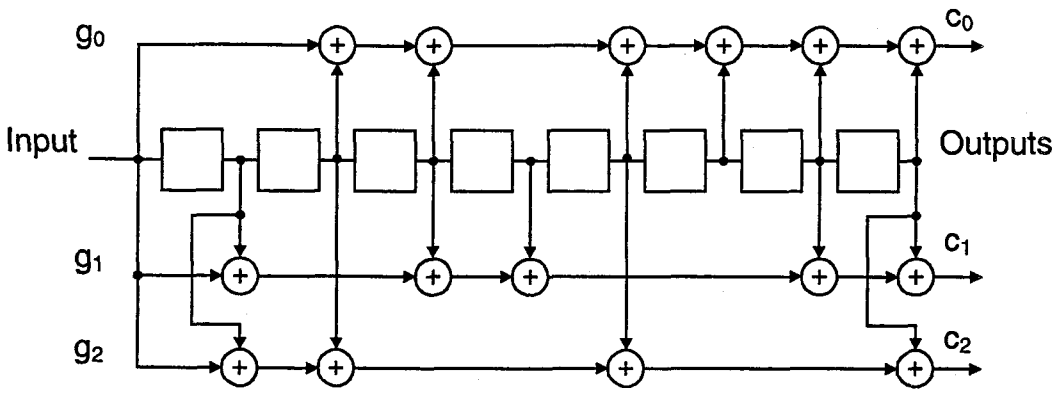


Figure 1.6: IS-95 uplink convolutional encoder.

The constraint length K of this code is therefore 9. The generator polynomials (g_0, g_1, and g_2) completely specify the structure of the convolutional code. They determine which of the bits in the shift-register are modulo-2 added[4] in order to generate the three convolutionally coded bits. The generator polynomials for this code are $g_0=(557)_8$, $g_1=(663)_8$, and $g_2=(711)_8$. The binary representations of these numbers indicate the presence or absence of a connection of each shift-register position (from right to left in Figure 1.6). As each data bit travels through the shift-register, it affects the value of 18 convolutionally coded bits.

A convolutional encoder is a finite-state machine. The encoder state can be defined as the binary number formed by the contents of the K-1 leftmost shift-register positions. The encoder considered here has 256 possible states. At the beginning of each frame it is assumed that the encoder is in the zero-state, i.e., that the shift-register positions are zero. The zero tail-bits that are

[4] If binary '0' and '1' are mapped into '1' and '-1' respectively, modulo-2 addition is replaced by multiplication.

added to the frame data force the encoder back to the zero-state at the end of each frame. Therefore frame decoding at the receiver begins in the zero-state and should also end in the zero-state. The encoder state transitions and the generated convolutionally coded bits depend only on the input data bit and the current encoder state. This is the structure that is used in the decoder to determine the most likely transmitted sequence of data bits.

1.4.3.2. Uplink Interleavers

The interleavers used in the IS-95 uplink were designed to randomize correlated outputs of the mobile radio channel at the convolutionally coded bit level. This is important for good performance of the convolutional code described in the previous section.

A (32,18) block interleaver is used to scramble the convolutionally coded bits. The interleaver matrix (Figure 1.7) is filled by columns and emptied by rows. Different row scrambling is used for the access channel and also for the different data rates of the traffic channel.

1	33	65	97	129	161	193	225	257	289	321	353	385	417	449	481	513	545
2	34	66	98	130	161	194	226	258	290	322	354	386	418	450	482	514	546
3	35	67	99	131	162	195	227	259	291	323	355	387	419	451	483	515	547
4	36	68	100	132	163	196	228	260	292	324	356	388	420	452	484	516	548
5	37	69	101	133	164	197	229	261	293	325	357	389	421	453	485	517	549
6	38	70	102	134	165	198	230	262	294	326	358	390	422	454	486	518	550
7	39	71	103	135	166	199	231	263	295	327	359	391	423	455	487	519	551
8	40	72	104	136	167	200	232	264	296	328	360	392	424	456	488	520	552
9	41	73	105	137	168	201	233	265	297	329	361	393	425	457	489	521	553
10	42	74	106	138	169	202	234	266	298	330	362	394	426	458	490	522	554
11	43	75	107	139	170	203	235	267	299	331	363	395	427	459	491	523	555
12	44	76	108	140	171	204	236	268	300	332	364	396	428	460	492	524	556
13	45	77	109	141	172	205	237	269	301	333	365	397	429	461	493	525	557
14	46	78	110	142	173	206	238	270	302	334	366	398	430	462	494	526	558
15	47	79	111	143	174	207	239	271	303	335	367	399	431	463	495	527	559
16	48	80	112	144	175	208	240	272	304	336	368	400	432	464	496	528	560
17	49	81	113	145	176	209	241	273	305	337	369	401	433	465	497	529	561
18	50	82	114	146	177	210	242	274	306	338	370	402	434	466	498	530	562
19	51	83	115	147	178	211	243	275	307	339	371	403	435	467	499	531	563
20	52	84	116	148	179	212	244	276	308	340	372	404	436	468	500	532	564
21	53	85	117	149	180	213	245	277	309	341	373	405	437	469	501	533	565
22	54	86	118	150	181	214	246	278	310	342	374	406	438	470	502	534	566
23	55	87	119	151	182	215	247	279	311	343	375	407	439	471	503	535	567
24	56	88	120	152	183	216	248	280	312	344	376	408	440	472	504	536	568
25	57	89	121	153	184	217	249	281	313	345	377	409	441	473	505	537	569
26	58	90	122	154	185	218	250	282	314	346	378	410	442	474	506	538	570
27	59	91	123	155	186	219	251	283	315	347	379	411	443	475	507	539	571
28	60	92	124	156	187	220	252	284	316	348	380	412	444	476	508	540	572
29	61	93	125	157	188	221	253	285	317	349	381	413	445	477	509	541	573
30	62	94	126	158	189	222	254	286	318	350	382	414	446	478	510	542	574
31	63	95	127	159	190	223	255	287	319	351	383	415	447	479	511	543	575
32	64	96	128	160	191	224	256	288	320	352	384	416	448	480	512	544	576

Figure 1.7: IS-95 uplink block interleaver matrix.

For the full-rate traffic channel (9.6 kbits/s) the rows are read in their original order. It follows then from Figure 1.7 that the convolutionally coded bits are transmitted in the following order: 1, 33, 65, 97, and so on. As a result, consecutive convolutionally coded bits are separated 18 convolutionally coded bits or 625 μs during transmission. Also, severe fades of the channel that cover less than 19 convolutionally coded bit times are separated 31 convolutionally coded bits after deinterleaving.

1.4.3.3. Orthogonal Walsh Modulation

The interleaved convolutionally coded bits are 'modulated' using 64-ary orthogonal Walsh modulation.

In general, M-ary orthogonal Walsh modulation is performed by replacing a group of $\log_2 M$ binary symbols with one of M Walsh sequences. The Walsh sequences can be generated using the recursion

$$H_1 = 1, \quad H_M = \begin{bmatrix} H_{M/2} & H_{M/2} \\ H_{M/2} & -H_{M/2} \end{bmatrix}, \quad M = 2, 4, 8, 16, \ldots$$

The Walsh sequences or Walsh codes are given by the rows or columns of the matrix H_M. They are mutually orthogonal, i.e.,

$$\sum_{i=1}^{M} h_{ij} \cdot h_{ik} = \begin{cases} M & j = k \\ 0 & j \neq k \end{cases}, \quad 1 \leq j, k \leq M,$$

where h_{ij} is the element in the i-th row and the j-th column of the matrix H_M. Detection of such a signal at the receiver is performed using correlation, which does not require a coherent receiver front-end, i.e., the signal phase is not needed to demodulate the interleaved convolutionally coded bits[5].

Here, groups of six interleaved convolutionally coded bits are replaced by one of sixty-four Walsh codes. Each Walsh code consists of 64 Walsh code chips. A group of interleaved convolutionally coded bits that selects a transmitted Walsh code will be referred to as a Walsh

[5] Performance is degraded for a noncoherent receiver front-end due to the inherent squaring operation.

group. There are 96 Walsh groups in each 20 ms frame. The interleaved convolutionally coded bits in a Walsh group will also be referred to as the Walsh bits of the Walsh group.

Selection of the transmitted Walsh codes is done by interpreting the Walsh bits in each Walsh group as a binary number (least significant bit first). The corresponding Walsh code (row or column) from the matrix H_{64} is transmitted.

As a result of the 64-ary orthogonal modulation, the 576 interleaved convolutionally coded bits are replaced by 6144 Walsh code chips. In addition to allowing noncoherent detection of the signal, this is equivalent with block encoding of the transmitted interleaved convolutionally coded bits. Therefore, the IS-95 uplink employs an interleaved concatenated coding scheme, which is the basis for the decision feedback decoding (DFD) method that is presented in the next chapter.

The remainder of this dissertation is organized as follows:

The concept of DFD is presented in Chapter 2, together with a summary of prior results and an analysis of DF effectiveness.

Chapter 3 describes the nature of the CDMA signal and introduces the simulation model, which was used to obtain the simulation results presented in the later chapters.

In Chapter 4, the performance of the DF decoder is assessed in unfaded as well as Rayleigh fading multipath propagation in additive white Gaussian noise (AWGN) interference. Simulation results using coherent and noncoherent detection are presented for both independent

Rayleigh fading and Rayleigh fading with an applied Doppler spectrum according to Clarke's model [39]. The results show improved performance compared to conventional non-DF receivers using the same decoding metric, which is a prerequisite for application of the DF decoder in an actual mobile communications environment.

The effectiveness of the initial DF decoder design, as it is applied to IS-95 based systems, is studied. It is found that the effectiveness of the DF decoder is determined by the decoding delay of the convolutional decoder and the interleaver specification. Based on these findings, two methodologies to improve the effectiveness of the DF decoder are investigated. In Chapter 5, the average decoding delay is reduced using sub-optimal convolutional decoding. In Chapter 6, the combination of a new block interleaver design and the DF decoder is considered. Simulation results of average decoding delay, bit error rate (BER) and frame error rate (FER) are presented for coherent and noncoherent detection of unfaded and Rayleigh fading multipath signals. It is shown that both approaches result in better system performance, which can further improve the quality of service and/or capacity of an IS-95 based system.

A simplified analysis of the DF decoder performance is presented in Chapter 7.

Finally, Chapter 8 summarizes the results and contributions of this dissertation and indicates areas for future research.

Chapter 2. Decision Feedback Decoding

2.1. Prior Work and Results

One of the characteristics of the IS-95 uplink signal processing is the concatenation of convolutional coding, interleaving, and orthogonal Walsh modulation. As for other product or concatenated codes, e.g., 'Turbo' codes, this can be exploited using iterative decoding techniques [40], which improves the performance of the IS-95 uplink as demonstrated in [41][42].

The non-iterative decision feedback (DF) receiver/decoder that is presented in this dissertation was first introduced by Rabinowitz [24] as a method to combine the orthogonal and convolutional decoding of IS-95 uplink traffic channel frames.

In addition to an introduction into the topics of code division multiple access (CDMA), the IS-95 standard, and Viterbi decoding, [24] presented simulation results for synchronous multiple access interference (MAI), but no multipath and fading effects were considered.

Several decoding metrics used within the convolutional decoder were also studied. From those, the one resulting in the best decoder performance is used here.

The simulation results showed consistently better bit error rate (BER) and frame error rate (FER) performance than conventional non-DF receivers that use the same decoding metric. This was the motivation for the continued studies in this dissertation.

Finally, [24] outlined the DF decoder implementation complexity and gave a qualitative analysis of the DF decoder performance.

2.2. Decoder Description

A simple single-user detector performs the inverse steps of the transmitter. A coherent or noncoherent receiver front-end provides the correlations of the received signal corresponding to a transmitted Walsh group. Decoding metrics for the interleaved convolutionally coded bits are computed using those correlations. After the entire frame has been received the metrics that were computed for the interleaved convolutionally coded bits are deinterleaved and used as the input to a convolutional decoder using the Viterbi algorithm [43][44].

A block diagram of the DF decoder design is shown in Figure 2.1. Instead of decoding the orthogonal Walsh codes and the convolutionally coded bits separately, the matrix $\boldsymbol{W}$ containing the Walsh code correlations of all 96 Walsh groups are used as the input of the convolutional decoder. The decoder performs 192 decoding steps (one for each data bit). During each decoding step, the required branch metrics are computed based on the correlation values of the three Walsh groups containing the current three convolutionally coded bits. One-to-one mappings from convolutionally coded bit number to Walsh group number and Walsh group bit number, stored in lookup tables, can be used here. In a practical decoder implementation, it might be advantageous to perform a pre-computation of the decoding metrics during frame reception. Then, only metric updates are needed during the decoding operation.

Generally, for a tailed-off convolutional code, the data bits are determined by performing all 192 decoding steps and then using the stored bit-history for the all-zeros state.

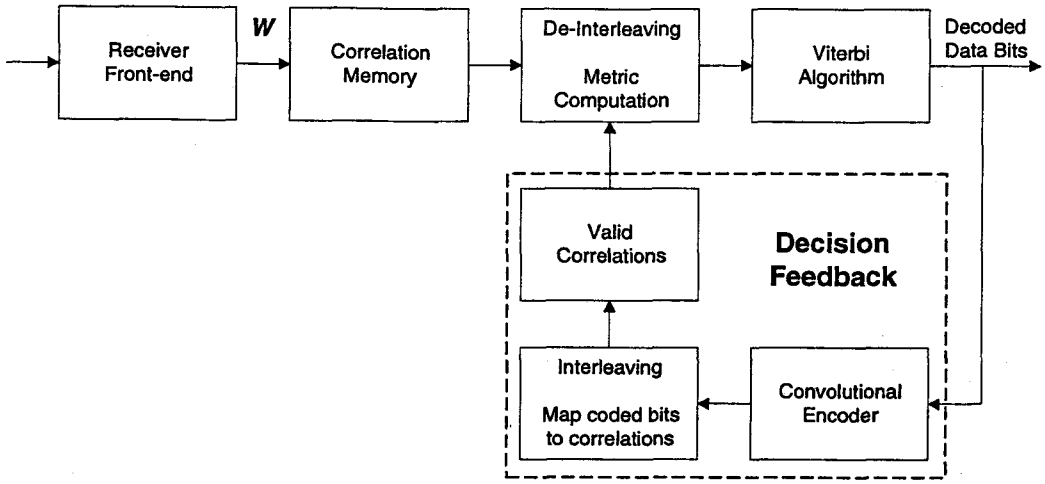


Figure 2.1: Decision feedback receiver block diagram.

However, this method does not accommodate the non-iterative decision feedback decoding (DFD) technique.

Instead, the Viterbi decoder decides on the value of a data bit as soon as all 256 bit-histories of the decoder path-memory converge to the same data bit value or at the end of the frame[6]. The decoder bit-histories are examined after each decoding step to determine if they converge. If so, a decision on one or more data bits can be made. This way data bit decisions are made as soon as possible in order to allow DF metric updates during frame decoding without any performance degradation.

Initially, all 64 Walsh code correlation values are considered (equally likely) for the computation of the decoding metrics. Decoded data bits are convolutionally re-encoded starting in the zero state at the beginning of each frame. Walsh code correlation values that are not

[6] The remaining undecided data bits at the end of the frame are chained back from the all-zeros state as usual.

consistent with the convolutionally reencoded bits are invalidated in the corresponding Walsh groups for the remainder of the frame. Assuming that the data bit decision was correct the number of valid correlation values is reduced by half each time[7].

The decoding metrics are then updated using only the remaining valid Walsh code correlation values. Although wrong data bit decisions invalidate correct Walsh code correlation values for at least K (constraint length of the convolutional code) decoding steps, on average the quality of the decoding metrics is improved by the DF metric updates.

Because convolutionally coded bits in a Walsh group are separated by the interleaving operation, the convolutional decoder is able to use additional information from the DF metric updates. This results in a reduction of incorrect data bit decisions.

The obtained performance improvement is a direct result of Shannon's information theory [16]. In [45], Viterbi put it in the form of a fundamental lesson: "Never discard information prematurely that may be useful in making a decision until after all decisions related to that information have been completed."

The reader should refer to [24] for a more detailed description of DFD.

[7] Ideally, if the last convolutionally coded bit contained in a Walsh group is considered and data bit decisions that determine the other convolutionally coded bits have been made, only two correlation values remain for metric computation.

2.3. Decoder Metrics

The performance of the convolutional decoder is much dependent on the decoding metrics that are used. For optimum performance in a memoryless channel, the branch metrics are proportional to the log-likelihood of the branch transitions in the trellis diagram of the convolutional code [46].

If binary decisions on the convolutionally coded bits are made based on the received signal and then passed to the convolutional decoder, the decoder performs hard decision decoding. The branch metric used in this case is simply the Hamming distance of the received convolutionally coded bits and the convolutionally coded bits of the branch transition under consideration.

For better performance, the convolutional decoder uses quantized values of the received signal to compute the branch metrics resulting in soft decision decoding, which is superior to hard decision decoding by about 2-3 dB of required bit-energy-to-interference-density (E_b/N_0) ratio [39].

The decoding metrics that were considered in [24] are:

1) hard decisions
2) log probability of maximum correlation
3) value of maximum correlation of expected symbol
4) square of maximum correlation
5) "dual-maxima" metric
6) square of "dual-maxima" metric

It was found that the metrics using squares (4 and 6) resulted in poorer performance compared to the metrics

they are based on (3 and 5). The log probability metric requires knowledge of the interference level (parametric) with little performance improvement compared to using metric 3. Simulation results were presented for metrics 1 and 3.

For metric 1, the Walsh code with the highest correlation value is chosen as the most likely transmitted and mapped into the corresponding Walsh group of interleaved convolutionally coded bits. The simplest form of a soft metric is obtained by scaling with the maximum correlation value, which improves performance over hard (binary) decisions by about 1 dB of required E_b/N_0.

Metric 3 is similar to the "dual-maxima" metric used in this dissertation (see below). The branch metric Λ_{st} (transition from encoder state s to encoder state t) used in the convolutional decoder approximates the required log-likelihood using only the correlation values $\boldsymbol{W}$ (nonparametric).

Let $\underline{u}_{st} = \left(u_{st}^1, u_{st}^2, u_{st}^3\right)$ be the three convolutionally coded bits that correspond to a state transition then

$$\Lambda_{st} = \sum_{k=1}^{3} \underset{j \in U_k}{Max}\left(w_{ij}\right),$$

where U_k is the set of (valid) Walsh code numbers having u_{st}^k as the value of the p-th Walsh group bit in the i-th Walsh group. The Walsh group number i and the Walsh group bit number p are functions of the convolutionally coded bit number that are determined by the interleaver.

The simulation results presented in this dissertation use the "dual-maxima" metric for block codes [47][48] that is

$$\Lambda_{st} = \sum_{k=1}^{3} \left[\underset{j \in U_k}{Max}(w_{ij}) - \underset{j \in \overline{U}_k}{Max}(w_{ij}) \right],$$

where $\overline{U}_k$ is the set of (valid) Walsh code numbers having $-u_{st}^k$ as the value of the p-th Walsh group bit in the i-th Walsh group. This nonparametric metric is known to be a good approximation of the optimal metric for L equal-strength Rayleigh fading multipath signals [47]. As pointed out in [24][49] the "dual-maxima" metric has basically the same performance as the "single-maxima" metric, which is metric 3 from above.

The effects of quantization of the correlation values $\boldsymbol{W}$ were also studied in [24]. It was found that relatively few bits of resolution are needed for good performance as long as the analog-to-digital (A/D) converter has a range of zero to about twice the expected correlation value (noncoherent case). Quantization of the decoding metrics was not considered in this dissertation.

2.4. Effectiveness of Decision Feedback

In this section, the concept of DF effectiveness is introduced. The term DF effectiveness has two meanings, both of which are used in this dissertation. First, it is used to describe the performance improvement (in E_b/N_0) obtained through the use of DFD. In this sense, effectiveness is interpreted as performance improvement.

Second, effective-ness also refers to the ability of the decoder to take advantage of the decoding metric updates, which occur during the decoding of a frame. This is discussed here. Both meanings are closely related. An analytical justification of the performance improvement obtained with DF is delayed until Chapter 7.

As a result of the 64-ary Walsh modulation used in the IS-95 uplink, 6 interleaved convolutionally coded bits are transmitted within one Walsh modulation symbol (Walsh code). Therefore a DF metric update can occur up to five times for each Walsh group during frame decoding.

The best possible (optimum) scenario is that from the 576 decoding metrics used during frame decoding exactly 96 each have no prior DF update, 1 DF update, and so on. This requires that a DF metric update occurs each time before a Walsh group is reused by the decoder. If this could be achieved, DFD should be most effective, i.e., result in the largest BER and FER improvement.

Exactly how well this can be achieved depends on the interleaver specification and the data bit decoding delays of the convolutional decoder.

The interleaver specification determines the separation of the convolutionally coded bits within Walsh groups, or equivalently, the number of decoding steps after which the Walsh groups are reused during frame decoding. Using the interleaver specified in IS-95 for the 9.6 kbits/s rate reverse traffic channel, as described earlier (Section 1.4.3.2), consecutive convolutionally coded bits are separated by 3 Walsh groups during transmission. The convolutionally coded bits in each Walsh group are separated by 32 such bits. As a result, the correlations

of each Walsh group are being reused after either 10 or 11 decoding steps[8].

Another effect of the IS-95 interleaver is the separation of the frame decoding process into 3 distinct stages. Each stage exclusively uses only 32 Walsh groups for metric computations. More specifically, during the first 64 decoding steps only 32 Walsh groups are being used. These 32 Walsh groups are not used in the remaining decoding steps. Similarly, decoding steps 65-128 and 129-192 exclusively use 32 Walsh groups. Each of the three stages begins with no DF information available. At the beginning of a new stage, DF updates of Walsh groups that are no longer used have no benefit and can be skipped in the decoder implementation. This slightly reduces the effectiveness of DFD because some information has to be discarded.

The decoding delay of a data bit is defined as follows:

$$d(n) = m - n,$$

where n is the data bit and decoding step number, and m is the decoding step after which a decision on data bit n can be made ($m \geq n$).

The decoding delays of the data bits in a frame depend on the received signal level and the interference. Therefore they are discrete random variables with integer values ranging from under one constraint length K up to $192-n$. The decoding delays are not independent since the decoder waits until data bit n has been determined before

[8] It is interesting to note that, since the number of rows of the (32,18) block interleaver is not an integer multiple of 3 but instead 32 / 3 = 10.333…, approximately 1/3 of the Walsh groups are reused after 10 decoding steps and approximately 2/3 of the Walsh groups reused after 11 decoding steps.

checking if a decision on data bit $n+1$, $n+2$, and so on, can be made.

These two determining factors of DF effectiveness suggest the performance improvements of the DF decoder presented in Chapter 5 and Chapter 6.

Chapter 3. Simulation Model

3.1. QPSK CDMA Analysis and Assumptions

The transmitted signal of a mobile is given by

$$x(t) = \mathrm{Re}\left\{ \left[\sum_n \sqrt{E_c}\, x[n]\, a[n] \left(a_I[n] - j a_Q[n]\right) h(t - nT_c) \right] e^{j(2\pi f_0 t + \Theta_c)} \right\},$$

where E_C is the chip energy, $x[n]$ is the Walsh code chip sequence (indexed at chip rate, i.e., remains constant over 4 chips corresponding to one Walsh code chip), $a[n]$ is the long code spreading sequence, $a_I[n]$ and $a_Q[n]$ are the short code in-phase and quadrature spreading sequences, respectively, T_C is the chip time of the spreading sequences, f_0 is the carrier frequency, and Θ_0 is the carrier phase.

$h(t)$ is the impulse response of the pulse-shaping (lowpass) filter used to contain the signal inside the assigned channel bandwidth. It is normalized so that

$$\int_{-\infty}^{\infty} |H(f)|^2 df = 1 .$$

The signal is received at the base station with amplitude α and time delay t_d, i.e.,

$$y_r(t) = \mathrm{Re}\left\{ \left[\sum_n \alpha\sqrt{E_c}\, x[n]\, a[n] \left(a_I[n] - j a_Q[n]\right) h(\tilde{t} - nT_c) \right] e^{j(2\pi f_0 \tilde{t} + \Theta)} \right\},$$

where $\tilde{t} = t - t_d$ and $\Theta = \Theta_0 - 2\pi f_0 t_d$. It is down-converted to baseband by multiplication with a locally generated version of the carrier frequency that is characterized by a frequency error Δf and a phase error $\Delta\Theta$. The resulting baseband signal is then given by

$$y(\tilde{t}) = \mathrm{Re}\left\{ \left[\sum_n \frac{\alpha\sqrt{E_c}}{2} x[n]\, a[n] \left(a_I[n] - j a_Q[n]\right) h(\tilde{t} - nT_c) \right] e^{-j\left(2\pi\Delta f_0 \tilde{t} + \Delta\Theta\right)} \right\}.$$

The factor 0.5 represents the fact that half of the signal is shifted to twice the carrier frequency by the downconversion operation. These signal components are removed by appropriate filtering. Here they are filtered out by the following lowpass filter.

The frequency error Δf can be assumed to be small, since there are strict requirements specified in the IS-95 standard for the carrier frequency of the mobile. Specifically, "the mobile station transmit carrier frequency shall be 45.0 MHz ± 300 Hz lower than the frequency of the base station transmit signal as measured at the mobile station receiver" [36]. This small error will contribute slow time variability to the phase error $\Delta\Theta$ and therefore the assumption $\Delta f = 0$ is justified.

With this assumption

$$y(\tilde{t}) = \mathrm{Re}\left\{ \left[\sum_n \frac{\alpha\sqrt{E_c}}{2} x[n]\, a[n] \left(a_I[n] - j a_Q[n]\right) h(\tilde{t} - nT_c) \right] e^{-j\Delta\Theta} \right\}.$$

Next, the signal is matched filtered with $h^*(-t)$ (complex conjugate of $h(-t)$) and sampled at the chip rate. The corresponding convolution operation is given by

$$
\begin{aligned}
& h(\tilde{t} - nT_c) * h^*(-\tilde{t})\Big|_{\tilde{t}=mT_c+\Delta\tilde{t}} \\
= & \int_{-\infty}^{\infty} H(f)\, e^{-j2\pi f n T_c} H^*(f)\, e^{j2\pi f \tilde{t}}\, df \Big|_{\tilde{t}=mT_c+\Delta\tilde{t}} , \\
= & \int_{-\infty}^{\infty} |H(f)|^2 \cos\left(2\pi f\left((m-n)T_c + \Delta\tilde{t}\right)\right) df
\end{aligned}
$$

where $\Delta\tilde{t}$ is the synchronization error between the received spreading sequences and the locally generated spreading sequences. If synchronization has been achieved ($\Delta\tilde{t} = 0$) using spreading sequence timing acquisition and tracking circuitry, the received complex signal sequence is given by

$$
y[n] = \frac{\alpha\sqrt{E_c}}{2}\, x[n]\, a[n]\left(a_I[n] - j a_Q[n]\right) e^{-j\Delta\Theta} + n_{ICI}[n] , \qquad (3.1)
$$

where the inter-chip interference is given by

$$
n_{ICI}[n] = \frac{\alpha\sqrt{E_c}}{2} \sum_{\substack{m \\ m\neq n}} \left[x[m]\, a[m]\left(a_I[m] - j a_Q[m]\right) \int_{-\infty}^{\infty} |H(f)|^2 \cos\left(2\pi f (m-n) T_c\right) df \right]
$$

The inter-chip interference is usually very small compared to the other interference (interference from other users, background noise) and can be neglected. For example, MATLAB simulations that were performed using the lowpass filter specified in the IS-95 standard [36] indicated that at a bit-energy-to-interference-density (E_b/N_0) ratio of 6

dB the interference created by the filter results in E_b/N_0 degradation on the order of 10^{-5} dB.

If the cascade of the transmitter and receiver lowpass filters satisfies the Nyquist criterion the inter-chip interference is zero. This can be achieved by using a root Nyquist filter at the transmitter and receiver [50].

3.2. Interference

The previous section described the received signal of the user of interest including only the inter-chip interference. The main source of interference in the CDMA system is the signals of other users. In addition to that, there is interference from background noise, receiver-generated thermal noise, and other sources.

The interference by other users can be divided into interference from users in the same cell and interference from users in neighboring cells. Due to power control, the signals of all users in the cell of interest are received with approximately the same signal amplitude. Assuming the system operates at capacity, the signals from the same cell users will dominate the interference.

Assume for a moment that all same cell user signals arrive at the base station synchronized. Their received complex signal sequences $y_i[n]$ follow from equation (3.1):

$$y_i[n] = \frac{\alpha_i \sqrt{E_c}}{2} x_i[n]\, a_i[n] \left(a_I[n] - j a_Q[n]\right) e^{-j\Delta\Theta_i}, \qquad (3.2)$$

where α_i, $x_i[n]$, $a_i[n]$, and $\Delta\Theta_i$ are, respectively, the signal amplitude, Walsh chip sequence, long code spreading sequence, and phase error of the i-th user.

If it is further assumed that all user signal amplitudes are equal (perfect power control) and that the phase errors are zero, then (3.2) simplifies to

$$y_i[n] = \frac{\alpha\sqrt{E_c}}{2} x_i[n]\, a_i[n] \left(a_I[n] - j a_Q[n]\right).$$

This is the type of interference that was studied in [24]. When the correlation values of the user of interest are computed, the contributions of the remaining same cell users consist of binomial random variables, i.e., sums of binary random variables.

Assuming that the number of interfering users is not too small, the sum of binomial random variables closely approximates a Gaussian random variable.

Under these assumptions, the interference of the other same cell users can be modeled as additive white Gaussian noise (AWGN). This is known as the *Gaussian approximation* for spread spectrum CDMA systems. The Central Limit Theorem (CLT) [51] is the mathematical basis for this approximation. It can be extended to the asynchronous case with imperfect power control including the interference from users in neighboring cells [39].

Based on the *Gaussian approximation*, the simulation that is used in this dissertation models the interference as AWGN.

3.3. Received Signal Model

For the computer simulations, based on the results of the previous sections, it is assumed that the transmitted signal reaches the receiver over L resolvable paths, i.e., the multipath signals are all separated in time by more than a chip time T_C of the spreading sequences. The receiver performs ideal downconversion to complex baseband of the received signal and a RAKE [19] receiver structure with L fingers is used. The individual fingers acquire the spreading sequence timings and track the multipath signals perfectly. The multipath signals are time-delay adjusted, matched filtered and sampled at the chip rate.

The received complex signal sequence of the l-th multipath signal is then given by

$$y_l[n] = \tfrac{1}{2}\,\alpha_l\sqrt{E_C}\,x[n]\,a[n]\left(a_I[n] - j\,a_Q[n]\right)\cdot e^{-j\Delta\Theta_l} + n_{Il}[n] + j\,n_{Ql}[n],$$

where α_l is the signal amplitude, E_C is the chip energy, $x[n]$ is the Walsh code chip sequence (indexed at chip rate, i.e., remains constant over 4 chips corresponding to one Walsh code chip), $a[n]$ is the long code spreading sequence, $a_I[n]$ is the in-phase spreading sequence, $a_Q[n]$ is the quadrature spreading sequence, $\Delta\Theta_l$ is the difference between the phase of the local carrier and the phase of the multipath signal, $n_{Il}[n]$ and $n_{Ql}[n]$ are independent Gaussian sequences with variance $\sigma_c^2/2$ modeling all the interference (thermal noise, other multipath signals, other users same cell, other users other cells, intersymbol).

The multipath signal amplitudes α_l are normalized so that

$$\sum_{l=1}^{L} \alpha_l^2 = 1 .$$

3.4. Fading

The multipath signals are characterized by the signal amplitudes α_l and phases $\Delta\Theta_l$. In the unfaded cases the multipath parameters remain constant.

In the fading cases the multipath parameters are time varying. It is assumed however that they remain constant for the duration of each Walsh group (208.3 μs). This can be justified by the slowly time-varying nature of the mobile channel impulse response, which is characterized by channel Doppler spread and coherence time.

The Doppler spread B_D of the channel is related to the maximum Doppler shift f_m which is given by

$$f_m = \frac{v_{max}}{\lambda} = \frac{v_{max} \cdot f_c}{c} ,$$

where v_{max} is the maximum relative velocity between transmitter and receiver, λ is the wavelength of the signal, f_c is the carrier frequency, and c is the speed of light (299792458 m/s). Common values of f_m for outdoor environments are in the range of 5 Hz - 200 Hz.

The channel coherence time T_{coh} is a statistical measure of the time over which the mobile channel impulse response

remains constant. [39] gives a popular rule of thumb for modern digital communications as

$$T_{coh} = \sqrt{\frac{9}{16\pi f_m^2}} = \frac{0.423}{f_m} \quad .$$

Using a maximum Doppler shift of 100 Hz this formula suggest a channel coherence time of 4.23 ms. [17] gives an even higher value of 11 ms for outdoor channels.

Two types of Rayleigh fading are considered. The first case will be referred to as *independent Rayleigh fading* and assumes that the multipath parameters of different Walsh groups are independent. As just stated above, this is not the case for a mobile channel. However, this assumption is often made to obtain analytical results or performance bounds (, e.g., [47]), and is also the basis of many published simulation results. It will allow comparisons of the results obtained in this dissertation with other results.

In order to model the *independent Rayleigh fading* a complex Gaussian random sequence $c_l[n]$ with variance σ_f^2 is generated for each multipath signal. The amplitude r of $c_l[n]$ has the desired Rayleigh probability density function (pdf)

$$f_r(x) = \frac{x}{\sigma_f^2} \cdot e^{-\frac{x^2}{2\sigma_f^2}}, \quad x > 0,$$

and the phase is uniformly distributed between zero and 2π .

The average power of the fading amplitude is given by

$$E\left[r^2\right] = 2\sigma_f^2$$

and is be set to unity by the choice $\sigma_f^2 = 1/2$. Finally, each multipath signal is multiplied with $c_l[n]$.

The second type of Rayleigh fading will be referred to as *correlated Rayleigh fading* or *Rayleigh fading with Doppler spectrum*. Using a Fast-Fourier-Transform (FFT) approach a baseband Doppler spectrum of the form

$$S(f) = \frac{1}{\pi f_m \sqrt{1 - \left(\frac{f}{f_m}\right)^2}}, \quad |f| \le f_m, \text{ zero elsewhere,}$$

is applied to the *independent Rayleigh fading* sequences $c_l[n]$. The corresponding autocorrelation function or correlation coefficient (since it is normalized) is then given by

$$R(\tau) = \rho(\tau) = J_0(2\pi f_m \tau),$$

where $J_0(x)$ is the Bessel function of the first kind of zero order, and the fading amplitude correlation coefficient is given by

$$\rho_r(\tau) = J_0^2(2\pi f_m \tau).$$

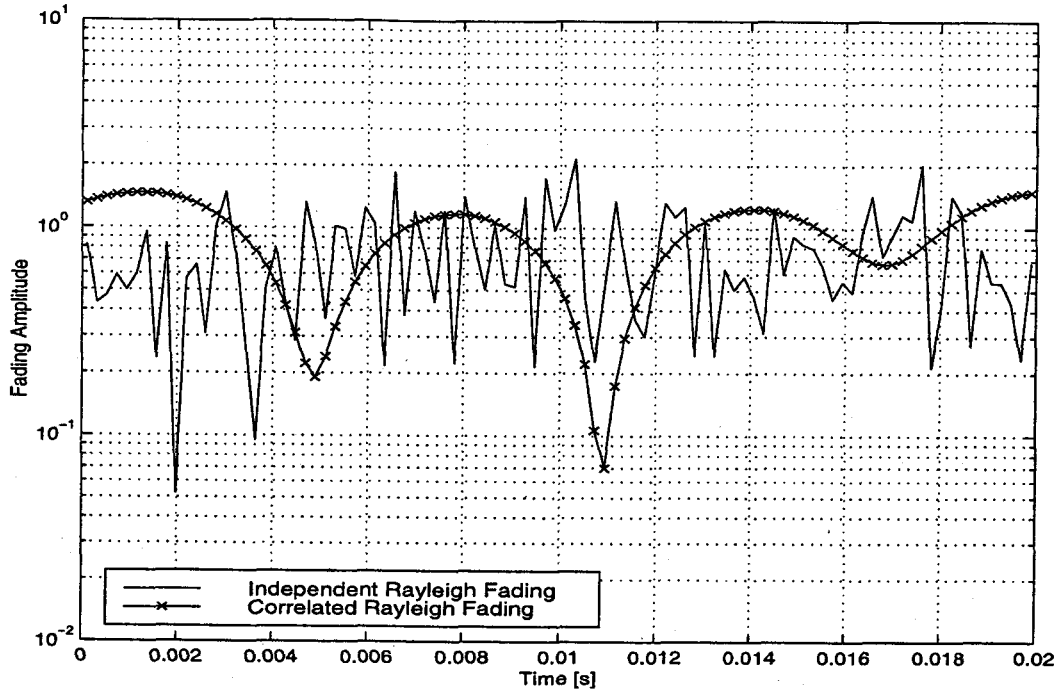


Figure 3.1: Comparison of *independent Rayleigh fading* and *correlated Rayleigh fading* (f_m=100 Hz) over the duration of a 20 ms frame.

This form of Doppler spectrum is used to model the slow fading characteristics of the mobile communications channel. It is a special case of Clarke's model [39] and very similar to the channel model that is used to test base station receivers [52][53]. The results give a good indication of the performance of the DF decoder design under extreme fading conditions.

For *correlated Rayleigh fading*, the multipath parameters of a Walsh group are no longer independent of the multipath parameters of neighboring Walsh groups. The rate of change depends on the maximum Doppler frequency f_m used in the model.

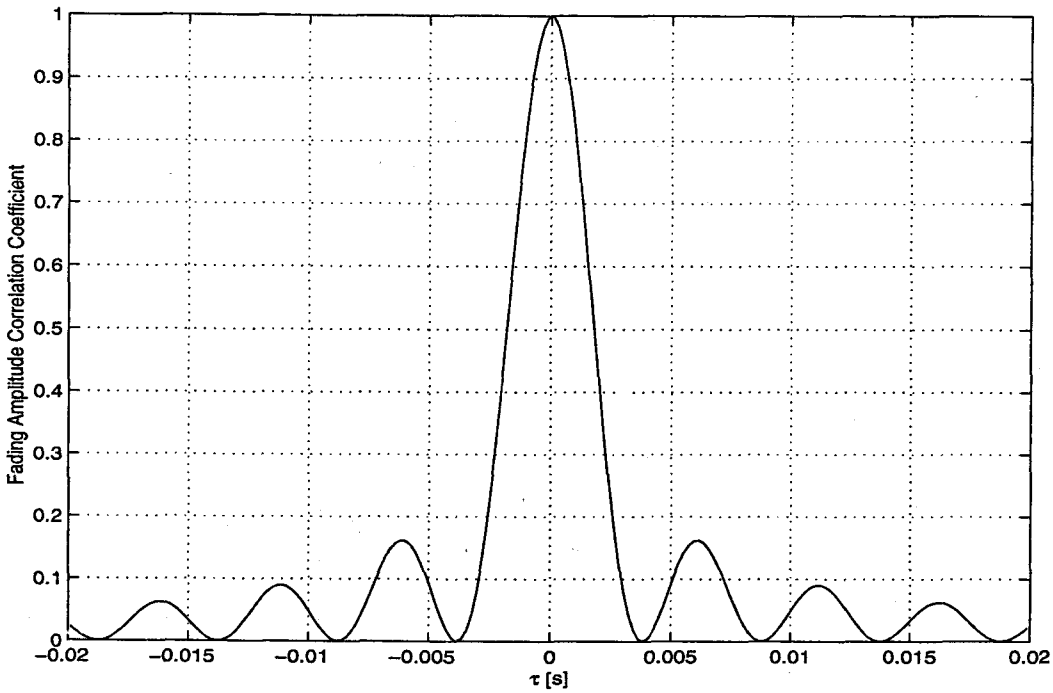


Figure 3.2: Fading amplitude correlation coefficient for *correlated Rayleigh fading* (f_m=100 Hz).

For the computer simulations a sampling rate of 4.8 kHz is used for the multipath parameters resulting in exactly 96 samples for every 20 ms frame. The maximum Doppler frequency f_m is chosen to be 100 Hz, which corresponds to a velocity of approximately 82 mph (131 km/h) at cellular frequencies (824-849 MHz) and 36 mph (57 km/h) at PCS frequencies (1850-1910 MHz). Figure 3.1 shows a comparison of realizations of *independent Rayleigh fading* and *Rayleigh fading with Doppler spectrum* for this value of the maximum Doppler frequency. Figure 3.2 shows the corresponding fading amplitude correlation coefficient.

If the received multipath signals are independently fading, the use of a RAKE receiver results in additional performance improvement. This is illustrated in Figure 3.3.

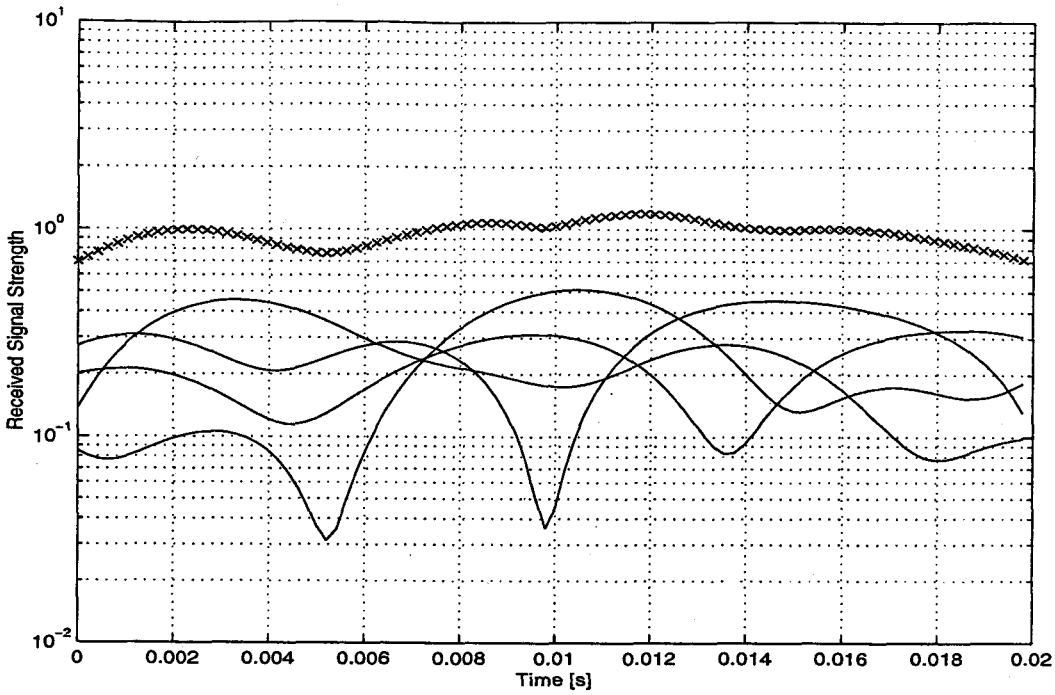


Figure 3.3: Multipath diversity gain for 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) with a combined signal power of 1.

In addition to improving the average received signal strength, the variation in the combined signal is also reduced. This is referred to as *multipath diversity gain.*

3.5. Receiver Signal Processing

3.5.1. Coherent Detection

In the coherent case, each finger of the RAKE receiver has knowledge of the multipath parameters α_l and $\Delta\Theta_l$. Although the IS-95 uplink was not designed for coherent detection, estimation of the multipath parameters could be

accomplished using pilot symbols [54][55] or decision aided techniques [56][57].

Coherent signal processing consists of multiplication with $\alpha_1 \cdot e^{j\Delta\Theta_1}$, quadrature despreading with the short code sequences, summation of in-phase and quadrature signals (real and imaginary parts), and despreading with the long code sequence resulting in

$$z_1[n] = \alpha_1^2 \sqrt{E_c} x[n] + \alpha_1 n_1[n],$$

where $n_1[n]$ is a Gaussian sequence with variance σ_c^2. This assumes that the scrambling of the noise components does not change their statistics (ideal binary random sequences). Adding these multipath signals results in maximum ratio combining [58]. The Walsh correlations $\boldsymbol{W}$ are computed, e.g., by a Fast-Hadamard-Transform (FHT), and passed to the decoder.

3.5.2. Noncoherent Detection

In the noncoherent case, the multipath parameters are unknown. Only knowledge of the (average) multipath strengths is assumed. The in-phase and quadrature sequences of the received signal contain information of both the transmitted in-phase and quadrature signal. A crosscombination of these sequences mathematically represented by multiplication with $a[n] \cdot (a_I[n] + ja_Q[n])$ results in two new in-phase and quadrature signals

$$\tilde{y}_1[n] = \alpha_1 \sqrt{E_c} x[n](\cos \Delta\Theta_1 - j \sin \Delta\Theta_1) + \tilde{n}_{I1}[n] + j\tilde{n}_{Q1}[n]$$

(the noise components now having variance σ_c^2). The FHT is performed for each of these new in-phase and quadrature sequences and the results are square-law combined. The matrix $\boldsymbol{W}$ is obtained by adding the noncoherent multipath correlations scaled by their (average) multipath strengths.

3.6. Simulation Statistics

In order to gather sufficient statistics for reliable bit error rate (BER) and frame error rate (FER) estimates, extensive computer simulations of different propagation scenarios were performed.

In the following chapters, the performance of the original decision feedback (DF) receiver/decoder design and the proposed design modifications is evaluated for the following propagation scenarios:

1) unfaded single-path signal
2) *independent Rayleigh fading* single-path signal
3) 4 equal-strength unfaded multipath signals
4) 4 equal-strength *independent Rayleigh fading* multipath signals
5) 4 equal-strength *correlated Rayleigh fading* multipath signals

For all these cases, coherent and noncoherent detection in additive white Gaussian noise (AWGN) interference is considered.

To simulate an IS-95 uplink traffic channel frame, 184 random data bits are generated and used to generate the

corresponding Walsh chip sequence according to the IS-95 standard. The combination of the long code and short code spreading sequences is also modeled by binary random sequences.

All the binary sequences are generated using a uniform random number generator. The interference and the fading variables are generated using a normally distributed random number generator. Both random number generators were initialized to the same state for each E_b/N_0 value.

The unfaded single-path results are based on the simulation of 100,000 IS-95 uplink traffic channel frames. Due to time and resource constraints, all other results are based on the simulation of 10,000 IS-95 uplink traffic channel frames.

The addition of multipath, *independent Rayleigh fading* and *correlated Rayleigh fading* requires simulation of more and more frames in order to cover most of the possible realizations of interference and fading variables. The simulation results of the more complex propagation scenarios show signs of the effects of insufficient statistics. This is especially true for scenario 2) (due to lack of multipath diversity) and 5). It is the author's belief that those results are reliable down to a BER between 10^{-3} and 10^{-4}, and continue to show the performance trends beyond that.

Chapter 4. Multipath and Rayleigh Fading Performance

In the original work of Rabinowitz [24], the performance of the decision feedback (DF) receiver/decoder that was presented in the previous chapter was evaluated in the presence of synchronous multiple access interference (MAI).

However, the mobile radio channel is characterized by multipath and fading effects. Multipath is caused by the fact that the transmitted signal reaches the receiver antenna via many routes. The different signal paths are created by scattering, reflection, and diffraction of the transmitted signal due to obstacles in the environment. These multipath signals are received with different time delays.

One of the advantages of the CDMA signal waveform is that multipath components with a relative time delay greater than the chip time T_C can be resolved and constructively combined at the receiver.

The resolvable multipath components consist themselves of a number of multipath components with small relative time delays. These signals combine constructively and destructively resulting in random phase and amplitude variations. This effect is referred to as fading.

In the remainder of this chapter, the simulation model of Chapter 3 is used to evaluate the performance of the DF receiver/decoder for the IS-95 uplink traffic channel (9.6 kbits/s) in the presence of these multipath and fading effects.

4.1. Unfaded Single-Path Signal

Before multipath and fading is considered, the performance of the DF receiver/decoder is evaluated in additive white Gaussian noise (AWGN) interference alone. The results can be used for comparison with other published results. They are also useful to determine the degradation caused by the multipath and fading effects that are considered in the following sections.

The bit error rate (BER) and frame error rate (FER) for coherent detection is shown in Figure 4.1. It can be seen that the use of decision feedback decoding (DFD) results in a consistent improvement of the BER and FER for all values of the bit-energy-to-interference-density (E_b/N_0) ratio shown.

Without DF an E_b/N_0 of 3.61 dB is required to achieve a BER of 10^{-3}. Use of DF results in a performance improvement of 0.09 dB, and reduces the required E_b/N_0 to achieve the same BER to 3.52 dB. The FER corresponding to this BER is on the order of 1-2%, which is in agreement with the requirement for above average voice quality stated in [35].

For 2% FER the required E_b/N_0 is reduced by 0.09 dB from 3.52 dB to 3.43 dB. Similarly, for 1% FER the reduction is 0.1 dB from 3.70 dB to 3.60 dB. This is illustrated in more detail in Figure 4.2.

Note that the improvements for the BER and the FER are approximately the same at the considered performance levels.

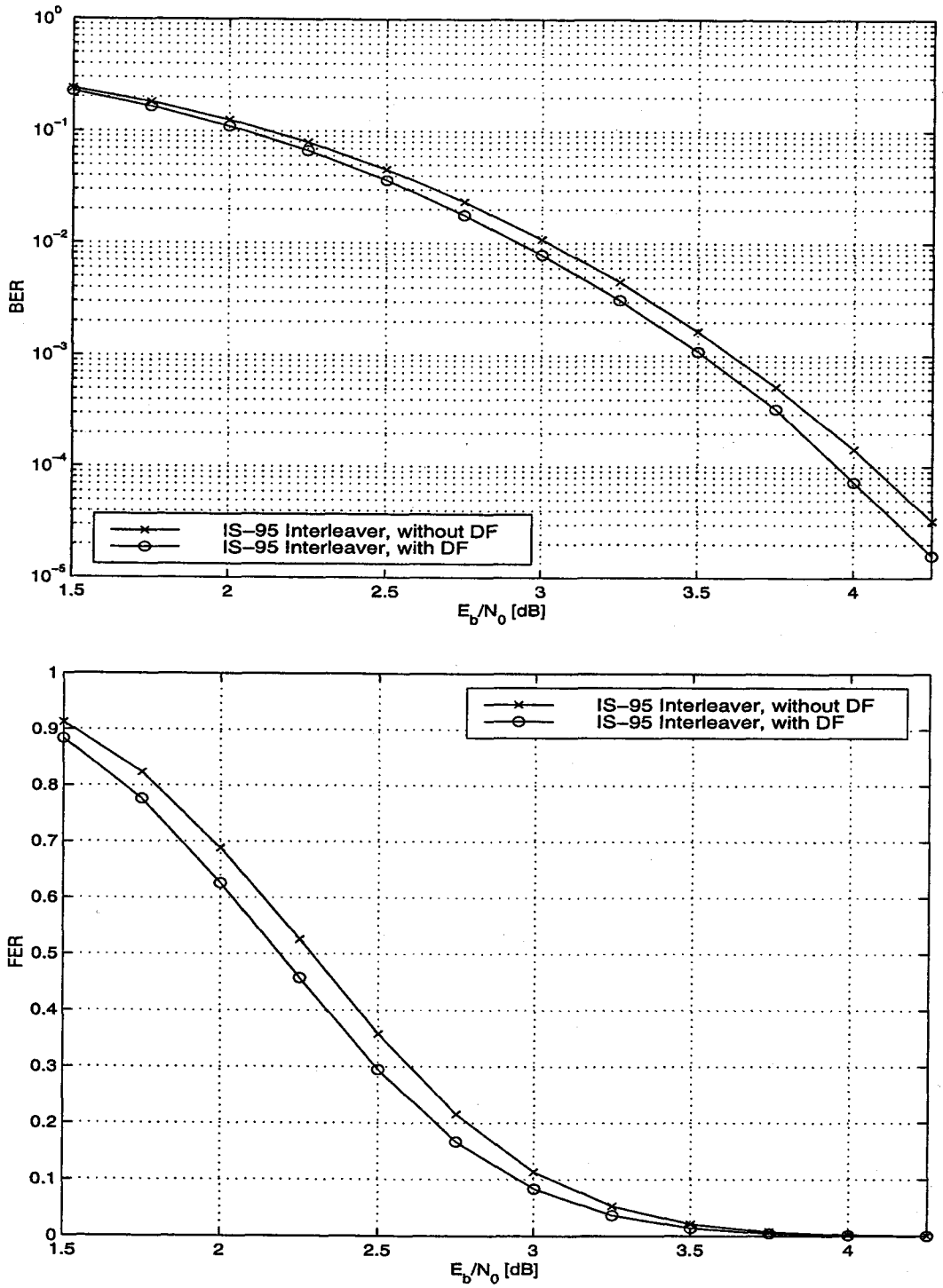


Figure 4.1: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of unfaded single-path signal in AWGN.

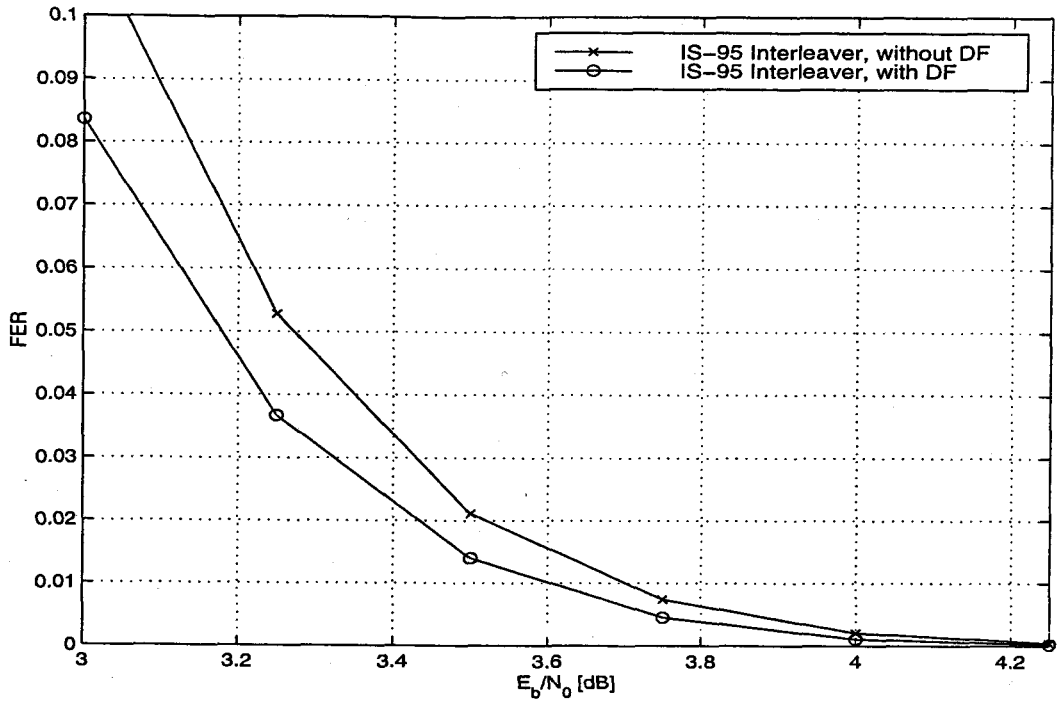


Figure 4.2: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of unfaded single-path signal in AWGN.

The results for noncoherent detection are shown in Figure 4.3. The degradation due to noncoherent combination of the in-phase and quadrature components of the received signal is approximately 1.4 dB. In this case, DF also results in consistent performance improvement, but the improvement is smaller.

To achieve a BER of 10^{-3}, the required E_b/N_0 is 5.01 dB without the use of DF, and 4.95 dB with DF.

At 2% FER, the required E_b/N_0 is reduced by 0.06 dB as well, from 4.94 dB to 4.88 dB. Similarly at 1 % FER, the reduction is 0.09 dB, from 5.09 dB to 5.00 dB (Figure 4.4).

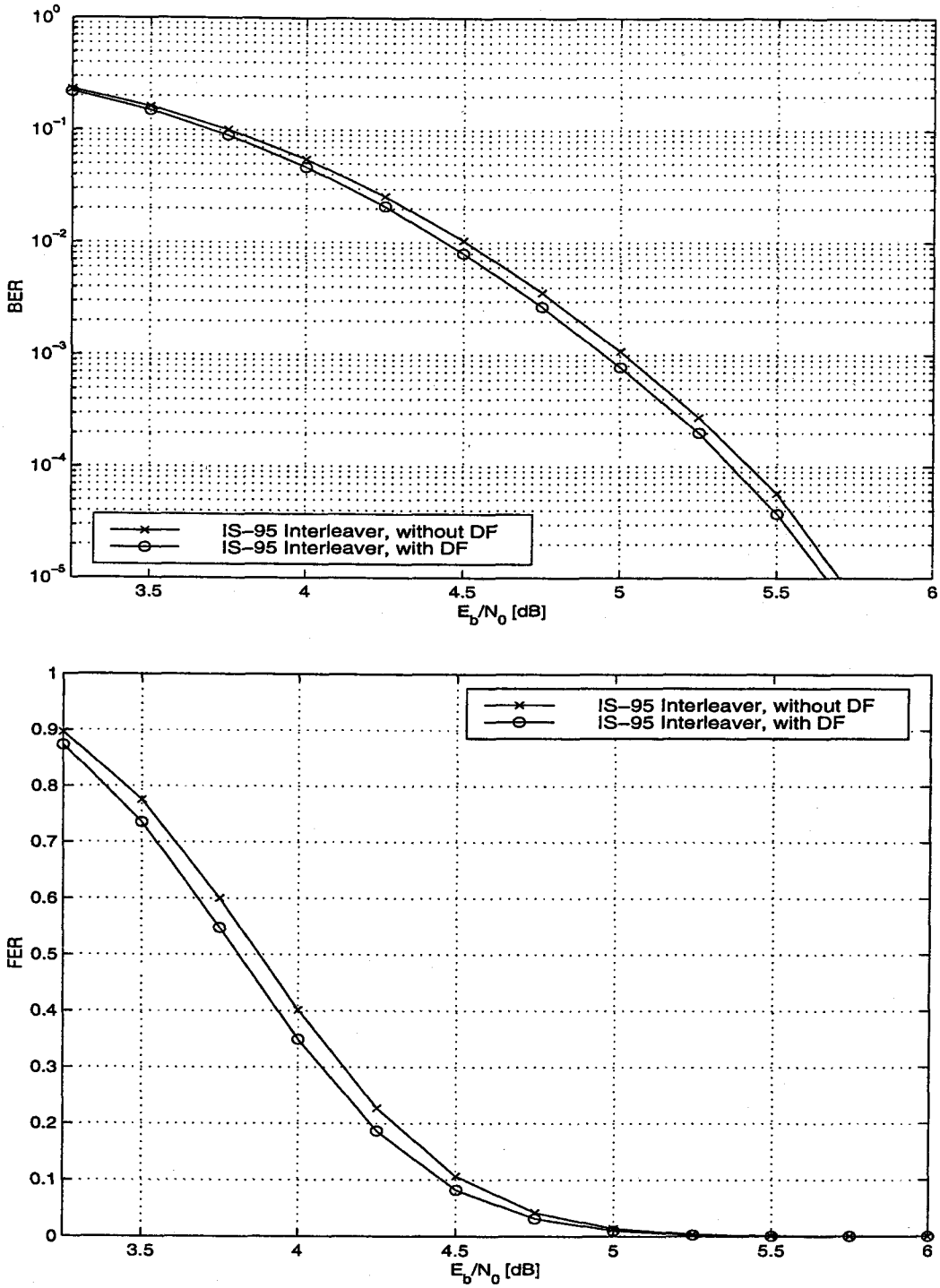


Figure 4.3: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of unfaded single-path signal in AWGN.

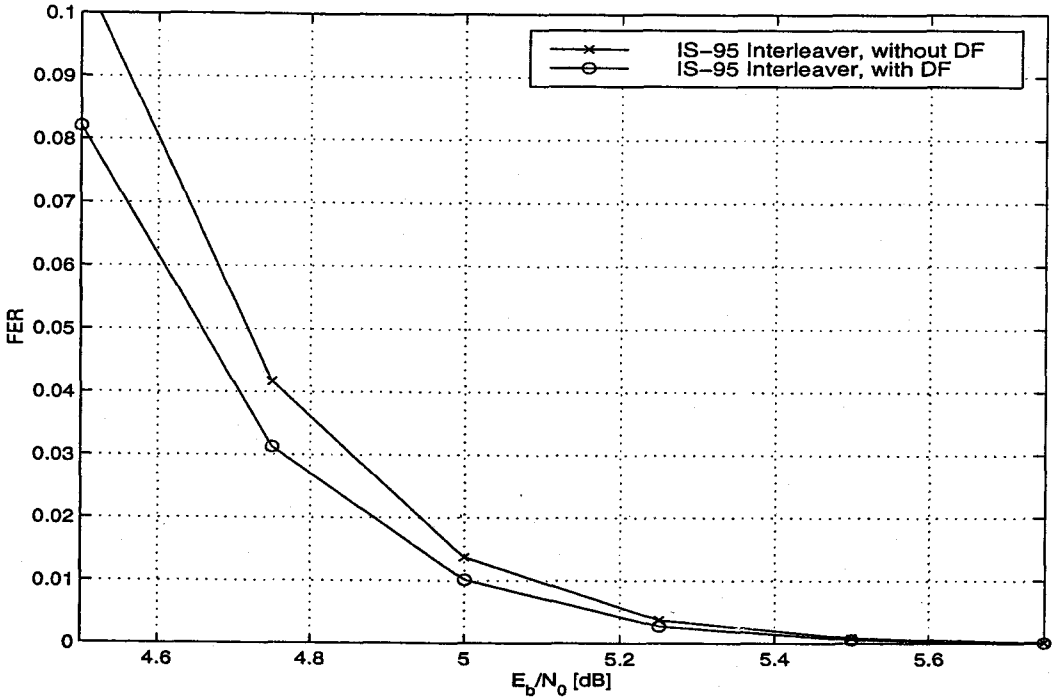


Figure 4.4: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of unfaded single-path signal in AWGN.

Again, the performance improvement for both the BER and FER is approximately the same of the considered values.

Since the synchronous MAI studied by Rabinowitz [24] can be approximated by AWGN (see Section 3.2), the results presented here are expected. The use of DF results in consistent performance improvement for coherent and noncoherent detection of an unfaded single-path signal in AWGN.

Also, the BER results in this section that do not use DFD are almost identical to the ones presented by Herzog et al. [41][42]. This validates the computer simulation.

4.2. Rayleigh Fading Single-Path

In this section, the performance of the DF receiver/decoder is evaluated for the case of an *independent Rayleigh fading* (Section 3.4) single-path signal in AWGN interference. The results will indicate if the DF decoder can tolerate a randomly varying signal amplitude or equivalently a randomly varying E_b/N_0 within a frame. Strictly speaking, the E_b/N_0 in this and all the other fading cases is the average value of the E_b/N_0.

Coherent signal detection is considered first. The resulting BER and FER are shown in Figure 4.5. DF improves the performance for all E_b/N_0 values shown.

A comparison with Figure 4.1 shows that the performance at a BER of 10^{-3} is degraded by 1.83 dB due to the fading of the signal amplitude. The DF performance improvement is 0.08 dB at this BER and remains almost the same compared to 0.09 dB in the unfaded case. The required E_b/N_0 with and without DF is 5.44 dB and 5.36 dB, respectively.

Another effect due to fading can also be observed. In the unfaded case, the BER is reduced from 10^{-2} to 10^{-3} by an E_b/N_0 increase of only 0.59 dB. In the fading case here, the E_b/N_0 has to be increased by 1.13 dB to achieve the same reduction.

The FER shown in Figure 4.5 and, in more detail, in Figure 4.6 also shows consistent performance improvement through DFD. Here, a FER close to 1% is required for a BER of 10^{-3}. For this FER, the required E_b/N_0 is reduced by 0.08 dB from 5.53 dB to 5.45 dB.

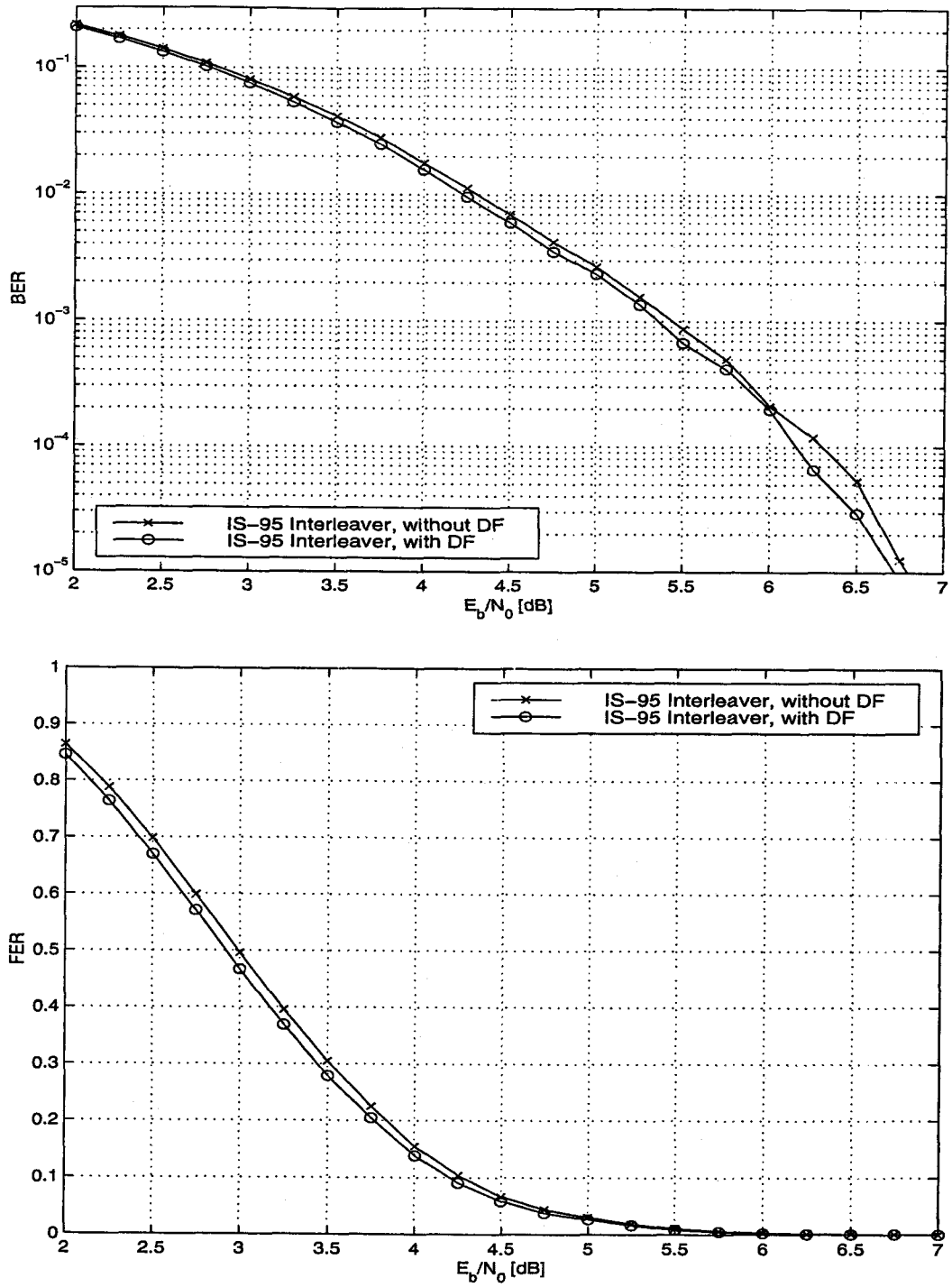


Figure 4.5: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of *independent Rayleigh fading* single-path signal in AWGN.

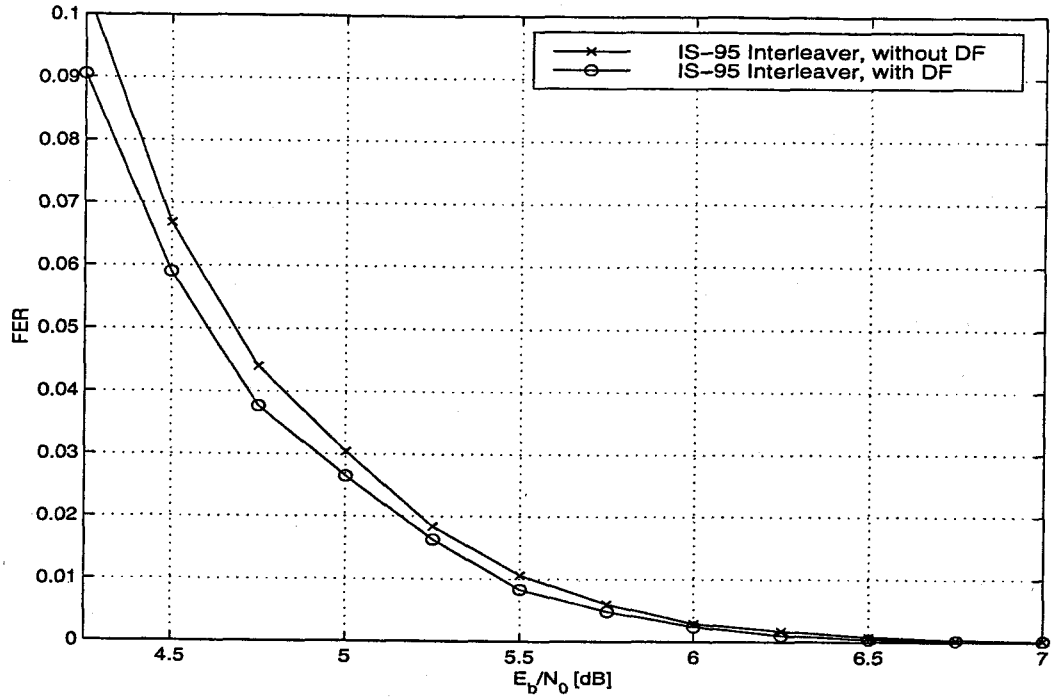


Figure 4.6: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of *independent Rayleigh fading* single-path signal in AWGN.

The receiver performance for noncoherent detection is shown in Figure 4.7 and Figure 4.8. As in the coherent case, DF improves the performance for all E_b/N_0 values shown.

Comparing the results here with the unfaded results in Figure 4.3 shows a 2.42 dB performance degradation due to fading of at BER 10^{-3}. At this BER, DF reduces the required E_b/N_0 by 0.06 dB from 7.43 dB to 7.37 dB. Similarly, at 1% FER the reduction is 0.07 dB from 7.50 dB to 7.43 dB.

An E_b/N_0 increase of 1.09 dB is required to reduce the BER from 10^{-2} to 10^{-3}. In the unfaded case, the required E_b/N_0 increase was only 0.52 dB.

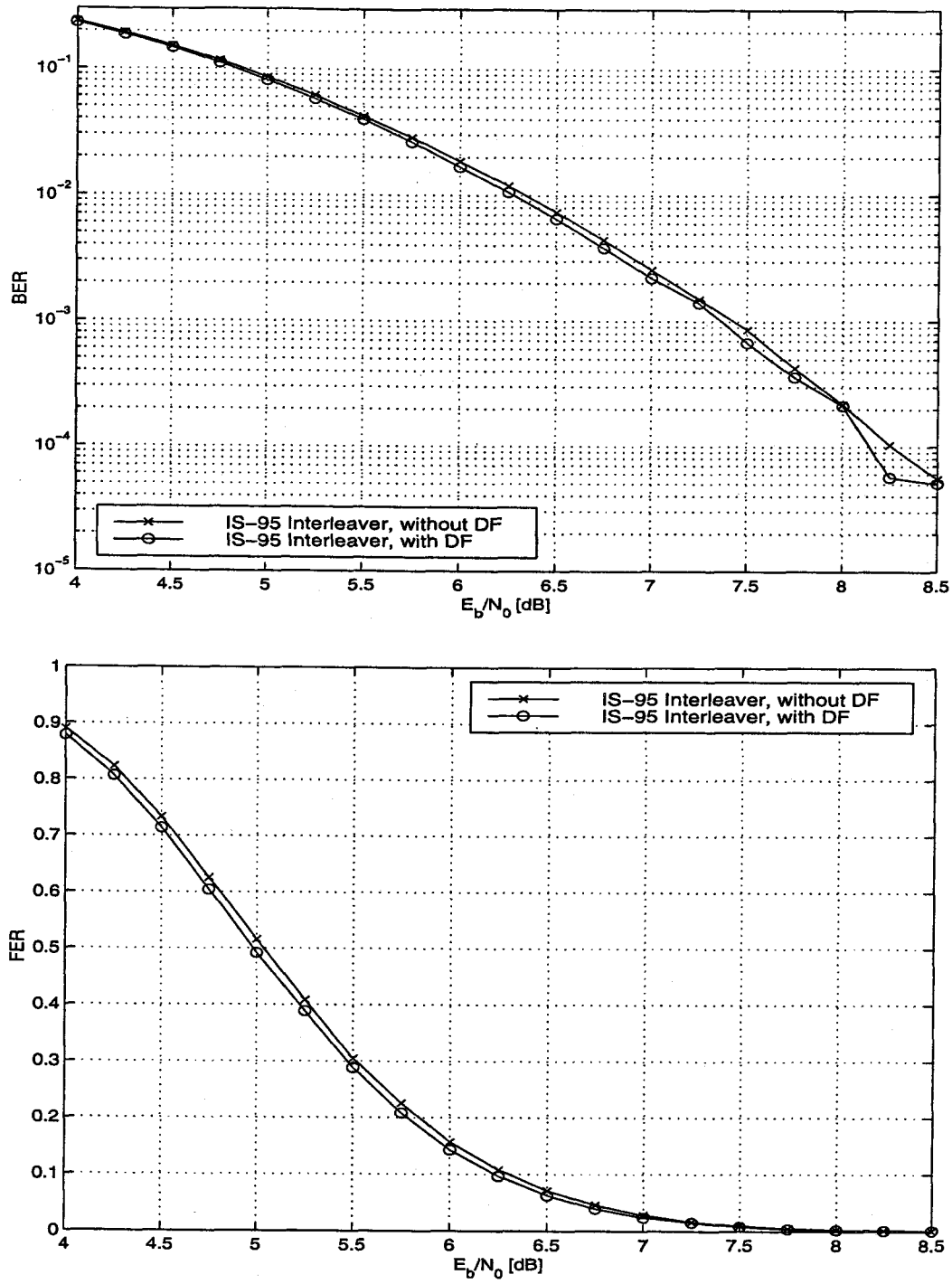


Figure 4.7: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of *independent Rayleigh fading* single-path signal in AWGN.

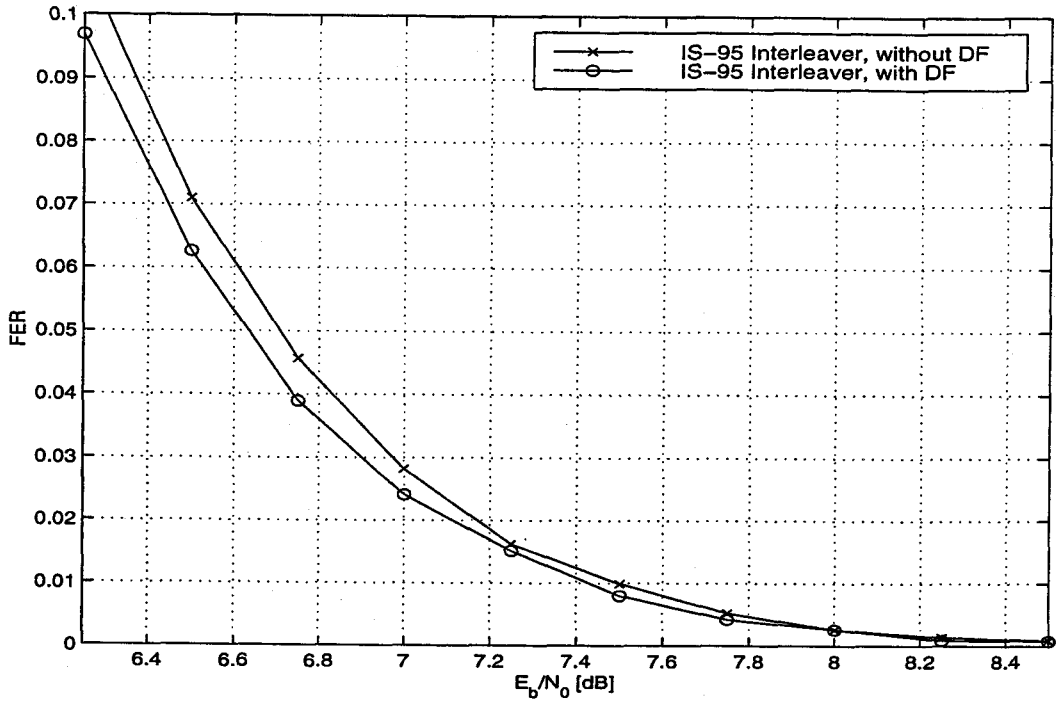


Figure 4.8: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of *independent Rayleigh fading* single-path signal in AWGN.

In summary, the results presented in this section confirm that the concept of DF continues to improve the receiver performance for coherent and noncoherent detection of an *independent Rayleigh fading* single-path signal.

While the improvement through DF is somewhat degraded, especially in the noncoherent case, the DF receiver/decoder consistently improves the BER and FER performance by similar reductions of the required E_b/N_0 to achieve a given level of performance.

4.3. Unfaded Multipath

The results in this section are provided to demonstrate the effect of the RAKE [19] receiver structure, that is the ability to constructively combine several multipath components and thereby improve the E_b/N_0 of the received signal. In the case of coherent detection, a RAKE receiver with perfect knowledge of the multipath parameters has the same performance as an unfaded single-path signal. This is assuming that the combined signal energy of the multipath components is the same as the signal energy of the single-path signal.

4 equal-strength unfaded multipath signals are considered here. The corresponding BER and FER are shown in Figure 4.9. As expected, they are basically identical to the results shown in Figure 4.1.

For noncoherent detection, the RAKE receiver is still able to capture more signal energy than would be possible with a receiver that can only demodulate one signal (the strongest of the multipath signals). In this case however, the multipath components have to be combined in a noncoherent fashion. This causes the performance degradation known as *noncoherent combining loss*.

The BER and FER results in Figure 4.10 and Figure 4.11 show the expected degradation. Compared to the noncoherent detection of a single-path signal (Figure 4.3), the noncoherent combining loss at a BER of 10^{-3} is on the order of 1.76 dB.

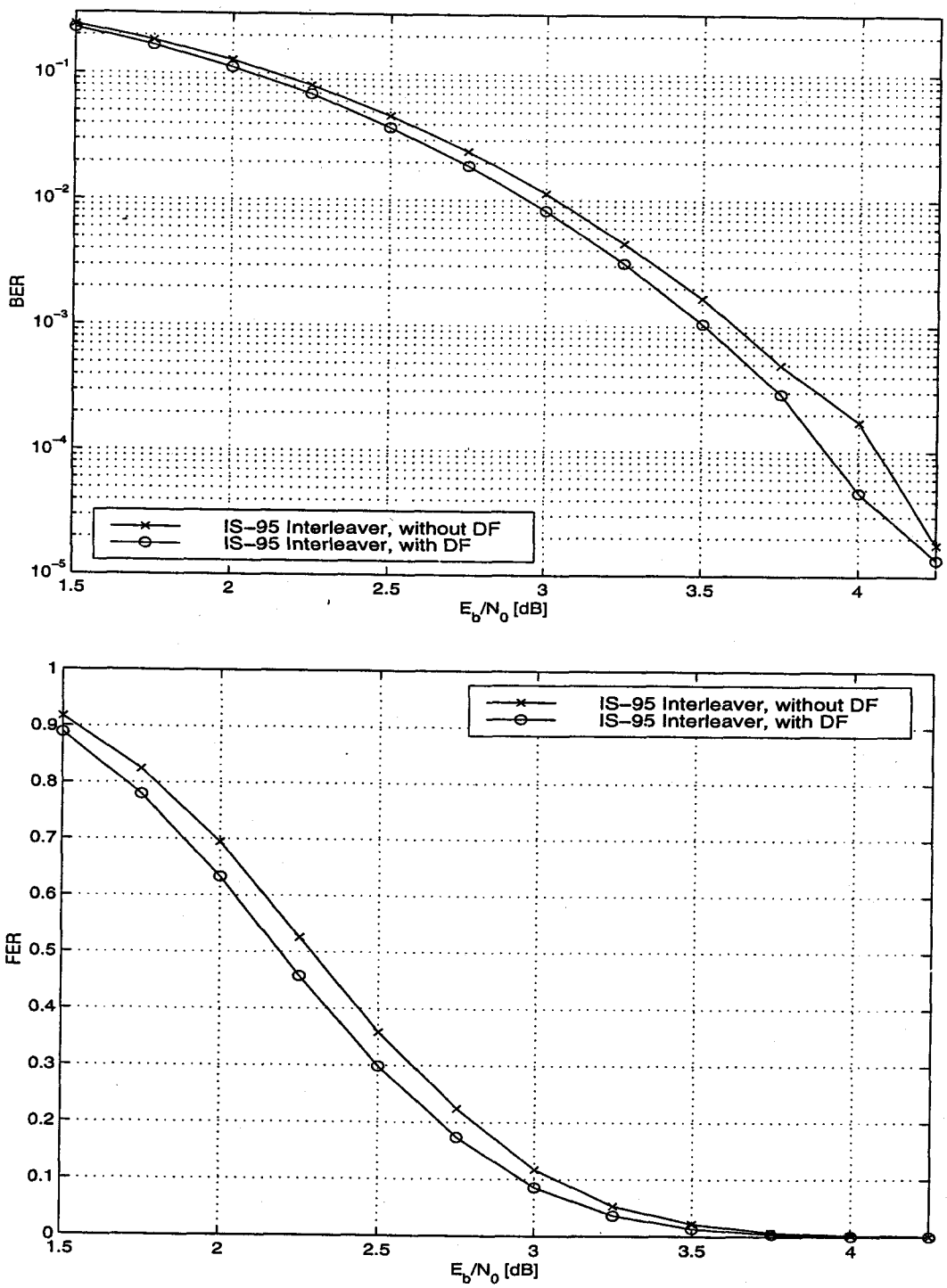
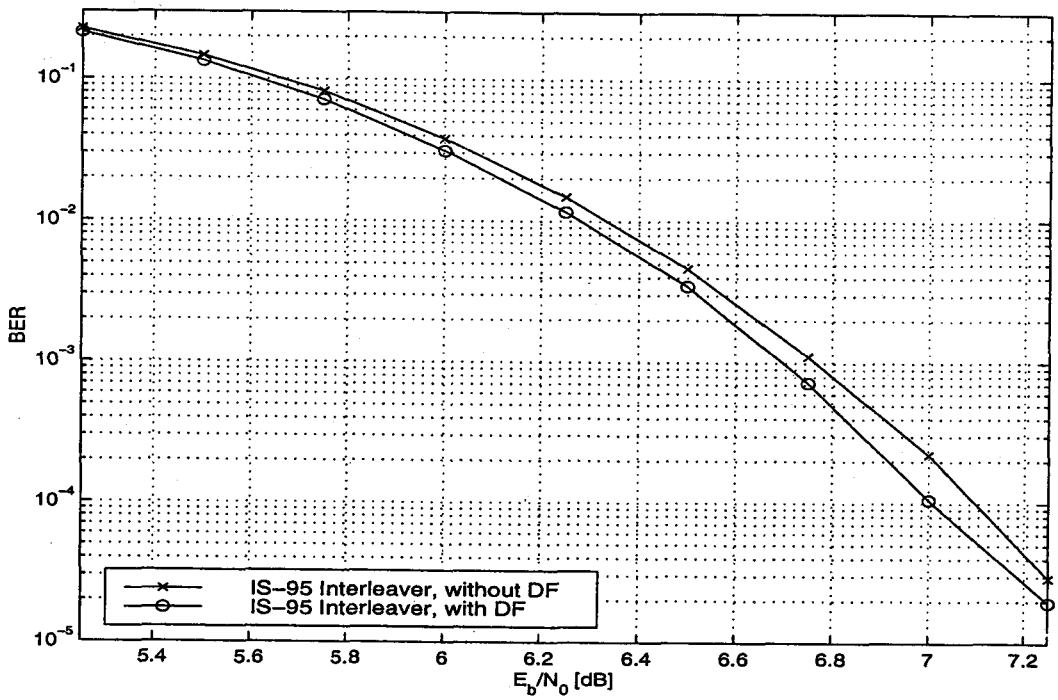


Figure 4.9: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of 4 equal-strength unfaded multipath signals in AWGN.



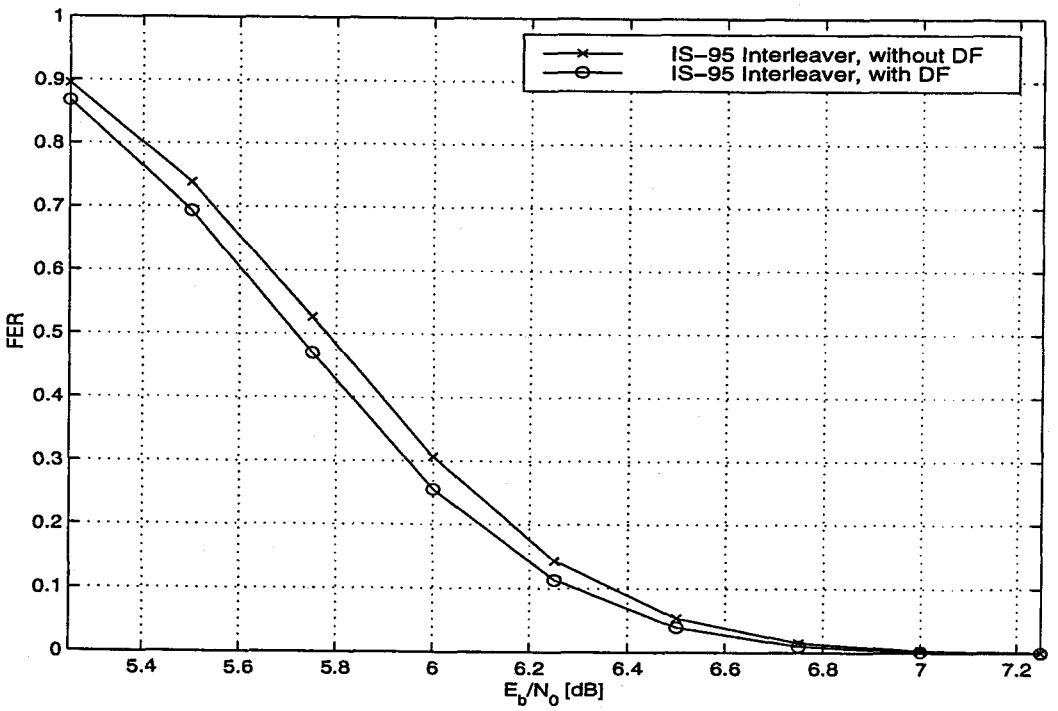


Figure 4.10: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of 4 equal-strength unfaded multipath signals in AWGN.

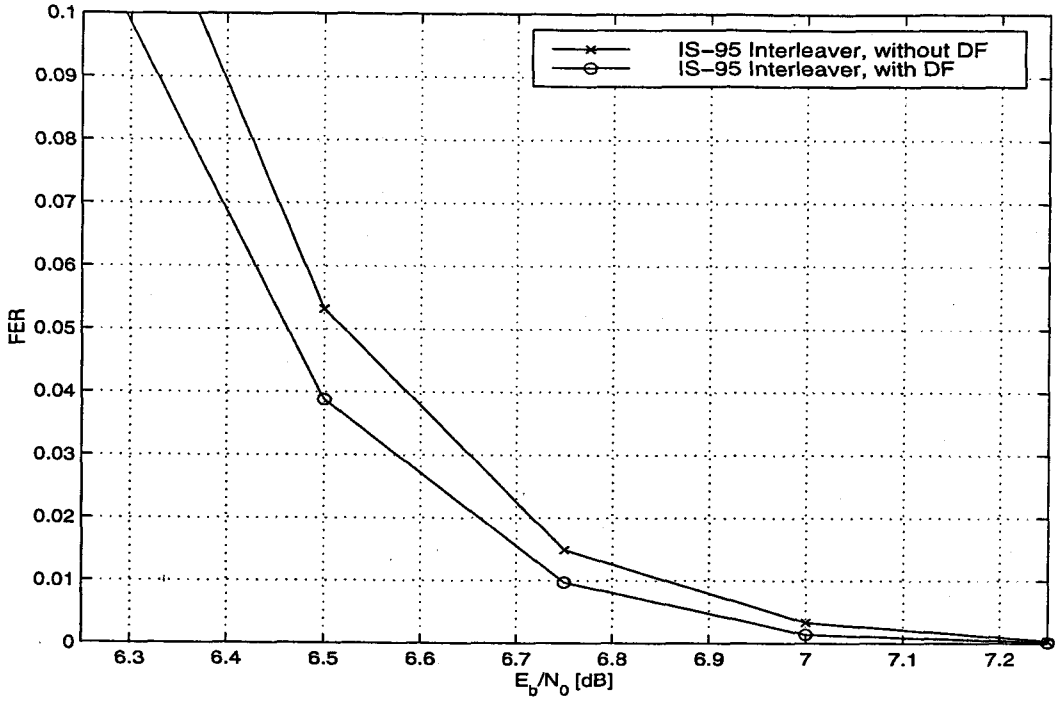


Figure 4.11: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of 4 equal-strength unfaded multipath signals in AWGN.

The performance improvement of DFD remains as in the single-path case. At BER 10^{-3}, the required E_b/N_0 is reduced by 0.07 dB from 6.77 dB to 6.70 dB. For a FER of 1% the reduction is 0.11 dB from 6.86 dB to 6.75 dB.

Herzog et al. [41][42] also present comparable BER results for noncoherent detection of equal-strength unfaded multipath signals. Again, these results are in good agreement with the results shown in this section.

4.4. Rayleigh Fading Multipath

In this section, the effects of Rayleigh fading and multipath are combined. In that case, performance is improved by the *multipath diversity gain*.

As illustrated in Section 3.4, the combination of several fading multipath signals increases the average value of the received signal strength. At the same time, the variation of the combined signals is reduced.

Again, reception of 4 equal-strength multipath signals is considered. The amplitude of each signal is fading. Following the discussion in Section 3.4, the performance of the DF decoder is evaluated for two types of fading. First, *independent Rayleigh fading* is considered. This is followed by results for *correlated Rayleigh fading*.

4.4.1. Independent Rayleigh fading

The BER and FER results for coherent detection are shown in Figure 4.12 and Figure 4.13. The results for noncoherent detection are shown in Figure 4.14 and Figure 4.15. The performance improvement of DFD can be seen in all figures.

In the coherent case, DF reduces the required E_b/N_0 for a BER of 10^{-3} by 0.08 dB from 4.00 dB to 3.92 dB. At 1% FER the reduction is 0.09 dB from 4.07 dB to 3.98 dB.

Comparing these results to the results in Figure 4.5 shows that the multipath diversity gain is on the order of 1.44 dB.

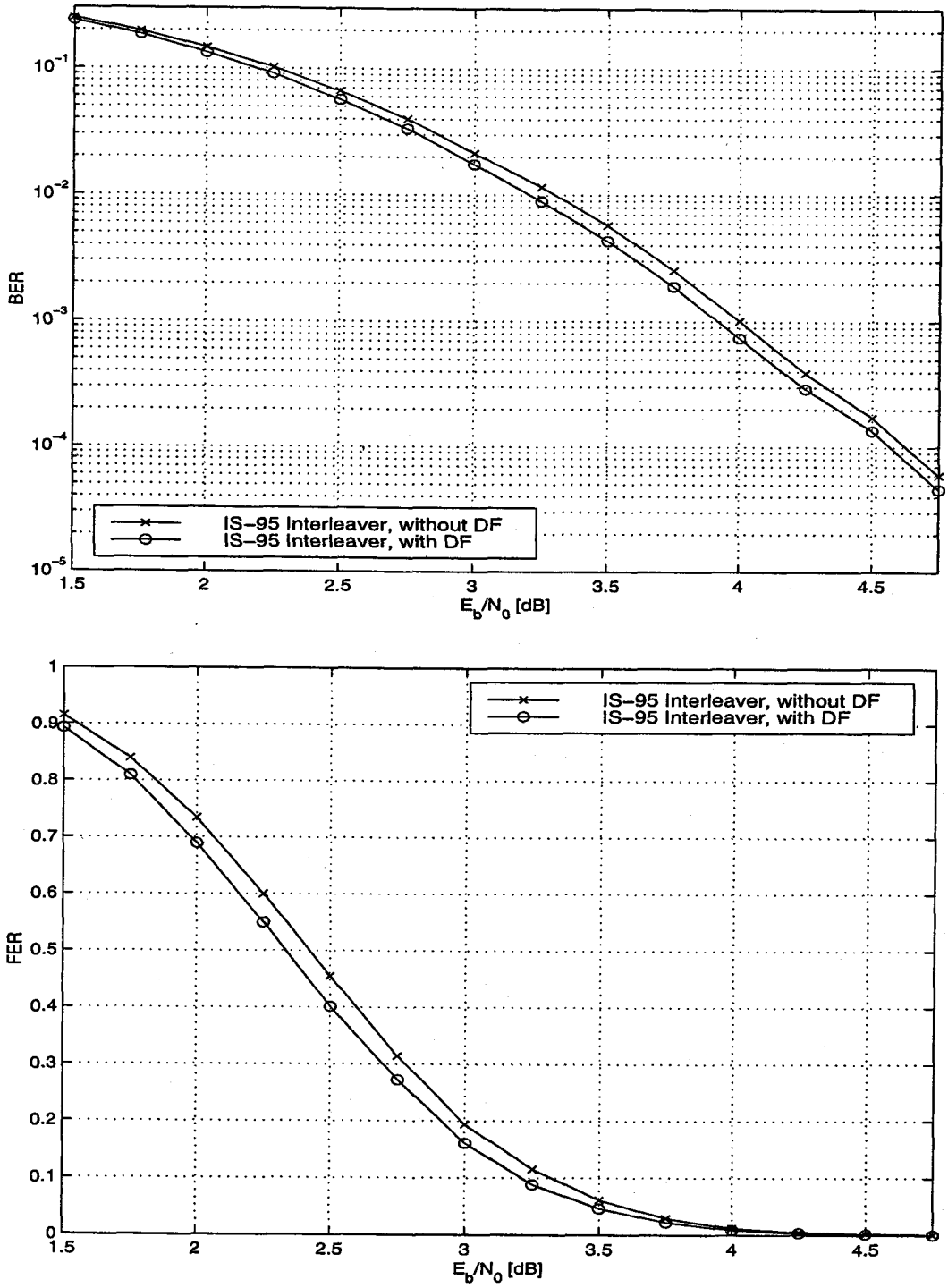


Figure 4.12: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

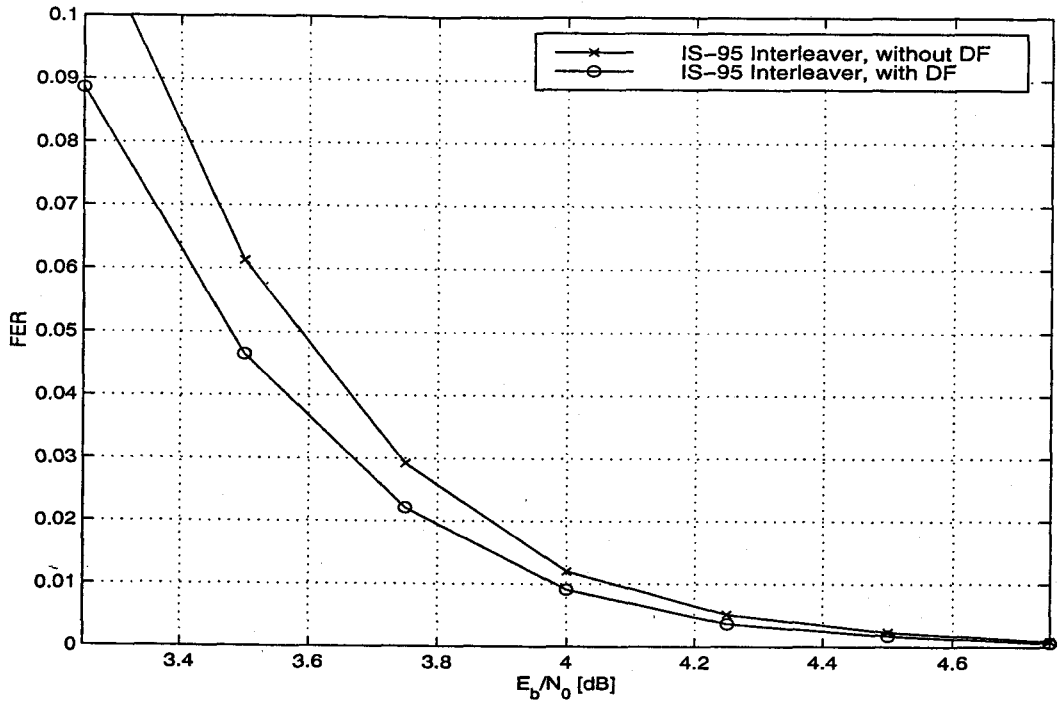


Figure 4.13: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

Using noncoherent detection, DF reduces the required E_b/N_0 for a BER of 10^{-3} by 0.06 dB from 7.37 dB to 7.31 dB. Similarly, at 1% FER the reduction is also 0.06 dB from 7.44 dB to 7.38 dB.

A comparison with the noncoherent results for *independent Rayleigh fading* of a single-path signal (Figure 4.7) shows that performance is improved by a small amount (about 0.06 dB), i.e., the multipath diversity gain is greater than the noncoherent combining loss. Recall that the noncoherent combining loss in the unfaded case (Figure 4.9) was approximately 1.76 dB.

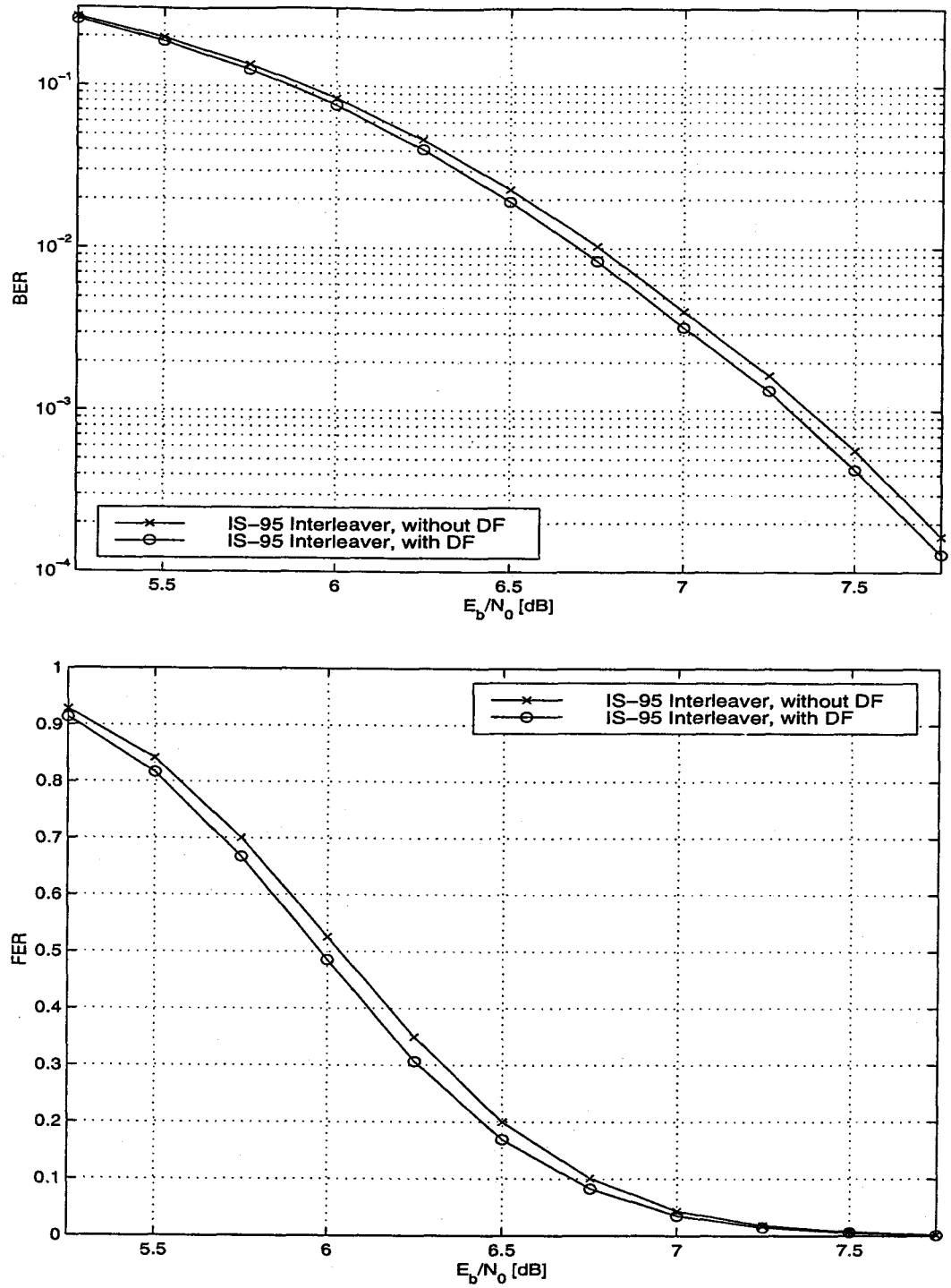


Figure 4.14: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

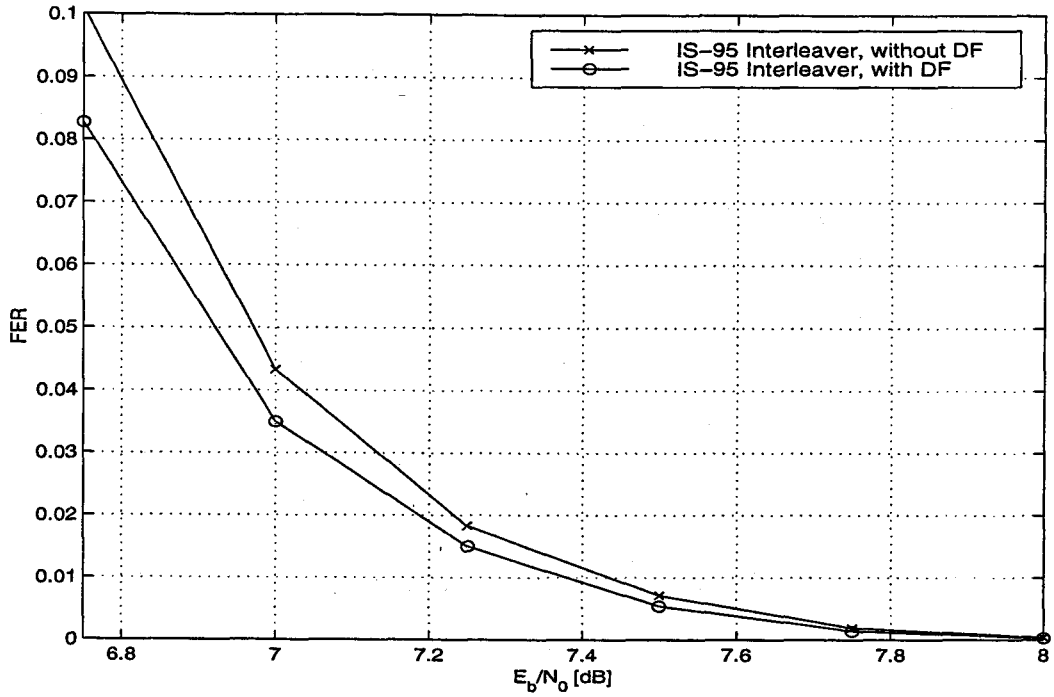
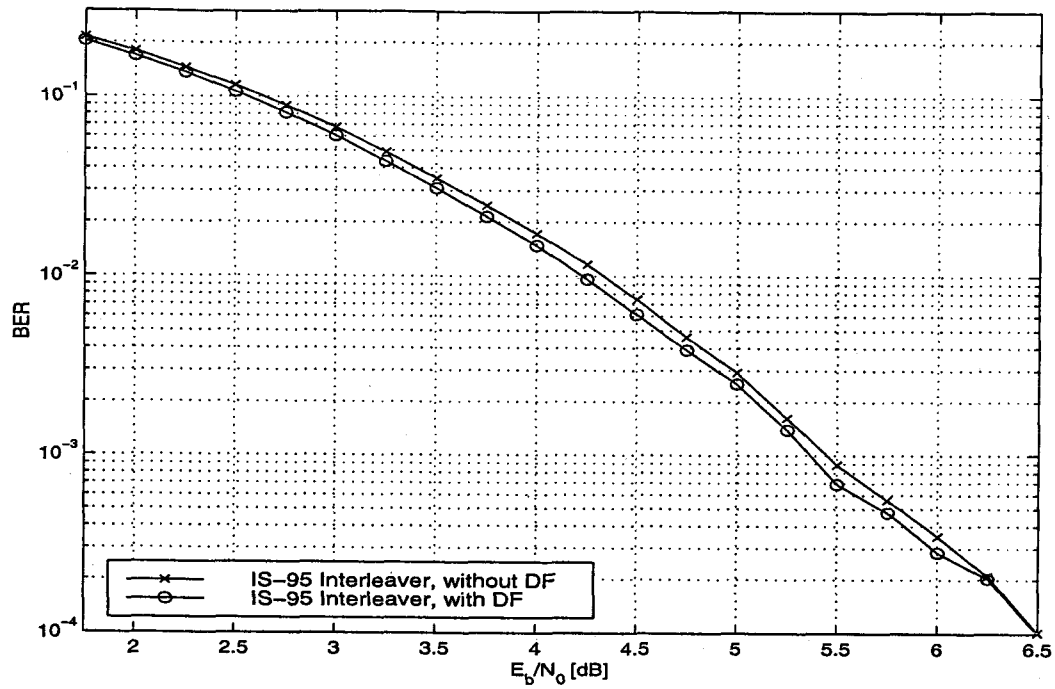


Figure 4.15: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

4.4.2. Correlated Rayleigh fading

In this last propagation scenario, the *independent Rayleigh fading* is replaced with *correlated Rayleigh fading* using a maximum Doppler frequency f_m of 100 Hz (Section 3.4).

The coherent BER and FER results are shown in Figure 4.16 and Figure 4.17. The noncoherent results are shown in Figure 4.18 and Figure 4.19. DFD improves the performance in both cases.



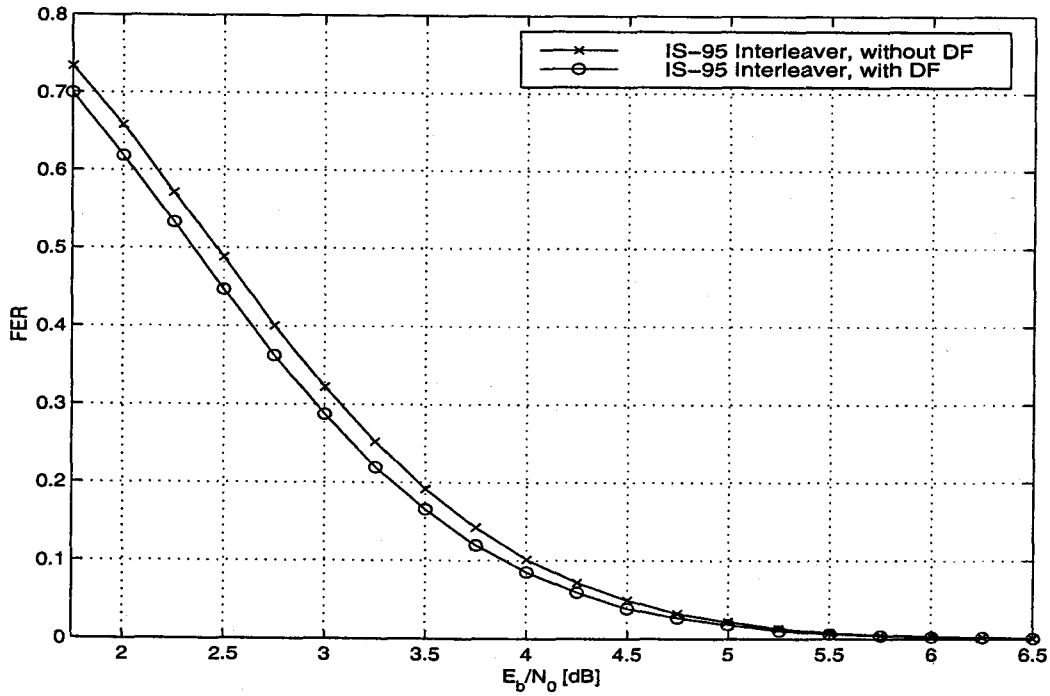


Figure 4.16: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.

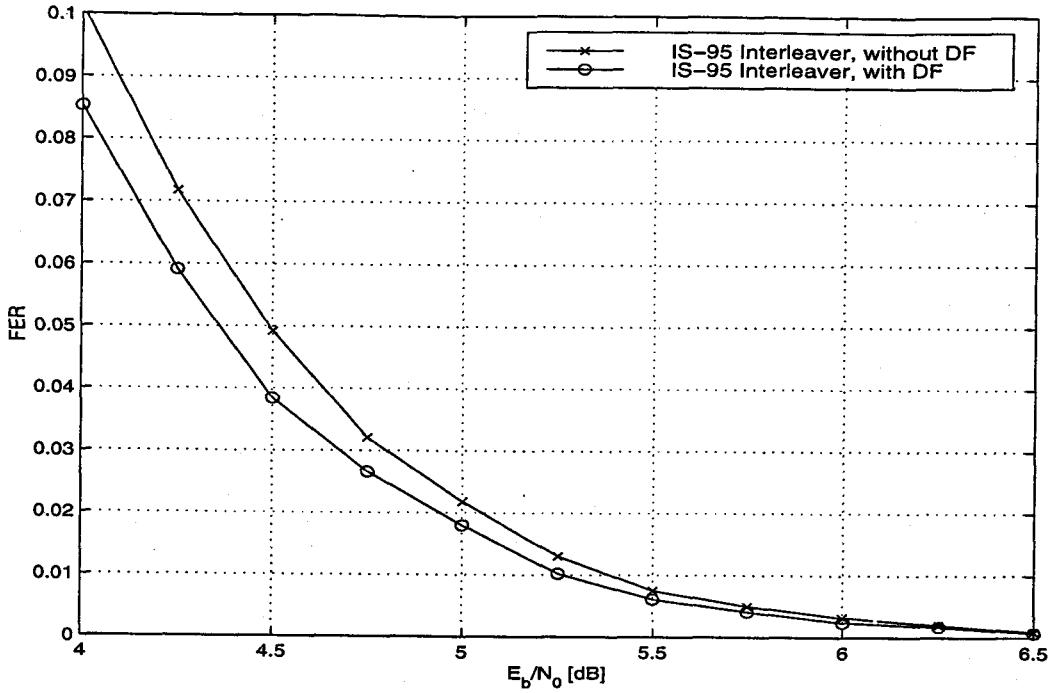


Figure 4.17: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Coherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.

Using coherent detection, decision feedback (DF) reduces the required E_b/N_0 for a BER of 10^{-3} by 0.08 dB from 5.45 dB to 5.37 dB. Similarly, for a FER of 1% the reduction is 0.13 dB from 5.39 dB to 5.26 dB.

The time correlation of the multipath parameters degrades the performance an additional 1.45 dB compared to *independent Rayleigh fading* (Figure 4.12).

The additional degradation for noncoherent detection is 1.66 dB. Here, DF reduces the required E_b/N_0 for a BER of 10^{-3} by 0.08 dB from 9.03 dB to 8.95 dB. For a FER of 1% the reduction is also 0.08 dB from 8.86 dB to 8.78 dB.

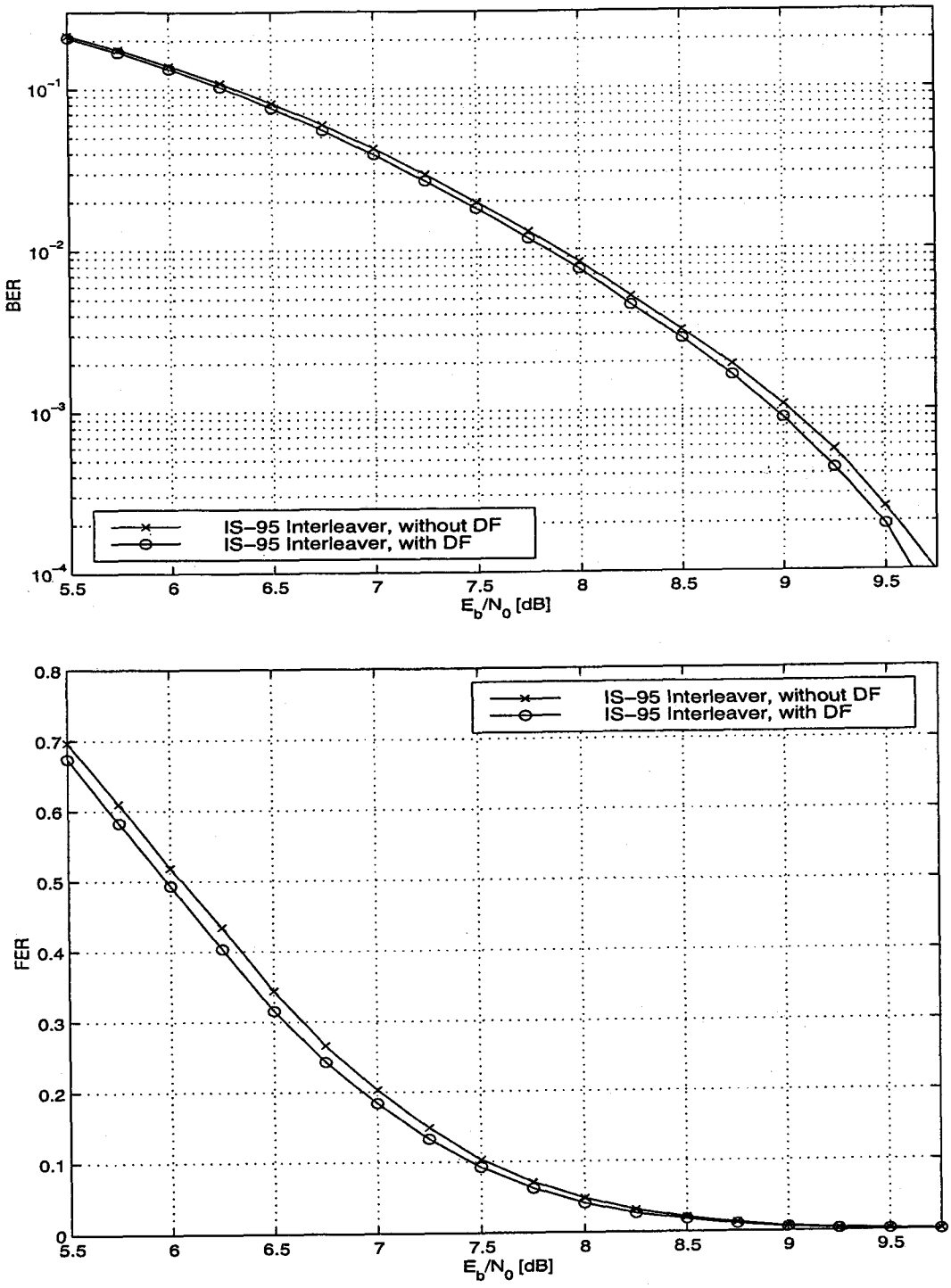


Figure 4.18: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.

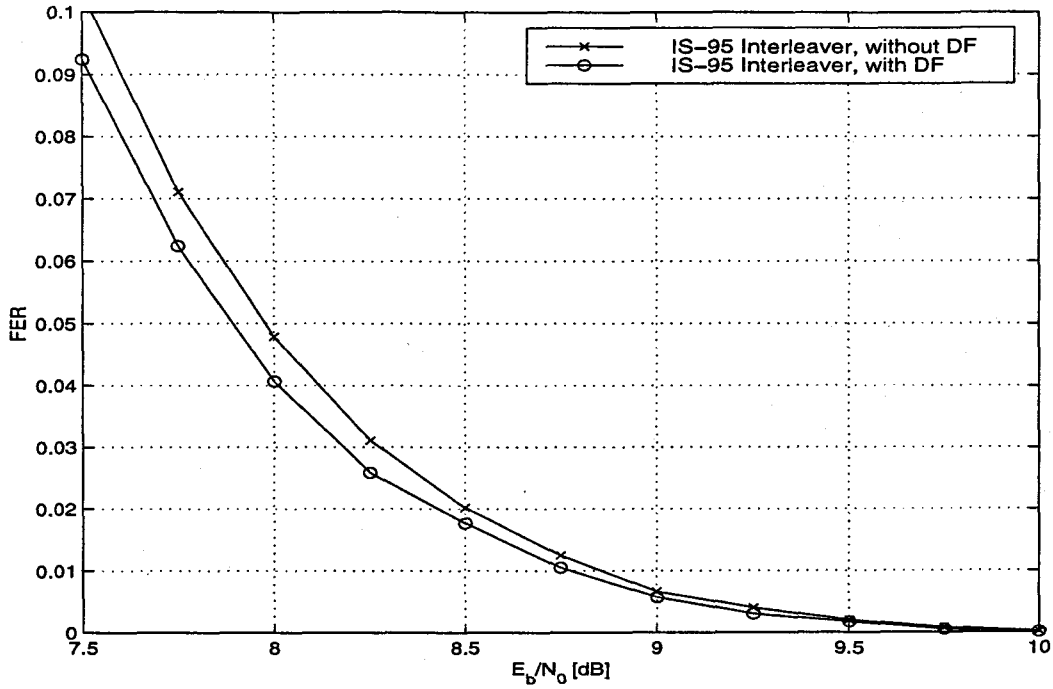


Figure 4.19: FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF. Noncoherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.

4.5. Conclusions

It has been shown in this chapter, at least through computer simulation, that decision feedback decoding (DFD) of IS-95 uplink frames results in small but consistent improvements of bit error rate (BER) and frame error rate (FER) performance. The concept of decision feedback (DF) can therefore be applied to both static and Rayleigh fading multipath propagation in additive white Gaussian noise (AWGN) interference.

The benefit of DF is reduced somewhat by the effects of multipath and fading. The highest reductions are for the cases of *independent Rayleigh fading* without diversity and

for equal-strength *correlated Rayleigh fading* multipath. Nonetheless, the benefit of DFD is a reduction of the required bit-energy-to-interference-density ratio (E_b/N_0) ranging between 0.06-0.10 dB for a BER of 10^{-3} and between 0.06-0.13 dB for a FER of 1%.

Due to the closed-loop power control used in the uplink of an IS-95 system (Section 1.4.2), much of the fading variation is alleviated. Expected system performance should be somewhere close to the results for unfaded multipath presented in section 4.3 of this chapter. Indeed, [35] reported an average required E_b/N_0 of 6.8 dB for the initial field trials of an IS-95 system. Although not directly comparable, this is close to the E_b/N_0 required for a BER of 10^{-3} and a FER of 1% for noncoherent detection of 4 equal-strength multipath signals.

The results of this chapter are used as a basis for comparison for two separate improvements of DFD, which are presented in the following two chapters.

Chapter 5. Performance Improvement with *Earlier Decisions*

In this chapter, a methodology to improve the effectiveness of the decision feedback (DF) receiver/decoder is studied. It is based on reducing the average decoding delay of the convolutional decoder, which is one of the factors that determine it. The other factor, i.e., the interleaver specification remains unchanged. Consequently, the results presented here are useful because they can be applied directly to an IS-95 system.

The average decoding delay of the convolutional decoder is reduced by a new data bit decision criterion that allows faster data bit decisions.

The performance of the modified DF decoder is evaluated for coherent and noncoherent detection of several propagation scenarios. Results of average decoding delay, bit error rate (BER) and frame error rate (FER) are presented. These results will be compared to the original DF decoder design.

Instead of the usual performance degradation it is demonstrated that the DF decoder benefits from some forms of sub-optimal convolutional decoding. This additional performance gain can further improve the quality of service and/or the capacity of systems that are based on the IS-95 standard.

5.1. New Data Bit Decision Criterion

As already mentioned in Section 2.4, data bit decisions with a delay of less than 10 or 11 decoding steps are required for DF to be most effective. However, for the

convolutional code used in the IS-95 uplink this is not achieved in practice at the bit error rates (BERs) of interest. The results presented later in this chapter indicate that at the BER of interest for voice communications (10^{-3}) the average decoding delay is on the order of 21-35 decoding steps (63-105 convolutionally coded bits).

Due to the standardized interleaver specification, the data bit decoding delays of the convolutional decoder determine the effectiveness of decision feedback decoding (DFD). If the decoding delays are reduced, the effectiveness of DFD is improved, i.e., the number of decoding metrics that benefit from DF updates is increased.

A reduction of the decoding delay can be achieved by using some form of sub-optimal data bit decision criterion within the convolutional decoder. The trade-off is, of course, performance degradation because these data bit decisions have a higher probability of error. In the case of the DF decoder, however, a net performance improvement can result if the performance improvement of more efficient DF exceeds this degradation.

A well-known method of sub-optimal Viterbi decoding is to use a fixed chain-back length, i.e., the size of the decoder path-memory is limited to a fixed length (smaller than the frame length). This is used in practical applications to reduce the memory requirements of the convolutional decoder [44]. As soon as the decoder path-memory is filled, one data bit is output after each decoding step. This method guarantees a fixed decoding delay for each data bit of the frame. However, a path-history length of several constraint lengths K is required for good performance of the convolutional decoder.

According to [59], five constraint lengths are usually sufficient. Application of this sub-optimal decoding technique to the DF decoder does not improve the effectiveness of DFD because it allows for at most 1 DF metric update for each Walsh group. Lowering the chain-back length, in order to guarantee a higher number of DF metric updates, degrades the convolutional code performance too much to be beneficial [60].

Another way to reduce the decoding delay of the convolutional decoder is investigated here. The goal is to reduce the *average* decoding delay while maintaining good performance of the convolutional code. This is achieved by relaxing the criterion for making data bit decisions.

Instead of waiting until all the bit histories converge to the same data bit value, the decoder only waits for a certain number M of bit histories (including the one with the highest state metric) to converge. It remains likely, but is no longer guaranteed, that those data bit decisions are identical with the data bit decisions obtained by chaining-back from the all-zeros state at the end of the frame. As a result, performance degradation is expected when DF is not used.

This method of sub-optimal Viterbi decoding is referred to as *earlier decisions*. With an appropriate choice of the parameter M the decoder continues to make reliable data bit decisions. At the same time the average decoding delay is reduced, resulting in a higher number of DF metric updates. It is shown, by means of computer simulation, that this increases the performance gain of the DF decoder.

5.2. Performance Evaluation

Average values of the decoding delay, as well as bit error rate (BER) and frame error rate (FER) have been determined for unfaded single-path signals, Rayleigh fading single-path signals, unfaded multipath signals, and Rayleigh fading multipath signals. The parameter M = 221 has been chosen heuristically for the cases using *earlier decisions*.

The figures showing the average data bit decoding delays establish the values that have to be expected for the different propagation scenarios. They also illustrate the effect of DF on the average decoding delays and their reduction when the new data bit decision criterion is used.

The figures showing BER and FER are used to determine the additional performance improvement of *earlier decisions* over the original DF receiver/decoder design. They also show the degradation that the sub-optimal convolutional decoding causes for a non-DF decoder.

5.2.1. Unfaded Single-Path

In this section, the performance of the new data bit decision criterion is evaluated in the presence of additive white Gaussian noise (AWGN) interference. Figure 5.1 and Figure 5.2 show the results for coherent and noncoherent detection, respectively. The average decoding delay of each data bit is plotted as a function of the bit-energy-to-interference-density (E_b/N_0) ratio with and without DF using regular and *earlier decisions*.

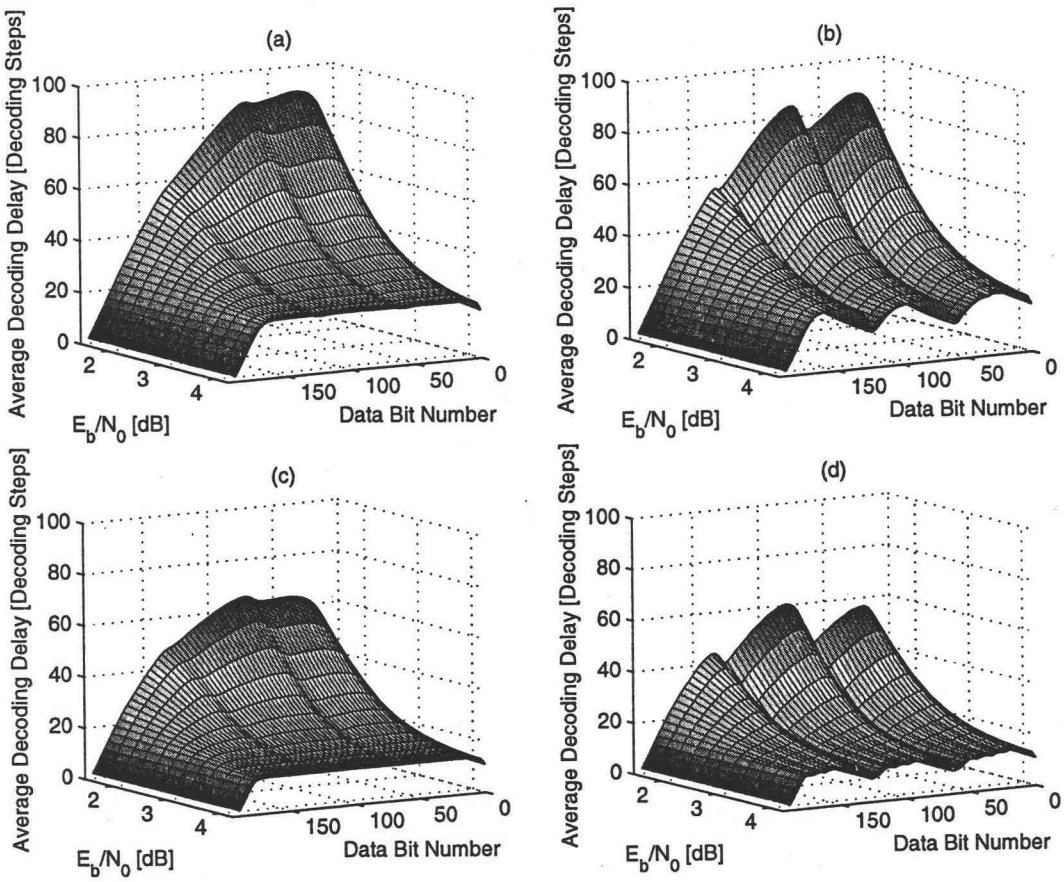


Figure 5.1: Average data bit decoding delays as a function of E_b/N_0. Coherent detection of unfaded single-path signal in AWGN using the IS-95 interleaver (Uplink, 9.6 kbits/s). (a) without DF, (b) with DF, (c) *earlier decisions*, without DF, (d) *earlier decisions*, with DF.

Note that, except for the expected degradation in E_b/N_0, the results for noncoherent detection are very similar to the coherent results. This indicates that the average decoding delay is not so much determined by the used receiver front-end but by the used decoder and the BER or FER at which the system is operating.

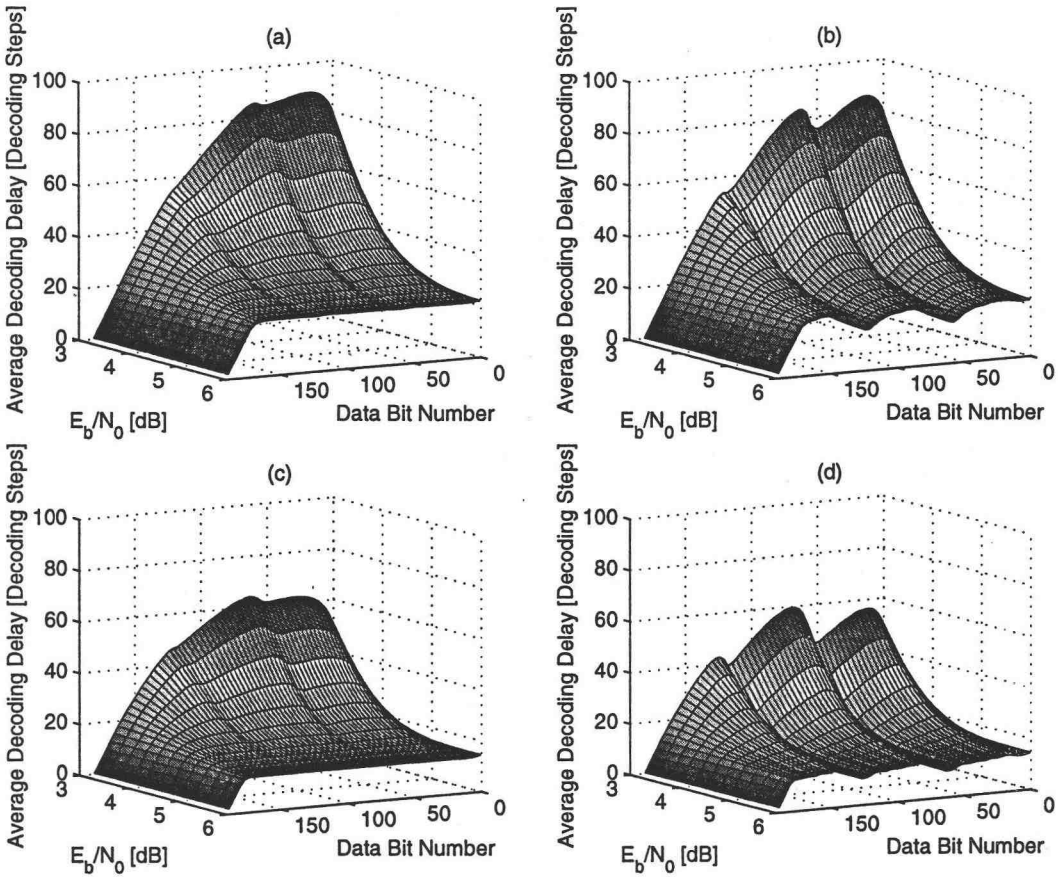


Figure 5.2: Average data bit decoding delays as a function of E_b/N_0. Noncoherent detection of unfaded single-path signal in AWGN using the IS-95 interleaver (Uplink, 9.6 kbits/s). (a) without DF, (b) with DF, (c) *earlier decisions*, without DF, (d) *earlier decisions*, with DF.

Without DF an approximately constant average decoding delay of is visible (for a fixed value of E_b/N_0). The average decoding delay decays with increasing E_b/N_0 and levels out at about 15 decoding steps. The linear drop (slope -1) at the end of the frame is due to data bit decisions that are made after the last decoding step. This is more pronounced for lower E_b/N_0 values indicating that in these cases more and more data bits are left undetermined until the convolutional code is tailed-off.

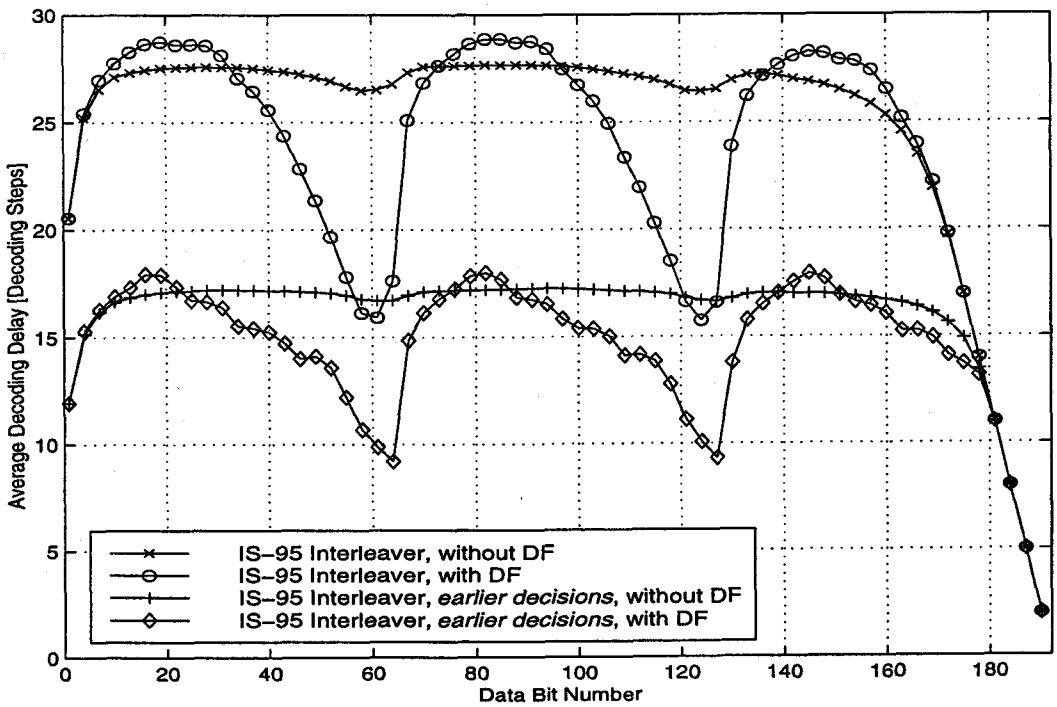


Figure 5.3: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Coherent detection of unfaded single-path signal in AWGN. E_b/N_0=3.5 dB.

Note the small reductions of the average decoding delay for several data bits right before data bits 65 and 129. These mark the three stages of the frame decoding process that are due to the IS-95 specified interleaver (Section 1.4.3.2).

With DF the three stages of the frame decoding process are clearly visible. Initially, the average decoding delay remains at a slightly higher value than without DF. It then continuously drops until a new decoding stage begins.

Figure 5.3 and Figure 5.4 show the average decoding delay for a fixed E_b/N_0 value of 3.5 dB in the case of coherent detection and 5 dB in the case of noncoherent

detection. This corresponds to a BER close to 10^{-3} in both cases.

The average decoding delay without DF is found to be around 27 decoding steps. Note again the small reductions before data bits 65 and 129.

The plots using DF match the plots without DF at the beginning and end of the frame. In between, similar behavior is observed in each of the three frame decoding stages. Initially, the average decoding delay rises to about 29 decoding steps and remains there until approximately the middle of the decoding stage (data bits 32, 96 and 160).

For the first two decoding stages a sharp, almost linear drop to 16 decoding steps follows this. In the third decoding stage the plot merges the (slope -1) end of frame behavior observed without DF.

This can be interpreted as a sign for improved performance using DFD. As more and more DF information becomes available, the decoder path-histories converge faster (on average) to a unique data bit value. When a new decoding stage begins, no DF information is available and the average decoding delay increases.

The results using *earlier decisions* show a significant reduction of the average decoding delay as expected due to the new data bit decision criterion. Besides that, the plots show similar characteristics than the ones using regular data bit decisions. With DF the reduction of the average decoding delay toward the end of the three decoding stages is again much more pronounced than without DF. At a BER of 10^{-3} the average decoding delay is reduced to 17 decoding steps, a reduction of 10 decoding steps, for both coherent and noncoherent detection.

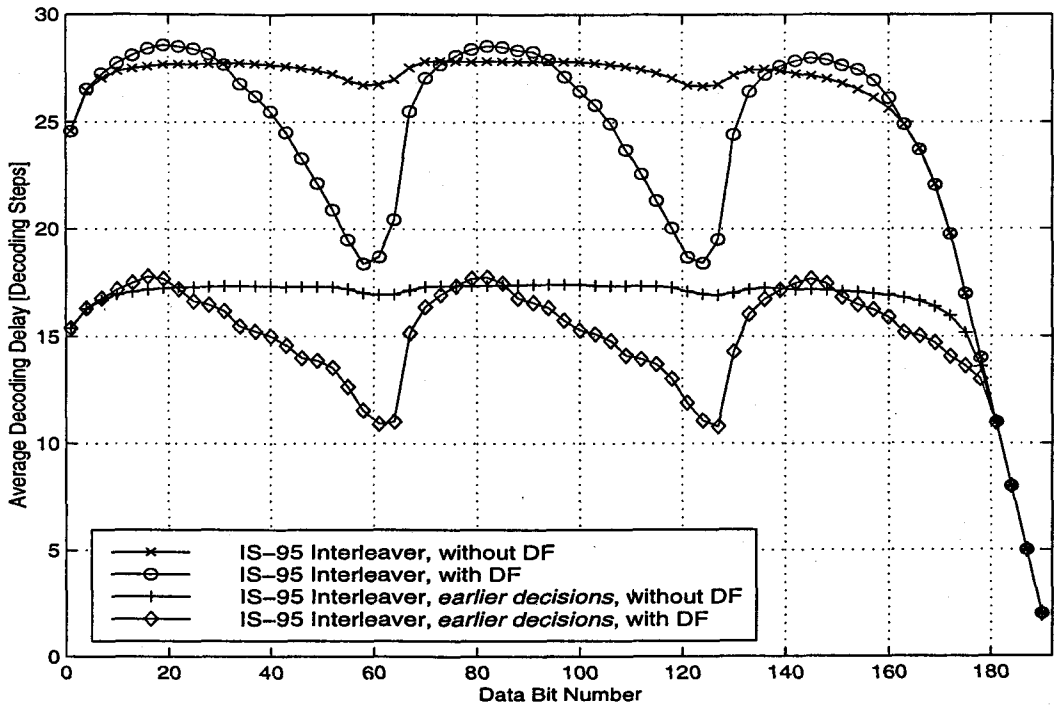


Figure 5.4: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Noncoherent detection of unfaded single-path signal in AWGN. E_b/N_0=5 dB.

Note that the use of *earlier decisions* slightly changes the shape of the average decoding delay within the three decoding stages. As before, the average decoding delay increases continuously, but only during the first quarter of each decoding stage with a peak approximately one decoding step above the plot without DF. In the middle part of each decoding stage, this is followed by an almost staircase like (3 steps are visible) descent down to about 14 decoding steps. Finally, there is an almost linear, steeper decrease down to about 9 decoding steps at the end of the first two decoding stages and, as before, down to zero at the end of the frame. The staircase like descent of the decoding delay marks the reuse of Walsh groups with

added DF feedback information in each of the stages of the frame decoding process.

So far, it has been demonstrated that *earlier decisions* effectively reduce the average decoding delay of the convolutional decoder as intended.

The impact of this on the effectiveness of DFD can be seen in the following figures, which show BER and FER as a function of E_b/N_0 in AWGN interference. The coherent results are shown in Figure 5.5 and the noncoherent results are shown in Figure 5.6.

In both cases, *earlier decisions* without DF lead to small performance degradation. This is expected due to the sub-optimality of the decoder. *Earlier decisions* with DF, on the other hand, result in a clearly visible performance improvement for all E_b/N_0 values shown. This improvement is mostly larger than the degradation incurred without the use of DF.

For coherent detection at BER 10^{-3} the degradation due to *earlier decisions* without DF is 0.02 dB. The performance improvement for *earlier decisions* with DF is 0.06 dB. The performance gain of the original DF decoder design is increased from 0.09 dB to 0.15 dB, a factor of 1.67. The required E_b/N_0 to achieve this BER is now 3.46 dB.

The FER plot shows an additional gain of about 0.1 dB by using *earlier decisions* for all E_b/N_0 values. At 1% FER the simulation data shows a performance degradation due to *earlier decisions* without DF of 0.06 dB. The performance improvement for *earlier decisions* with DF is 0.08 dB. The effectiveness of the original DF decoder design is increased from 0.1 dB to 0.18 dB, a factor of 1.8. The required E_b/N_0 to achieve this FER is 3.52 dB.

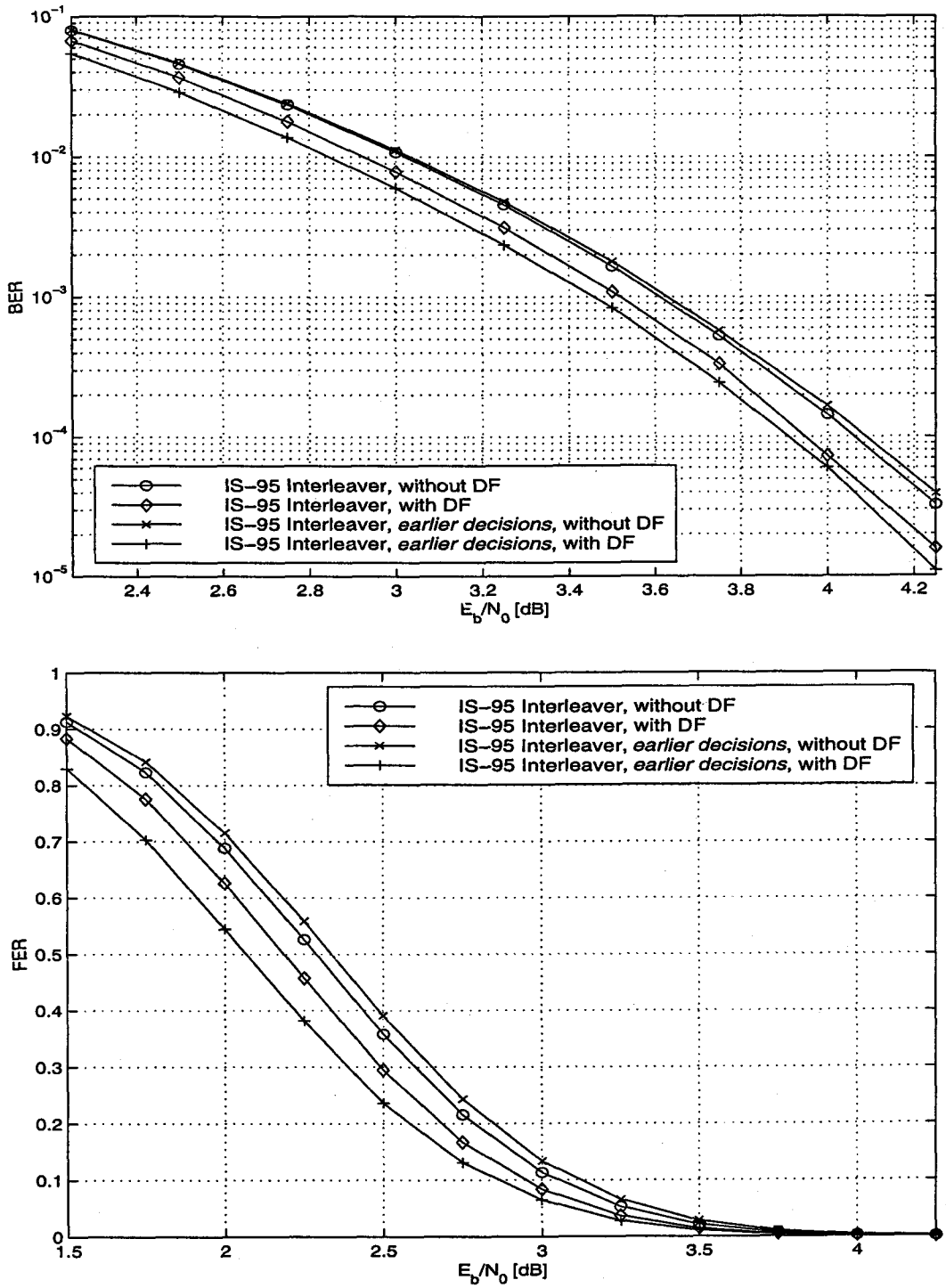


Figure 5.5: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Coherent detection of unfaded single-path signal in AWGN.

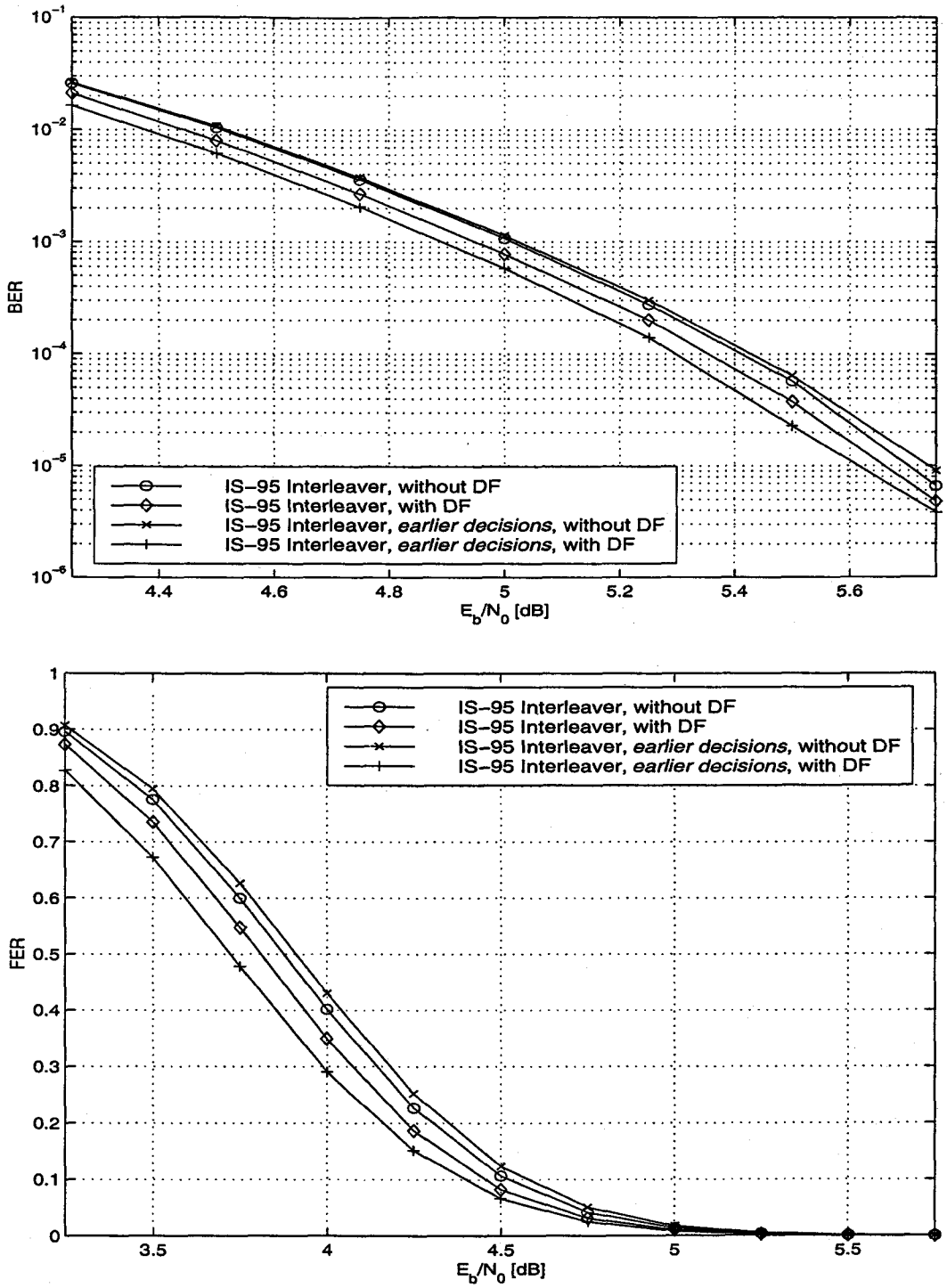


Figure 5.6: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Noncoherent detection of unfaded single-path signal in AWGN.

The results for noncoherent detection show similar performance improvement due to *earlier decisions*.

At BER 10^{-3}, performance is improved by an additional 0.06 dB to a total of 0.12 dB. The required E_b/N_0 to achieve this BER is 4.89 dB. The performance degradation for *earlier decisions* without DF is only very small here, about 0.01 dB.

The FER improvement that is visible in the plot is almost 0.1 dB. At the 1% FER however, the additional improvement is only 0.03 dB resulting in a total improvement of 0.12 dB. The required E_b/N_0 to achieve this FER is 4.97 dB. In this case the performance degradation for *earlier decisions* without DF is 0.06 dB.

5.2.2. Rayleigh Fading Single-Path

The results for the average data bit decoding delays for an unfaded *independent Rayleigh fading* single-path signal have a very similar shape compared to the unfaded results.

Figure 5.7 shows the coherent results for an E_b/N_0 value of 5.25 dB and Figure 5.8 shows the noncoherent results for an E_b/N_0 of 7.25 dB. Both figures correspond to a BER close to 10^{-3}.

The degrading effects caused by the fading of the signal amplitude are seen in an increase of the average decoding delay to about 34 decoding steps.

The end of the 3 decoding stages is again marked by a reduction of the average decoding delay. Compared to the unfaded cases (Figure 5.3 and Figure 5.4), the reduction is more pronounced without DF and decreased with DF.

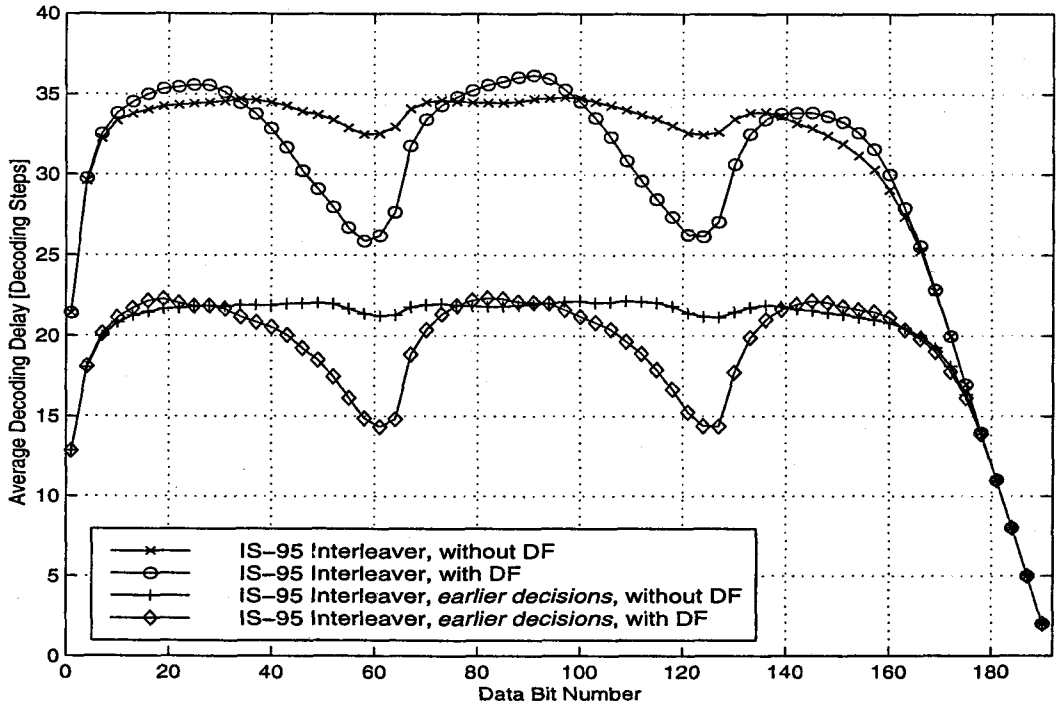


Figure 5.7: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Coherent detection of *independent Rayleigh fading* single-path signal in AWGN. E_b/N_0=5.25 dB.

The shape of the decoding delay in this case is much smoother, i.e., the staircase like decrease of the average decoding delay is not observed here. This is another result of the variation of the received signal strength caused by the fading.

The use of *earlier decisions* reduces the average decoding delay by about 12 decoding steps to a value of 22 decoding steps.

The BER and FER are shown in Figure 5.9 and Figure 5.10, respectively, for coherent and noncoherent detection.

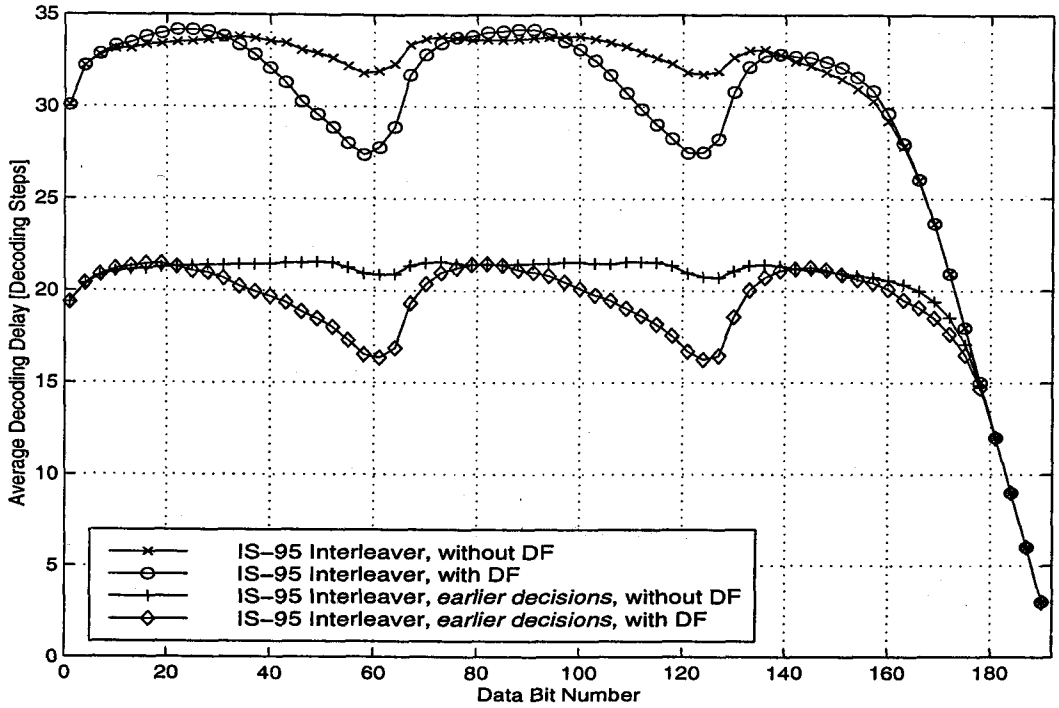


Figure 5.8: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Noncoherent detection of *independent Rayleigh fading* single-path signal in AWGN. E_b/N_0=7.25 dB.

In this case, the BER degradation of *earlier decisions* without DF is very small. However, a clearly noticeable improvement is seen with the use of DF.

The higher values of the FER also show the performance degradation and improvement due to *earlier decisions*.

At a BER of 10^{-3}, the required E_b/N_0 is reduced by an additional 0.11 dB for coherent and 0.07 dB for noncoherent detection. The combined E_b/N_0 reduction by DF and *earlier decisions* is, respectively, 0.19 dB and 0.13 dB. The required E_b/N_0 to achieve this BER is, respectively, 5.25 dB and 7.30 dB.

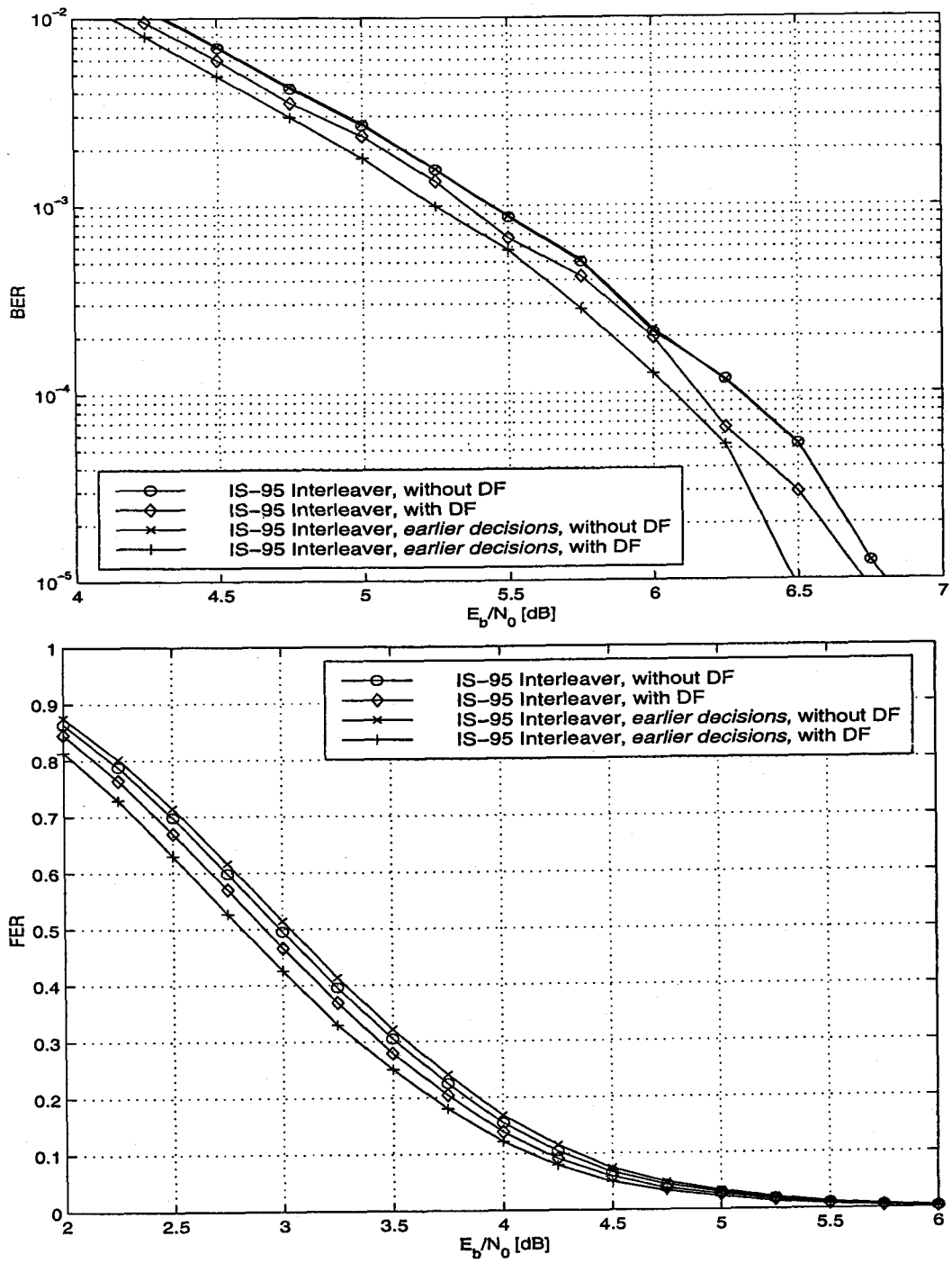


Figure 5.9: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Coherent detection of *independent Rayleigh fading* single-path signal in AWGN.

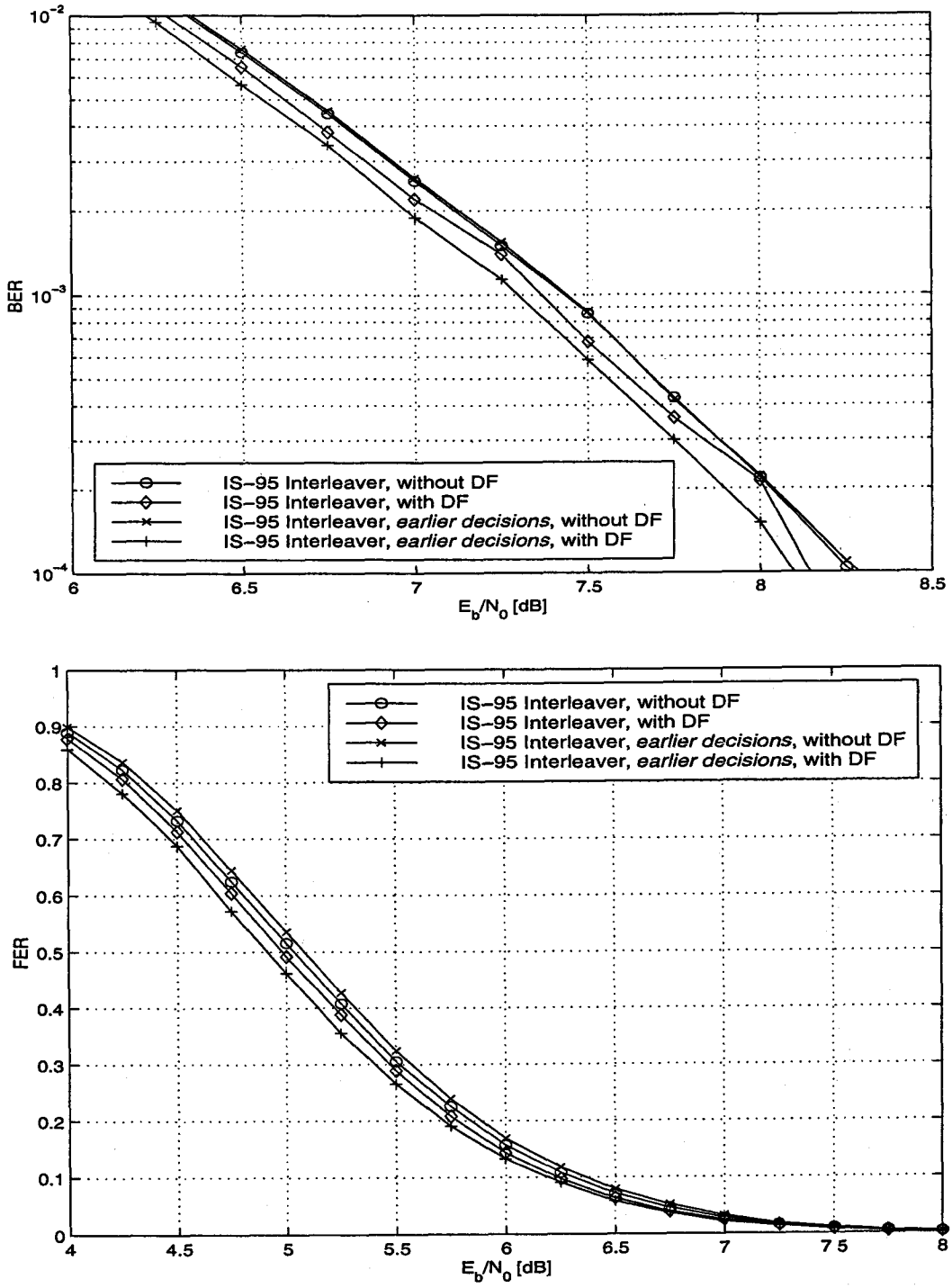


Figure 5.10: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Noncoherent detection of *independent Rayleigh fading* single-path signal in AWGN.

Similarly, at a FER of 1%, the required E_b/N_0 is reduced by an additional 0.05 dB for coherent and 0.04 dB for noncoherent detection. The combined E_b/N_0 reduction by DF and *earlier decisions* is, respectively, 0.13 dB and 0.11 dB. The required E_b/N_0 to achieve this FER is, respectively, 5.40 dB and 7.39 dB.

5.2.3. Unfaded Multipath

The results in this section are for 4 equal-strength unfaded multipath signals in AWGN interference.

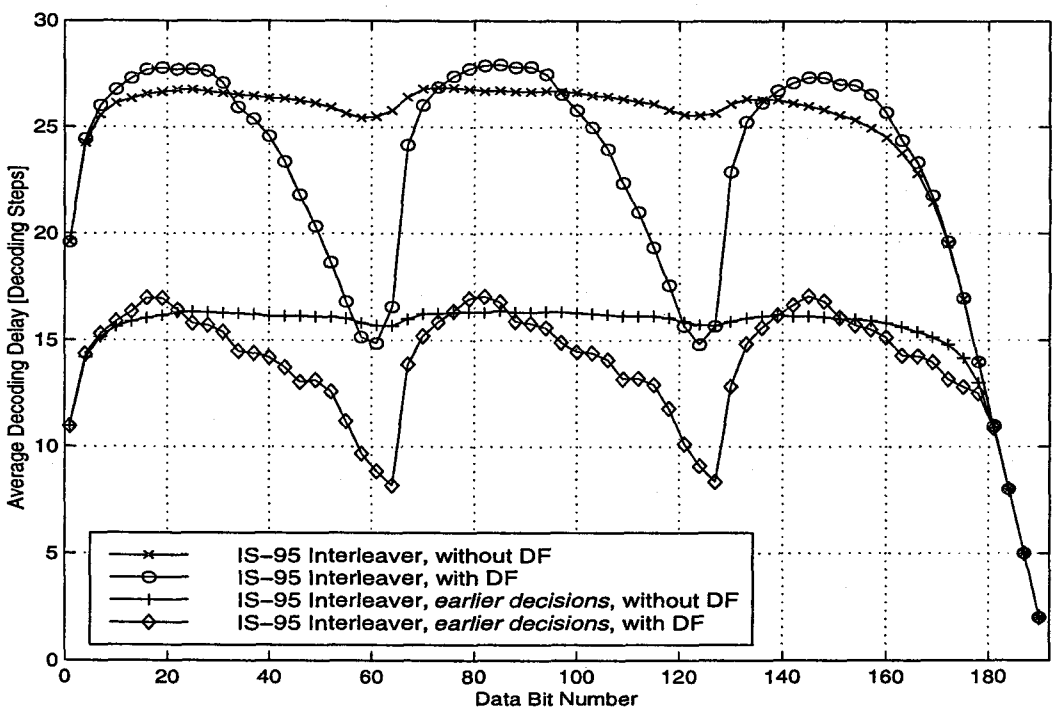


Figure 5.11: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Coherent detection of 4 equal-strength unfaded multipath signals in AWGN. E_b/N_0=3.50 dB.

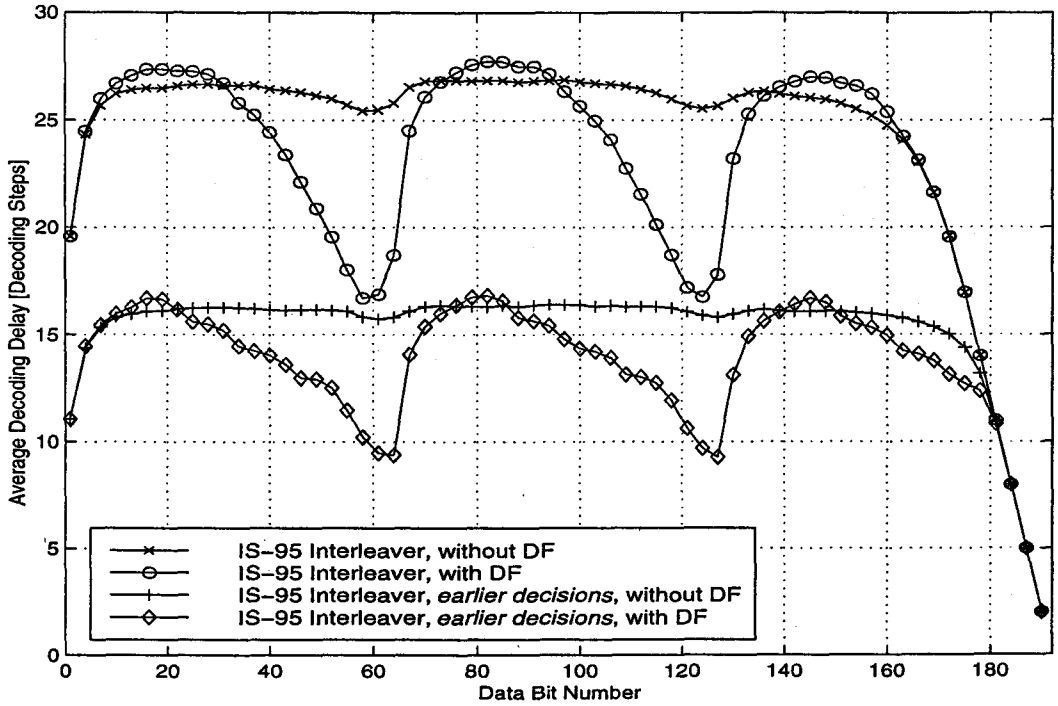


Figure 5.12: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Noncoherent detection of 4 equal-strength unfaded multipath signals in AWGN. E_b/N_0=6.75 dB.

The average data bit decoding delays are shown in Figure 5.11 and Figure 5.12. The E_b/N_0 is 3.50 dB for the coherent results and 6.75 dB for the noncoherent results. The use of *earlier decisions* reduces the average decoding delay by 11 decoding steps for both coherent and noncoherent detection.

As in Chapter 4, the results for coherent detection (Figure 5.11 and Figure 5.13) are basically identical to the single-path case (Figure 5.3 and Figure 5.5). The small differences are due to the smaller number of simulated frames and the different realizations of the interference.

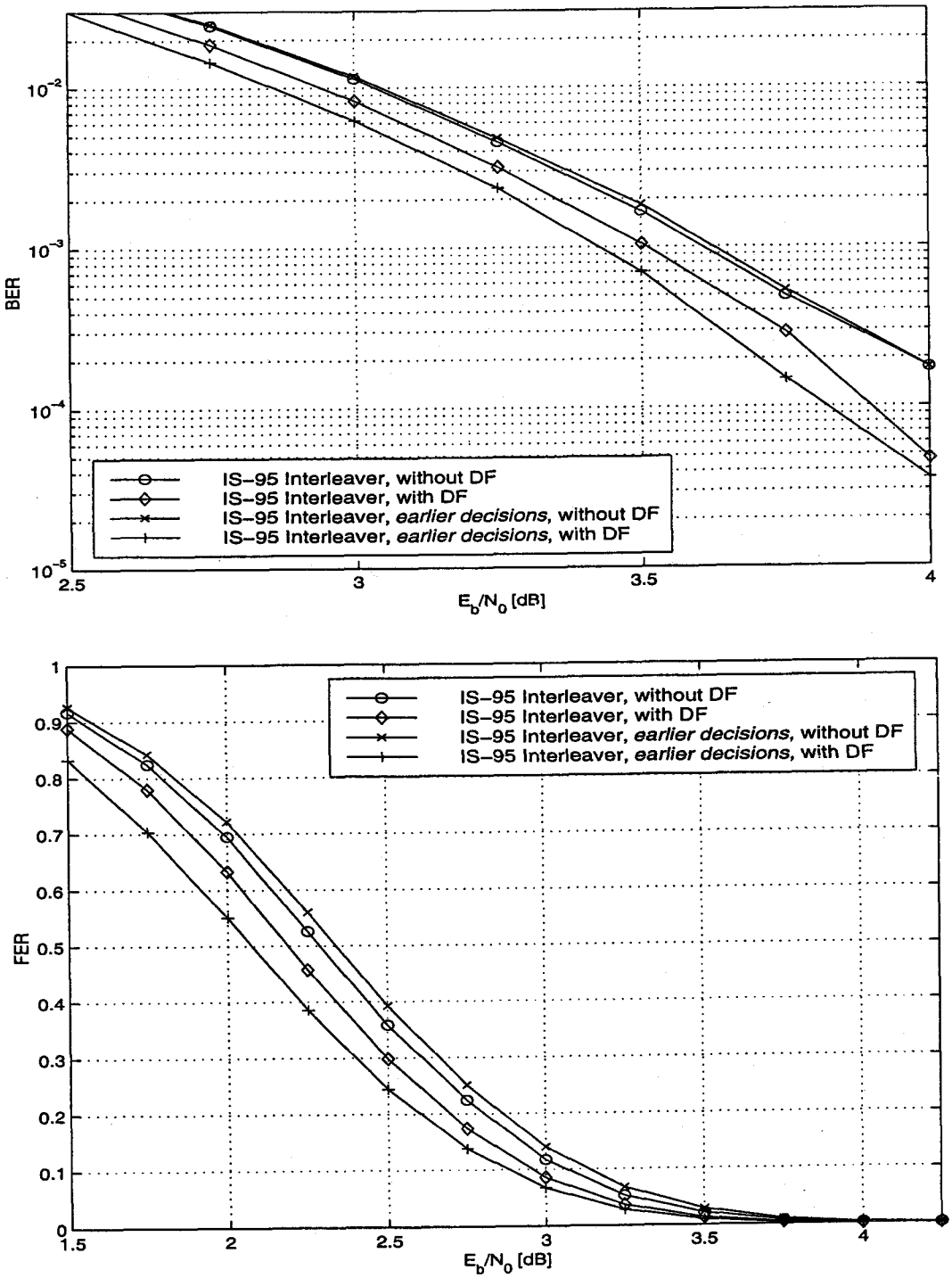


Figure 5.13: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Coherent detection of 4 equal-strength unfaded multipath signals in AWGN.

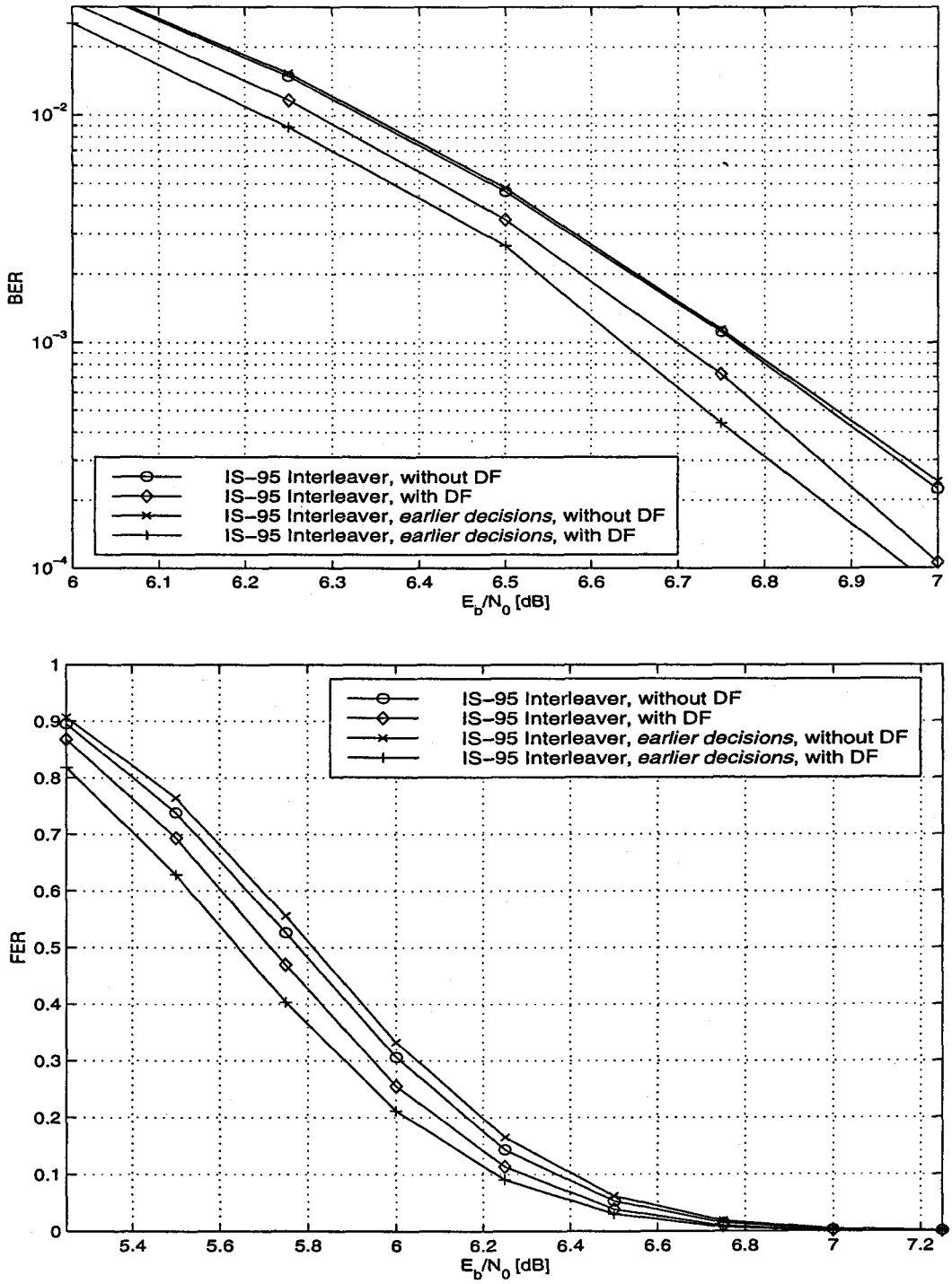


Figure 5.14: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Noncoherent detection of 4 equal-strength unfaded multipath signals in AWGN.

The noncoherent BER and FER results in Figure 5.14 show a noncoherent combining loss of about 1.76 dB compared to the single-path case (Figure 5.6). Accounting for this, the effect of *earlier decisions* on the BER and FER performance is almost the same as in the single-path case.

At a BER of 10^{-3}, the required E_b/N_0 is reduced by an additional 0.06 dB resulting in a combined reduction of 0.13 dB by DF and *earlier decisions*.

The degradation without DF is smaller than 0.01 dB in this case. The E_b/N_0 value required to achieve this BER is 6.64 dB.

The FER improvement that is visible in the plot is between 0.07-0.08 dB. At 1% FER however, the additional improvement is only 0.03 dB resulting in a total improvement of 0.14 dB. The required E_b/N_0 to achieve this FER is 6.72 dB. In this case the performance degradation for *earlier decisions* without DF is 0.05 dB.

5.2.4. Rayleigh Fading Multipath

The results in this section show the results for 4 equal-strength Rayleigh fading multipath signals in AWGN interference. Again, two cases are considered: *independent Rayleigh fading* and *correlated Rayleigh fading* (Section 3.4).

5.2.4.1. Independent Rayleigh Fading

The availability of multipath diversity is beneficial to the average data bit decoding delays. Figure 5.15 and Figure 5.16 show the average decoding delay for a fixed

E_b/N_0 value of 4.00 dB in the case of coherent detection and 7.25 dB in the case of noncoherent detection (corresponding to a BER close to 10^{-3}).

For a Rayleigh fading single-path signal (Figure 5.7 and Figure 5.8), the average decoding delay was increased by about 8 decoding steps and the delay reductions caused by the IS-95 interleaver were much less and smoother compared to the unfaded case (Figure 5.3 and Figure 5.4). Here, the average decoding delay is increased by only about 1-2 decoding steps and is very similar in shape. *Earlier decisions* reduce the average decoding delay by about 11 decoding steps.

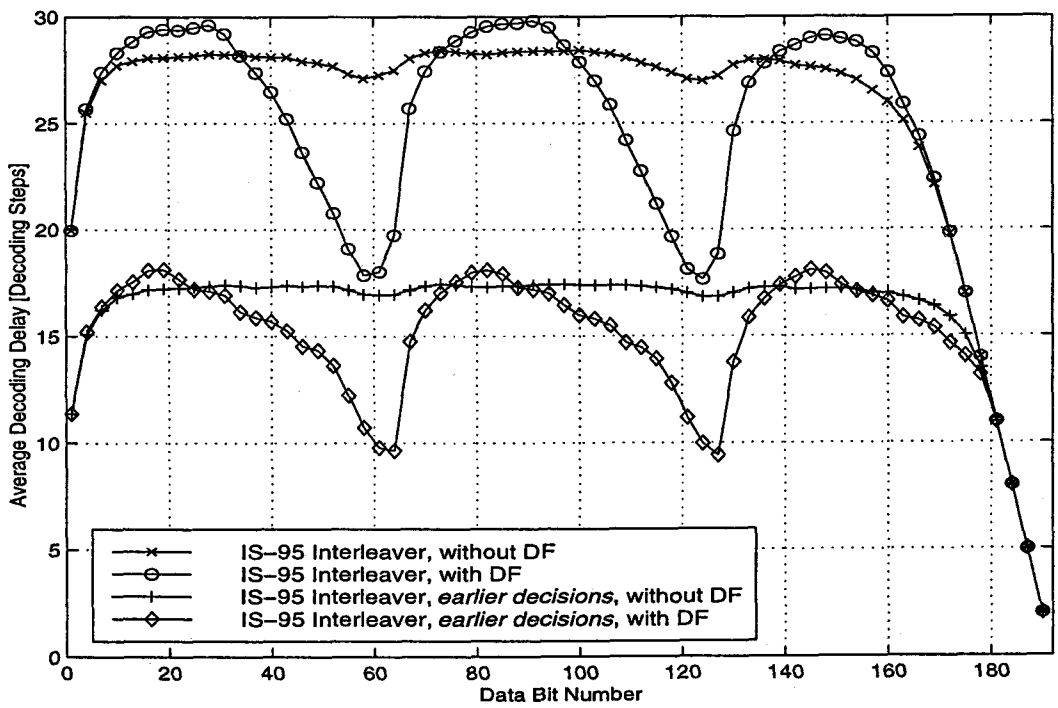


Figure 5.15: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Coherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN. E_b/N_0=4 dB.

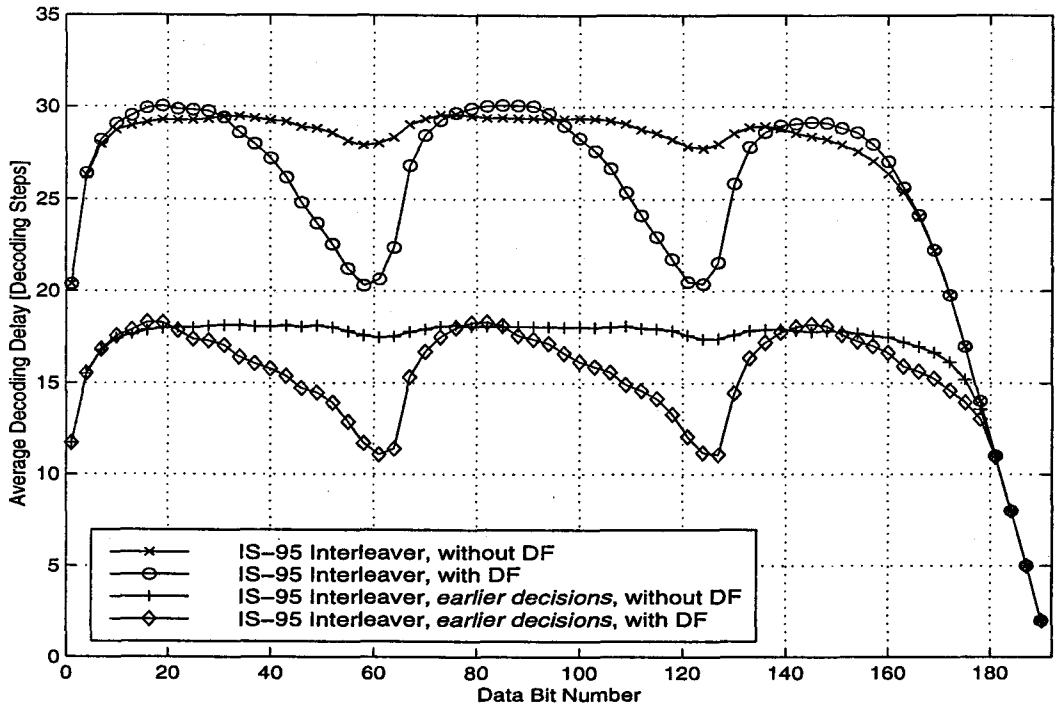


Figure 5.16: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Noncoherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN. E_b/N_0=7.25 dB.

The BER and FER results show the performance improvement resulting from the reduced data bit decoding delays.

In the coherent case (Figure 5.17), the required E_b/N_0 is reduced an additional 0.06 dB at BER 10^{-3}. An E_b/N_0 of 3.86 dB is needed to achieve this BER. The combined E_b/N_0 reduction by DF and *earlier decisions* is 0.14 dB.

At 1% FER, the total E_b/N_0 reduction is improved by 0.05 dB from 0.09 dB to 0.14 dB. The E_b/N_0 required to achieve this FER is 3.93 dB.

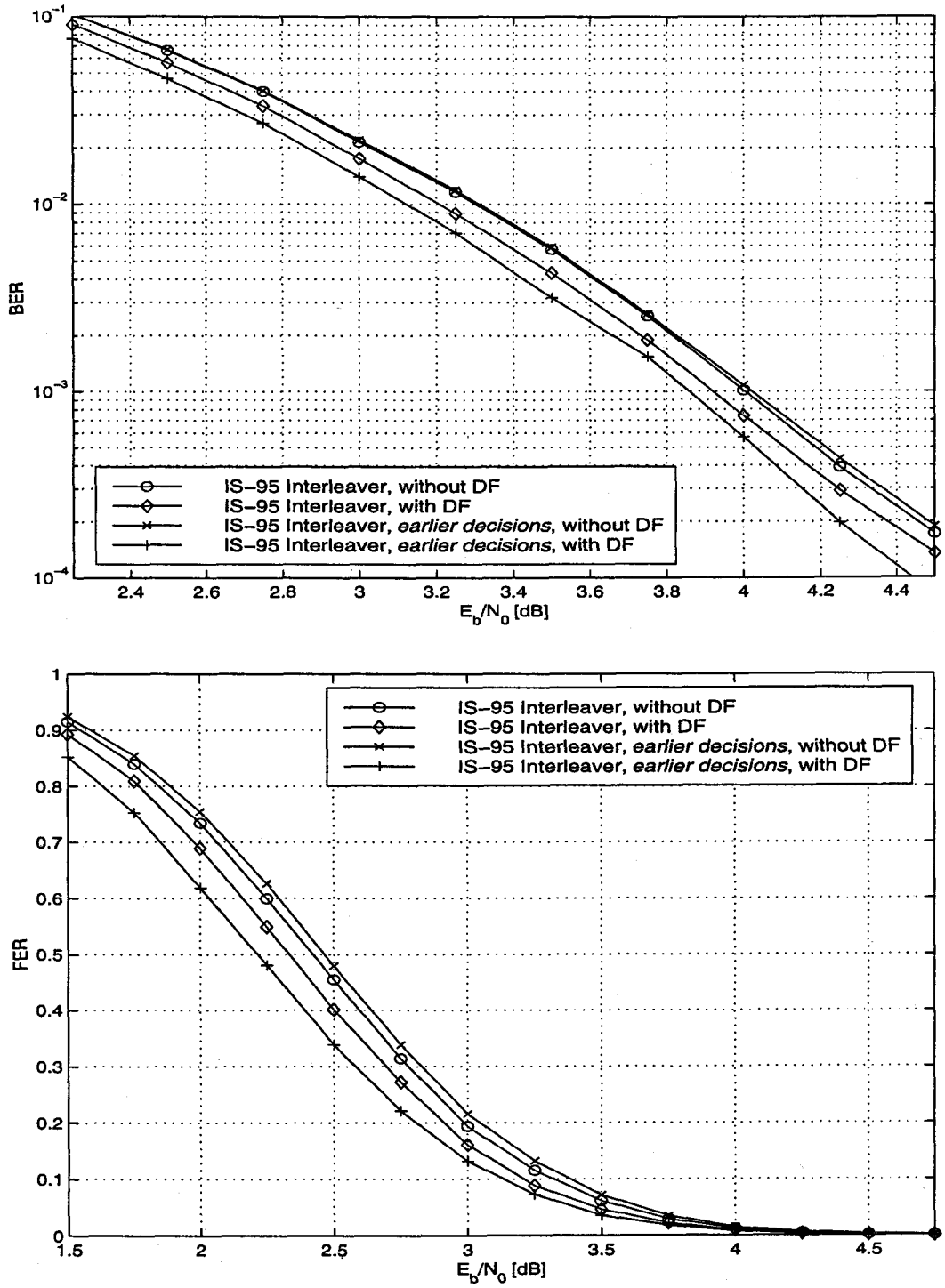


Figure 5.17: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Coherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

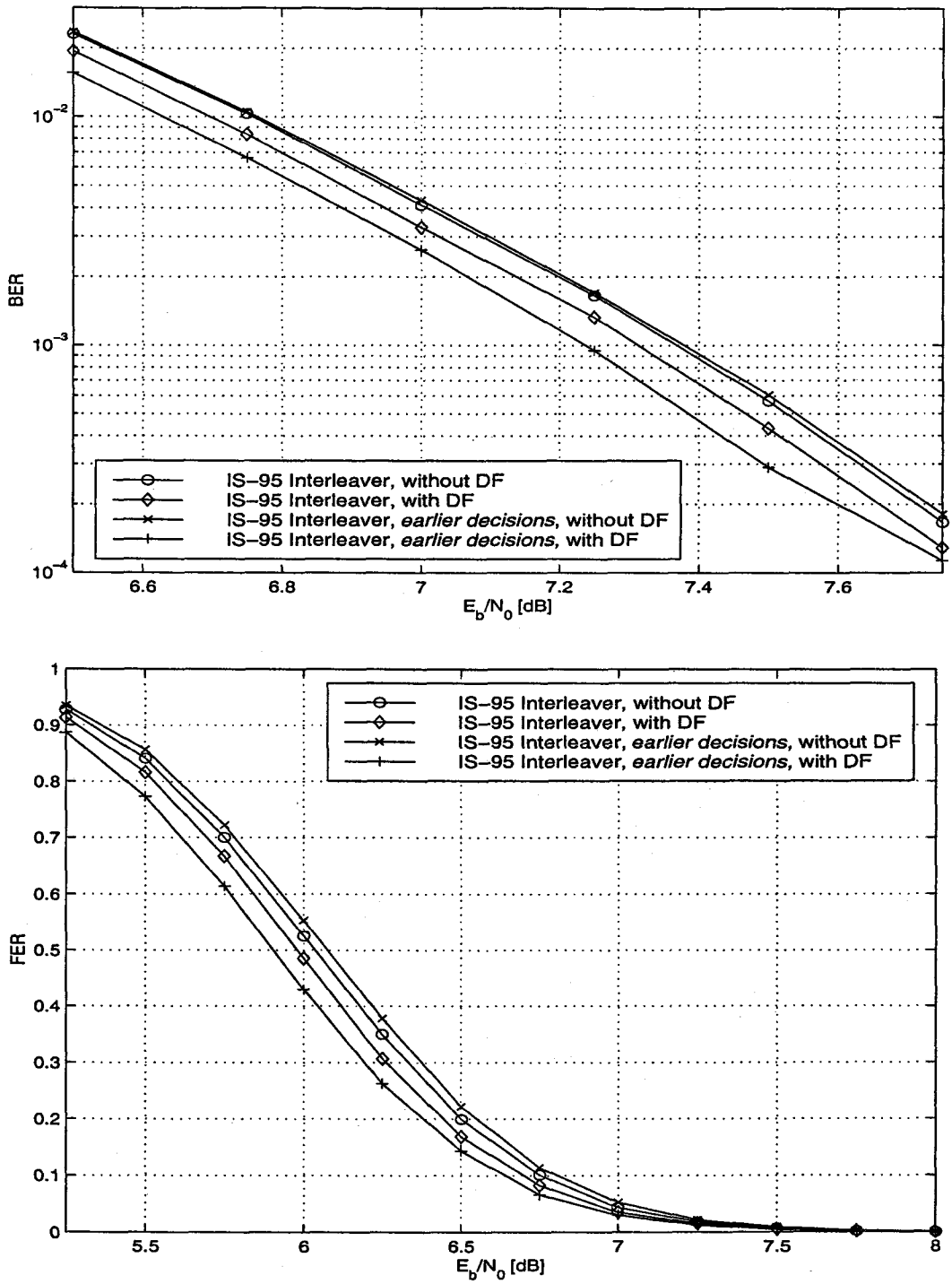


Figure 5.18: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Noncoherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

Using noncoherent detection (Figure 5.18), the required E_b/N_0 for a BER of 10^{-3} is reduced an additional 0.07 dB to 7.24 dB. The combined E_b/N_0 reduction by DF and *earlier decisions* is 0.13 dB. Similarly, for a FER of 1%, the required E_b/N_0 is reduced an additional 0.07 dB to 7.31 dB. The combined E_b/N_0 reduction is also 0.13 dB.

5.2.4.2. Correlated Rayleigh Fading

The average data bit decoding delays at a BER of approximately 10^{-3}, are shown in Figure 5.19 for coherent detection with an E_b/N_0 value of 5.25 dB and in Figure 5.20 for noncoherent detection with an E_b/N_0 value of 9.00 dB.

In the coherent case, *earlier decisions* reduce the average decoding delay by 9 decoding steps from 23 to 14. In the noncoherent case, the average decoding delay is reduced by 7 decoding steps from 21 to 14.

The average decoding delays are lower here than in the case of *independent Rayleigh fading* (Figure 5.15 and Figure 5.16). This is because a greater E_b/N_0 value is required to achieve a BER of 10^{-3}, while the average decoding delay as a function of E_b/N_0 remains approximately the same.

For coherent detection (Figure 5.21), the decoder performance at a BER of 10^{-3} is improved an additional 0.11 dB from 5.37 dB to 5.26 dB. The combined E_b/N_0 reduction by DF and *earlier decisions* is improved from 0.08 dB to 0.19 dB.

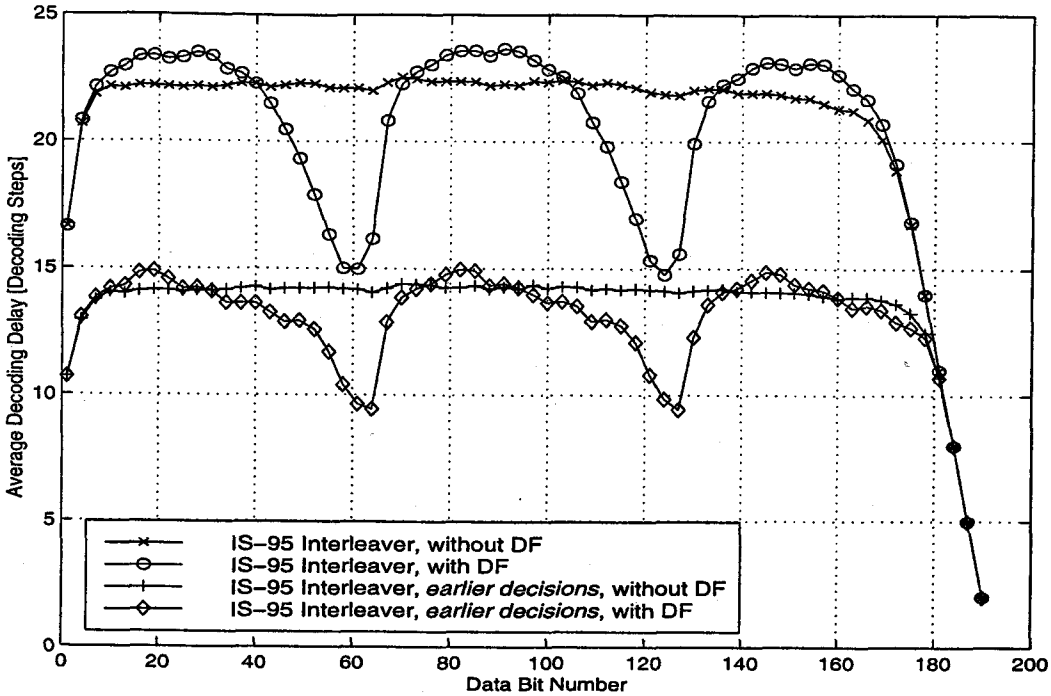


Figure 5.19: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Coherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN. E_b/N_0=5.25 dB.

The required E_b/N_0 for a FER of 1% is reduced an additional 0.08 dB from 5.26 dB to 5.18 dB. The use of *earlier decisions* increases the original DF performance improvement from 0.13 dB to 0.21 dB.

For noncoherent detection (Figure 5.22), the decoder performance at a BER of 10^{-3} is improved an additional 0.07 dB from 8.95 dB to 8.88 dB. The combined E_b/N_0 reduction by DF and *earlier decisions* is improved from 0.08 dB to 0.15 dB.

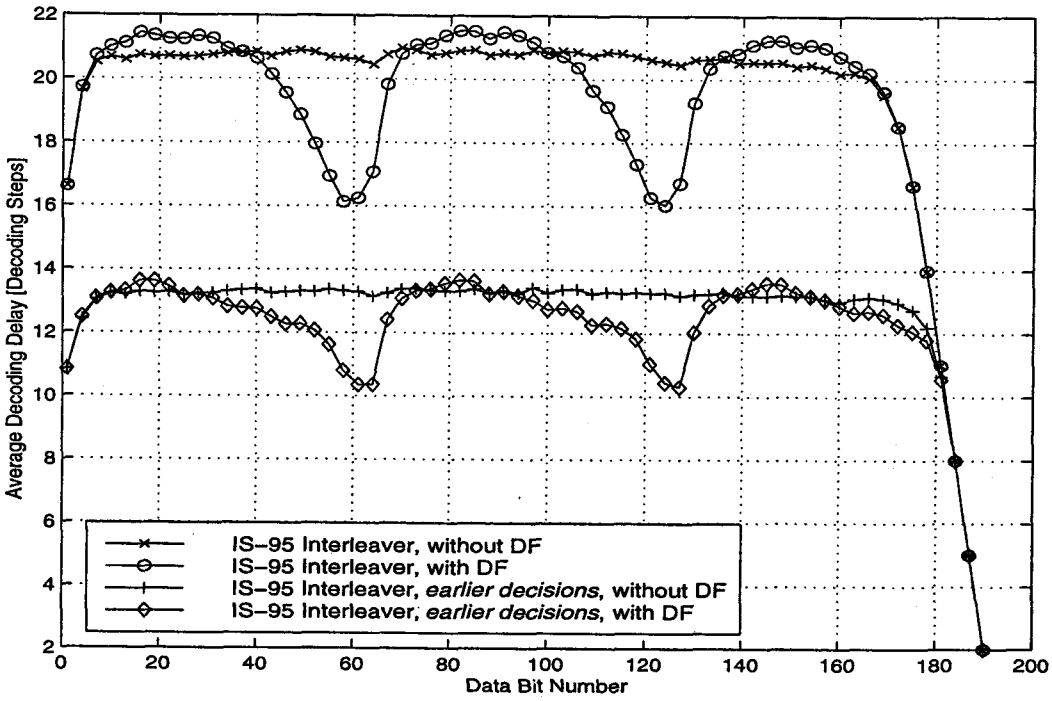


Figure 5.20: Average data bit decoding delays for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier* decisions. Noncoherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN. E_b/N_0=9 dB.

The required E_b/N_0 for a FER of 1% is reduced an additional 0.08 dB from 8.78 dB to 8.70 dB. The use of *earlier decisions* increases the DF performance improvement of 0.08 dB by a factor of 2.

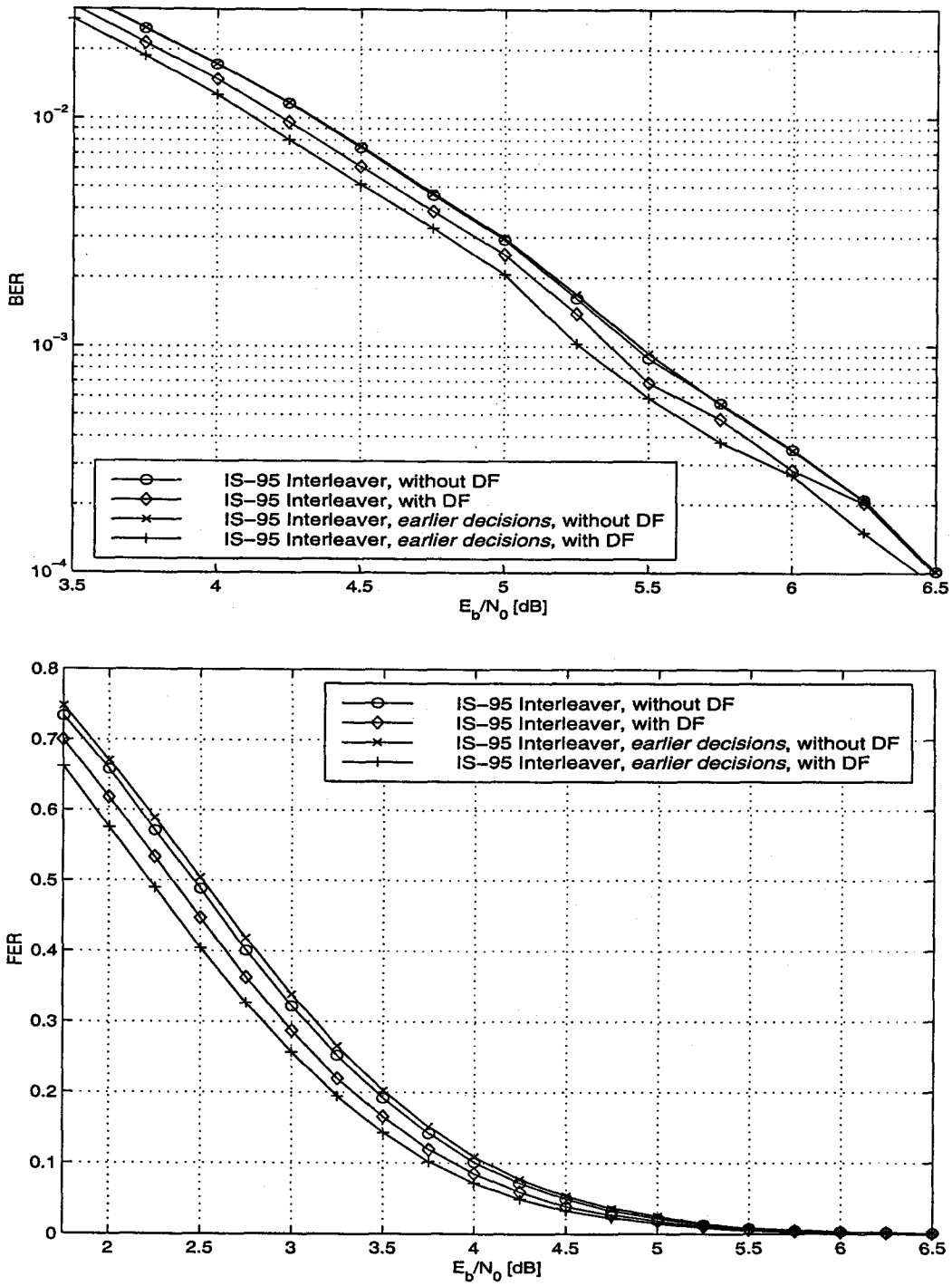
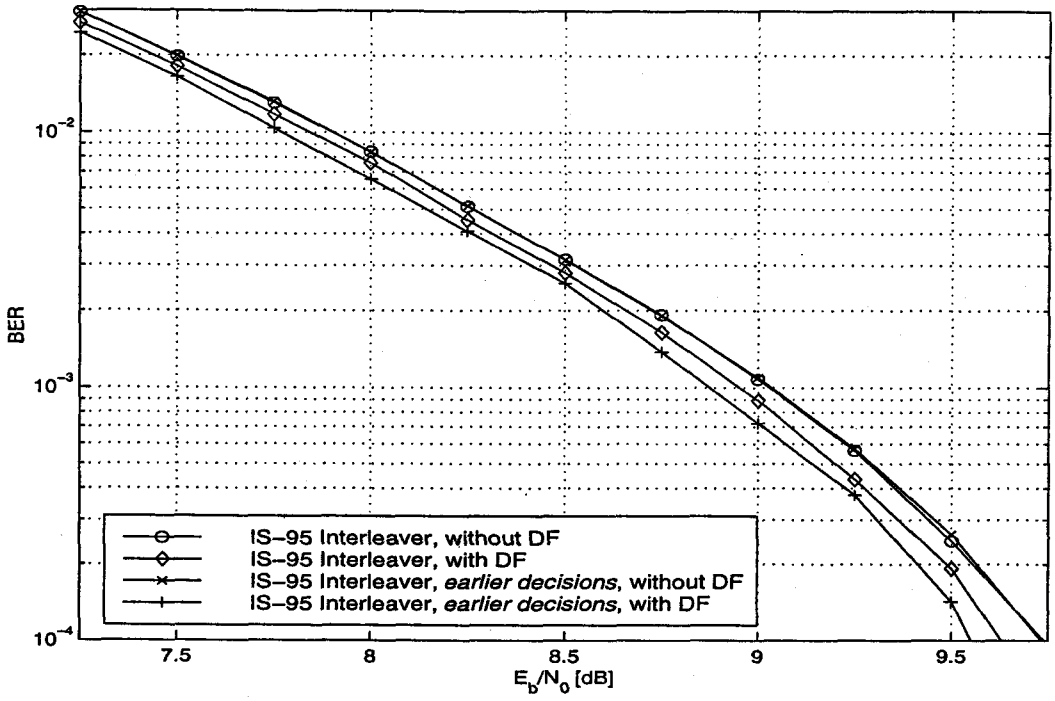


Figure 5.21: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Coherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.



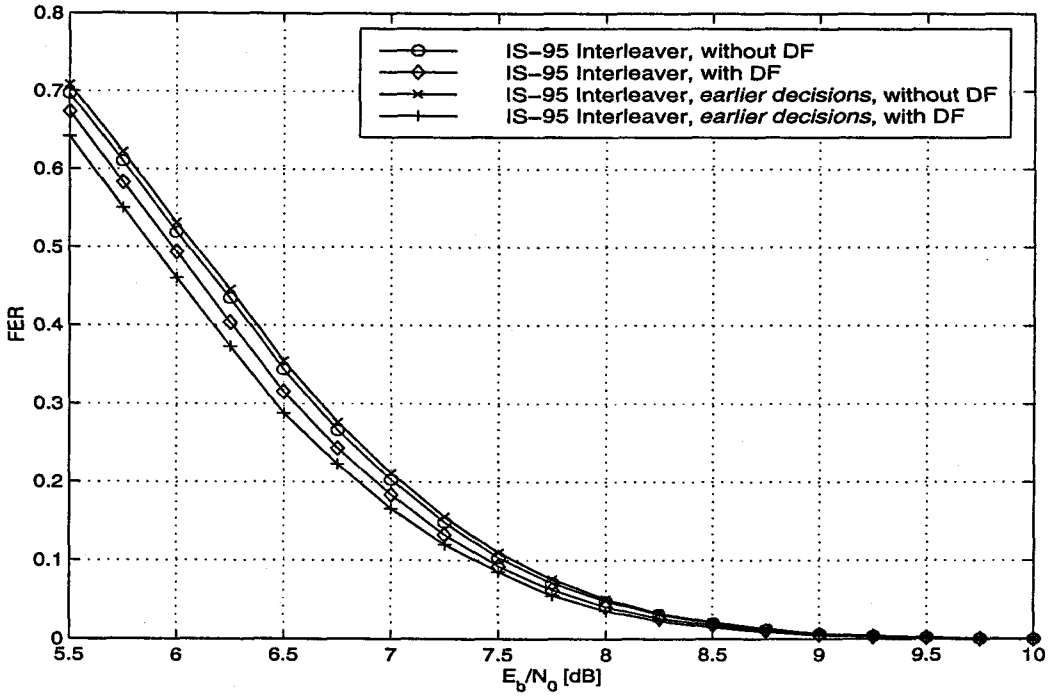


Figure 5.22: BER and frame error rate FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) with/without DF using regular and *earlier decisions*. Noncoherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.

5.3. Conclusions

In this chapter, it was shown that sub-optimal convolutional decoding using a new data bit decision criterion, *earlier decisions*, improves the effectiveness of the original decision feedback (DF) receiver/decoder design. As a result, the bit error rate (BER) and frame error rate (FER) performance of the DF decoder was improved.

The performance of the modified decoder was evaluated for coherent and noncoherent detection of unfaded and Rayleigh fading single-path as well as multipath signals in additive white Gaussian noise (AWGN) interference.

Earlier decisions reduce the average data bit decoding delays of the convolutional decoder for all E_b/N_0 values. At a BER of 10^{-3}, these reductions range from 7-12 decoding steps in all studied propagation scenarios. The BER and FER performance with DF was improved due to a higher number of decoding metrics that benefit from DF updates. Also, performance degradation for a non-DF decoder was observed.

For a BER of 10^{-3}, the required E_b/N_0 values were decreased an additional 0.06-0.11 dB. The performance improvement of the original DF decoder design was thereby increased by a factor of 1.67-2.38, which is considerable. Similar, although smaller, improvements of 0.03-0.08 dB resulted for a FER of 1%.

Consequently, a data bit decision criterion like *earlier decisions* should therefore be used in a practical implementation of the DF decoder in an IS-95 based system.

Chapter 6. New Block Interleaver Design

In the previous chapter, the effectiveness of the original decision feedback (DF) receiver/decoder design (Section 2.2) was improved by a reduction of the average decoding delay of the convolutional decoder. The interleaver specification, which is the second factor that determines DF effectiveness, was not modified, mainly because it is part of the IS-95 standard.

In this chapter, this restriction is dropped and a new block interleaver design is presented. Unlike the IS-95 specified interleaver (Section 1.4.3.2), this new design is carried out specifically with the concept of decision feedback decoding (DFD) in mind.

The design considerations and parameter selection for the new interleaver are followed by a performance evaluation. Simulation results of average decoding delay, bit error rate (BER) and frame error rate (FER) are presented for coherent and noncoherent detection of several propagation scenarios. These results show a considerable performance improvement over the IS-95 interleaver even without the use of DF. In addition, the effectiveness of the DF receiver/decoder is also improved as intended by the new interleaver design.

Given these performance improvements, the effort of a revision of the IS-95 standard to accommodate the new block interleaver design might be well justified.

6.1. Design Considerations

It has been demonstrated by means of computer simulation that DFD results in small but consistent performance improvements for both coherent and noncoherent detection of the IS-95 uplink modulation (Chapter 4).

The interleaver is an integral part of the DF concept and the choice of interleavers in the IS-95 standard was done without this consideration. The question, whether this choice is still the best possible when the DF concept is applied, could be asked for this reason alone.

With some additional insight, it can be concluded that DFD as applied to the IS-95 uplink cannot result in optimal performance for two reasons: the decoding delay of the convolutional decoder and the order in which Walsh groups are being used during frame decoding (Section 2.4).

Because the frame is decoded only once, the additional information provided by DF cannot be used most efficiently if the decoding delay is too large. Ideally, DF should occur each time before a Walsh group is reused. With the IS-95 interleaver, as described earlier (Section 1.4.3.2), this cannot be achieved since it would require data bit decisions of the convolutional decoder with a delay of less than 10 or 11 decoding steps. The simulation results indicate that at the BER of interest for voice communications (10^{-3}) the average decoding delay is on the order of 21-35 decoding steps. The use of *earlier decisions* reduces these numbers to 13-22 decoding steps, which is still not enough for DF to be most effective (Chapter 5). Overall, use of the IS-95 interleaver has the average potential of 3 DF metric updates per Walsh group for regular data bit decisions and 4 DF updates per Walsh group for *earlier decisions*.

An appropriate change of the interleaver specification, on the other hand, can result in an even better improvement of the DF effectiveness. The main design consideration is to achieve maximum separation of the convolutionally coded bits that are transmitted within the same Walsh group, so that on average DF occurs each time before a Walsh group is reused. This way, DFD should be most effective. A second important consideration must be that most of the functionality of the interleaving operation is maintained, i.e., randomization of burst errors.

6.2. Parameter Selection

In the IS-95 uplink traffic channel (Figure 1.5), the fixed duration of a frame (20 ms) imposes a restriction on the design of the interleaver. Although it would be possible to use interleavers that span several frames, this would further complicate implementation issues and is not considered in this dissertation.

There are 576 convolutionally coded bits in a frame and 6 convolutionally coded bits in each Walsh group. Thus, the maximum possible separation of the convolutionally coded bits that can be achieved on a frame basis is 96. This can be easily achieved by replacing the (32,18) block interleaver with a (96,6) block interleaver. With this selection, an average decoding delay of 32 decoding steps still results in a DF metric update before a Walsh group is reused during the frame decoding process (on average).

However, an examination of the resulting Walsh groups reveals that consecutive convolutionally coded bits are now transmitted in consecutive Walsh groups. This might

result in severe performance degradation under correlated fading conditions. To avoid this, the matrix rows are scrambled by block interleaving the row numbers. One possibility is a (4,24) block interleaver[9], which separates consecutive convolutionally coded bits by 24 or 71 Walsh groups.

1	97	193	289	385	481	34	130	226	322	418	514	67	163	259	355	451	547
5	101	197	293	389	485	38	134	230	326	422	518	71	167	263	359	455	551
9	105	201	297	393	489	42	138	234	330	426	522	75	171	267	363	459	555
13	109	205	301	397	493	46	142	238	334	430	526	79	175	271	367	463	559
17	113	209	305	401	497	50	146	242	338	434	530	83	179	275	371	467	563
21	117	213	309	405	501	54	150	246	342	438	534	87	183	279	375	471	567
25	121	217	313	409	505	58	154	250	346	442	538	91	187	283	379	475	571
29	125	221	317	413	509	62	158	254	350	446	542	95	191	287	383	479	575
33	129	225	321	417	513	66	162	258	354	450	546	4	100	196	292	388	484
37	133	229	325	421	517	70	166	262	358	454	550	8	104	200	296	392	488
41	137	233	329	425	521	74	170	266	362	458	554	12	108	204	300	396	492
45	141	237	333	429	525	78	174	270	366	462	558	16	112	208	304	400	496
49	145	241	337	433	529	82	178	274	370	466	562	20	116	212	308	404	500
53	149	245	341	437	533	86	182	278	374	470	566	24	120	216	312	408	504
57	153	249	345	441	537	90	186	282	378	474	570	28	124	220	316	412	508
61	157	253	349	445	541	94	190	286	382	478	574	32	128	224	320	416	512
65	161	257	353	449	545	3	99	195	291	387	483	36	132	228	324	420	516
69	165	261	357	453	549	7	103	199	295	391	487	40	136	232	328	424	520
73	169	265	361	457	553	11	107	203	299	395	491	44	140	236	332	428	524
77	173	269	365	461	557	15	111	207	303	399	495	48	144	240	336	432	528
81	177	273	369	465	561	19	115	211	307	403	499	52	148	244	340	436	532
85	181	277	373	469	565	23	119	215	311	407	503	56	152	248	344	440	536
89	185	281	377	473	569	27	123	219	315	411	507	60	156	252	348	444	540
93	189	285	381	477	573	31	127	223	319	415	511	64	160	256	352	448	544
2	98	194	290	386	482	35	131	227	323	419	515	68	164	260	356	452	548
6	102	198	294	390	486	39	135	231	327	423	519	72	168	264	360	456	552
10	106	202	298	394	490	43	139	235	331	427	523	76	172	268	364	460	556
14	110	206	302	398	494	47	143	239	335	431	527	80	176	272	368	464	560
18	114	210	306	402	498	51	47	243	339	435	531	84	180	276	372	468	564
22	118	214	310	406	502	55	151	247	343	439	535	88	184	280	376	472	568
26	122	218	314	410	506	59	155	251	347	443	539	92	188	284	380	476	572
30	126	222	318	414	510	63	159	255	351	447	543	96	192	288	384	480	576

Figure 6.1: New block interleaver matrix (arranged in 3 columns).

[9] This is only one of many possibilities. No claim is made that this is the best possible solution.

Unlike the IS-95 interleaver, the new interleaver does not result in a separation of the frame decoding process into three distinct stages. Instead, all of the Walsh groups are used once for metric computations before any are reused. The additional information from the DF metric updates becomes available gradually over the frame decoding process. This is clearly seen in the results that are presented in the next section. The results also indicate average decoding delays of 31-37 decoding steps at BER 10^{-3} for most simulation scenarios. At the same time the results show that, within a frame, DF results in a fast reduction of the average decoding delay below the threshold value of 32 decoding steps needed for most effective DF.

As a result, the decoding metrics now have an average potential of at least 4 DF updates (5 in some cases), which is a significant improvement from using the IS-95 interleaver.

6.3. Performance Evaluation

The results presented in this section show a considerable improvement of BER and FER performance when using the proposed new interleaver design. This is true even without the use of DF. Additionally, the effectiveness of DF is increased by the new interleaver design.

An explanation for a performance improvement due to the use of a modified interleaver design is the use of orthogonal Walsh modulation. 6 interleaved convolutionally coded bits are transmitted within one modulation symbol. As a result, the metrics for these 6 interleaved

convolutionally coded bits used for convolutional decoding are based on the same correlation values and therefore correlated. The simplest example for this is the use of the maximum correlation value in each Walsh group as the metric value and the corresponding Walsh code as the most likely transmitted one. In this case, the same metric values are used throughout the decoding process, 6 times each, at decoding steps determined by the interleaver.

It is well known that uncorrelated decoder inputs are required for optimal performance of a convolutional decoder [44]. The interleaver makes correlated communications channel outputs appear uncorrelated. This is achieved by separating consecutive coded symbols by an adequately designed number of symbols during transmission, so that the channel affects them differently. Since the orthogonal modulation generates correlated inputs to the convolutional decoder even in a memoryless channel, there are now two effects that cause correlated decoding metrics. The one that has the most detrimental effect on the performance of the convolutional decoder should drive the interleaver design.

As in the previous chapters, five different propagation scenarios are evaluated in the presence of additive white Gaussian noise (AWGN) interference: unfaded single-path signal, *independent Rayleigh fading* single-path signal, 4 equal-strength unfaded multipath signals, 4 equal-strength *independent Rayleigh fading* multipath signals, and 4 equal-strength *correlated Rayleigh fading* multipath signals.

6.3.1. Unfaded Single-Path

The average data bit output delays for coherent and noncoherent detection with and without DF for this case are shown in Figure 6.2. Compared with the same plots for the IS-95 interleaver (Figure 5.1, Figure 5.2), there are several differences.

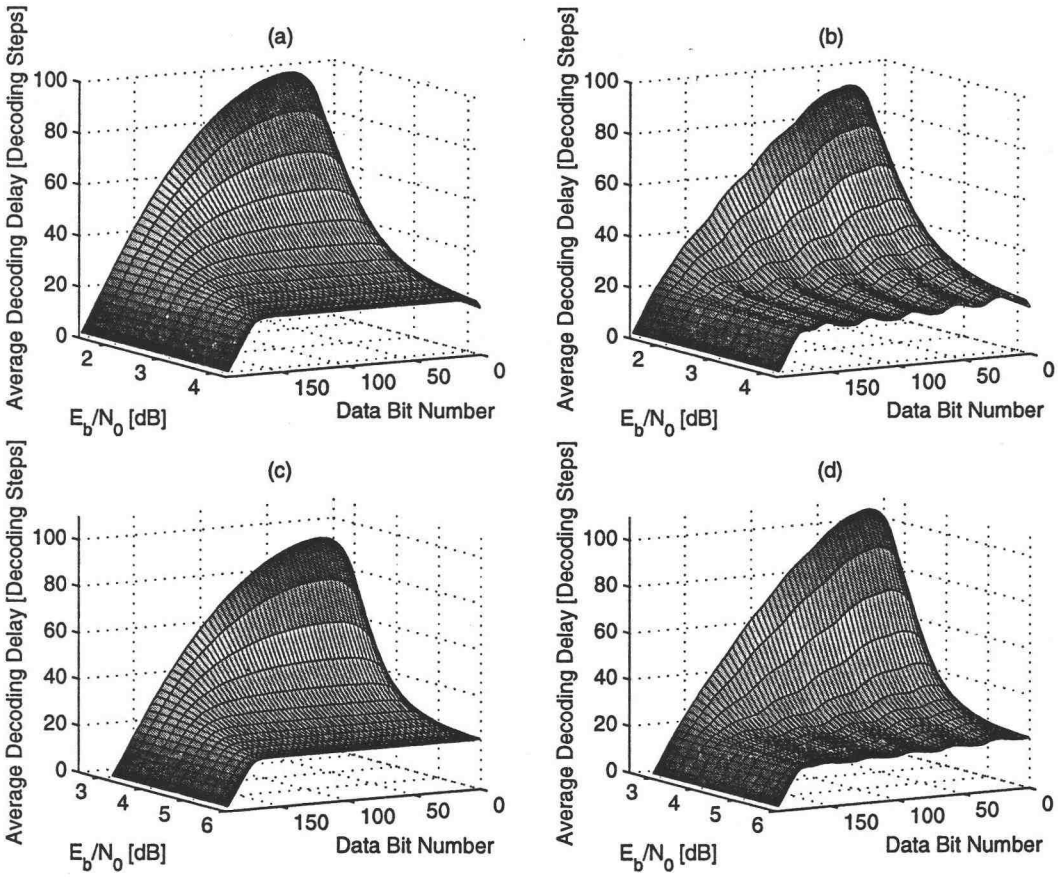


Figure 6.2: Average data bit output delays as a function of E_b/N_0. Detection of unfaded single-path signal in AWGN using the new interleaver. (a) coherent, without DF, (b) coherent, with DF, (c) noncoherent, without DF, (d) noncoherent, with DF.

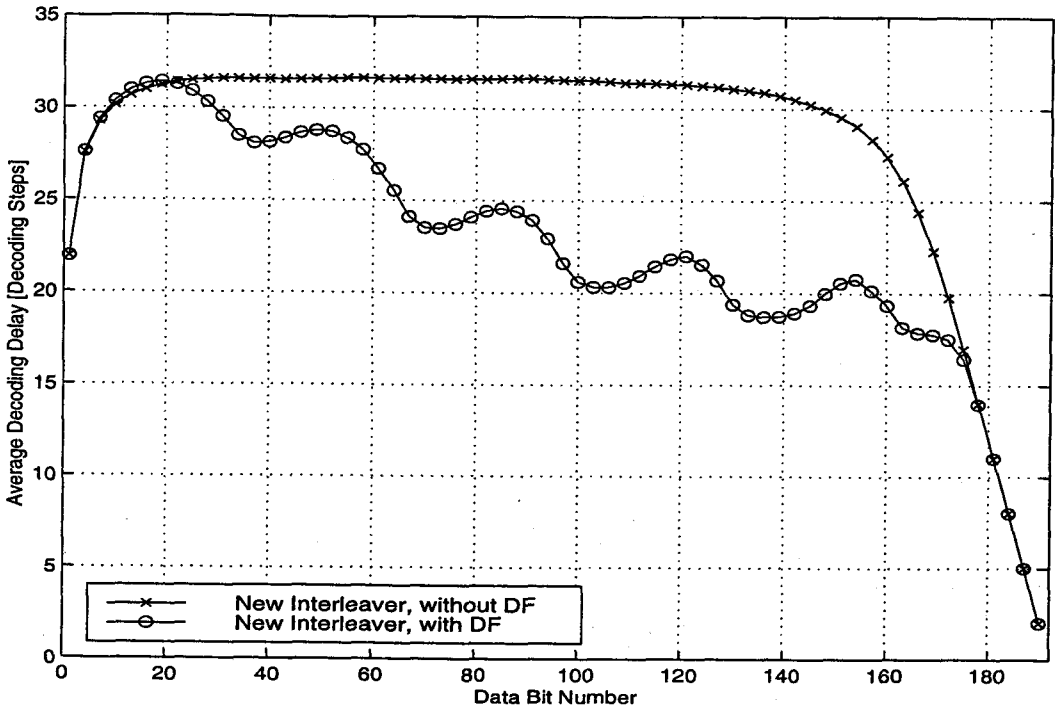


Figure 6.3: Average data bit decoding delays for the new interleaver with/without DF. Coherent detection of unfaded single-path signal in AWGN. E_b/N_0=3 dB.

First, the new interleaver design results in a small increase of the average decoding delay, which is noticeable at low E_b/N_0 values. Also at low E_b/N_0 values, DF reduces the average decoding delay for coherent detection. For noncoherent detection it is increased.

Second, with the IS-95 interleaver, the frame decoding process is divided into three stages. Each stage begins with no DF information available and DF information is added over the duration of such a stage. Here, there is only one stage and DF information is added over the duration of the entire frame. This is seen in the figures. Without DF, there is no more reduction of the average decoding delay that marks the three decoding stages in Figure 5.1 and Figure 5.2.

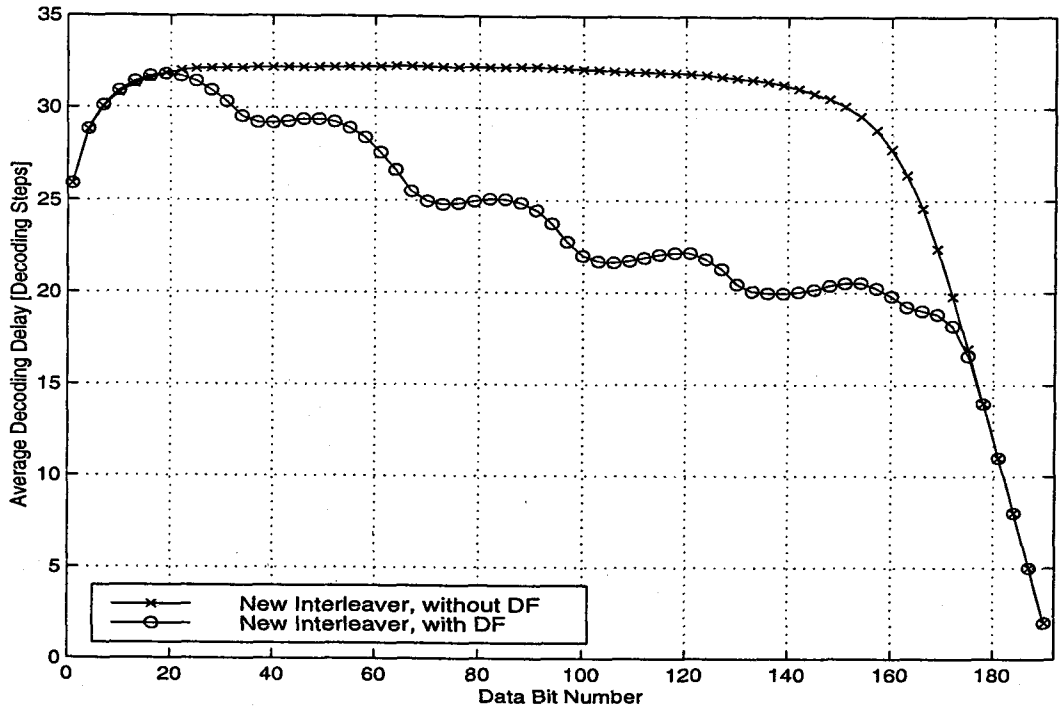


Figure 6.4: Average data bit decoding delays for the new interleaver with/without DF. Noncoherent detection of unfaded single-path signal in AWGN. E_b/N_0=4.5 dB.

With DF, the reuse of the Walsh groups (6 times) is clearly noticeable and the average decoding delay is gradually reduced over the frame duration.

Looking at fixed E_b/N_0 values illustrates this even better. Figure 6.3 shows the average decoding delay for coherent detection with an E_b/N_0 of 3 dB. Figure 6.4 shows noncoherent detection with an E_b/N_0 of 4.5 dB. For both plots the corresponding BER is approximately 10^{-3}. The reader should compare these plots to Figure 5.3 and Figure 5.4.

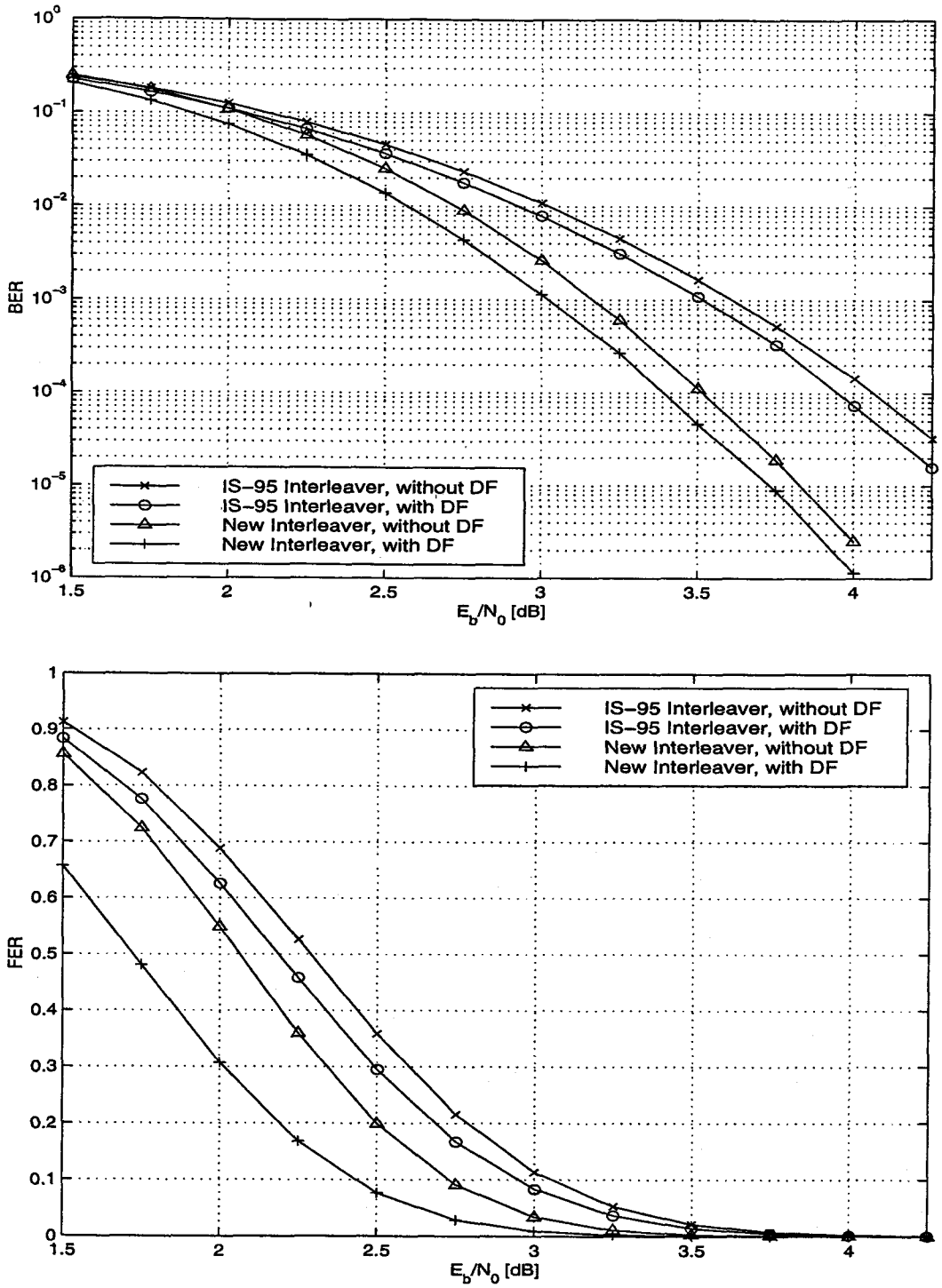


Figure 6.5: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Coherent detection of unfaded single-path signal in AWGN.

In both cases the average decoding delay without DF is about 32 decoding steps, which is 5 decoding steps higher than when the IS-95 interleaver is used. Use of DF gradually reduces the decoding delay to about 20 decoding steps.

On average, a DF metric update occurs each time before the convolutional decoder reuses a Walsh group. DFD should therefore be more effective when the new interleaver design is used.

This is confirmed by the BER and FER results shown in Figure 6.5 and Figure 6.6, respectively, for coherent and noncoherent detection.

The new interleaver design results in a higher coding gain of the concatenation of the convolutional code and the orthogonal Walsh modulation even without DF. At the same time, the performance improvement through the use of DF (the effectiveness of DF) is greater than it is for the IS-95 interleaver.

In the coherent case, the new interleaver with DF achieves a performance improvement over the IS-95 specified interleaver without DF of 0.59 dB to achieve a BER of 10^{-3}. The effectiveness of DF for the new interleaver is 0.15 dB, for the IS-95 interleaver it is 0.09 dB. The required E_b/N_0 for this BER is 3.02 dB.

Comparing the 1% FER, the new interleaver with DF achieves a 0.72 dB performance improvement over the IS-95 interleaver without DF. In this case the effectiveness of DF for the new interleaver is 0.27 dB and only 0.11 dB for the IS-95 interleaver. The required E_b/N_0 to achieve this FER is 2.99 dB. Note that the E_b/N_0 reduction through DF at the 1% FER is almost twice the E_b/N_0 reduction at BER 10^{-3}.

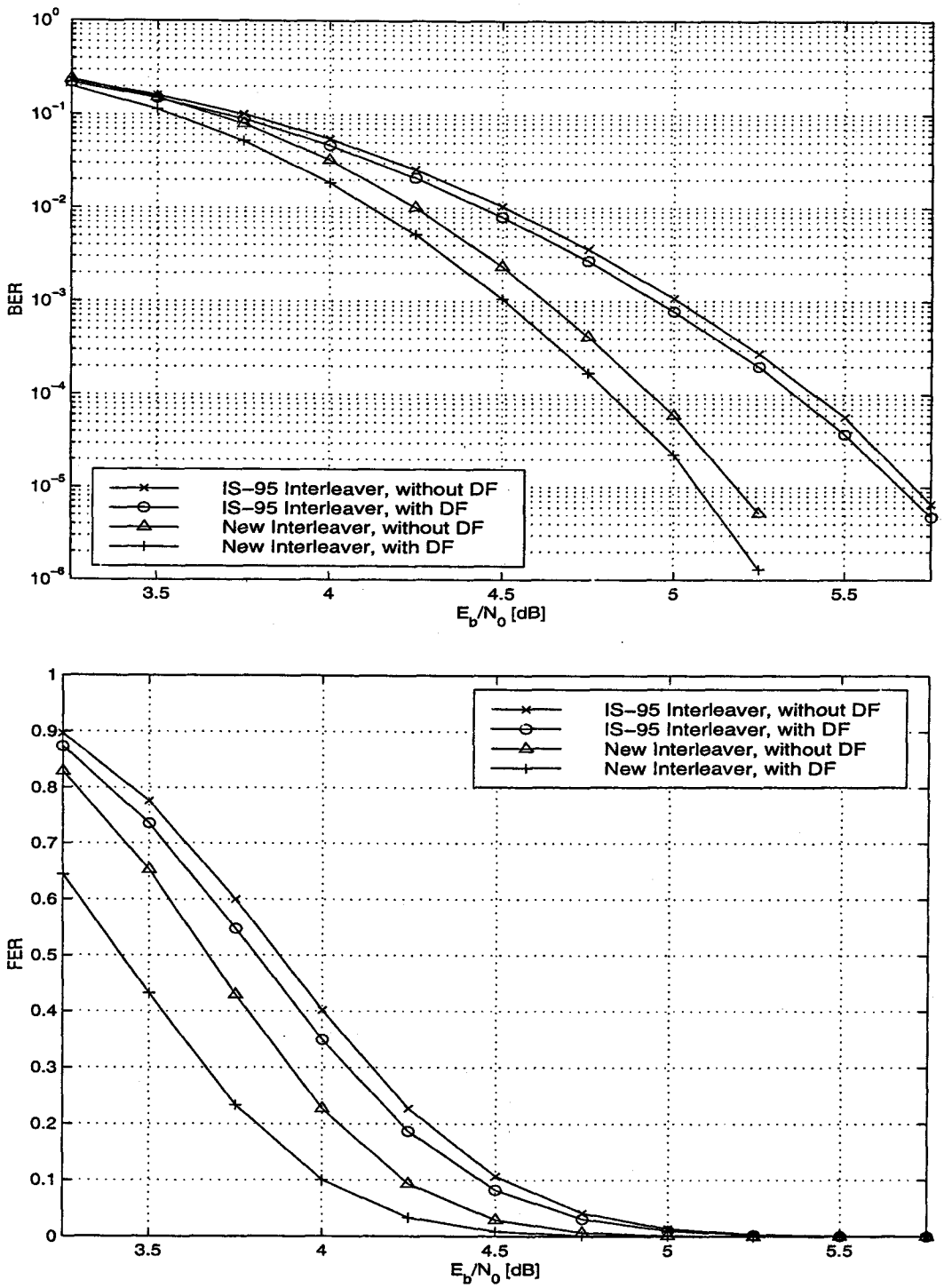


Figure 6.6: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Noncoherent detection of unfaded single-path signal in AWGN.

E_b/N_0 [dB] required to achieve		BER 10^{-2}	BER 10^{-3}	BER 10^{-4}	FER 5%	FER 2%	FER 1%
Coherent	IS-95 Interleaver, without DF	3.02	3.61	4.06	3.27	3.52	3.70
	IS-95 Interleaver, with DF	2.93	3.52	3.95	3.18	3.43	3.60
	New Interleaver, without DF	2.72	3.17	3.52	2.93	3.15	3.27
	New Interleaver, with DF	2.57	3.02	3.39	2.64	2.86	2.99
Noncoherent	IS-95 Interleaver, without DF	4.51	5.01	5.41	4.72	4.94	5.09
	IS-95 Interleaver, with DF	4.44	4.95	5.35	4.67	4.88	5.00
	New Interleaver, without DF	4.25	4.62	4.93	4.42	4.60	4.72
	New Interleaver, with DF	4.12	4.51	4.82	4.19	4.39	4.49

Table 6.1: Performance summary for detection of unfaded single-path signal in AWGN.

For noncoherent detection, the performance improvement of the new interleaver with DF over the IS-95 interleaver without DF (BER 10^{-3}) is 0.54 dB. The effectiveness of DF for the new interleaver is 0.14 dB, for the IS-95 interleaver it is 0.05 dB. The E_b/N_0 required to achieve this BER is 4.51 dB.

At 1% FER, the improvement due to the new interleaver and DFD is 0.63 dB with a DF effectiveness of 0.25 dB for the new interleaver and 0.07 dB for the IS-95 interleaver. Again, the DF effectiveness at the 1% FER exceeds the DF effectiveness at a BER of 10^{-3}.

Table 6.1 summarizes the required E_b/N_0 values to achieve a specified performance in terms of BER and FER in this case.

6.3.2. Rayleigh Fading Single-Path

The results in this section are for an *independent Rayleigh fading* single-path signal in AWGN interference.

The average data bit decoding delays for this case are shown in Figure 6.7 for coherent detection with an E_b/N_0 of 4.25 dB and in Figure 6.8 for noncoherent detection with an E_b/N_0 of 6.25 dB. As before, this corresponds to a BER of approximately 10^{-3}.

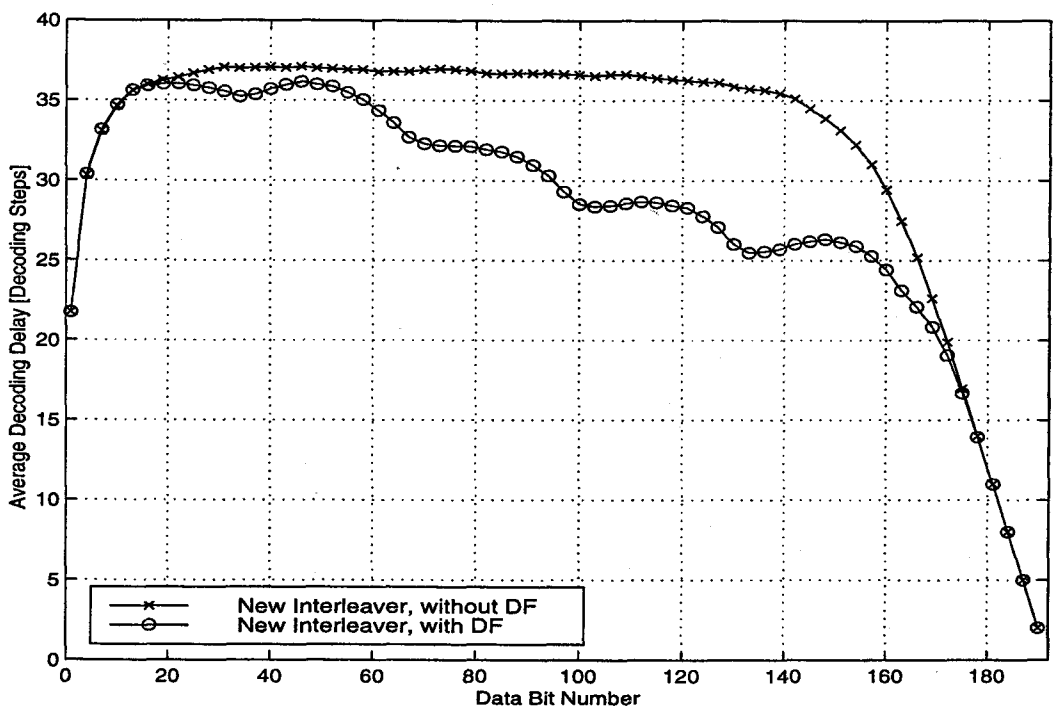


Figure 6.7: Average data bit decoding delays for the new interleaver with/without DF. Coherent detection of *independent Rayleigh fading* single-path signal in AWGN. E_b/N_0=4.25 dB.

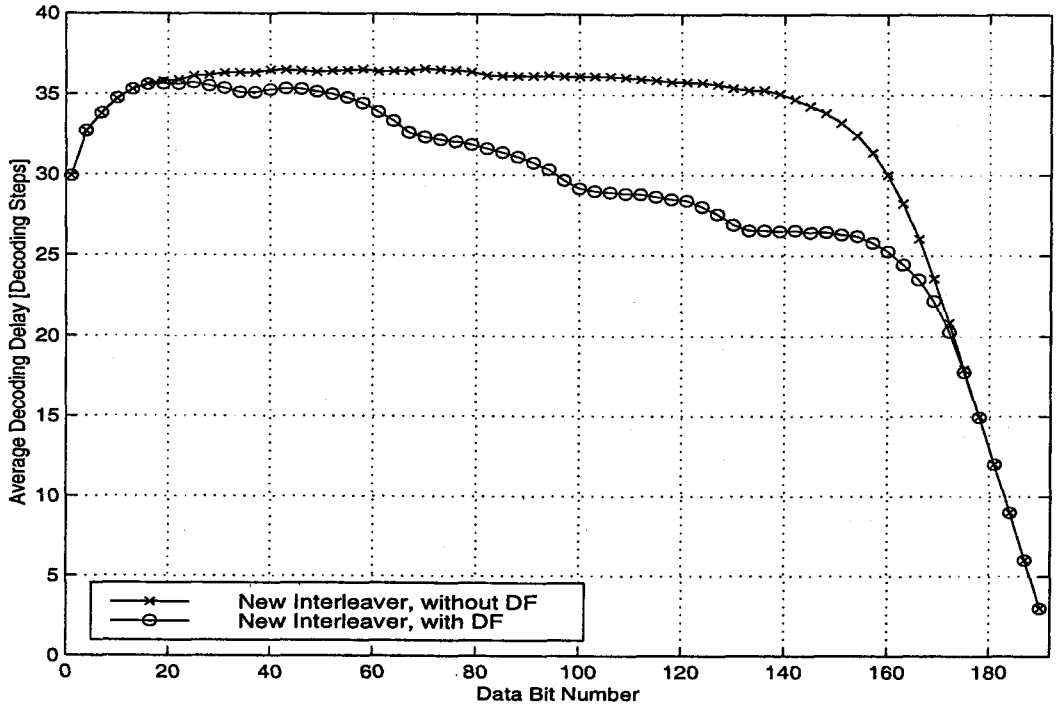


Figure 6.8: Average data bit decoding delays for the new interleaver with/without DF. Noncoherent detection of *independent Rayleigh fading* single-path signal in AWGN. E_b/N_0=6.25 dB.

The average decoding delay without DF is approximately 36 decoding steps. With DF, the delay is gradually reduced to about 25 decoding steps. However, it takes longer for this reduction to begin than in the unfaded case (Figure 6.3, Figure 6.4). This is because the decoding delay is greater than 32 decoding steps at the beginning of the frame and, on average, no DF metric update is available when the Walsh groups are reused for the first time.

The corresponding BER and FER plots for coherent and noncoherent detection are shown in Figure 6.9 and Figure 6.10, respectively. Substantial performance improvement is shown in these plots.

In the coherent case, the new interleaver with DF results in a 1.14 dB reduction of the required E_b/N_0 at a BER of 10^{-3}. The reduction of the required E_b/N_0 due to DF is 0.18 dB for the new interleaver and 0.08 dB for the IS-95 interleaver. An E_b/N_0 of 4.30 dB is required for this BER.

Even greater improvement is obtained at a FER of 1%. The new interleaver with DF results in a 1.35 dB reduction of the required E_b/N_0. The effectiveness of DF is 0.32 dB for the new interleaver and 0.08 dB for the IS-95 interleaver. An E_b/N_0 of 4.18 dB is required for this FER.

Similar results are obtained in the noncoherent case. Compared to the IS-95 interleaver without DF, the new interleaver with DF reduces the required E_b/N_0 for a BER of 10^{-3} by 1.16 dB from 7.43 dB to 6.27 dB. The effectiveness of DF at this BER for the new interleaver and the IS-95 interleaver is, respectively, 0.20 dB and 0.06 dB.

To achieve a FER of 1%, the new interleaver with DF requires an E_b/N_0 of 6.21 dB. This is a reduction of 1.29 dB compared to the IS-95 interleaver without DF. The DF effectiveness of the new interleaver is 0.29 dB. For the IS-95 interleaver it is 0.07 dB.

Table 6.2 summarizes the required E_b/N_0 values to achieve a specified performance in terms of BER and FER in this case.

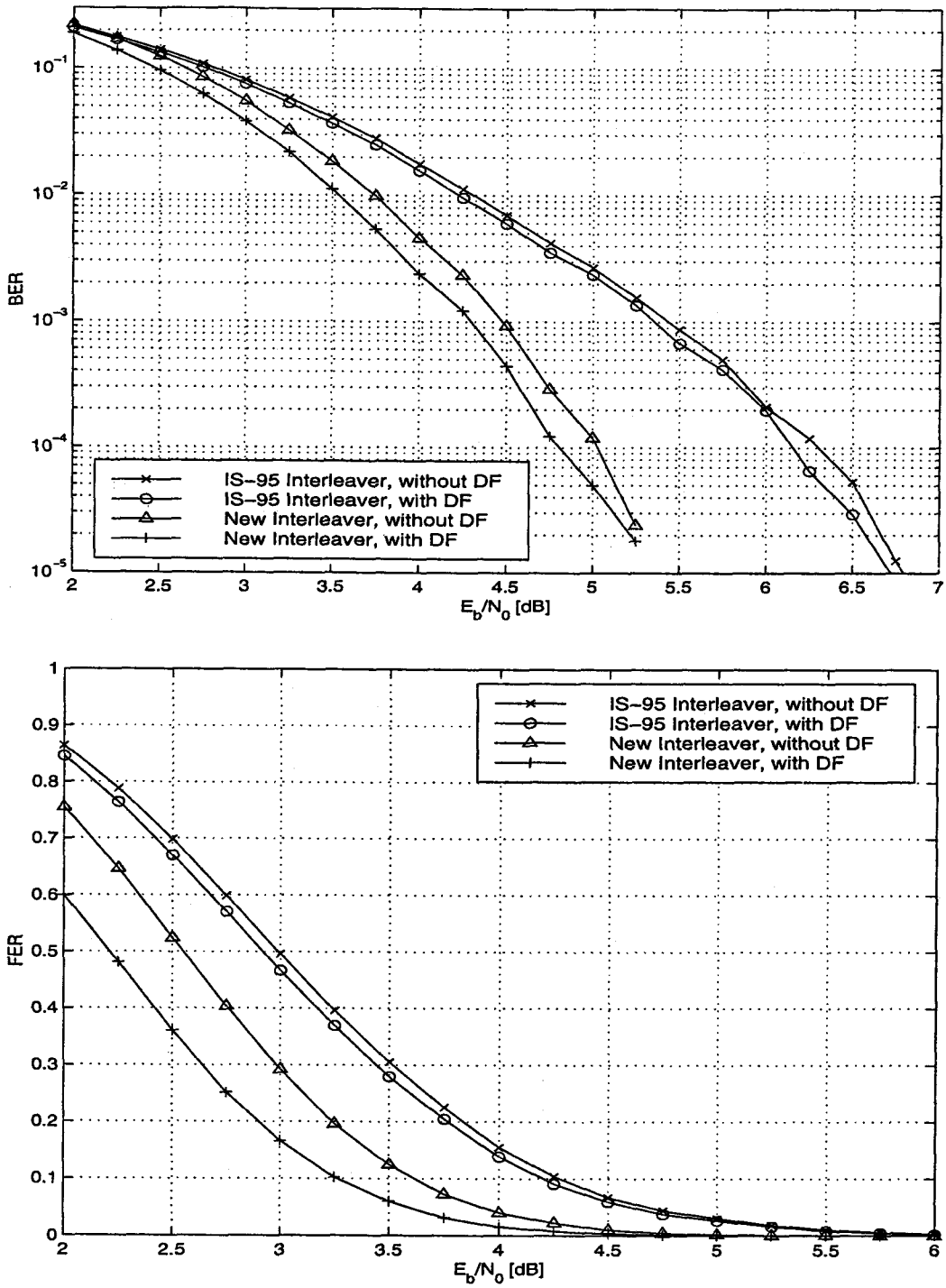


Figure 6.9: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Coherent detection of *independent Rayleigh fading* single-path signal in AWGN.

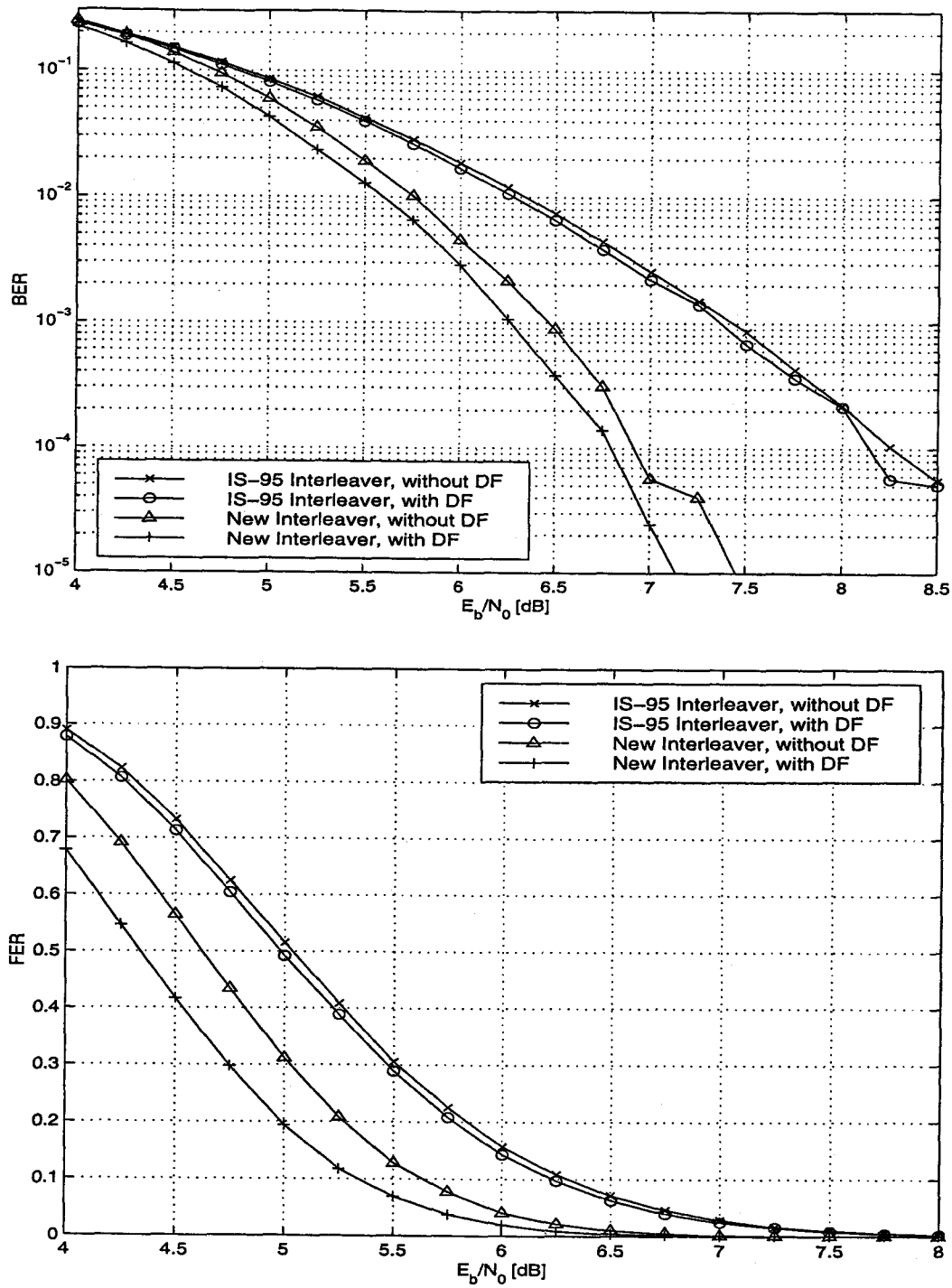


Figure 6.10: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Noncoherent detection of *independent Rayleigh fading* single-path signal in AWGN.

E_b/N_0 [dB] required to achieve		BER 10^{-2}	BER 10^{-3}	BER 10^{-4}	FER 5%	FER 2%	FER 1%
Coherent	IS-95 Interleaver, without DF	4.31	5.44	6.30	4.68	5.22	5.53
	IS-95 Interleaver, with DF	4.23	5.36	6.15	4.61	5.16	5.45
	New Interleaver, without DF	3.74	4.48	5.03	3.92	4.29	4.50
	New Interleaver, with DF	3.54	4.30	4.81	3.59	3.92	4.18
Noncoherent	IS-95 Interleaver, without DF	6.34	7.43	8.26	6.71	7.17	7.50
	IS-95 Interleaver, with DF	6.28	7.37	8.14	6.63	7.11	7.43
	New Interleaver, without DF	5.76	6.47	6.92	5.94	6.28	6.50
	New Interleaver, with DF	5.59	6.27	6.80	5.65	5.99	6.21

Table 6.2: Performance summary for *independent Rayleigh fading* single-path signal in AWGN.

6.3.3. Unfaded Multipath

The results in this section are for 4 equal-strength unfaded multipath signals AWGN interference.

The average data bit decoding delays are shown in Figure 6.11 and Figure 6.12. The E_b/N_0 is 3 dB for the coherent results and 6.25 dB for the noncoherent results. Both figures are very similar to the corresponding figures in the unfaded single-path case (Figure 6.3, Figure 6.4). The average decoding delay is about 31 decoding steps for coherent detection and 33 decoding steps for noncoherent detection. DF reduces the decoding delay gradually over the frame down to 16 decoding steps.

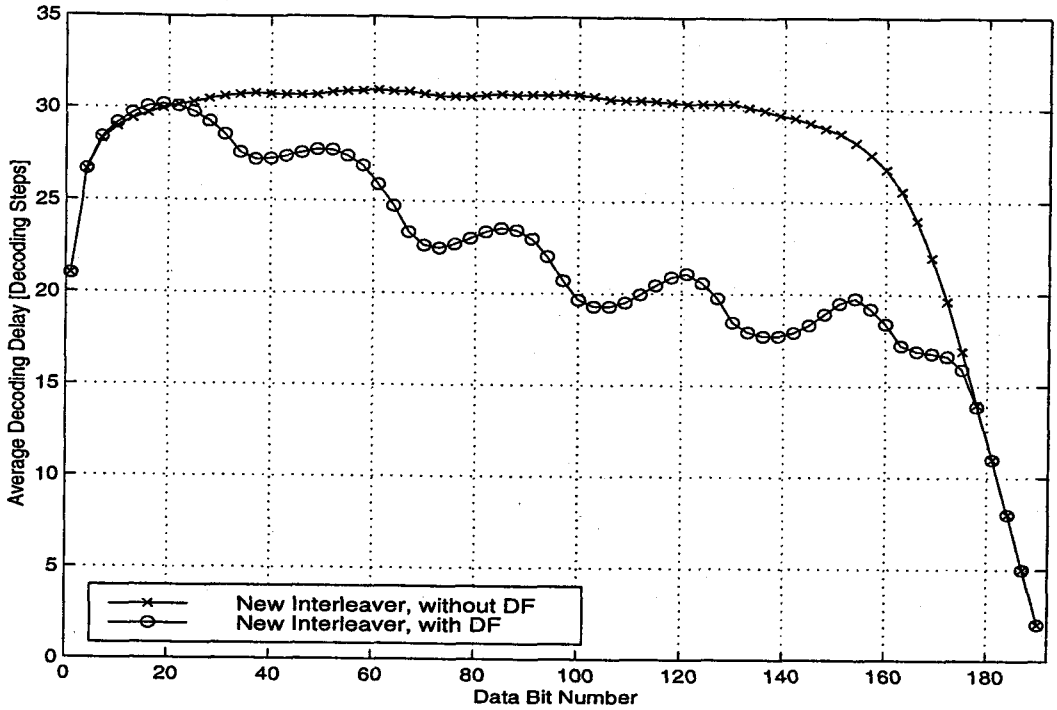


Figure 6.11: Average data bit decoding delays for the new interleaver with/without DF. Coherent detection of 4 equal-strength unfaded multipath signals in AWGN. E_b/N_0=3.00 dB.

The coherent and noncoherent BER and FER results are shown in Figure 6.13 and Figure 6.14, respectively.

For coherent detection, the performance is, as expected, almost identical to unfaded single-path (Figure 6.5, Table 6.1). The small differences can be attributed to the higher number of frames simulated in the single-path case (100,000 versus 10,000) and the different realization of the interference between the two cases.

The noncoherent results show a noncoherent combining loss of about 1.75 dB for the IS-95 interleaver and 1.80 dB for the new interleaver (BER 10^{-3}).

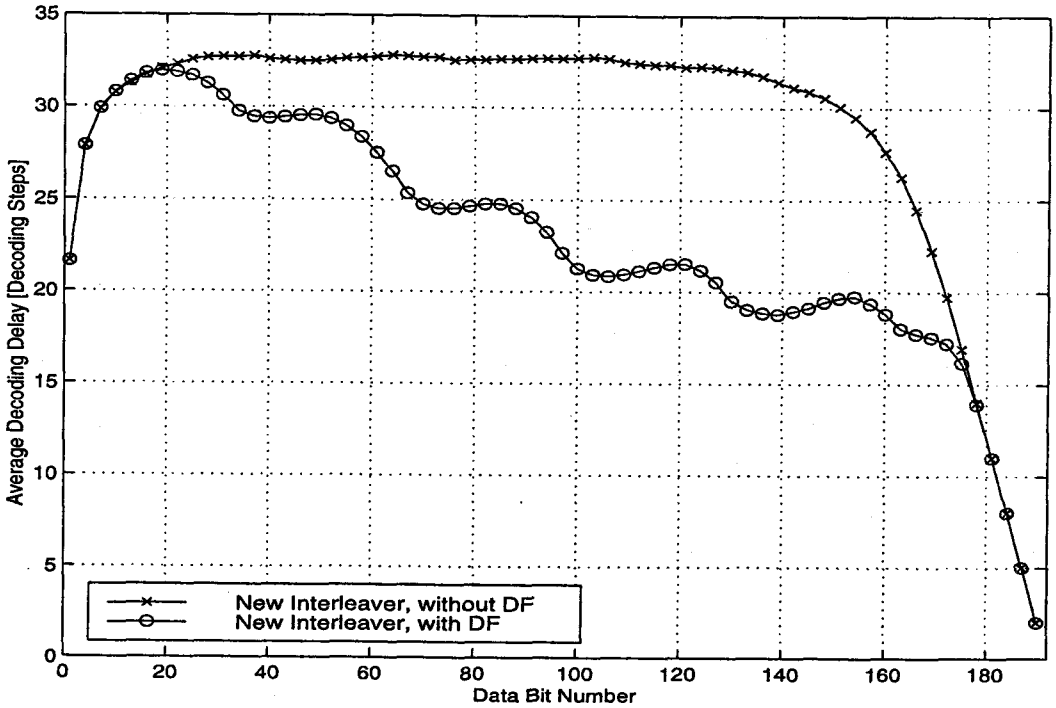


Figure 6.12: Average data bit decoding delays for the new interleaver with/without DF. Noncoherent detection of 4 equal-strength unfaded multipath signals in AWGN. E_b/N_0=6.25 dB.

Aside from that, the performance improvement between the new interleaver with DF and the IS-95 interleaver without DF at BER 10^{-3} and 1% FER remains, respectively, 0.47 dB and 0.57 dB.

The DF effectiveness using the new interleaver is 0.13 dB (BER 10^{-3}) and 0.22 dB (1% FER); for the IS-95 interleaver it is 0.07 dB (BER 10^{-3}) and 0.11 dB (1% FER). Taking into account the uncertainty in the noncoherent combining losses, the performance improvements of the new interleaver and DFD are almost unchanged from the unfaded single-path case. The E_b/N_0 required to achieve a BER of 10^{-3} and a FER of 1% is, respectively, 6.30 dB and 6.29 dB.

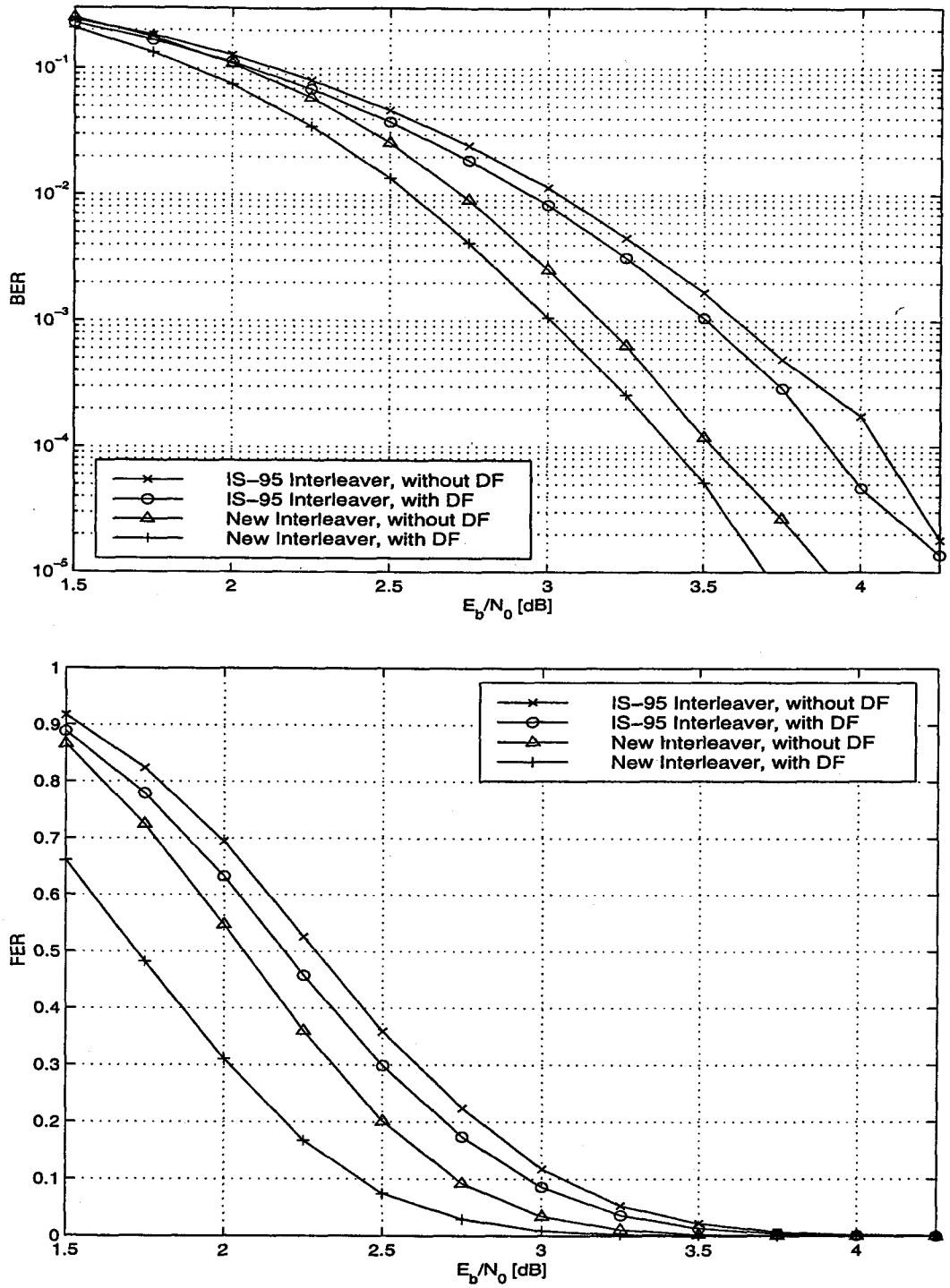


Figure 6.13: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Coherent detection of 4 equal-strength unfaded multipath signals in AWGN.

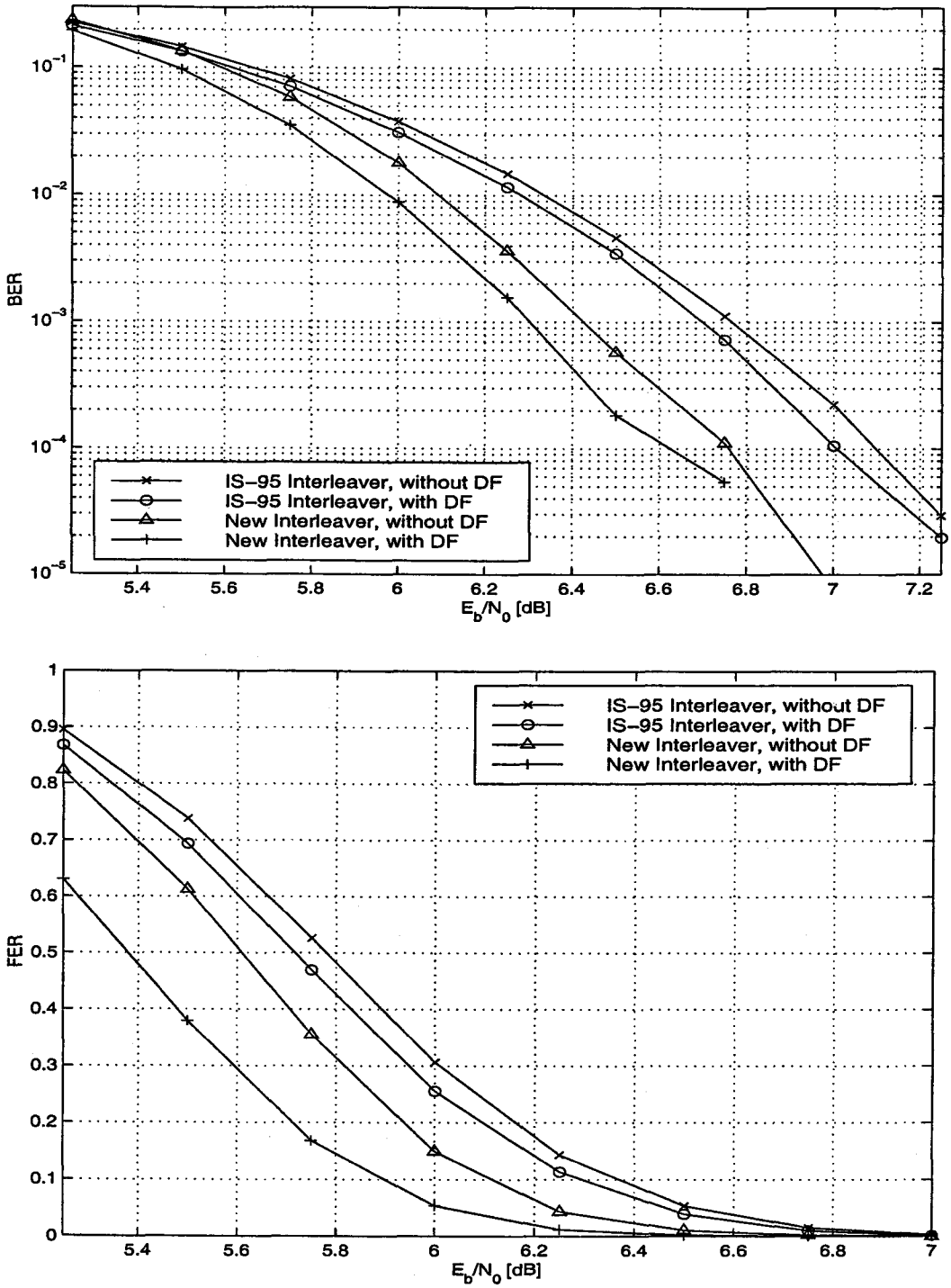


Figure 6.14: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Noncoherent detection of 4 equal-strength unfaded multipath signals in AWGN.

E_b/N_0 [dB] required to achieve		BER 10^{-2}	BER 10^{-3}	BER 10^{-4}	FER 5%	FER 2%	FER 1%
Coherent	IS-95 Interleaver, without DF	3.03	3.61	4.06	3.27	3.53	3.70
	IS-95 Interleaver, with DF	2.94	3.51	3.90	3.18	3.43	3.59
	New Interleaver, without DF	2.73	3.17	3.53	2.93	3.15	3.28
	New Interleaver, with DF	2.56	3.01	3.40	2.63	2.86	2.98
Noncoherent	IS-95 Interleaver, without DF	6.33	6.77	7.10	6.52	6.72	6.86
	IS-95 Interleaver, with DF	6.28	6.70	7.01	6.46	6.66	6.75
	New Interleaver, without DF	6.09	6.43	6.76	6.23	6.43	6.51
	New Interleaver, with DF	5.98	6.30	6.62	6.02	6.20	6.29

Table 6.3: Performance summary for detection of 4 equal-strength unfaded multipath signals in AWGN.

Table 6.3 shows the E_b/N_0 that is required in this case to achieve a specified performance in terms of BER and FER.

6.3.4. Rayleigh Fading Multipath

The results in this section show performance for 4 equal-strength Rayleigh fading multipath signals in AWGN interference. Two cases are considered: *independent Rayleigh fading* and *correlated Rayleigh fading* (Section 3.4).

6.3.4.1. Independent Rayleigh Fading

The average data bit decoding delays for this case are shown in Figure 6.15 and Figure 6.16, respectively, for coherent detection with an E_b/N_0 of 3.25 dB and noncoherent detection with an E_b/N_0 of 6.75 dB (BER approximately 10^{-3}).

In both cases the average decoding delay is about 33 decoding steps without the use of DF. DF gradually reduces the delay down to 16 decoding steps over the frame.

The coherent and noncoherent BER and FER results are shown in Figure 6.17 and Figure 6.18, respectively.

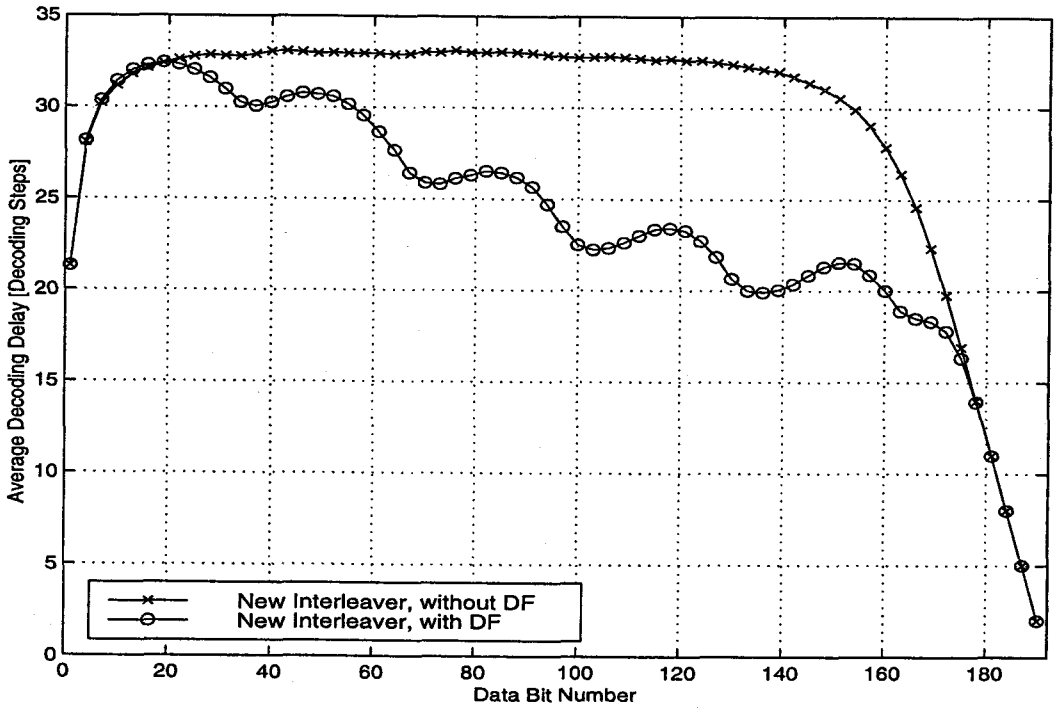


Figure 6.15: Average data bit decoding delays for the new interleaver with/without DF. Coherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN. E_b/N_0=3.25 dB.

The fading of the multipath signals leads to performance degradation in both cases compared to the unfaded multipath results. The new interleaver outperforms the IS-95 interleaver for all E_b/N_0 values shown. Also, the additional performance improvement of DFD is greater for the new interleaver design.

The coherent results show a required E_b/N_0 of 3.31 dB for the new interleaver with DF to achieve a BER of 10^{-3}. The improvement over the IS-95 interleaver without DF is 0.69 dB in this case. The DF improvement is 0.18 dB for the new interleaver and 0.08 dB for the IS-95 interleaver.

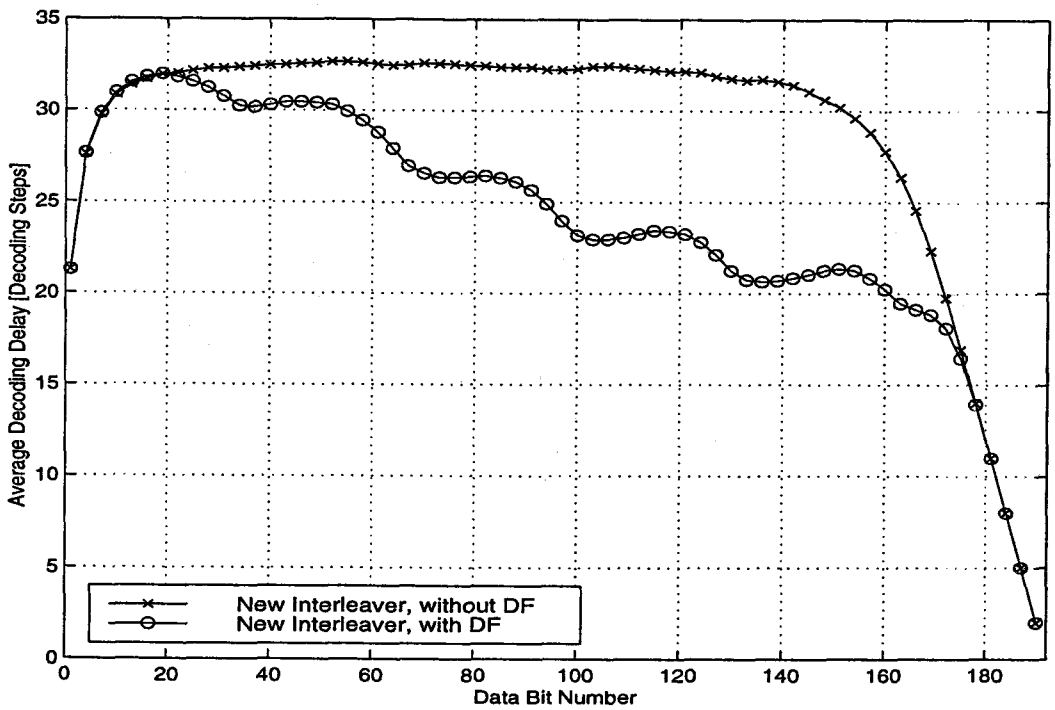


Figure 6.16: Average data bit decoding delays for the new interleaver with/without DF. Noncoherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN. E_b/N_0=6.75 dB.

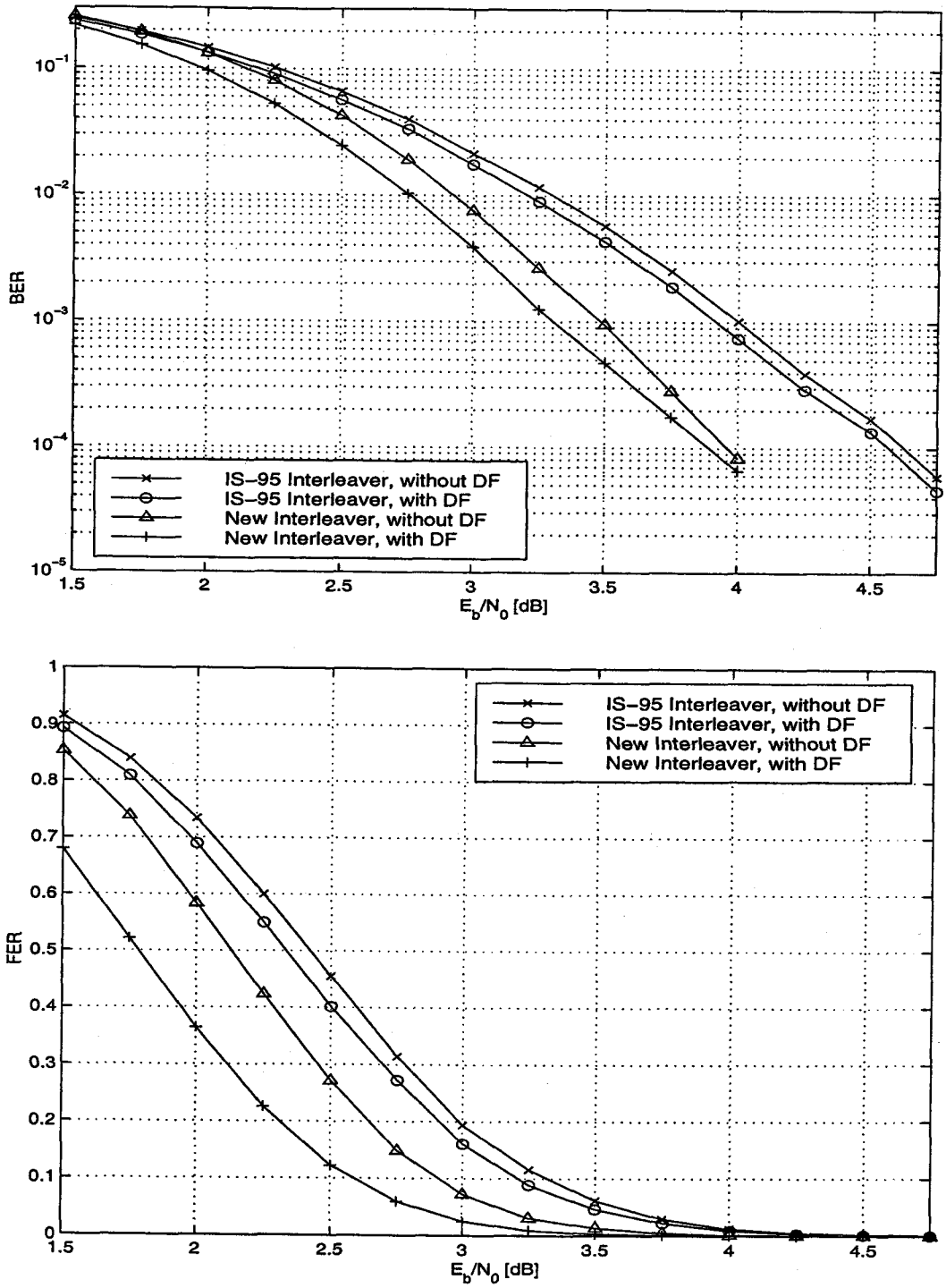


Figure 6.17: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Coherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

For a FER of 1%, the required E_b/N_0 is 3.23 dB, an improvement of 0.84 dB over the IS-95 interleaver without DF. The effectiveness of DF is 0.37 dB for the new interleaver and 0.09 dB for the IS-95 interleaver.

A comparison with the Rayleigh fading single-path results (Table 6.2) and the unfaded multipath results (Table 6.3) indicates a diversity gain of approximately 1 dB for the new interleaver and 1.4 dB for the IS-95 interleaver (BER 10^{-3}). At the same time, the degradation due to Rayleigh fading is about 0.4 dB for the IS-95 interleaver and 0.3 dB for the new interleaver (BER 10^{-3}). These numbers vary with BER and FER.

The noncoherent results show a required E_b/N_0 of 6.74 dB for the new interleaver with DF to achieve a BER of 10^{-3}. The improvement over the IS-95 interleaver without DF is 0.63 dB in this case. The DF effectiveness is 0.11 dB for the new interleaver and 0.06 dB for the IS-95 interleaver.

For an FER of 1%, the required E_b/N_0 is 6.73 dB, an improvement of 0.71 dB over the IS-95 interleaver without DF. The effectiveness of DF is 0.23 dB for the new interleaver and 0.06 dB for the IS-95 interleaver.

A comparison with the unfaded multipath results (Table 6.3) indicates a degradation due to Rayleigh fading of approximately 0.44 dB for the new interleaver and 0.60 dB for the IS-95 interleaver (BER 10^{-3}). In the single-path case, Rayleigh fading resulted in a performance degradation of about 1.80 dB for the new interleaver and 2.42 dB for the IS-95 interleaver (Table 6.2). The multipath diversity gain is therefore 1.36 dB for the new interleaver and 1.82 for the IS-95 interleaver. Again these numbers vary with BER and FER.

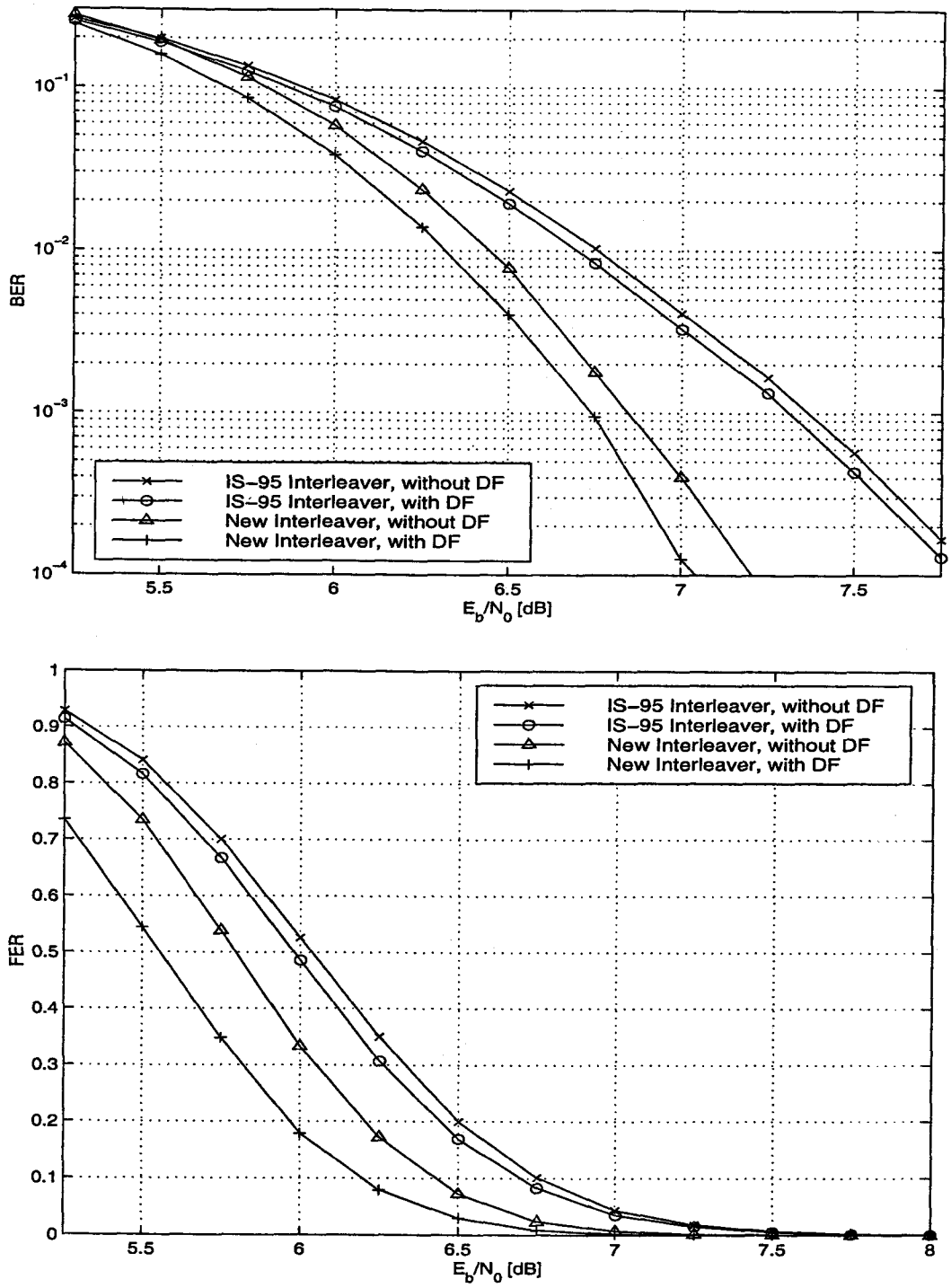


Figure 6.18: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Noncoherent detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

E_b/N_0 [dB] required to achieve		BER 10^{-2}	BER 10^{-3}	BER 10^{-4}	FER 5%	FER 2%	FER 1%
Coherent	IS-95 Interleaver, without DF	3.30	4.00	4.63	3.59	3.88	4.07
	IS-95 Interleaver, with DF	3.21	3.92	4.57	3.48	3.79	3.98
	New Interleaver, without DF	2.93	3.49	3.96	3.13	3.40	3.60
	New Interleaver, with DF	2.76	3.31	3.89	2.82	3.08	3.23
Noncoherent	IS-95 Interleaver, without DF	6.76	7.37	7.80	6.97	7.23	7.44
	IS-95 Interleaver, with DF	6.70	7.31	7.78	6.92	7.19	7.38
	New Interleaver, without DF	6.44	6.85	7.21	6.61	6.80	6.96
	New Interleaver, with DF	6.32	6.74	7.04	6.40	6.61	6.73

Table 6.4: Performance summary for detection of 4 equal-strength *independent Rayleigh fading* multipath signals in AWGN.

Table 6.4 shows the E_b/N_0 that is required in this case to achieve a specified performance in terms of BER and FER.

6.3.4.2. Correlated Rayleigh Fading

For this propagation scenario, the *independent Rayleigh fading* is replaced with *correlated Rayleigh fading* using a maximum Doppler frequency f_m of 100 Hz.

The average data bit decoding delays for a BER of approximately 10^{-3} are shown in Figure 6.19 and Figure 6.20, respectively, for coherent detection with an E_b/N_0 of 5.00 dB and noncoherent detection with an E_b/N_0 of 8.75 dB.

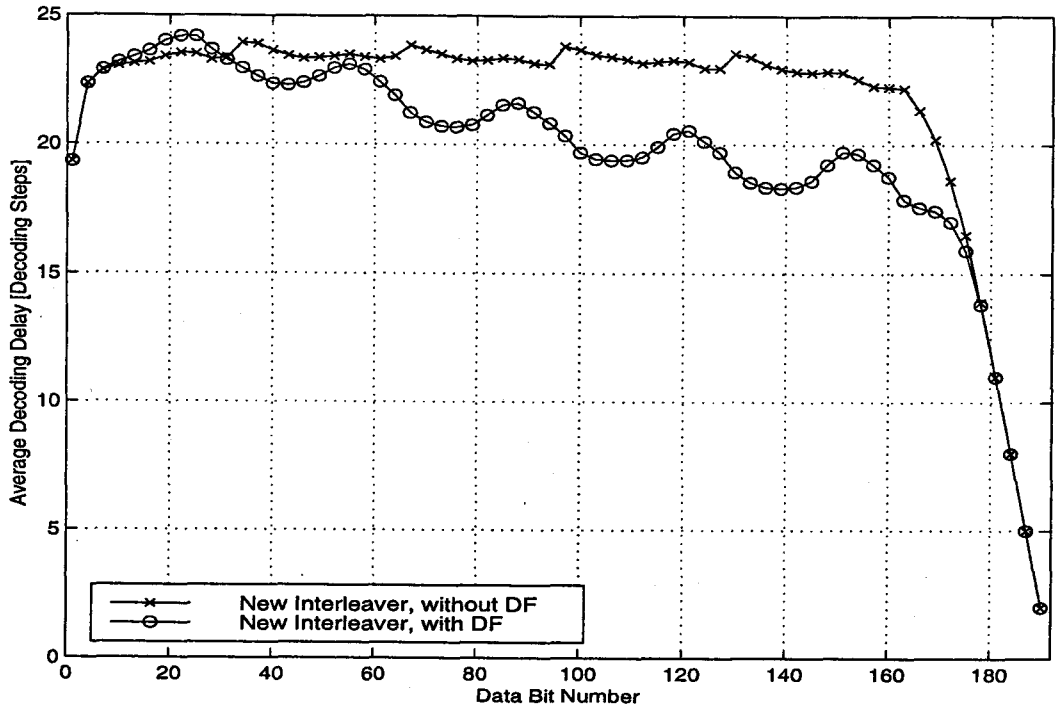


Figure 6.19: Average data bit decoding delays for the new interleaver with/without DF. Coherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN. E_b/N_0=5.00 dB.

For coherent detection, the delay without DF is 24 decoding steps. DF gradually reduces the delay to about 17 decoding steps over the frame. For noncoherent detection, the delay without DF is even lower, 22 decoding steps. DF also reduces it to 17 decoding steps over the frame.

The BER and FER performance is shown in Figure 6.21 and Figure 6.22, respectively, for coherent and noncoherent detection. As expected, performance is degraded compared to *independent Rayleigh fading.*

For coherent detection, the performance degradation is 1.76 dB at a BER of 10^{-3}. The required E_b/N_0 to achieve this BER is now 5.07 dB for the new interleaver with DF.

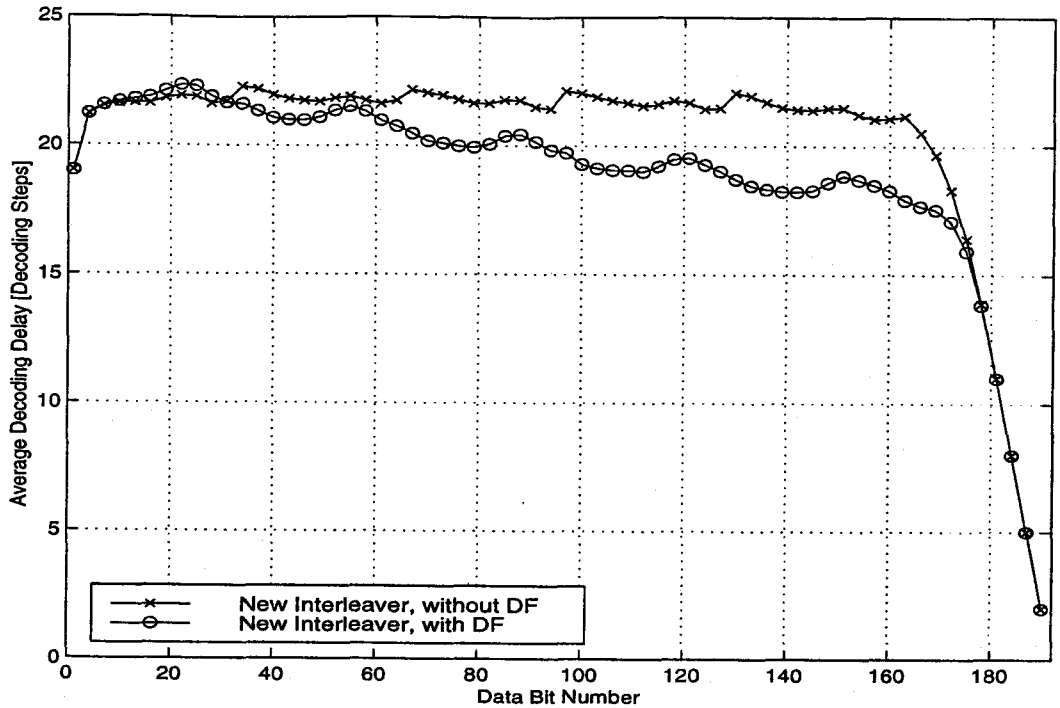


Figure 6.20: Average data bit decoding delays for the new interleaver with/without DF. Noncoherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN. E_b/N_0=8.75 dB.

This is 0.38 dB better than the E_b/N_0 required for the IS-95 interleaver without DF to achieve the same BER. The DF effectiveness for the new interleaver is 0.19 dB and for the IS-95 interleaver it is 0.08 dB.

To achieve a FER of 1%, the new interleaver with DF requires an E_b/N_0 of 4.76 dB, an improvement of 0.63 dB over the IS-95 interleaver without DF. The additional E_b/N_0 reduction through DF is 0.33 dB for the new interleaver and 0.13 dB for the IS-95 interleaver.

Similarly, for noncoherent detection, the performance degradation compared to *independent Rayleigh fading* is about 2.00 dB at a BER of 10^{-3}.

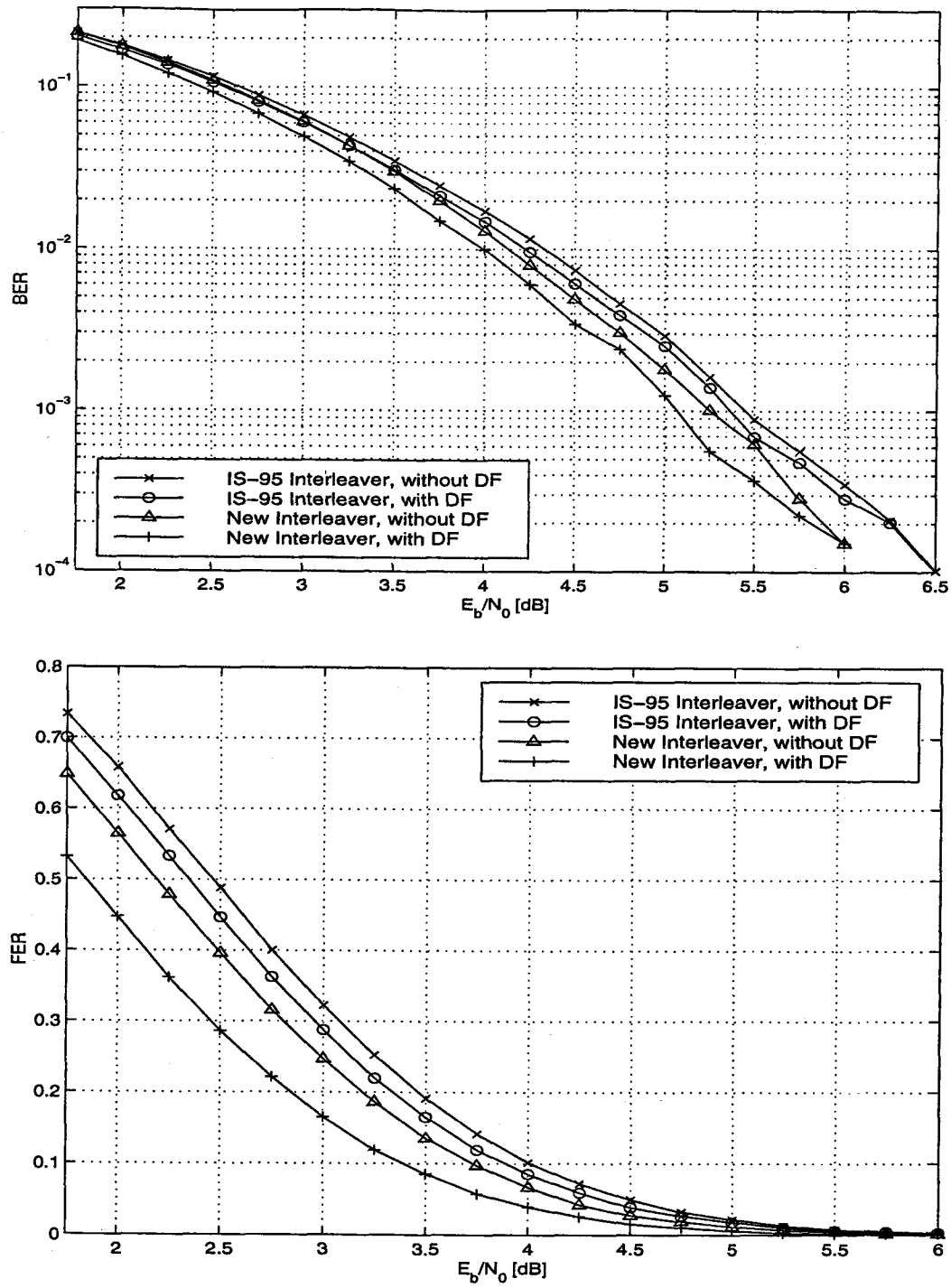


Figure 6.21: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Coherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.

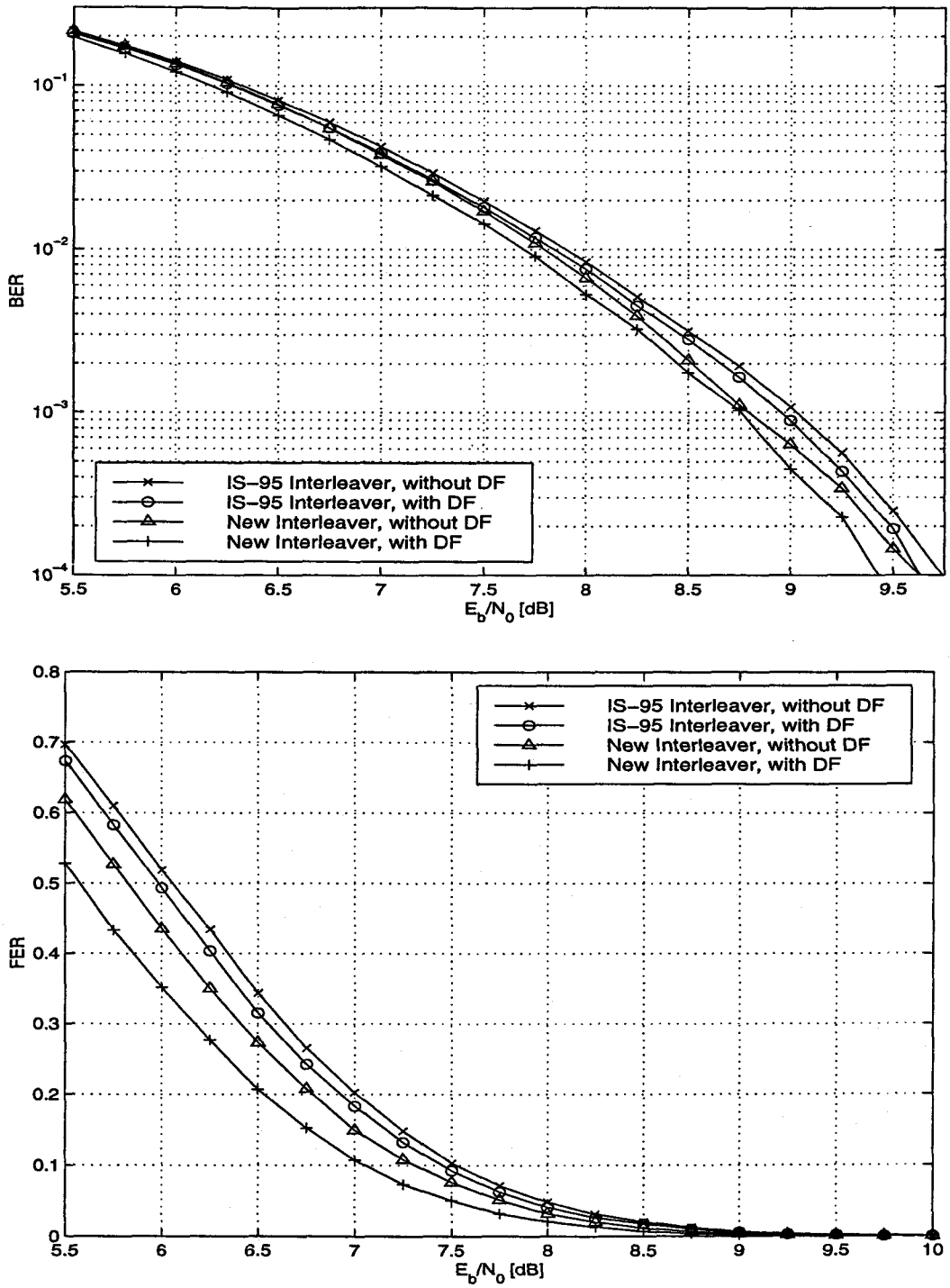


Figure 6.22: BER and FER as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s) and the new interleaver with/without DF. Noncoherent detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.

E_b/N_0 [dB] required to achieve		BER 10^{-2}	BER 10^{-3}	BER 10^{-4}	FER 5%	FER 2%	FER 1%
Coherent	IS-95 Interleaver, without DF	4.33	5.45	6.50	4.49	5.05	5.39
	IS-95 Interleaver, with DF	4.23	5.37	6.50	4.36	4.94	5.26
	New Interleaver, without DF	4.13	5.26	6.18	4.17	4.71	5.09
	New Interleaver, with DF	4.00	5.07	6.14	3.85	4.38	4.76
Noncoherent	IS-95 Interleaver, without DF	7.90	9.03	9.74	7.98	8.51	8.86
	IS-95 Interleaver, with DF	7.84	8.95	9.63	7.89	8.43	8.78
	New Interleaver, without DF	7.79	8.80	9.63	7.76	8.25	8.58
	New Interleaver, with DF	7.70	8.76	9.43	7.50	8.02	8.37

Table 6.5: Performance summary for detection of 4 equal-strength *correlated Rayleigh fading* multipath signals (f_m=100 Hz) in AWGN.

The required E_b/N_0 to achieve this BER is now 8.76 dB for the new interleaver with DF. This is 0.27 dB better than the E_b/N_0 required for the IS-95 interleaver without DF to achieve the same BER.

The DF effectiveness for the new interleaver is 0.04 dB and for the IS-95 interleaver it is 0.08 dB. This is the only simulation scenario where the effectiveness of DFD was less for the new interleaver at this BER.

To achieve a FER of 1%, the new interleaver with DF requires an E_b/N_0 of 8.37 dB, an improvement of 0.49 dB over the IS-95 interleaver without DF. The additional E_b/N_0 reduction through DF is 0.21 dB for the new interleaver and 0.08 dB for the IS-95 interleaver.

It should be noted that the simulation results of this section show signs of insufficient statistics. Therefore these results are given with less confidence.

Table 6.5 shows the E_b/N_0 that is required in this case to achieve a specified performance in terms of BER and FER.

6.4. New Interleaver and *Earlier Decisions*

The average data bit decoding delays for the new interleaver design are close to the value of 32 that is required for most efficient DF performance. This suggests that *earlier decisions,* the sub-optimal data bit decision criterion for the convolutional decoder presented in the previous chapter, could result in additional performance improvement for the new interleaver design. In this section, this is investigated using the example of noncoherent detection of an unfaded single-path signal. As before, the parameter M is chosen to be 221.

Without DF, Figure 6.23 shows that the average data bit decoding delays for the new interleaver with *earlier decisions* are reduced by about 12 decoding steps (BER about 10^{-3}). With DF, the reduction is 12 decoding steps at the beginning of the frame and about 8 at the end of the frame. The reduction of the decoding delay within the frame (due to DF) is smaller when *earlier decisions* are used.

The BER and FER performance are shown in Figure 6.24. A small performance improvement is seen for higher values of the BER. For a BER of 10^{-3}, the required E_b/N_0 is reduced an additional 0.01 dB when *earlier decisions* are used. However, for BERs below $3 \cdot 10^{-4}$ the performance is actually degraded.

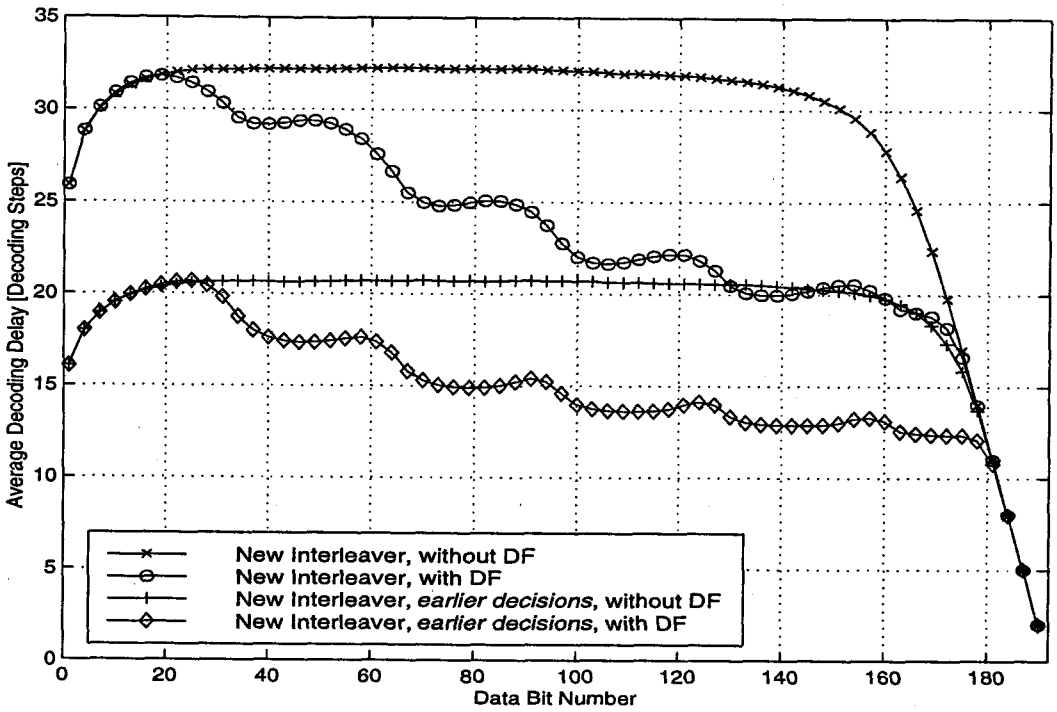


Figure 6.23: Average data bit decoding delays for the new interleaver using regular/*earlier decisions*, with/without DF. Noncoherent detection of unfaded single-path signal in AWGN. E_b/N_0=4.50 dB.

Similar behavior is observed for the FER performance. For high FERs, the performance is improved. However, for a FER below 4% no performance improvement is visible. The simulation data shows performance degradation for a FER below 1%.

These observations are not surprising. Just like in the case of the IS-95 interleaver, the use of *earlier decisions* results in additional DF information that is made available to the convolutional decoder. However, the additional improvement is much smaller here. The data bit decoding delays for a BER of 10^{-3} are such that the new interleaver with regular data bit decisions leads to almost optimal DF effectiveness.

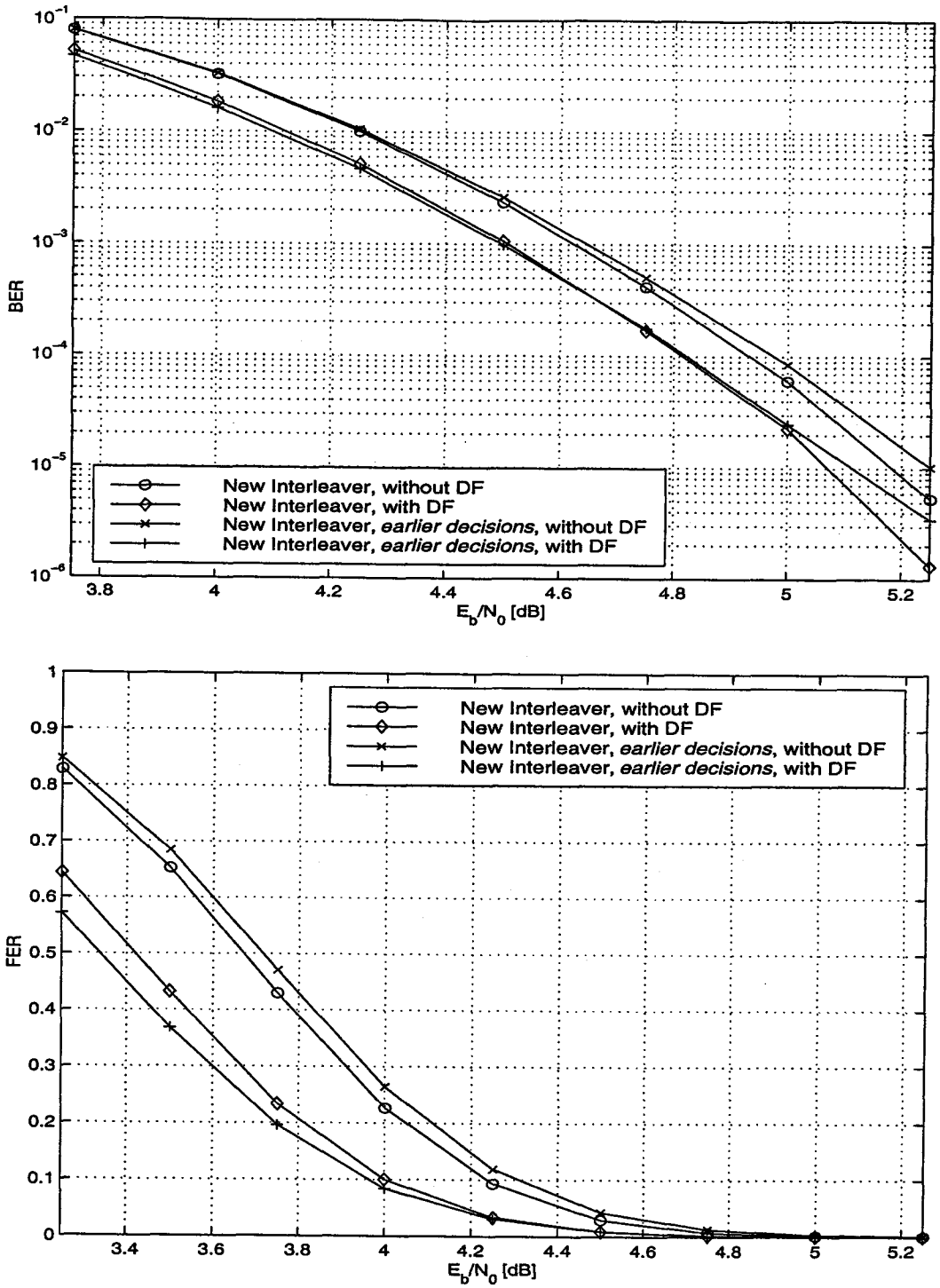


Figure 6.24: BER and FER as a function of E_b/N_0 for the new interleaver using regular/*earlier decisions* with/without DF. Noncoherent detection of unfaded single-path signals in AWGN.

For even higher E_b/N_0 values, the decoding delay is further reduced to a point where almost no additional information can be obtained with *earlier decisions*. Then the sub-optimality of *earlier decisions* becomes dominant and degrades the performance. Similar behavior was observed for the other simulation scenarios.

Generally, the combination of the new interleaver and *earlier decisions* will result in small performance improvement for BERs above 10^{-3} (FER above 2%). Some cases showed improvement even at a BER of 10^{-4} (FER of 1%). For E_b/N_0 values where the average decoding delay is at or below the value of 32 decoding steps, the performance improvement is minimal. If the delay is significantly below 32 decoding steps, performance degradation eventually occurs when the sub-optimality of *earlier decisions* becomes the dominant factor.

6.5. Conclusions

In this chapter, a new block interleaver design for the IS-95 uplink (9.6 kbits/s data rate frames) was presented. The design motivation was to improve the effectiveness of decision feedback decoding (DFD) (Section 2.4) by increasing the separation of convolutionally coded bits that are transmitted in the same Walsh groups.

However, additional performance improvement was obtained even without DFD. This additional improvement is explained by the systematically correlated metrics used by the convolutional decoder, which are separated more in time by the new interleaver design.

Performance was evaluated by computer simulation in additive white Gaussian noise (AWGN), with and without

multiple signal paths. Unfaded and Rayleigh faded signals were considered. Tables 6.1-6.5 show the E_b/N_0 required to achieve specific bit error rates (10^{-2}, 10^{-3}, and 10^{-4}) and frame error rates (5%, 2%, and 1%) for each of the five simulation scenarios.

In summary, the new interleaver with DF outperforms all other decoder cases, especially the IS-95 interleaver without DF, at BERs between 10^{-2} and 10^{-4} and FERs between 1% and 5% in each of the considered simulation scenarios. The total performance gain in E_b/N_0 afforded by the new interleaver with DF for a BER of 10^{-3} varies between 0.27 dB and 1.16 dB. Looking at the 1% FER comparisons the performance gain of the new interleaver with DF varies between 0.49 dB and 1.35 dB.

Performance improvement in all these cases suggests that use of the new interleaver with DFD could improve the capacity and/or the quality of service of a modified cellular IS-95 system or a system using a similar concatenation of convolutional coding, interleaving, and orthogonal Walsh modulation.

The effectiveness of DFD was improved by the new interleaver design as intended by increasing the number of decoding metrics that benefit from previous DF updates. Most notably for the FER comparisons where the effectiveness of DFD improved by a minimum factor of 2 compared to the IS-95 interleaver in all simulation scenarios and decoder cases. The average improvement factor was 3.8 and the peak improvement factor observed was 15.5. For the BER comparisons the average and peak improvement factor of the DFD effectiveness were, respectively, 2.12 and 8.5. There were some data points where the DFD effectiveness of the IS-95 interleaver was greater than for the new interleaver. This was observed

only at low BER (10^{-4}). These few exceptions can be attributed to insufficient statistics in those cases.

Compared with the simulation results for the iterative decoding technique applied to IS-95 in [41][42], the new interleaver with DFD achieves the same performance for coherent detection in AWGN as the iterative decoder after five iterations at BER 10^{-3}. For noncoherent detection the performance lies somewhere between the first and fifth iteration. Finally, for noncoherent detection of 4 equal-strength unfaded multipath signals the performance of the new interleaver with DFD is within 0.1 dB of the fifth iteration of the iterative decoder. It should be pointed out again that the DF concept does not require iterations.

The sub-optimal data bit decision criterion introduced in Chapter 5 can also be applied in conjunction with the new interleaver design. The additional performance improvement is small, since the new interleaver design results in very efficient DFD even with regular data bit decisions. At high E_b/N_0 values performance degradation may occur.

Chapter 7. Analysis of Decision Feedback

The extensive performance evaluations of the decision feedback (DF) decoder and the proposed improvements are based on computer simulations. The reason for this is the inherent difficulty to obtain closed-form analytical results that quantify the performance improvement obtained when decision feedback decoding (DFD) is used.

This chapter presents some analytical justification, approximations, and experiments pertaining to the DF decoder performance. First, a qualitative analytical justification of the DF performance improvement is presented. It is shown that correct DF results in better decoding metrics by reducing the error probability of the convolutionally coded bits. Then, the effect of incorrect DF is evaluated experimentally. The decoder is forced to make errors and the resulting bit error rate (BER) degradation due to DF is obtained. At the end of this chapter, a method to approximate the average number of DF metric updates is presented. Simulation results verify that the average data bit decoding delay and knowledge of the interleaver specification are sufficient to estimate the average number of decoding metrics that receive no DF update, 1 DF update, and so on.

7.1. Qualitative Analytical Justification

An exact analysis of the IS-95 uplink BER performance is difficult even without the use of DFD. Most results use simplifying assumptions and upper bounds of error probability [19][22][61]. Analysis of the improvements

obtained using DFD is still more complicated. A similar qualitative analysis of DFD than the one presented here can be found in [24] and also [49].

As an aide to understanding how the proposed DF decoder can improve the BER performance, consider the simple case where the maximum correlation value of each Walsh group is used to determine the Walsh code that was most likely transmitted. The corresponding interleaved convolutionally coded bits (possibly scaled with the maximum correlation value) are used as the metrics for convolutional decoding.

It can be shown that, in the case of noncoherent detection of an unfaded single-path signal in additive white Gaussian noise (AWGN) interference, the probability density functions (pdf's) of the correct and incorrect correlation values are given by (Appendix A)

$$p_C(x) = \frac{1}{2\sigma_C^2}\, e^{-\frac{x+\alpha^2 N^2 E_C}{2\sigma_C^2}}\, I_0\left(\frac{\sqrt{\alpha^2 N^2 E_C x}}{\sigma_C^2}\right) \cdot u(x),$$

$$p_I(x) = \frac{1}{2\sigma_C^2}\, e^{-\frac{x}{2\sigma_C^2}} \cdot u(x),$$

where α is the signal amplitude, N is the number of chips that are added during the correlation (in this case N=256), E_C is the chip energy, and $I_0(x)$ is the modified Bessel function of the first kind of zero order. $\sigma_C^2 = 256 \cdot \sigma^2$, where σ^2 is the variance of the chip interference. $u(x)$ is the unit step function indicating that the correlation values for noncoherent detection are nonnegative.

The correct correlation value and all the incorrect correlation values are mutually independent random variables. This is because the interference is modeled as AWGN and the fact that the Walsh codes are mutually orthogonal.

Initially, all $M = 64$ correlation values are considered by the DF decoder. The distribution function of the $M - 1$ incorrect correlation values is

$$F_I(x) = \int_{-\infty}^{x} p_I(z)\, dz = \left[1 - e^{-\frac{x}{2\sigma_c^2}}\right] \cdot u(x)$$

and, by independence, the distribution function of the maximum incorrect correlation value is

$$F_{IMAX}(x) = F_I(x)^{M-1}.$$

$F_{IMAX}(x)$ is the probability that the maximum incorrect correlation value is smaller or equal to x.

Let P_W be the probability of choosing an incorrect Walsh code when selecting the Walsh code with the maximum correlation value. It is obtained by averaging over all possible values of the correct correlation value

$$\begin{aligned} P_W &= \int_0^{\infty} \left[1 - F_{IMAX}(x)\right] \cdot p_C(x)\, dx \\ &= \int_0^{\infty} \left[1 - F_{IMAX}(x)\right] \cdot \frac{1}{2\sigma_C^2}\; e^{-\frac{x+N^2E_C}{2\sigma_{C2}}}\, I_0\left(\frac{\sqrt{N^2E_Cx}}{\sigma_C^2}\right) dx \end{aligned}$$

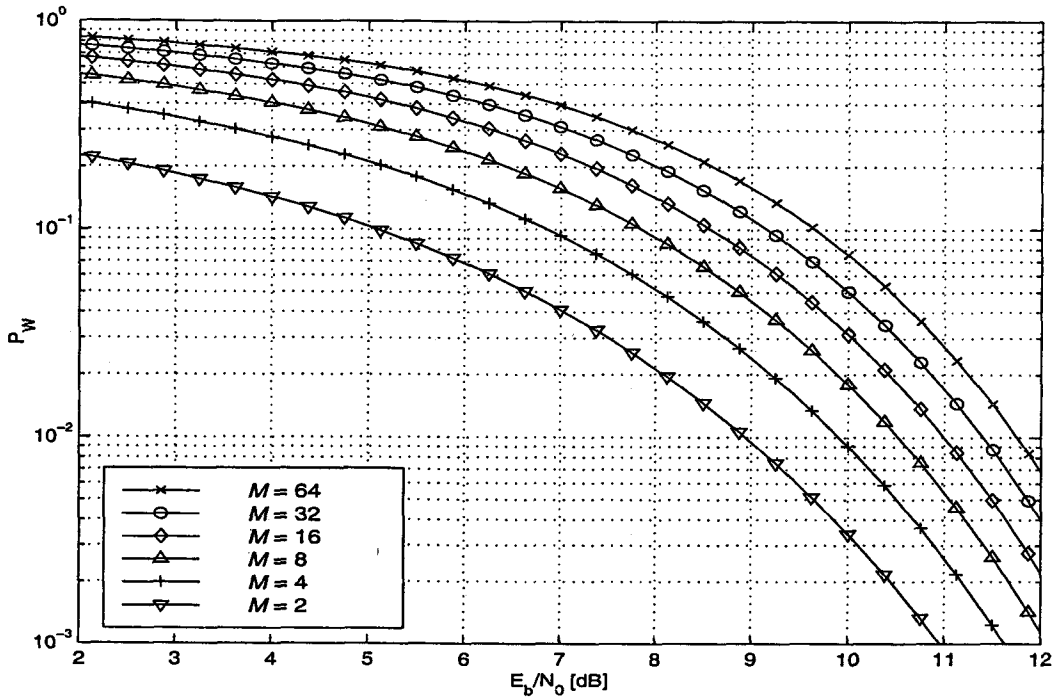


Figure 7.1: Reduction of the probability P_W by correct DF. Noncoherent detection of an unfaded single-path signal in AWGN.

Each correct data bit decision of the convolutional decoder reduces M by half (in the three Walsh groups that contain the corresponding interleaved convolutionally coded bits) resulting in a decrease of P_W as shown in Figure 7.1.

This can be interpreted as an improvement in bit-energy-to-interference-density (E_b/N_0) ratio for the remaining interleaved convolutionally coded bits that are encoded in these Walsh groups.

The same argument applies for (non)coherent detection of L (un)faded multipath signals using the appropriate pdf's. For example, if noncoherent detection of L equal-strength unfaded multipath signals with equal-gain combining is considered, the (non-)central chi-squared

densities with 2 degrees of freedom above are replaced by (non-)central chi-squared densities with $2L$ degrees of freedom [41][42].

The most likely transmitted interleaved convolutionally coded bits are determined by the Walsh code with the maximum correlation value. The error probability of these interleaved convolutionally coded bits P_E follows directly from P_W, since exactly half of the Walsh codes encode the correct value:

$$P_E = 1 - \left(\frac{\frac{1}{2} M - 1}{M - 1} \cdot P_W + (1 - P_W) \right) = \frac{\frac{3}{2} M - 1}{M - 1} \cdot P_W$$

Thus, the error probability of the convolutionally coded bits is reduced by correct DF metric updates. This also reduces the probability of data bit errors.

In cases where the maximum correlation value does not correspond to the transmitted Walsh code, it is still possible that DF invalidates the incorrect correlation value. This is because the convolutional decoder considers the correlation values of many Walsh groups before data bit decisions are made. Compared to a non-DF decoder, these events improve the performance of the DF decoder even further.

7.2. Effect of Incorrect Data Bit Decisions

Incorrect data bit decisions reduce the improvement in decoder performance because the convolutional encoder will use a different path in the code trellis [46]. If no additional errors are made this incorrect path rejoins the correct path after $K = 9$ decoding steps (as long as the

wrong data bit remains in the convolutional encoder shift-register). From the corresponding $K/r = 27$ convolutionally reencoded bits 9 agree with the actually transmitted ones while 18 are different (except for the encoder tail-bits at the end of the frame).

The 18 different convolutionally reencoded bits will invalidate the correct correlation value in their corresponding Walsh groups. Therefore, future metric computations from these Walsh groups consider only identically distributed incorrect correlation values providing no useful information to the decoder. However, since three different Walsh groups are considered in each decoding step, it is likely that the decoder still obtains some useful information from the other two Walsh groups. Since the BER of interest is relatively low (10^{-3}), incorrect data bit decisions occur infrequently so that, on the average, DFD results in performance improvement.

In order to examine the effect of incorrect data bit decisions on the performance of the DF decoder, the following experiment is performed: The Walsh chip sequence of a frame is directly used as the input of a Walsh correlator resulting in interference free Walsh correlations at the input of the DF decoder. There are no errors under normal conditions, but here the decoder is forced to make incorrect data bit decisions at predetermined positions within the frame.

Assuming consecutive data bit errors, two different deterministic error patterns are chosen creating a worst case scenario, respectively, for the new interleaver and the IS-95 interleaver. With the new interleaver, the decoder uses the correlation values of all Walsh groups evenly over the entire frame. Therefore in the first case, the errors are generated at the beginning of the frame.

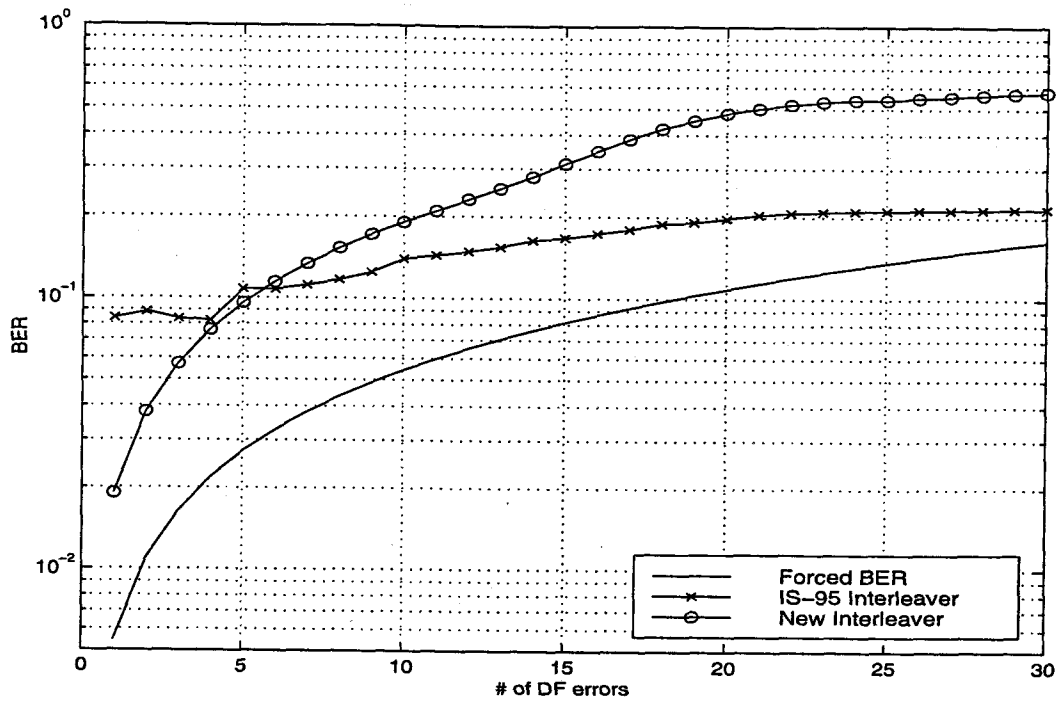


Figure 7.2: Effect of forced DF errors. Worst case scenario for new interleaver.

As already discussed, the IS-95 interleaver design creates three decoding stages, that each use different Walsh groups. Prior incorrect data bit decisions are inconsequential when a new decoding stage begins because the decoder uses another set of Walsh groups. Therefore, in the second case the errors are generated, three at a time, at the beginning of the three decoding stages.

In a third scenario, the data bit errors are created at random locations within the frame. Note, that BERs above $1/184=5.43\cdot10^{-3}$ are considered for this experiment.

The results are shown in Figure 7.2, Figure 7.3, and Figure 7.4. The BER that corresponds to the number of forced incorrect data bit decisions without DF is compared to the BER when DF is used.

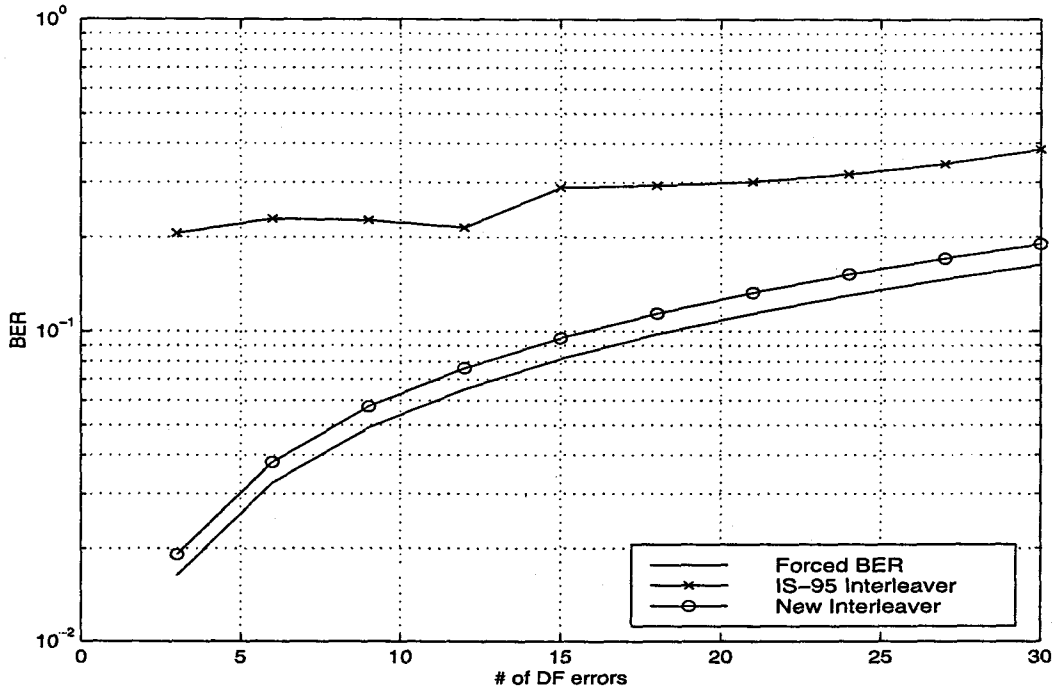


Figure 7.3: Effect of forced DF errors. Worst case scenario for IS-95 interleaver.

As expected, performance degradation occurs because the DF of incorrect data bits invalidates correct Walsh codes. Even under worst case conditions (Figure 7.2), the new interleaver results in a lower BER degradation up to the occurrence of 5 data bit errors. In Figure 7.3, worst case for the IS-95 interleaver, the new interleaver design is clearly superior.

When more than 3 data bit errors occur at random locations within a frame, the new interleaver design will, on average, result in a higher BER degradation than the IS-95 interleaver does (Figure 7.4). This explains the fact that the new interleaver exhibits a higher FER improvement than BER improvement compared to the IS-95 interleaver.

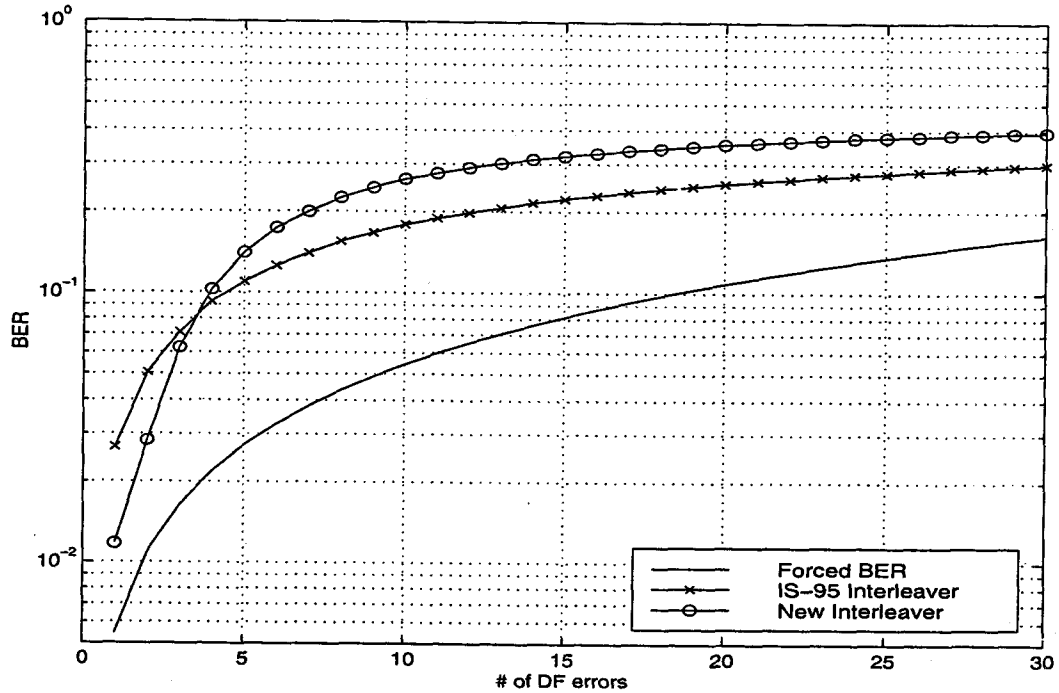


Figure 7.4: Effect of forced DF errors. Data bit errors at random locations within the frame.

The performance advantage of the new interleaver stems from the separation of systematically correlated decoding metrics and less degradation in situations when only up to 3 data bit errors occur.

In conclusion, incorrect data bit decisions that are fed back result in additional degradation. The degradation is however not catastrophic in the sense that the performance of the convolutional code only deteriorates gradually with an increasing number of incorrect data bit decisions. This is what was expected, based on the argument at the beginning of this section.

7.3. Approximate Number of Updated Decoding Metrics

In this section, a closer look is taken at the actual number of decoding metrics which have no DF update, 1 DF update, and so on. This might be useful for analysis of the DF decoder. As already discussed (Section 2.4), these numbers are determined by the data bit decoding delays and the interleaver specification.

Using average values, it is possible to approximate the average number of decoding metrics which have no DF update, 1 DF update, and so on, up to five DF updates. The case of coherent and noncoherent detection of an unfaded single-path signal is used as an example here.

For the IS-95 specified interleaver (Section 1.4.3.2), the average decoding delay using regular decisions and no DF is 27 decoding steps (Figure 5.3, Figure 5.4). It follows then from the IS-95 interleaver specification, that the 32 Walsh groups of each decoding stage are used three times before a DF metric update occurs. After that, a DF metric update occurs each time before the Walsh groups are reused, 3 times all together.

The conclusion in this case is that there are, on average, 3*3*32=288 decoding metrics with no prior DF update, 3*32=96 metrics with 1 DF update, 96 metrics with 2 DF updates, and 96 metrics with 3 DF updates.

Similarly, if the average decoding delay is reduced using *earlier decisions* to 17 decoding steps, then the 32 Walsh groups of each decoding stage are used twice before a DF metric update occurs. Then, a DF metric occurs each time before the Walsh groups are reused, 4 times all together.

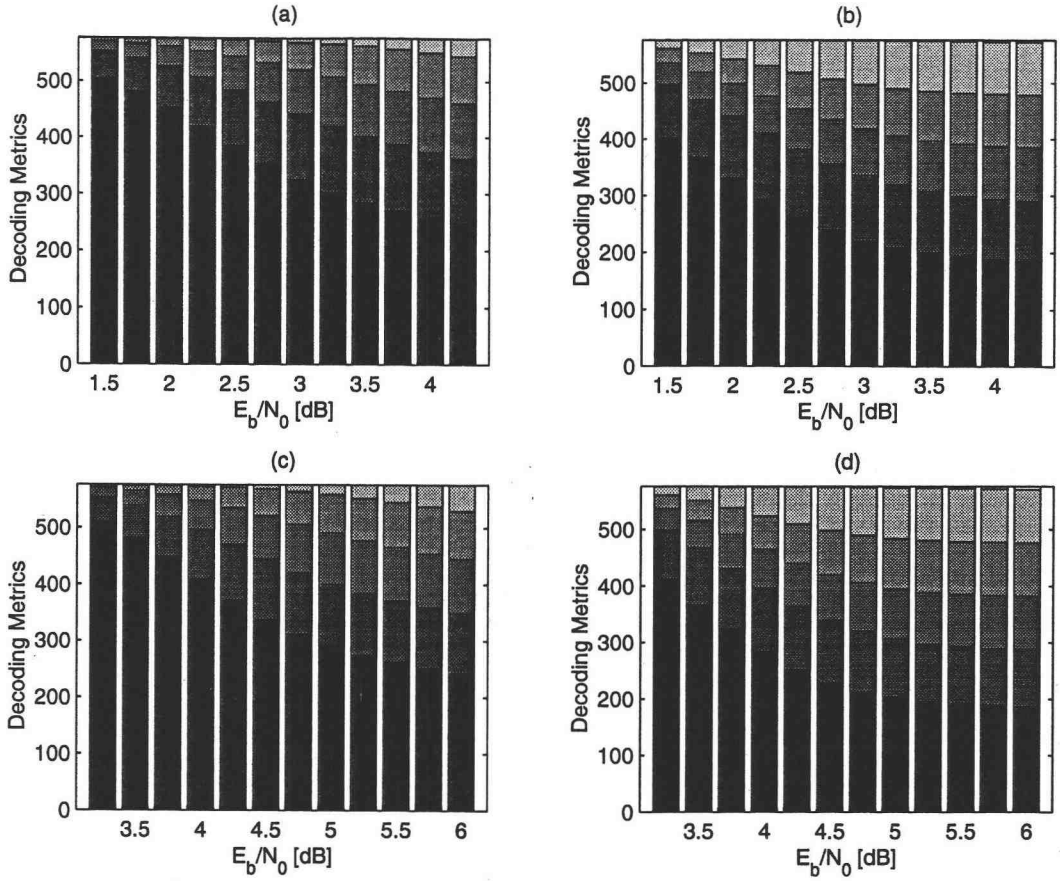


Figure 7.5: Average DF decoding metric updates as a function of E_b/N_0 for the IS-95 interleaver (Uplink, 9.6 kbits/s). Unfaded single-path signal in AWGN. (a) coherent detection, regular decisions, (b) coherent detection, *earlier decisions*, (c) noncoherent detection, regular decisions, (d) noncoherent detection, *earlier decisions*. From the bottom up, the bars indicate the number of decoding metrics with no prior DF update, 1 DF metric update, and so on.

Consequently there are, on average, 2*3*32=182 decoding metrics with no prior DF update, 3*32=96 metrics with 1 DF update, 96 metrics with 2 DF updates, 96 metrics with 3 DF updates, and 96 metrics with 4 DF updates.

On average, *earlier decisions* result in an additional DF metric update for each Walsh group resulting in the observed performance improvement.

This can be verified by computer simulation by actually counting for each frame the number of decoding metrics with no DF update, 1 DF update, and so on. Average results are shown in Figure 7.5. As expected, the reduction of the average decoding delay with increasing E_b/N_0 results in more metrics with DF updates in all cases. Also, the expected effect of *earlier decisions* is clearly demonstrated in the plots.

For coherent detection, the actual numbers (rounded to the nearest integer) for an E_b/N_0 of 3.5 dB, corresponding to a BER close to 10^{-3}, are as follows:

Using regular decisions there are, on average, 285 decoding metrics with no prior DF update, 117 metrics with 1 DF update, 92 metrics with 2 DF updates, 67 metrics with 3 DF updates, and 14 metrics with 4 DF updates.

Using *earlier decisions* there are, on average, 199 decoding metrics with no prior DF update, 109 metrics with 1 DF update, 88 metrics with 2 DF updates, 88 metrics with 3 DF updates, 89 metrics with 4 DF updates, and 2 metrics with 5 DF updates.

Similarly, for noncoherent detection, the actual numbers (rounded to the nearest integer) for an E_b/N_0 of 5 dB, corresponding to a BER close to 10^{-3}, are as follows:

Using regular decisions there are, on average, 291 decoding metrics with no prior DF update, 110 metrics with 1 DF update, 91 metrics with 2 DF updates, 68 metrics with 3 DF updates, and 17 metrics with 4 DF updates.

Using *earlier decisions* there are, on average, 203 decoding metrics with no prior DF update, 104 metrics with

1 DF update, 89 metrics with 2 DF updates, 89 metrics with 3 DF updates, 90 metrics with 4 DF updates, and 2 metrics with 5 DF updates.

These results match the approximations reasonably well. The discrepancies are due to the random nature of the decoding delays.

When the new interleaver design of Chapter 6 is used, the average decoding delay without DF is about 32 decoding steps (Figure 6.3, Figure 6.4). Almost optimum effectiveness of DF is expected in this case, i.e., 96 decoding metrics without prior DF update, 96 metrics with 1 DF update, and so on.

Using *earlier decisions* in conjunction with the new interleaver design, the average decoding delay can be reduced even further as shown in Section 6.4. However, only a small improvement of the effectiveness of DF is expected in this case. The results in Figure 7.6 verify this for both coherent and noncoherent detection.

The actual numbers (rounded to the nearest integer) for an E_b/N_0 of 3.0 dB, corresponding to a BER close to 10^{-3}, are as follows:

Using regular decisions there are, on average, 131 decoding metrics with no prior DF update, 87 metrics with 1 DF update, 79 metrics with 2 DF updates, 89 metrics with 3 DF updates, 95 metrics with 4 DF updates, and 95 metrics with 5 DF updates.

Using *earlier decisions* there are, on average, 102 decoding metrics with no prior DF update, 92 metrics with 1 DF update, 94 metrics with 2 DF updates, 96 metrics with 3 DF updates, 96 metrics with 4 DF updates, and 96 metrics with 5 DF updates.

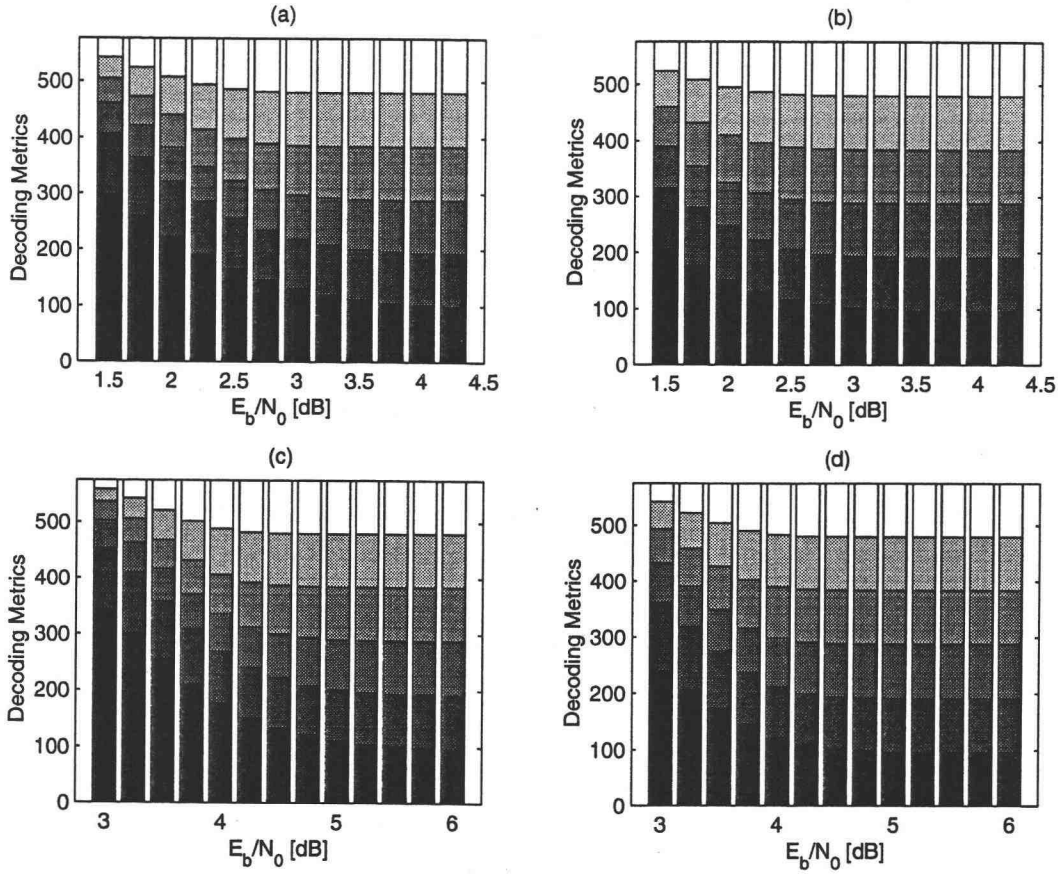


Figure 7.6: Average DF decoding metric updates as a function of E_b/N_0 for the new interleaver. Unfaded single-path signal in AWGN. (a) coherent detection, regular decisions, (b) coherent detection, *earlier decisions*, (c) noncoherent detection, regular decisions, (d) noncoherent detection, *earlier decisions*. From the bottom up, the bars indicate the number of decoding metrics with no prior DF update, 1 DF metric update, and so on.

Similarly, for noncoherent detection, the actual numbers (rounded to the nearest integer) for an E_b/N_0 of 4.5 dB, corresponding to a BER close to 10^{-3}, are as follows:

Using regular decisions there are, on average, 134 decoding metrics with no prior DF update, 88 metrics with 1 DF update, 78 metrics with 2 DF updates, 87 metrics with

3 DF updates, 94 metrics with 4 DF updates, and 95 metrics with 5 DF updates.

Using *earlier decisions* there are, on average, 102 decoding metrics with no prior DF update, 92 metrics with 1 DF update, 94 metrics with 2 DF updates, 96 metrics with 3 DF updates, 96 metrics with 4 DF updates, and 96 metrics with 5 DF updates.

Again, these results match the approximations reasonably well. The discrepancies are due to the random nature of the decoding delays. For regular decisions, the discrepancies are more pronounced here because the average decoding delay is just at the threshold value of 32 that is required for most efficient DF.

7.4. Conclusions

In this chapter, a qualitative analytical justification of the decision feedback (DF) decoder performance was presented. The improvement of the decoding metrics by correct DF was demonstrated using the example of noncoherent detection of an unfaded single-path signal in additive white Gaussian noise (AWGN) interference.

The effect of incorrect data bit decisions was also considered. An experiment showed that incorrect DF results only in gradual performance degradation. On average, DF improves the performance.

An approximation for the average number of decoding metric updates was also presented. It is based on the average decoding delay of the convolutional decoder and the interleaver specification.

Chapter 8. Summary and Open Research Areas

This dissertation presented a new and comprehensive evaluation of a decision feedback (DF) receiver/decoder design for efficient demodulation of IS-95 uplink traffic channels, as well as two very significant enhancements to improve its performance.

First, an evaluation of the DF decoder performance in a multipath and fading environment was presented. Continued performance improvement under such conditions is a requirement for the implementation of the DF decoder in a mobile communications environment. This was demonstrated in all studied cases.

The concept of the effectiveness of DF was also introduced in this work. It relates to the ability of the decoder to use the additional information provided by DF. The decoding delay of the convolutional decoder and the IS-95 interleaver specification were identified as limiting factors of DF effectiveness.

Under the constraint of the IS-95 interleaver specification, it was shown that DF is more efficient when a sub-optimal data bit decision criterion is used within the convolutional decoder. The presented decision criterion, *earlier decisions*, reduces the average decoding delay of the convolutional decoder. It results in more effective DF and consequently in additional performance improvement compared to the original DF decoder design. This is in contrast to non-DF decoders, for which a sub-optimal data bit decision criterion results in performance degradation. Performance improvement of *earlier decisions* can be expected whenever the interleaver specification is such that Walsh groups are reused after a smaller number

of decoding steps than the average decoding delay of the convolutional decoder.

A new block interleaver design was presented next. The design effort was intended to further improve the effectiveness of decision feedback decoding (DFD). This was achieved. In addition, the performance without the use of DF was also improved significantly. In the IS-95 uplink, interleaved convolutionally coded bits are transmitted encoded in Walsh codes. Thus, the channel output corresponding to a Walsh code is used to compute decoding metrics for several convolutionally coded bits. This results in systematically correlated decoding metrics and therefore degradation of the convolutional code performance. The new interleaver design separates the correlated metrics more in time. This increases the effectiveness of DF as well as the coding gain of the convolutional code. An important conclusion is that, in this case, the interleaver design should consider not only the channel effects but also the used modulation.

Finally, additional analytical and experimental results that explain the performance improvement of DF were also presented.

It should be noted that the essence of DFD is the concatenation of convolutional code, interleaver, and orthogonal modulation. The results presented in this dissertation can therefore also be used in other applications using a similar concatenated coding scheme.

The contributions of this work do not complete the study of DF. Indeed, there is an opportunity for further research in several directions.

The performance and the performance improvements of the DF decoder design were focused on the highest data rate traffic channel of the IS-95 uplink. The performance for

the lower rate frames should be evaluated as well. Due to a smaller number of data bits and consequently less opportunity for DF, a smaller performance improvement has to be expected there. Also, the data rates introduced in the IS-95-A revision [36] and ANSI J-STD-008-1996 [38] (rate set 2), especially the highest data rate of 14.4 kbits/s, should be considered.

The results suggest that the new interleaver design leads to a higher number of frames that are received without errors compared to the interleaver specified in IS-95. This could be especially useful for data applications and warrants simulation runs for BERs of 10^{-6} and lower. At the same time, error frames contain, on average, more bit errors, as indicated by the lower BER improvements compared to the FER improvements in the simulation results. For voice communications, the impact on the voice quality of these error events should be investigated. Implementation issues resulting from a change of the block interleaver also need to be addressed.

With respect to convolutional decoding, there is the possibility of bi-directional Viterbi decoding. Since the convolutional code is tailed-off to the zero state, the Viterbi algorithm can also be applied working from the end of the frame toward the beginning. If this is done in parallel with the forward DFD, additional performance improvement might be possible. Also, the DF decoder could be allowed to backtrack in cases where the decoding delay of the convolutional decoder is too big. This way DF could be made most effective. The price for this would be added complexity and possibly more time needed to complete the frame decoding.

The concept of DFD decoding could be combined with the iterative decoding techniques presented in [40]. For this,

the regular Viterbi algorithm has to be replaced with a soft-output Viterbi algorithm (SOVA) [62]. The performance improvement of the DF decoder could effectively reduce the number of iterations required to achieve a certain performance.

Bibliography

[1] Perry La Forge, "A Look Ahead," Mobile Communications International *CDMA Spectrum*, Issue 3, pp. 4-6, December 1997.

[2] V. H. MacDonald, "The Cellular Concept," *The Bell Systems Technical Journal*, Vol. 58, No. 1, pp. 15-41, January 1979.

[3] J. R. Boucher, *Voice Teletraffic Systems Engineering*. Artech House. 1988.

[4] J. Jacobsmeyer, "Improving Throughput and Availability of Cellular Digital Packet Data," *Virginia Tech 4th Symposium on Wireless Personal Communications*, pp. 18.1-18.12, June 1994.

[5] TIA/EIA/IS-54 Interim Standard, "Cellular System Dual Mode Mobile Station - Land Station Compatibility Specifications," Electronic Industries Association, IS-54, May 1990.

[6] European Telecommunications Standards Institute, "GSM Specifications," ETSI TC-SMG, Sophia Antipolis, France, 1991.

[7] Robert A. Scholtz, "The Origins of Spread spectrum Communications," *IEEE Trans. on Communications*, Vol. COM-30, No. 5, pp. 822-854, May 1982.

[8] *IEEE Trans. on Communications*, Vol. COM-25, August 1977. Special Issue on spread spectrum.

[9] *IEEE Trans. on Communications*, Vol. COM-30, May 1982. Special Issue on spread spectrum.

[10] *IEEE J. on Selected Areas in Communications*, Vol. 8, May 1990. Special Issue on spread spectrum.

[11] *IEEE J. on Selected Areas in Communications*, Vol. 8, June 1990. Special Issue on spread spectrum.

[12] *IEEE J. on Selected Areas in Communications*, Vol. 10, May 1992. Special Issue on spread spectrum.

[13] R. Price and P. E. Green, Jr., "A Communication Technique for Multipath Channels," *Proc. IRE*, Vol. 46, pp. 555-570, March 1958.

[14] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein, "Theory of Spread spectrum Communications - A Tutorial," *IEEE Trans. on Communications*, Vol. COM-30, No. 5, pp. 855-884, May 1982.

[15] Roger L. Peterson, Rodger E. Ziemer, and David E. Borth, *Introduction to Spread Spectrum Communications*. Englewood Cliffs, NJ: Prentice-Hall, ISBN: 0-02-431623-7, 1995.

[16] C. E. Shannon, "A Mathematical Theory of Communications," *The Bell Systems Technical Journal*, Vol. 27, pp. 379-423 and 623-656, 1948.

[17] Peter Jung, Paul Walter Baier, and Andreas Steil, "Advantages of CDMA and spread spectrum techniques over FDMA and TDMA in cellular mobile radio applications," *IEEE Trans. On Vehicular Technology*, Vol. 42, pp. 357-364, August 1993.

[18] Raymond L. Pickholtz, Laurence B. Milstein, and Donald L. Schilling, "Spread Spectrum for Mobile Communications," *IEEE Trans. on Vehicular Technology*, Vol. 40, No.2, pp. 313-322, May 1991.

[19] G. L. Turin, "Introduction to Spread spectrum Antimultipath Techniques and Their Application to

Urban Digital Radio," *Proc. IEEE*, Vol. 68, No. 3, pp. 328-353, March 1980.

[20] P. T. Brady, "A Statistical Analysis of On-Off Patterns in 16 Conversations," *The Bell Systems Technical Journal*, Vol. 47, pp. 73-91, January 1968.

[21] William C. Y. Lee, "Overview of Cellular CDMA," *IEEE Trans. on Vehicular Technology*, Vol. 40, No.2, pp. 291-302, May 1991.

[22] Klein S. Gilhousen, Irwin M. Jacobs, Roberto Padovani, Andrew J. Viterbi, Lindsay A. Weaver, Jr., and Charles E. Wheatley III, "On the Capacity of a Cellular CDMA System," *IEEE Trans. on Vehicular Technology*, Vol. 40, No. 2, pp. 303-312, May 1991.

[23] Floyd Simpson, and Jack M. Holtzman, "Direct Sequence CDMA Power Control, Interleaving, and Coding," *IEEE J. on Selected Areas in Communications*, Vol. 11, no. 7, pp. 1085-1095, September 1993.

[24] D. Rabinowitz, "Joint Convolutional and Orthogonal Decoding of Interleaved-Data Frames for IS-95 CDMA Communications," Ph.D. dissertation, Electrical and Computer Engineering, Oregon State University, February 1996.

[25] S. Verdú, "Minimum Probability of Error for Asynchronous Gaussian Multiple-Access Channels," *IEEE Trans. on Information Theory*, Vol. IT-32, No.1, pp. 85-96, January 1986.

[26] R. Lupas, and S. Verdú, "Near-far Resistance of Multiuser Detectors in Asynchronous Channels," *IEEE Trans. on Communications*, Vol. 38, pp. 496-508, April 1990.

[27] M. K. Varanasi, and B. Aazhang, "Multistage Detection in Asynchronous Code-Division Multiple access

Communications," *IEEE Trans. on Communications*, Vol. 38, pp. 509-519, April 1990.

[28] Z. Xie, R. T. Short, and C. K. Rushforth, "Suboptimum Coherent Detection of Direct-Sequence Multiple access Signals," *Proc. 1989 IEEE Military Communications Conference*, Boston MA, October 1989.

[29] Mahesh K. Varanasi, and Subramanian Vasudevan, "Multiuser Detectors for Synchronous CDMA Communcations over Non-selective Rician Fading Channels," *IEEE Trans. on Communications*, Vol. 42, pp. 711-722, Feb./Mar./Apr. 1994.

[30] Mahesh K. Varanasi, and Subramanian Vasudevan, "Optimum Diversity Combiner based Multiuser Detection for Time-dispersive Rician Fading CDMA Channels," *IEEE J. on Selected Areas in Communications*, Vol. 12, pp. 580-592, May 1994.

[31] F. S. Gutleber, "Non-orthogonal Mobile Subscriber Multiple access System," U.S. Patent No. 4,470,138, September 1984.

[32] T. Masamura, "Spread spectrum Multiple access System with Intrasystem Interference Cancellation," *Trans. IEICE*, Vol. E71, pp. 224-231, March 1988.

[33] R. S. Mowbray, R.D. Pringle, and P. M. Grant, "Increased CDMA System Capacity through Adaptive Cochannel Interference Regeneration and Cancellation," *IEEE Proc.*, Vol. 139, Pt. I, No. 5, pp. 515-524, October 1992.

[34] Burton S. Abrams, Andrew E. Zeger, and Thomas E. Jones, "Efficiently structured CDMA receiver with near-far immunity," *IEEE Trans. on Vehicular Technology*, Vol. 44, pp. 1-12, February 1995.

[35] Qualcomm, Inc., "An Overview of the Application of Code Division Multiple Access (CDMA) to Digital

Cellular Systems and Personal Cellular Networks," May 21, 1992.

[36] "Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System," TIA/EIA Interim Standard 95 (IS-95), Washington DC: Telecommunications Industry Association, July 1993 (amended as IS-95-A in May 1995).

[37] Jack Scanlon, "IS-95: progress, performance, and prospects," Mobile Communications International *CDMA Spectrum*, Issue 1, pp. 29-30, June 1997.

[38] ANSI J-STD-008-1996, "Personal Station - Base Station Compatibility Requirements for 1.8 to 2.0 GHz Code Division Multiple Access (CDMA) Personal Communications Systems, " ANSI, 1996.

[39] Theodore S. Rappaport, *Wireless Communications Principles & Practice.* Upper Saddle River, NJ: Prentice Hall PTR, ISBN: 0-13-375536-3, 1996.

[40] Joachim Hagenauer, Elke Offer, and Lutz Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. on Information Theory*, Vol. 42, No. 2, pp. 429-445, March 1996.

[41] R. Herzog, J. Hagenauer, and A. Schmidbauer, "Soft-In/Soft-Out Hadamard Despreader for Iterative Decoding in the IS-95(A) System," *Proc. IEEE VTC*, pp. 1219-1222, 1997.

[42] R. Herzog, A. Schmidbauer, and J. Hagenauer, "Iterative Decoding and Despreading improves CDMA-Systems using M-ary Orthogonal Modulation and FEC," *Proc. IEEE International Conference on Communications*, pp. 909-913, 1997.

[43] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm,"

IEEE Trans. on Information Theory, Vol. IT-13, pp. 260-269, April 1967.

[44] G. D. Forney, Jr., "The Viterbi Algorithm," *Proc. IEEE*, Vol. 61, No 3., pp. 268-278, March 1973.

[45] Andrew J. Viterbi, "Wireless Digital Communication: A View Based on Three Lessons Learned," *IEEE Communications Magazine*, September 1991.

[46] A. J. Viterbi, "Convolutional Codes and Their Performance in Communication Systems," *IEEE Trans. on Information Theory*, Vol. IT-17, pp. 751-772, October 1971.

[47] Andrew J. Viterbi, *CDMA, Principles of Spread Spectrum Communication*. Reading, MA: Addison-Wesley, ISBN 0-201-63374-4, 1995.

[48] Andrew J. Viterbi, "An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes," *IEEE J. of Selected Areas in Communications*, Vol. 16, No. 2, pp. 260-264, February 1998.

[49] David Rabinowitz and Mario E. Magaña, "Decision Feedback for IS-95 Uplink Frame Decoding," *IEEE Trans. on Vehicular Technology*, Vol. 48, No. 2, pp. 527-532, March 1999.

[50] Seiichi Sampei, *Applications of Digital Wireless Technologies to Global Wireless Communications*. Upper Saddle River, NJ: Prentice Hall PTR, ISBN: 0-13-214272-4. 1997.

[51] Henry Stark and John W. Woods, *Probability, Random Processes, and Estimation Theory for Engineers*. Second Edition. Englewood Cliffs, NJ: Prentice-Hall, ISBN: 0-13-728791-7. 1994.

[52] "Recommended Minimum Performance Standards for Base Stations Supporting Dual-Mode Wideband Spread Spectrum Cellular Mobile Stations," TIA/EIA Interim Standard 97 (IS-97), Washington DC: Telecommunications Industry Association, December 1994.

[53] ANSI J-STD-019-1996, "Recommended Minimum Performance Requirements for Base Stations Supporting 1.8 to 2.0 GHz Code Division Multiple Access (CDMA) Personal Stations ," ANSI, 1996.

[54] C. D'Amours, M. Moher, A. Yongacoglu, and J. Wang, "RAKE Receiver Structures for Differential and Pilot Symbol-Assisted Detection of DS-CDMA Signals in Frequency-Selective Rayleigh Fading Channels," *Proc. IEEE Globecom*, pp. 1798-1802, 1993.

[55] F. Ling, "Coherent Detection with Reference-Symbol Based Channel Estimation for Direct-Sequence CDMA Uplink Communications," *Proc. IEEE VTC*, pp. 400-403, 1993.

[56] Rod Walton and Mark Wallace, "Near Maximum Likelihood Demodulation for M-ary Orthogonal Signaling," *Proc. IEEE VTC*, pp. 5-8, 1993.

[57] P. Schramm and J. Huber, "Coherent Demodulation for IS-95 Uplink," *ISSSTA*, pp. 1073-1077, 1996.

[58] J. G. Proakis, *Digital Communications*, 2nd ed., New York: McGraw-Hill, 1989.

[59] Michel C. Jeruchim, Philip Balaban, and K. Sam Shanmugan, *Simulation of Communications Systems*. New York and London: Plenum Press, ISBN 0-306-43989-1, 1992.

[60] Farhad Hemmati and Daniel J. Costello, "Truncation Error Probability in Viterbi Decoding," *IEEE Trans. on Communications*, pp. 530-532, May 1977.

[61] E. K. Hong, K. J. Kim, and K. C. Wang, "Performance Evaluation of DS-CDMA System with M-ary Orthogonal Signaling," *IEEE Trans. On Vehicular Technology*, Vol. 45, No. 1, pp. 57-63, 1996.

[62] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and Its Applications," *Proc. IEEE GLOBECOM '89*, Dallas, TX, pp. 47.1.1-7, November 1989.

Appendices

Appendix A. Probability Density Functions of Sum of Squares of Independent Gaussian Random Variables with Equal Variance

The noncoherent receiver signal processing for the simulation model used in this dissertation (Section 3.5.2) gives rise to random variables, which are the sum of squares of two independent Gaussian random variables with equal variance σ^2. More specifically, the Walsh correlation values, which are the input to the decision feedback (DF) decoder, are such random variables in the case of noncoherent detection of a single-path signal in additive white Gaussian noise (AWGN) interference.

For the correct correlation value, the means of the independent Gaussian random variables are non-zero, resulting in a non-central chi-square probability density function with two degrees of freedom. For the incorrect correlation values, the means of the independent Gaussian random variables are zero, resulting in an exponential probability density function.

The following derivation of these probability density functions has several interesting probability functions as intermediate results, namely the Ricean and Rayleigh probability density functions.

These are accepted models for the signal envelope of fading signals in a mobile propagation environment. The Rayleigh probability density function arises if the received signal consists of a large number of independent multipath components that arrive at the receiving antenna with random phases. For the Ricean probability density, a fixed unfaded (line-of-sight) signal component is added to the received signal.

Consider two independent Gaussian random variables X and Y with the joint probability density function (pdf)

$$p_{X,Y}(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-a)^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-b)^2}{2\sigma^2}} .$$

Now let $Z = \sqrt{X^2 + Y^2}$. The probability distribution function of the random variable Z is given by

$$F_Z(z) = \iint_{\sqrt{x^2+y^2} \le z} p_{X,Y}(x, y)\, dxdy, \quad z \ge 0 .$$

It is convenient to change to polar coordinates in order to evaluate the above integral, i.e.,

$$x = z\cos\Theta, \quad y = z\sin\Theta, \quad dx\,dy = z\,dz\,d\Theta .$$

Then it follows, that

$$\begin{aligned} F_Z(z) &= \int_0^z \int_0^{2\pi} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z\cos\Theta-a)^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z\sin\Theta-b)^2}{2\sigma^2}} z\, d\Theta\, dz \\ &= \int_0^z \int_0^{2\pi} \frac{z}{2\pi\sigma^2} e^{-\frac{z^2+(a^2+b^2)}{2\sigma^2}} e^{\frac{z(a\cos\Theta+b\sin\Theta)}{\sigma^2}} d\Theta\, dz \qquad , \ z \ge 0, \\ &= \int_0^z \frac{z}{\sigma^2} e^{-\frac{z^2+(a^2+b^2)}{2\sigma^2}} I_0\left(\frac{\sqrt{(a^2+b^2)}\, z}{\sigma^2}\right) dz \end{aligned}$$

where $I_0(x)$ is the modified Bessel function of the first kind of order zero defined as

$$I_0(x) = \frac{1}{2\pi} \int_0^{2\pi} e^{x\cdot\cos\Theta} d\Theta .$$

The corresponding pdf is

$$p_Z(z) = \frac{z}{\sigma^2} e^{-\frac{z^2+(a^2+b^2)}{2\sigma^2}} I_0\left(\frac{\sqrt{(a^2 + b^2)}\, z}{\sigma^2}\right), \quad z \geq 0 .$$

The random variable Z has a Ricean probability density function.

The modified Bessel function of the first kind of order zero is a monotonically increasing function of its argument. For large arguments it can be approximated by

$$I_0(z) \approx \frac{e^z}{\sqrt{2\pi z}}, \quad z >> 1$$

Then the random variable Z is approximately Gaussian about the mean with probability density function

$$p_Z(z) = \frac{\sqrt{z / \sqrt{a^2 + b^2}}}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(z+\sqrt{a^2+b^2}\right)^2}{2\sigma^2}}, \quad z \geq 0 .$$

If the random variables X and Y have zero means (a=b=0), then the random variable Z has a Rayleigh probability density function.

$$p_Z(z) = \frac{z}{\sigma^2} e^{-\frac{z^2}{2\sigma^2}}, \quad z \geq 0$$

Now let $W = Z^2$. The probability density function of the random variable W is easily obtained using the theory of function of random variables [51]:

$$p_W(w) = \frac{1}{2\sigma^2} e^{-\frac{w+(a^2+b^2)}{2\omega^2}} I_0\left(\frac{\sqrt{(a^2 + b^2)w}}{\sigma^2}\right), \quad w \geq 0 .$$

This is known as a non-central chi-square probability density function with two degrees of freedom. It is the probability density function of the correct correlation values for noncoherent detection of an unfaded single-path signal in AWGN.

If the mean values of the random variables X and Y are zero ($a=b=0$) the probability density function of W becomes central chi-squared with two degrees of freedom, which is the same as an exponential probability density function:

$$p_W(w) = \frac{1}{2\sigma^2} e^{-\frac{w}{2\sigma^2}}, \quad w \geq 0 .$$

This is the probability function of the incorrect correlation values for noncoherent detection of an unfaded single-path signal in AWGN.

Appendix B. MATLAB™ Source Code

Source code for the MATLAB™ scripts and functions used to generate the simulation results. Time critical routines are implemented as MEX-functions in C. This source code should be used with MATLAB™ version 5.1 or higher. Throughout binary '0' and '1' are mapped to '1' and '-1' respectively.

B.1. simulation.m - Main Simulation

```
%function simulation % Define as function for profiling!

% Discrete multipath simulation model for IS-95 Uplink.

% SIMULATION DESCRIPTION
% Put description here.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

% SIMULATION PARAMETERS
FileName='simulation'; % Save results to this file.

% Probability estimate that will disable decoders.
StopProb=.2;

% Bit-energy-to-interference-density ratios to be simulated.
EbN0=[1.5:.25:8];

% Decoder command lines. Will be used by eval(). Note: df_decode1()
% has same source as df_decode(). df_decode() is initialized with
% IS-95 interleaver. df_decode1() is initialized with the new
% interleaver.
% Correspondingly, cwalshcorr and ncwalshcorr are the correlations
% for the IS-95 interleaver. cwalshcorr1 and ncwalshcorr1 are the
% correlations for the new interleaver.
% Make sure all lines have same length.

DecoderNames=[
   'df_decode(cwalshcorr,1,0,[],[],255)    ' % no DF, M=256
   'df_decode(nwalshcorr,1,1,[],[],220)    ' % DF, M=221
   'df_decode1(cwalshcorr1,1,0,[],[],255) '
   'df_decode1(nwalshcorr1,1,1,[],[],255) '
];

% Number of frames to simulate. Increase this number in order to add
```

```
% more statistics to a simulation.
NumberOfFrames=10000;

% Multipaths to consider.
StartPath=1; % >= 1 and <= EndPath
EndPath=3; % >= StartPath and <= 5

% Simulation type. 0=Both, 1=Coherent, 2=Noncoherent
SimulationType=0;

% Type of multipath. 1=Unfaded, 2=Rayleigh faded, 3=Rayleigh with
% Doppler
FadingType=2;

% Maximum Doppler frequency, needed for Doppler fading.
% Sampling frequency of tapgains is 4800 Hz fixed below.
% Tapgains remain constant over the duration of a Walsh group.
MaxDoppler=100;

% Modification below this line should not be necessary!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize decoders.
df_decode(w_corr(w_mod(intlv(c_code(mk_frame)))),1,0,intlv([0:575]),
dintlv([0:575]));
df_decode1(w_corr(w_mod(intlv1(c_code(mk_frame)))),1,0,intlv1([0:575]
),dintlv1([0:575]));

% Compute signal-to-noise ratio and corresponding variance for
% interference.
SNR=EbN0+3;
Variance= 256 * 10.^(-SNR/10);
StandDev=sqrt(Variance);

% Number of decoders.
[NDecoders tmp]=size(DecoderNames);
clear tmp;

% Number of paths.
NumberOfPaths=EndPath-StartPath+1;

% Initialize some variables.
Biterrors=zeros(length(Variance),NDecoders,5);
Frameerrors=zeros(length(Variance),NDecoders,5);
MaxTimeout=zeros(length(Variance),NDecoders,5,192);
MinTimeout=192*ones(length(Variance),NDecoders,5,192);
AvgTimeout=zeros(length(Variance),NDecoders,5,192);
DF_Check=zeros(length(Variance),NDecoders,5,6);

Start=length(Variance);

% Decoder status matrix.
% Decoders will be disabled during simulation if error probability
% estimate is greater than StopProb.
% Initially all decoders are enabled for all multipath cases.
DecoderEnabled=ones(NDecoders,length(Variance),5);

% Save NumberOfFrames to detect if more statistics are requested.
NumberOfFrames1=NumberOfFrames;
```

```
% Frame to start at. Used if adding statistics to simulation.
StartFrame=1;

% Random number generator seeds. Begin in zero-state for each EbN0.
RandState=0;
RandnState=0;

% Load fading variables if present.
if FadingType==3
   if exist([FileName '.dat'])
      eval(['load ' FileName '.dat -mat']);
   end
end

% Load data of previous or interrupted simulation.
% For new simulation the file needs to be deleted!
if exist([FileName '.mat'])
   eval(['load ' FileName]);
end

% Detect if more statistics are requested.
if NumberOfFrames1>NumberOfFrames
   Start=length(Variance); % reset Start
   StartFrame=NumberOfFrames+1; % set new StartFrame value
   NumberOfFrames=NumberOfFrames1;
end

% Initialize variables that are expected by mex-files.
risignal=zeros(1,24576);
rqsignal=zeros(1,24576);
risignal1=zeros(1,24576);
rqsignal1=zeros(1,24576);

% Begin at high Eb/N0 values because decoding is faster.
for j=Start:-1:1

   % Set random number generator to initial state.
   randn('state',RandnState);
   rand('state',RandState);

   for nframes=StartFrame:NumberOfFrames

      % Generate a random frame (192 bits) for transmission.
      frame=mk_frame;
      % Convolutionally encode the frame (576 code symbols).
      cframe=c_code(frame);
      % Interleave the frame (576 interleaved code symbols).
      iframe=intlv(cframe);
      iframe1=intlv1(cframe);

      % Walsh modulate the interleaved code symbols
      % (96 Walsh groups a 64 Walsh chips = 6144 Walsh chips).
      wframe=w_mod(iframe);
      wframe1=w_mod(iframe1);

      % Increase samples by a factor of 4 since each Walsh chip
      % will be spread by 4 pseudorandom code chips (24576 chips).
      chips=upfirdn(wframe,[1 1 1 1],4); % signal processing toolbox
      chips1=upfirdn(wframe1,[1 1 1 1],4);
```

```
% No long code spreading.

% Signal is separated into in-phase and quadrature sequences.
isignal=chips;
qsignal=chips;
isignal1=chips1;
qsignal1=chips1;

% Spread by random short-codes. Use binary random sequences.
icode=2*(rand(1,24576)>0.5)-1;
qcode=2*(rand(1,24576)>0.5)-1;
isignal=isignal.*icode;
qsignal=qsignal.*qcode;
isignal1=isignal1.*icode;
qsignal1=qsignal1.*qcode;

% Generate noise interference.
inoise=randn(5,24576);
mult(inoise,StandDev(j));
qnoise=randn(5,24576);
mult(qnoise,StandDev(j));

% MULTIPATH PARAMETERS
% in this simulation we simulate up to five equal-strength
% multipath signals that arrive at the receiver with more than
% a chip-time of time delay between each other. We assume that
% the receiver has aquired the timing of these multipath
% signals and is tracking them perfectly. Also the receiver is
% perfectly downconverting the signal from the carrier
% frequency. For the coherent detectors the phase of each
% multipath signal is also perfectly known. For equal-strength
% multipath signals the equal gain combination is equivalent
% with maximum ratio combining.

if FadingType==2 % Rayleigh fading

   rFading=sqrt(1/2).*randn(5,96);
   iFading=sqrt(1/2).*randn(5,96);

elseif FadingType==3 % Rayleigh fading with Doppler spectrum

   % First time compute fading variables.
   if j==length(Variance)

      % For Doppler Rayleigh Fading case generate Rayleigh
      % fading complex variables with average power of 1 and
      % Dopplerspectrum with fs=4800 Hz, fdmax=MaxDoppler Hz

      TempFading=tapgain(4800,MaxDoppler,5,1);

      % Pick the 5 fading envelopes from the long sequences at
      % random locations.
      for LoopFading=1:5
         Temp1Fading=ceil(rand(1,1)*(length(TempFading)-95));

      Fading(LoopFading,:)=TempFading(LoopFading,Temp1Fading:Te
      mpFading+95);
      end

      rFading=real(Fading);
```

```
        iFading=imag(Fading);
        realFading(nframes,:,:)=rFading;
        imagFading(nframes,:,:)=iFading;

    else

        rFading=squeeze(realFading(nframes,:,:));
        iFading=squeeze(imagFading(nframes,:,:));

    end
end

for npaths=StartPath:EndPath % For each multipath case.

    % Multipath strengths, sum of betas=1.
    betas=zeros(1,5);
    betas(1:npaths)=(1/npaths)*ones(1,npaths); % equal-strength
    alphas=sqrt(betas);

    if SimulationType<=1

        % Generate coherent correlations.

        % Initialize the correlations.
        cwalshcorr=zeros(64,96);
        cwalshcorr1=zeros(64,96);

        if FadingType==1

            for path=1:npaths

                risignal=alphas(1,path).*isignal;
                rqsignal=alphas(1,path).*qsignal;

                risignal=risignal + inoise(path,:);
                rqsignal=rqsignal + qnoise(path,:);

                cwalshcorr=cwalshcorr+alphas(path).*w_cdmod1(risign
                al,rqsignal,icode,qcode);

                risignal1=alphas(1,path).*isignal1;
                rqsignal1=alphas(1,path).*qsignal1;

                risignal1=risignal1 + inoise(path,:);
                rqsignal1=rqsignal1 + qnoise(path,:);

                cwalshcorr1=cwalshcorr1+alphas(path).*w_cdmod1(risi
                gnal1,rqsignal1,icode,qcode);

            end

        elseif FadingType>=2


            fadex1c(npaths,alphas,isignal,qsignal,icode,qcode,inoi
            se,qnoise,rFading,iFading,cwalshcorr);
            cwalshcorr=cwalshcorr';

            fadex1c(npaths,alphas,isignal1,qsignal1,icode,qcode,in
            oise,qnoise,rFading,iFading,cwalshcorr1);
```

```
      cwalshcorr1=cwalshcorr1';

   end
end

if SimulationType==0 | SimulationType==2

   % Generate noncoherent correlations.

   % Initialize the correlations.
   nwalshcorr=zeros(96,64);
   nwalshcorr1=zeros(96,64);

   if FadingType==1

      for path=1:npaths

         risignal=alphas(1,path).*isignal;
         rqsignal=alphas(1,path).*qsignal;
         risignal=risignal + inoise(path,:);
         rqsignal=rqsignal + qnoise(path,:);

         nwalshcorr=nwalshcorr+alphas(1,path).*w_ncdmo1(risi
         gnal,rqsignal,icode,qcode);

         risignal1=alphas(1,path).*isignal1;
         rqsignal1=alphas(1,path).*qsignal1;
         risignal1=risignal1 + inoise(path,:);
         rqsignal1=rqsignal1 + qnoise(path,:);

         nwalshcorr1=nwalshcorr1+alphas(1,path).*w_ncdmo1(ri
         signal1,rqsignal1,icode,qcode);

      end

   elseif FadingType>=2

      for path=1:npaths


         fadex(isignal,qsignal,rFading(path,:),iFading(path,
         :),alphas(path),risignal,rqsignal);
         risignal=risignal + inoise(path,:);
         rqsignal=rqsignal + qnoise(path,:);

         nwalshcorr=nwalshcorr+alphas(1,path).*w_ncdmo1(risi
         gnal,rqsignal,icode,qcode);


         fadex(isignal1,qsignal1,rFading(path,:),iFading(pat
         h,:),alphas(path),risignal1,rqsignal1);
         risignal1=risignal1 + inoise(path,:);
         rqsignal1=rqsignal1 + qnoise(path,:);

         nwalshcorr1=nwalshcorr1+alphas(1,path).*w_ncdmo1(ri
         signal1,rqsignal1,icode,qcode);

      end
   end
end
```

```
      % DECODING

      DecodedFrames=zeros(NDecoders,192);
      timeout=zeros(1,NDecoders,5,192);
      df_check=zeros(1,NDecoders,5,6);

      % Decode frame for each given decoder command line.
      for decoder=1:NDecoders
         if (DecoderEnabled(decoder,j,npaths))
               [DecodedFrames(decoder,:)
timeout(1,decoder,npaths,:)
                df_check(1,decoder,npaths,:)] =
                  eval(DecoderNames(decoder,:));
         end
      end

      % Find maximum and minimum decoding delays for each decoder
      % and update output if new maxima/minima were encountered.
      for decoder=1:NDecoders
         if (DecoderEnabled(decoder,j,npaths))

            idx=(timeout(1,decoder,npaths,:) >
                 MaxTimeout(j,decoder,npaths,:));
            MaxTimeout(j,decoder,npaths,idx) =
                 timeout(1,decoder,npaths,idx);
            idx=(timeout(1,decoder,npaths,:) <
                 MinTimeout(j,decoder,npaths,:));
            MinTimeout(j,decoder,npaths,idx) =
                 timeout(1,decoder,npaths,idx);

            AvgTimeout(j,decoder,npaths,:) =
               AvgTimeout(j,decoder,npaths,:) +
               timeout(1,decoder,npaths,:);
            DF_Check(j,decoder,npaths,:) =
               DF_Check(j,decoder,npaths,:) +
               df_check(1,decoder,npaths,:);

         end
      end

      % Count decoding errors for each case.
      DecodingErrors=zeros(1,NDecoders);
      for decoder=1:NDecoders
         if (DecoderEnabled(decoder,j,npaths))
            DecodingErrors(1,decoder) =
               sum((frame-DecodedFrames(decoder,:))~=0);
         end
      end

      % Add to total decoding errors for each case.
      Biterrors(j,:,npaths)=Biterrors(j,:,npaths) +
         DecodingErrors;
      Frameerrors(j,:,npaths)=Frameerrors(j,:,npaths) +
         (DecodingErrors>0);

   end
end

% Check probability estimate values and disable decoder case if
```

```
   % larger than StopProb.
   for decoder=1:NDecoders
      for npaths=StartPath:EndPath
         if DecoderEnabled(decoder,j,npaths)
            if (Biterrors(j,decoder,npaths)/(184*nframes)>StopProb)
               DecoderEnabled(decoder,1:j-1,npaths)=zeros;
            end
         end
      end
   end

   % Save intermediate results.
   Start=j-1;
   eval(['save ' FileName ' DF_Check StandDev MaxTimeout MinTimeout
         AvgTimeout Start NumberOfFrames FadingType MaxDoppler
         DecoderEnabled DecoderNames NDecoders nframes j SNR EbN0
         Variance Biterrors Frameerrors']);

   if FadingType==3
      if j==length(Variance)
         eval(['save ' FileName '.dat realFading imagFading
               NumberOfFrames MaxDoppler FadingType']);
      end
   end

   % Break loop if all decoders are disabled.
   if sum(sum(DecoderEnabled(:,j,StartPath:EndPath)))==0
      j=1; % To indicate that we are done and not interrupted!
      break;
   end
end

% Save results with random number seeds.
RandState=rand('state');
RandnState=randn('state');
eval(['save ' FileName ' RandState RandnState DF_Check StandDev
      MaxTimeout MinTimeout AvgTimeout Start NumberOfFrames
      FadingType MaxDoppler DecoderEnabled DecoderNames Ndecoders
      nframes j SNR EbN0 Variance Biterrors Frameerrors']);

% End of simulation!
```

B.2. mk_frame.m - Frame Generator

```
function frame=mk_frame()

% Input : ()
% Output: frame(1,192)
%
% Returns a random frame (192-bits) of '-1', '1' values corresponding
% to binary '1' and '0', using the MATLAB rand() number generator.
% The random number generator is not reset nor set to a specific
% seed! The frame quality indicator bits are random, i.e., are not
% generated according to the IS-95 standard.
%
% (C) Patrick Volz, 1997
```

```
% Generate 192 random +-1 's using the uniform generator
frame=2*((rand(1,192)>0.5))-1;

% The last eight bits of the frame are tail bits. They return the
% convolutional encoder that follows in the zero state at the end of
% each frame. Set them to 1.
frame(1,185:192)=ones(1,8);
```

B.3. c_code.c - Convolutional Encoder

```
/*
  Version 1.0, Copyright 1997, Patrick Volz
  Convolutional encoder for IS-95 uplink frame

  Usage: frameout=c_code(framein)

  framein(1,192)  data bits
  frameout(1,576) convolutionally coded bits

  Constraint length K=9, code rate r=1/3 convolutional code
  Generator functions: g0=557 , g1=663 , and g2=711 (octal)
  (see TIA/EIA/IS-95 6.1.3.1.3)
*/

#define TRUE 1
#define FALSE 0

#include <stdio.h>

/* MEX support function declarations and prototypes */
#include "mex.h"

/* ------------------------   COMPUTATION   -------------------------
*/
void computation(double framein[], double frameout[])
{
int    i,j;                        /* variables for loops */
unsigned int k,k1,k2,k3;           /* variables for state loops */

static int   BN_CSN[192][3];       /* code symbol number lookup */
static int   E_OUTPUTS[256][8];    /* encoder outputs */
static int   s_flag=FALSE;         /* indicate if E_OUTPUTS is computed
*/

int e_output[3];          /* store one encoder output */
int e_state=0;            /* store encoder state, begin in zero state
*/


/* --------------   first compute all static variables   ------------
-*/
if (s_flag==FALSE) {

   s_flag=TRUE;
```

```
/*
  First compute the possible encoder outputs for this code.
  The states of the encoder are numbered from 0 to 255. The state
  number corresponds to the 8 leftmost bits in the encoder shift
  register. The 9th bit is discarded when a new input bit is
  shifted in:

  shift register, after new bit is shifted in:

  | current input | b7 | b6 | b5 | b4 | b3 | b2 | b1 | -> b0:
                                                         discarded

  The old state number in binary is :  b7   b6 b5 b4 b3 b2 b1 b0
  The new state number in binary is : input b7 b6 b5 b4 b3 b2 b1

  Thus, if the new input is '1' (corresponds to binary '0'), the new
  state number is the old state divided by two, and if the new input
  is '-1' the new state number is the old state number divided by two
  plus 128.

  The encoder output for the '1' input is computed according to the
  generator polynomials of the convolutional code. Since the input
  bit is used in the computation of each output symbol, the output
  for the '-1' input is simply the negative of that. The encoder
  outputs are stored in positions 0, 1, and 2.

  The new state for an '1' and '-1' input are stored at positions 3
  and 4 of E_OUTPUTS so that they can be looked up when needed.

  The expected input bit and the two originating states for each
  state are stored in positions 5, 6, and 7.

*/


   /* for all states (numbered 0 ... 255) calculate elements of
      E_OUTPUTS */
   for (k=0;k<=255;k++) {

      /* initialize output to all ones */
      for (i=0;i<=2;i++) E_OUTPUTS[k][i]=1;

      /* use state and '1' (one) to determine actual outputs */
      /* first output: generator mask 1 0 1 1 0 1 1 1 1 */
      /* check bits 1,2,3,4,6,7 and toggle output if set */
      if (k & 0x01) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
      if (k & 0x02) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
      if (k & 0x04) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
      if (k & 0x08) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
      if (k & 0x20) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
      if (k & 0x40) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];

      /* second output: generator mask 1 1 0 1 1 0 0 1 1 */
      /* check bits 1,2,5,6,8 and toggle output if set */
      if (k & 0x01) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
      if (k & 0x02) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
      if (k & 0x10) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
      if (k & 0x20) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
      if (k & 0x80) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
```

```
      /* second output: generator mask 1 1 1 0 0 1 0 0 1 */
      /* check bits 1,4,7,8 and toggle output if set */
      if (k & 0x01) E_OUTPUTS[k][2]= -E_OUTPUTS[k][2];
      if (k & 0x08) E_OUTPUTS[k][2]= -E_OUTPUTS[k][2];
      if (k & 0x40) E_OUTPUTS[k][2]= -E_OUTPUTS[k][2];
      if (k & 0x80) E_OUTPUTS[k][2]= -E_OUTPUTS[k][2];

      /* Add the next state for a '1' and '-1' input in position 3
         and 4. For a '1' input the next state will be the current
         state shifted one bit to the right and for a '-1' input the
         next state will be the current state shifted one bit to the
         right + 128. */
      k1=(k >> 1);       /* k1 = k/2       */
      k2=(k1 | 0x80);    /* k1 = k/2 + 128 */
      E_OUTPUTS[k][3]=k1;
      E_OUTPUTS[k][4]=k2;

      /* Add the expected input bit in order to reach this state in
         position 5. */
      if (k<128)
         E_OUTPUTS[k][5]=1;
      else
         E_OUTPUTS[k][5]=-1;

      /* Add the two originating states at positions 6 and 7 */
      k1=(k << 1);    /* k1 = 2*k   */
      k2=k1+1;        /* k2 = 2*k+1 */
      k3=(k | 0x80); /* k3 = k+128 */
      if (k<128) {
         E_OUTPUTS[k][6]=k1;
         E_OUTPUTS[k3][6]=k1;
         E_OUTPUTS[k][7]=k2;
         E_OUTPUTS[k3][7]=k2;
      }
   }     /* E_OUTPUTS is now computed */

   /* compute bit number to code symbol number lookup */
      for (i=0;i<192;i++) {
         j=3*i;
         BN_CSN[i][0]=j;
         BN_CSN[i][1]=j+1;
         BN_CSN[i][2]=j+2;
      }
}

/* --------------------   ACTUAL ENCODING   --------------------- */

   for (i=0;i<192;i++) {
      k=BN_CSN[i][0];
      for (j=0;j<3;j++)
         frameout[k+j]=(framein[i]==1) ?
            E_OUTPUTS[e_state][j]:-E_OUTPUTS[e_state][j];
         e_state=(framein[i]==1) ?
            E_OUTPUTS[e_state][3]:E_OUTPUTS[e_state][4];
   }
}

/* --------------------  END OF COMPUTATION  -------------------- */

/* -------------   MEX FUNCTION GATEWAY ROUTINE   --------------- */
```

```
void mexFunction(
   int nlhs, mxArray *plhs[], /* # of, ptr on input arguments */
   int nrhs, const mxArray *prhs[]) /* # of, ptr on output arguments
*/
{
/* declare the function arguments (i.e. choose names for them here)*/
double *framein;      /* input arguments */

double *frameout;     /* output arguments */

/* check number of arguments */
if (nrhs > 1) {
   mexErrMsgTxt("Only 1 input argument  allowed!");
} else if (nlhs > 1) {
   mexErrMsgTxt("Only 1 output argument allowed!");
}

/* check input arguments */
if (!mxIsNumeric(prhs[0]) || mxIsComplex(prhs[0])
   || mxIsSparse(prhs[0]) || !mxIsDouble(prhs[0])
   || !((mxGetM(prhs[0]) == 1) && (mxGetN(prhs[0]) == 192))) {
   mexErrMsgTxt("Input argument must be a real 1 x 192 matrix!");
}

/* create matrices for output arguments */
plhs[0]=mxCreateDoubleMatrix(1, 576, mxREAL);

/* dereference the arguments and call computational routine */
frameout=mxGetPr(plhs[0]);
framein=mxGetPr(prhs[0]);

computation(framein,frameout);
}
/* ---------------------  END OF GATEWAY   ---------------------- */
```

B.4. intlv.m - IS-95 Uplink Interleaver

```
function frameout=intlv(framein)

% Input  : framein(1,576)
% Output : frameout(1,576)
%
% Performs the interleaving for a 9600 bps frame of IS-95 reverse
% link. The interleaver is an 32 x 18 array. It is filled by columns
% and read by rows. The order of the rows is as specified by the
% standard, i.e., 1,2,3,...,32. The input is a (1,576) vector of
% convolutionally encoded data bits plus overhead (quality indicators
% + tail bits) of one frame (192 bits). See TIA/EIA/IS-95 6.1.3.1.5.
%
% No size or content checks are performed on the input!
%
% (C) Patrick Volz, 1997

A=reshape(framein,32,18)'; % put input vector in matrix and transpose
```

```
frameout=reshape(A,1,576); % put matrix in output vector
```

B.5. dintlv.m - IS-95 Uplink Deinterleaver

```
function frameout=dintlv(framein)

% In : framein(1,576)
% Out: frameout(1,576)
%
% Inverse operation of intlv.m

A=reshape(framein,18,32)'; % put input vector in matrix and transpose

frameout=reshape(A,1,576); % put matrix in output vector
```

B.6. intlv1.m - New Interleaver

```
function frameout=intlv1(framein)

% In : framein(1,576)
% Out: frameout(1,576)
%
% New interleaver for testing of possible decision feedback
% improvement. Performs the interleaving for a 9600 bps frame of IS-
% 95 reverse link. The input is a (1,576) vector of convolutionally
% encoded data bits plus overhead (quality indicators + tail bits) of
% one frame (192 bits).
%
% No size or content checks are performed on the input!
%
% (C) Patrick Volz, 1998

A=reshape(framein,96,6); % put input vector in a matrix

% Row scrambling = block interleaving of row numbers with a (4,24)
% block interleaver.
idx=reshape(reshape([1:96],4,24)',1,96);
A=A(idx,:);

% put transposed matrix in output vector
frameout=reshape(A',1,576);
```

B.7. dintlv1.m - New Deinterleaver

```
function frameout=dintlv1(framein)

% In : framein(1,576)
% Out: frameout(1,576)
```

```
%
% Inverse operation of intlv1.m
%
% (C) Patrick Volz, 1998

A=reshape(framein,6,96)'; % put input vector in matrix and transpose

% Row scrambling = block interleaving of row numbers with a (24,4)
% block interleaver.
idx=reshape(reshape([1:96],24,4)',1,96);
A=A(idx,:);

frameout=reshape(A,1,576); % put matrix in output vector
```

B.8. w_mod.m - Walsh Modulator

```
function frameout=w_mod(framein)

% In : framein(1,576)
% Out: frameout(1,6144)
%
% Orthogonally modulates the input frame using Walsh functions
% according to the IS-95 uplink standard. The input frame is assumed
% to be 576 symbols long (a convolutionally encoded and interleaved
% frame). For every 6 input symbols one of 64 orthogonal Walsh codes
% is generated and output. The output frame is therefore 6144 Walsh
% chips long. A 64 by 64 Hadamard matrix is used to generate the
% Walsh sequences. The c0+c1*2+c2*4+c3*8+c4*16+c5*32 -th row is the
% output where c0,c1,c2,c3,c4,c5 are the six interleaved
% convolutionally coded bits that are encoded.
%
% (c) Patrick Volz, 1996,97,98

% generate the Hadamard matrix (uses 1,-1 convention)
global H
if isempty(H)
   H=hadamard(64);
end

framein=reshape(framein,6,96); % reshape input in columns of 6 bits
frameout(1:96,:)=H([1 2 4 8 16 32]*(framein(:,1:96)<0)+1,:);
frameout=reshape(frameout',1,6144); % reshape to output size
```

B.9. mult.c - Scalar/Matrix Multiplication

```
/*

  Version 1.0, Copyright 1998, Patrick Volz

  Real matrix multiplication with a real scalar. Somewhat faster that
  the MATLAB multiplication for this special case.
```

```
  Usage: mult(matrix,scalar)

  matrix(x,y)      real x by y matrix
  scalar(1,1)      a real scalar

  No return value! The first input argument is modified in the MATLAB
  workspace!

*/

/* MEX support function declarations and prototypes   */
#include "mex.h"

void mexFunction(
      int nlhs, mxArray *plhs[],  /* # of, ptr on input arguments */
      int nrhs, const mxArray *prhs[])  /* # of, ptr on output args
*/
{
unsigned int i,M,N,L;

/* declare the function arguments (i.e. choose names for them here)*/
double *matrix;   /* input arguments */
double *scalar;

/* check number of arguments */
if (nlhs > 0) {
mexErrMsgTxt("No output arguments! Function modifies the first input
argument.");
} else if (nrhs > 2) {
      mexErrMsgTxt("Only 2 imput arguments allowed!");
} else if (nrhs < 1) {
      mexErrMsgTxt("A scalar must be specified as the second input
argument!");
}

/* check input arguments */
M=mxGetM(prhs[0]); /* get number of rows of 1st input argument */
N=mxGetN(prhs[0]); /* get number of columns of 1st input argument */
L=M*N;

if (!mxIsNumeric(prhs[0]) || mxIsComplex(prhs[0])
      || mxIsSparse(prhs[0]) || !mxIsDouble(prhs[0])) {
      mexErrMsgTxt("The first input argument must be a real non-
sparse matrix or scalar!");
}

if (!mxIsNumeric(prhs[1]) || mxIsComplex(prhs[1])
      || mxIsSparse(prhs[1]) || !mxIsDouble(prhs[1])
      || !((mxGetM(prhs[1]) == 1) && (mxGetN(prhs[1]) == 1))) {
      mexErrMsgTxt("The first input argument must be a real scalar");
}

/* dereference the arguments */
matrix=mxGetPr(prhs[0]);
scalar=mxGetPr(prhs[1]);

for (i=0;i<L;i++) matrix[i]*= *scalar; /* the multiplication */

}
```

B.10. tapgain.m - Correlated Fading Generator

```
function output=tapgain(fs,fm,M,time)

% output=tapgain(fs,fm,M,time)
% fs   : sampling frequency of tap-gains
% fm   : max. Doppler frequency
% M    : number of tapgains
% time : time duration to be covered
%
% Create time-varying complex tap-gains that are consistent with the
% maximum Doppler frequency fm, i.e., have the correct time-
% correlation properties. A commonly used Doppler spectrum model is
% used as the scattering function (see below). The required minimum
% sampling rate fs for the tap gains is only twice the maximum
% Doppler frequency.

L=ceil(time*fs);      % number of samples required

output=zeros(M,L);    % initialize ouput

omegad=2*fm/fs;       % normalized max. Doppler frequency

N=200/omegad;         % to get nice frequency resolution

if N<L
   N=L;
end

N=2^nextpow2(N);          % use a power of two to get faster FFT's
N1=floor(omegad*N/2)+1; % first frequency component greater fm

% Normalized frequency vector, square root of Doppler spectrum.
% This is normalized so that the integral over the scattering
% function is equal to one. The resulting tap-gain sequence
% has an average power of one.
x1=[0:2/N:omegad];
S1=sqrt((1/(pi*omegad))./sqrt(1-(x1/omegad).^2));
if max(x1)==omegad
   % we had a division by zero, replace infinite result by a linear
   % continuation of S1!
   S1(1,N1)=2*S1(1,N1-1)-S1(1,N1-2);
   fprintf(1,'Caught division by zero!\n');
end

for m=1:M      % for each tap gain

% generate a 1 by N complex Gaussian random vector
IC=randn(1,N)+i*randn(1,N); % mean=0, variance=1

SIC=fft(IC);   % take FFT

% set frequency components > fm to zero
SIC(1,N1:N-N1+1)=zeros(1,length(N1:N-N1+1));

% Apply square root of Doppler spectrum to frequency components <= fm
SIC(1,1:N1)=SIC(1,1:N1).*S1;
SIC(1,N-N1+2:N)=SIC(1,N-N1+2:N).*S1(1,N1:-1:2);
```

```
IC1=ifft(SIC);                % inverse Fourier transform

output(m,:)=IC1(1,1:L);       % reduce to desired length

% average power of 1
output(m,:)=output(m,:)/sqrt(mean(output(m,:).*conj(output(m,:))));

end
```

B.11. fadex.c -Fade Multipath Signal

```
/*
  Version 1.0, Copyright 1998, Patrick Volz

  fadex()

  mex-file implementation of the following MATLAB commands:

  rsignal=alphas(path)*isignal-sqrt(-1)*alphas(path)*qsignal;
  rsignal=rsignal.*reshape(repmat((Fading(path,:)),256,1),1,24576);
  risignal=real(rsignal);
  rqsignal=-imag(rsignal); ,

  where isignal(1,24576), qsignal(1,24576), Fading(1,96) (complex),
  alphas(path) scalar, risignal(1,24576), rqsignal(1,24576).

  Usage:


fadex(isignal,qsignal,realFading,imagFading,alpha,risignal,rqsignal)

  In : isignal(1,24576)     In-phase signal of frame data
       qsignal(1,24576)     Quadrature signal of frame data
       realFading(1,96)     Real part of complex Fading
       imagFading(1,96)     Imaginary part of complex Fading
       alpha(1,1)           Relative multipath strength (sum of alphas
                            squared =1)
       risignal(1,24576)    Real part of faded signal
       rqsignal(1,24576)    Imaginary part of faded signal

  Out: No output arguments. The mex-file changes the contents of the
       Variables risignal and rqsignal in the MATLAB workspace!

  This function can also be used to do the inverse fading for
  coherent detection by using the negative of the imaginary part of
  the complex fading!! (This use is obsolete due to fadex1c.c!

  (C) Patrick Volz, 1998

  File history:

  6/12/1998: WARNING!! Due to previous change the output arguments
             MUST be different than the input arguments isignal and
             qsignal in the MATLAB workspace!!
```

```
  6/5/1998:  Removed output arguments and added the output arguments
             to the input argument. Now the variable in the MATLAB
             workspace are manipulated instead of regenerating new
             output variables on every function call. About 50%
             faster.

  6/4/1998:  Begin programming and finish initial implementation.

*/

/* MEX support function declarations and prototypes   */
#include "mex.h"

/*  ----------------------   COMPUTATION   ----------------------- */
void computation( double risignal[], double rqsignal[],
      double isignal[], double qsignal[], double realFading[],
      double imagFading[], double *alpha)
{
int    i,j;              /* variables for loops */

/* ---------------------   computations   ------------------------ */

/* loop through all the elements of the fading variables */
for (i=0;i<96;i++,realFading++,imagFading++) {

   /* loop through the Walsh group elements */
   for (j=0;j<256;j++,risignal++,rqsignal++,isignal++,qsignal++) {

      /* do the math */
      *risignal=*alpha*(*isignal*    *realFading    +    *qsignal    *
*imagFading);
      *rqsignal=*alpha*(*qsignal*    *realFading    -    *isignal    *
*imagFading);

   }
}
}
/* --------------------- END OF COMPUTATION -------------------- */

/* ---------------   MEX FUNCTION GATEWAY ROUTINE   -------------- */

void mexFunction(
   int nlhs, mxArray *plhs[], /* # of, ptr on input arguments */
   int nrhs, const mxArray *prhs[]) /* # of, ptr on output arguments
*/
{
/* declare the function arguments (i.e. choose names for them here)*/
/* choose the same names in the computational subroutine, or not!  */
double *isignal;     /* input arguments */
double *qsignal;
double *realFading;
double *imagFading;
double *alpha;
double *risignal;
double *rqsignal;

/* check number of arguments */

if (nrhs > 7) {
   mexErrMsgTxt("Maximum of 7 input arguments is allowed!");
```

```
} else if (nrhs <7) {
   mexErrMsgTxt("7 input arguments required!");
} else if (nlhs > 0) {
   mexErrMsgTxt("There are no output arguments!");
}

/* check input arguments */

if (!mxIsNumeric(prhs[0]) || mxIsComplex(prhs[0])
   || mxIsSparse(prhs[0]) || !mxIsDouble(prhs[0])
   || !((mxGetM(prhs[0]) == 1) && (mxGetN(prhs[0]) == 24576))) {
   mexErrMsgTxt("The first input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[1]) || mxIsComplex(prhs[1])
   || mxIsSparse(prhs[1]) || !mxIsDouble(prhs[1])
   || !((mxGetM(prhs[1]) == 1) && (mxGetN(prhs[1]) == 24576))) {
   mexErrMsgTxt("The second input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[2]) || mxIsComplex(prhs[2])
   || mxIsSparse(prhs[2]) || !mxIsDouble(prhs[2])
   || !((mxGetM(prhs[2]) == 1) && (mxGetN(prhs[2]) == 96))) {
   mexErrMsgTxt("The third input argument must be a real 1 x 96
vector!");
}

if (!mxIsNumeric(prhs[3]) || mxIsComplex(prhs[3])
   || mxIsSparse(prhs[3]) || !mxIsDouble(prhs[3])
   || !((mxGetM(prhs[3]) == 1) && (mxGetN(prhs[3]) == 96))) {
   mexErrMsgTxt("The fourth input argument must be a real 1 x 96
vector!");
}

if (!mxIsNumeric(prhs[4]) || mxIsComplex(prhs[4])
   || mxIsSparse(prhs[4]) || !mxIsDouble(prhs[4])
   || !((mxGetM(prhs[4]) == 1) && (mxGetN(prhs[4]) == 1))) {
   mexErrMsgTxt("The fifth input argument must be a real scalar!");
}

if (!mxIsNumeric(prhs[5]) || mxIsComplex(prhs[5])
   || mxIsSparse(prhs[5]) || !mxIsDouble(prhs[5])
   || !((mxGetM(prhs[5]) == 1) && (mxGetN(prhs[5]) == 24576))) {
   mexErrMsgTxt("The sixth input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[6]) || mxIsComplex(prhs[6])
   || mxIsSparse(prhs[6]) || !mxIsDouble(prhs[6])
   || !((mxGetM(prhs[6]) == 1) && (mxGetN(prhs[6]) == 24576))) {
   mexErrMsgTxt("The seventh input argument must be a real 1 x 24576
vector!");
}

/* create matrices for output arguments */
/* no output arguments */

/* dereference the arguments and call computational routine */
```

```
isignal=mxGetPr(prhs[0]);
qsignal=mxGetPr(prhs[1]);
realFading=mxGetPr(prhs[2]);
imagFading=mxGetPr(prhs[3]);
alpha=mxGetPr(prhs[4]);

risignal=mxGetPr(prhs[5]);
rqsignal=mxGetPr(prhs[6]);

computation(risignal, rqsignal, isignal, qsignal, realFading,
            imagFading, alpha);
}
/* --------------------   END OF GATEWAY   ----------------------- */
```

B.12. fadex1c.c - Coherent Multipath Correlator

```
/*

  Version 1.0, Copyright 1998, Patrick Volz

  Coherent multipath combiner for fading signals.

  Usage:


fadex1c(isignal,qsignal,realFading,imagFading,alpha,risignal,rqsignal
)

  In : npaths            Number of multipath signals (1-5)
       alphas(1,5)       Relative multipath strengths (sum of
                         alphas^2=1)
       isignal(1,24576)  In-phase signal of frame data
       qsignal(1,24576)  Quadrature signal of frame data
       icode(1,24576)    In-phase spreading sequence
       qcode(1,24576)    Quadrature spreading sequence
       inoise(5,24576)   In-phase interference
       qnoise(5,24576)   Quadrature interference
       rFading(5,96)     Real part of complex Fading
       iFading(5,96)     Imaginary part of complex Fading
       walshcorr(96,64)  Walsh correlations (Result of this function)

  Out: No output arguments. The mex-file changes the contents of the
       variable walshcorr!

  (C) Patrick Volz, 1998

*/

#define TRUE 1
#define FALSE 0

#include <stdio.h>

/* MEX support function declarations and prototypes   */
#include "mex.h"
```

```
/* global variables */
static int   s_flag=FALSE;    /* static variables are not computed */
static int   HADAMARD[64][64];   /* Hadamard Matrix size 64 */

/* ----------------------   COMPUTATION   ---------------------- */
static void computation(double *npaths, double alphas[],
      double isignal[], double qsignal[], double icode[],
      double qcode[], double inoise[], double qnoise[],
      double rFading[], double iFading[], double walshcorr[])
{
int    i,j,k,l,m;              /* variables for loops */
double *rFade[5],*iFade[5];    /* point to fading for a Walsh group */
double *corr_ptr;              /* used as pointer into walshcorr */
double *alpha;                 /* used as pointer into alphas */
double richip,richip1,         /* used for computations */
       rqchip,rqchip1,
       wchip,wsymbol;

/* ----------   first compute all static variables   ------------- */
if (s_flag==FALSE) {

   s_flag=TRUE;

   /* compute the HADAMARD matrix, e.g., required for correlations */

   /* initialize the algorithm for the HADAMARD matrix */
   HADAMARD[0][0]=1;

   /* run the algorithm until HADAMARD is full */
   for (i=1;i<=32;i+=i)
      for (j=0;j<i;j++)
         for (m=0;m<i;m++) {
            HADAMARD[j][m+i]=HADAMARD[j][m];
            HADAMARD[j+i][m]=HADAMARD[j][m];
            HADAMARD[j+i][m+i]=-HADAMARD[j][m];
         }
}

/* ----------------------   computations   ---------------------- */

/* Loop through all elements of isignal, qsignal, icode, and      */
/* qcode by looping through all Walsh groups, Walsh symbols,      */
/* and chips per Walsh symbol.                                    */

/* Loop through elements of the fading variables (Walsh groups). */
for (j=0,corr_ptr=walshcorr;j<96;j++,walshcorr=corr_ptr) {

   /* Set rFade[] and iFade[]. */
   for (k=0;k<5;k++,rFading++,iFading++) {
      rFade[k]=rFading;
      iFade[k]=iFading;
   }

   /* Loop through the Walsh symbols of the Walsh groups. */
   for (k=0;k<64;k++) {

      /* Loop through the chips of each Walsh symbol. */
      for (l=0,wsymbol=0;l<4;l++, isignal++, qsignal++, icode++,
            qcode++) {
```

```
        /* For each multipath considered compute contribution */
        /* to received signal for that Walsh chip and add.    */
        for
(i=0,wchip=0,alpha=alphas;i<*npaths;i++,alpha++,inoise++,
              qnoise++) {

           /* Multiplication of signal with complex fading. */
           richip = *alpha * (*isignal * *rFade[i] + *qsignal *
                    *iFade[i]);
           rqchip = *alpha * (*qsignal * *rFade[i] - *isignal *
                    *iFade[i]);

           /* Add the noise. */
           richip += *inoise;
           rqchip += *qnoise;

           /* Multiply with conjugate complex fading. */
           richip1 = *alpha * (richip * *rFade[i] - rqchip *
                     *iFade[i]);
           rqchip1 = *alpha * (rqchip * *rFade[i] + richip *
                     *iFade[i]);

           /* Despread with short codes and add. */
           wchip += richip1 * *icode + rqchip1 * *qcode;

           /* If long code specified despread with long code.    */
           /* Not implemented yet since not used in simulations! */

        } /* Next multipath. */

        /* Bring noise and fading to start of next Walsh chip. */
        while (i<5) {
           inoise++;
           qnoise++;
           i++;
        }

        /* Add contribution of wchip to the Walsh symbol. */
        wsymbol+=wchip;

     } /* Next Walsh chip */

     /* Compute contribution of wsymbol to correlations of       */
     /* this Walsh group.                                        */
     /* IMPORTANT: Correlation variable supplied by MATLAB       */

     /* workspace must be initialized with zeros!                */

     /* Multiply value of wsymbol by 0.5. */
      wsymbol*=0.5;

     /* Use corr_ptr to be able to go back to beginning of  */
     /* Walsh group of output variable.                     */
     for (m=0,corr_ptr=walshcorr;m<64;m++,corr_ptr++) {

        *corr_ptr+=wsymbol*HADAMARD[k][m];

     } /* Next element of HADAMARD sequence. */
```

```
   } /* Next Walsh symbol. */

} /* Next Walsh group. */

}
/* -------------------- END OF COMPUTATION --------------------- */


/* --------------- MEX FUNCTION GATEWAY ROUTINE -------------- */
void mexFunction(
   int nlhs, mxArray *plhs[],     /* # of, ptr on input arguments */
   int nrhs, const mxArray *prhs[]) /* # of, ptr on output args */
{
/* declare the function arguments (i.e. choose names for them here)*/
double *npaths;   /* input arguments */
double *alphas;
double *isignal;
double *qsignal;
double *icode;
double *qcode;
double *inoise;
double *qnoise;
double *rFading;
double *iFading;
double *walshcorr;

/* check number of arguments */

if (nrhs != 11) {
   mexErrMsgTxt("11 input arguments required!");
} else if (nlhs > 0) {
   mexErrMsgTxt("There are no output arguments!");
}

/* check input arguments */

if (!mxIsNumeric(prhs[0]) || mxIsComplex(prhs[0])
   || mxIsSparse(prhs[0]) || !mxIsDouble(prhs[0])
   || !((mxGetM(prhs[0]) == 1) && (mxGetN(prhs[0]) == 1))) {
   mexErrMsgTxt("The 1st input argument must be a real scalar!");
}

if (!mxIsNumeric(prhs[1]) || mxIsComplex(prhs[1])
   || mxIsSparse(prhs[1]) || !mxIsDouble(prhs[1])
   || !((mxGetM(prhs[1]) == 1) && (mxGetN(prhs[1]) == 5))) {
   mexErrMsgTxt("The 2nd input argument must be a real 1 x 5
vector!");
}

if (!mxIsNumeric(prhs[2]) || mxIsComplex(prhs[2])
   || mxIsSparse(prhs[2]) || !mxIsDouble(prhs[2])
   || !((mxGetM(prhs[2]) == 1) && (mxGetN(prhs[2]) == 24576))) {
   mexErrMsgTxt("The 3rd input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[3]) || mxIsComplex(prhs[3])
   || mxIsSparse(prhs[3]) || !mxIsDouble(prhs[3])
   || !((mxGetM(prhs[3]) == 1) && (mxGetN(prhs[3]) == 24576))) {
```

```
   mexErrMsgTxt("The 4th input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[4]) || mxIsComplex(prhs[4])
   || mxIsSparse(prhs[4]) || !mxIsDouble(prhs[4])
   || !((mxGetM(prhs[4]) == 1) && (mxGetN(prhs[4]) == 24576))) {
   mexErrMsgTxt("The 5th input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[5]) || mxIsComplex(prhs[5])
   || mxIsSparse(prhs[5]) || !mxIsDouble(prhs[5])
   || !((mxGetM(prhs[5]) == 1) && (mxGetN(prhs[5]) == 24576))) {
   mexErrMsgTxt("The 6th input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[6]) || mxIsComplex(prhs[6])
   || mxIsSparse(prhs[6]) || !mxIsDouble(prhs[6])
   || !((mxGetM(prhs[6]) == 5) && (mxGetN(prhs[6]) == 24576))) {
   mexErrMsgTxt("The 7th input argument must be a real 5 x 24576
matrix!");
}

if (!mxIsNumeric(prhs[7]) || mxIsComplex(prhs[7])
   || mxIsSparse(prhs[7]) || !mxIsDouble(prhs[7])
   || !((mxGetM(prhs[7]) == 5) && (mxGetN(prhs[7]) == 24576))) {
   mexErrMsgTxt("The 8th input argument must be a real 5 x 24576
matrix!");
}

if (!mxIsNumeric(prhs[8]) || mxIsComplex(prhs[8])
   || mxIsSparse(prhs[8]) || !mxIsDouble(prhs[8])
   || !((mxGetM(prhs[8]) == 5) && (mxGetN(prhs[8]) == 96))) {
   mexErrMsgTxt("The 9th input argument must be a real 5 x 96
matrix!");
}

if (!mxIsNumeric(prhs[9]) || mxIsComplex(prhs[9])
   || mxIsSparse(prhs[9]) || !mxIsDouble(prhs[9])
   || !((mxGetM(prhs[9]) == 5) && (mxGetN(prhs[9]) == 96))) {
   mexErrMsgTxt("The 10th input argument must be a real 5 x 96
matrix!");
}

if (!mxIsNumeric(prhs[10]) || mxIsComplex(prhs[10])
   || mxIsSparse(prhs[10]) || !mxIsDouble(prhs[10])
   || !((mxGetM(prhs[10]) == 64) && (mxGetN(prhs[10]) == 96))) {
   mexErrMsgTxt("The 11th input argument must be a real 64 x 96
matrix!");
}

/* dereference the arguments and call computational routine */
npaths=mxGetPr(prhs[0]);
alphas=mxGetPr(prhs[1]);
isignal=mxGetPr(prhs[2]);
qsignal=mxGetPr(prhs[3]);
icode=mxGetPr(prhs[4]);
```

```
qcode=mxGetPr(prhs[5]);
inoise=mxGetPr(prhs[6]);
qnoise=mxGetPr(prhs[7]);
rFading=mxGetPr(prhs[8]);
iFading=mxGetPr(prhs[9]);
walshcorr=mxGetPr(prhs[10]);

computation(npaths,alphas,isignal,qsignal,icode,qcode,inoise,qnoise,
            rFading,iFading,walshcorr);
}
/*  -------------------   END OF GATEWAY   ----- ----------------  */
```

B.13. w_cdmod1.c - Coherent Despreader/Correlator

```
/*

  Version 1.0, Copyright 1998, Patrick Volz

  corr=w_cdmod1(iframe,qframe,icode,qcode,lcode)

  In : iframe(1,24576) In-phase component of frame
       qframe(1,24576) Quadrature component of frame
       icode(1,24576)  Inphase shortcode
       qcode(1,24576)  Quadrature shortcode
       lcode(1,24576)  User longcode (optional)
  Out: corr(96,64)     Walsh correlations

  Coherent Despreader and Walsh correlator.
  Calculates correlations of input frame with all Walsh sequences.
  Includes factor 0.5 of signal downconversion!

  For coherent detection we need to perform the following operations:
  - multiply in-phase by icode, quadrature by qcode
  - add in-phase and quadrature components
  - multiply by lcode
  - add up over groups of 4 consecutive bits
  - correlate with Walsh sequences

  File History:

  5/23/1998: Finished Testing of version 1.0
             Comparison with m-file indicates some numerical
             differences that are on the order of 10e-15!!
             Considered negligible!

*/

#define TRUE 1
#define FALSE 0

#include <stdio.h>

/* MEX support function declarations and prototypes. */
#include "mex.h"

/* global variables */
```

```
static int   s_flag=FALSE;    /* static variables are not computed */
static int   HADAMARD[64][64];   /* Hadamard Matrix size 64 */

/* -------------------------   COMPUTATION   --------------------- */
void computation( double corr[], double iframe[], double qframe[],
                  double icode[], double qcode[], double lcode[])
{
int    i,j,k,m;                /* variables for loops */
double framein[24576];
double wframe[6144];
double tmp_corr;
int tmp;

/*   ---------   First compute all static variables.   ---------- */
if (s_flag==FALSE) {

   s_flag=TRUE;

   /* compute the HADAMARD matrix, e.g., required for correlations */

   /* initialize the algorithm for the HADAMARD matrix */
   HADAMARD[0][0]=1;

   /* run the algorithm until HADAMARD is full */
   for (i=1;i<=32;i+=i)
      for (j=0;j<i;j++)
         for (m=0;m<i;m++) {
            HADAMARD[j][m+i]=HADAMARD[j][m];
            HADAMARD[j+i][m]=HADAMARD[j][m];
            HADAMARD[j+i][m+i]=-HADAMARD[j][m];
         }
}

/* --------------------   Compute correlations.   ----------------- */

/* Compute sum of de-spreaded in-phase and quadrature signals.     */

for (i=0;i<24576;i++) {

   framein[i]=iframe[i]*icode[i]+qframe[i]*qcode[i];

}

/* If specified, despread with the long code sequence. */
if (lcode != NULL) for (i=0;i<24576;i++) framein[i]*=lcode[i];

/* Sum signal over 4 consecutive chips. The factor 0.5 represents  */
/* downconversion loss!                                             */

for (i=0;i<6144;i++) {

   for (j=0,tmp=4*i,wframe[i]=0;j<4;j++) wframe[i]+=framein[j+tmp];
   wframe[i]*=0.5;

}

/* Compute correlations by correlating with the HADAMARD matrix. */
for (i=0;i<96;i++) {

   for (j=0,tmp=i*64;j<64;j++) {
```

```
      for (k=0,tmp_corr=0;k<64;k++)
         tmp_corr+=wframe[k+tmp]*HADAMARD[j][k];
      corr[i+j*96]=tmp_corr;

   }
}
}
/* ---------------------  END OF COMPUTATION  ---=---------------- */

/* -----------------   MEX FUNCTION GATEWAY ROUTINE   ------------ */
void mexFunction(
   int nlhs, mxArray *plhs[],     /* # of, ptr on input arguments */
   int nrhs, const mxArray *prhs[]) /* # of, ptr on output args */
{
/* Declare function arguments (i.e. choose names for them here). */
double *iframe;          /* in-phase coherent received signal */
double *qframe;          /* quadrature coherent received signal */
double *icode;           /* in-phase short code */
double *qcode;           /* quadrature short code */
double *lcode;           /* long code */

double *corr;            /* output arguments */

/* Check number of arguments. */
if (nrhs > 5) {
   mexErrMsgTxt("Maximum of 5 input arguments is allowed!");
} else if (nrhs<4) {
   mexErrMsgTxt("At least 4 input arguments required!");
} else if (nlhs > 1) {
   mexErrMsgTxt("Only 1 output arguments allowed!");
}

/* Check input arguments. */

if (!mxIsNumeric(prhs[0]) || mxIsComplex(prhs[0])
   || mxIsSparse(prhs[0]) || !mxIsDouble(prhs[0])
   || !((mxGetM(prhs[0]) == 1) && (mxGetN(prhs[0]) == 24576))) {
   mexErrMsgTxt("The 1st input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[1]) || mxIsComplex(prhs[1])
   || mxIsSparse(prhs[1]) || !mxIsDouble(prhs[1])
   || !((mxGetM(prhs[1]) == 1) && (mxGetN(prhs[1]) == 24576))) {
   mexErrMsgTxt("The 2nd input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[2]) || mxIsComplex(prhs[2])
   || mxIsSparse(prhs[2]) || !mxIsDouble(prhs[2])
   || !((mxGetM(prhs[2]) == 1) && (mxGetN(prhs[2]) == 24576))) {
   mexErrMsgTxt("The 3rd input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[3]) || mxIsComplex(prhs[3])
   || mxIsSparse(prhs[3]) || !mxIsDouble(prhs[3])
   || !((mxGetM(prhs[3]) == 1) && (mxGetN(prhs[3]) == 24576))) {
```

```
   mexErrMsgTxt("The 4th input argument must be a real 1 x 24576
vector!");
}

if (nrhs>4) {
   if (!mxIsNumeric(prhs[4]) || mxIsComplex(prhs[4])
      || mxIsSparse(prhs[4]) || !mxIsDouble(prhs[4])
      || !((mxGetM(prhs[4]) == 1) && (mxGetN(prhs[4]) == 24576))) {
      mexErrMsgTxt("The 5th input argument must be a real 1 x 24576
vector!");
   }
}

/* Create matrices for output arguments. */
plhs[0]=mxCreateDoubleMatrix(96, 64, mxREAL);

/* Dereference the arguments and call computational routine. */
lcode=NULL; /* since optional */

corr=mxGetPr(plhs[0]);

iframe=mxGetPr(prhs[0]);
qframe=mxGetPr(prhs[1]);
icode=mxGetPr(prhs[2]);
qcode=mxGetPr(prhs[3]);
if (nrhs>4) lcode=mxGetPr(prhs[4]);

computation(corr,iframe,qframe,icode,qcode,lcode);
}
/* ----------------------   END OF GATEWAY   --------------------- */
```

B.14. w_ncdmo1.c - Noncoherent Despreader/Correlator

```
/*

  Version 1.0, Copyright 1998, Patrick Volz

  corr=w_ncdmo1(iframe,qframe,icode,qcode,lcode)

  In : iframe(1,24576) In-phase component of frame data
       qframe(1,24576) Quadrature component of frame data
       icode(1,24576) In-phase shortcode
       qcode(1,24576) Quadrature shortcode
       lcode(1,24576) User longcode (optional)
  Out: corr(96,64)    Walsh correlations

  Non-coherent De-spreader and Walsh Correlator.
  Calculates correlations of input frame with all Walsh sequences.
  Incorporates the factor of 0.25 due to downconversion!

  (C) Patrick Volz, 1998

  File history:

  5/23/1998: Finished testing of version 1.0
             Comparison with m-file indicates some numerical
```

```
            differences that are on the order of 10e-12!
            Considered negligible!


  For noncoherent detection we need to perform the following
  operations:
  - multiply (isignal+j*qsignal) by (icode-j*qcode)
  - multiply by lcode
  - add up over groups of 4 consecutive bits
  - correlate real and imaginary parts with Walsh sequences
    individually
  - square and add results

*/

#define TRUE 1
#define FALSE 0

#include <stdio.h>

/* MEX support function declarations and prototypes. */
#include "mex.h"

/* global variables */
static int  s_flag=FALSE;  /* Static variables are not computed. */
static int  HADAMARD[64][64]; /* Hadamard Matrix size 64. */

/* --------------------------   COMPUTATION  ------------------------
*/
void computation(double corr[], double iframe[], double qframe[],
                 double icode[], double qcode[], double lcode[])
{
int    i,j,k,m;          /* variables for loops */
double rp[24576];
double ip[24576];
double framein[24576];
double rwframe[6144];
double iwframe[6144];
double tmp_corr1;
double tmp_corr2;
int tmp;

/* -----------   First compute all static variables.  ------------ */
if (s_flag==FALSE) {

   s_flag=TRUE;

   /* Compute HADAMARD matrix, e.g., required for correlations. */

   /* Initialize the algorithm for the HADAMARD matrix. */
   HADAMARD[0][0]=1;

   /* Run the algorithm until HADAMARD is full. */
   for (i=1;i<=32;i+=i)
      for (j=0;j<i;j++)
         for (m=0;m<i;m++) {
            HADAMARD[j][m+i]=HADAMARD[j][m];
            HADAMARD[j+i][m]=HADAMARD[j][m];
            HADAMARD[j+i][m+i]=-HADAMARD[j][m];
         }
```

```
}

/* ------------------  Compute correlations.  -------------------- */

/* Compute real and imaginary parts of non-coherent cross          */
/* combination/despreading of received in-phase and quadrature     */
/* signals.                                                       */
for (i=0;i<24576;i++) {

   rp[i]=iframe[i]*icode[i]+qframe[i]*qcode[i];
   ip[i]=-iframe[i]*qcode[i]+qframe[i]*icode[i];

}

/* If long code is given, use it to despread the signals. */
if (lcode != NULL)
   for (i=0;i<24576;i++) {
      rp[i]*=lcode[i];
      ip[i]*=lcode[i];
   }


/* Sum in-phase and quadrature signals over 4 consecutive chips. */
for (i=0;i<6144;i++) {
   for (j=0,tmp=4*i,rwframe[i]=0,iwframe[i]=0;j<4;j++) {
      rwframe[i]+=rp[j+tmp];
      iwframe[i]+=ip[j+tmp];
   }
}

/* Compute correlations by correlating the in-phase and           */
/* quadrature signals with HADAMARD matrix and forming the sum    */
/* of the squares. The 0.0625 factor represents downconversion    */
/* loss.                                                          */
for (i=0;i<96;i++) {
   for (j=0,tmp=i*64;j<64;j++) {
      for (k=0,tmp_corr1=0,tmp_corr2=0;k<64;k++) {
         tmp_corr1+=rwframe[k+tmp]*HADAMARD[j][k];
         tmp_corr2+=iwframe[k+tmp]*HADAMARD[j][k];
      }
      corr[i+j*96]=0.0625*(tmp_corr1*tmp_corr1+tmp_corr2*tmp_corr2);
   }
}
}
/* ---------------------  END OF COMPUTATION  -------------------- */

/* -------------   MEX FUNCTION GATEWAY ROUTINE   --------------  */

void mexFunction(
   int nlhs, mxArray *plhs[], /* # of, ptr on input arguments */
   int nrhs, const mxArray *prhs[]) /* # of, ptr on output args */
{
/* declare function arguments (i.e. choose names for them here)  */
double *iframe;       /* in-phase coherent received signal */
double *qframe;       /* quadrature coherent received signal */
double *icode;        /* in-phase short code */
double *qcode;        /* quadrature short code */
double *lcode;        /* long code */

double *corr;         /* output arguments */
```

```
/* Check number of arguments. */
if (nrhs > 5) {
   mexErrMsgTxt("Maximum of 5 input arguments is allowed!");
} else if (nrhs<4) {
   mexErrMsgTxt("4 input arguments are required!");
} else if (nlhs > 1) {
   mexErrMsgTxt("Only 1 output arguments allowed!");
}

/* Check input arguments. */

if (!mxIsNumeric(prhs[0]) || mxIsComplex(prhs[0])
   || mxIsSparse(prhs[0]) || !mxIsDouble(prhs[0])
   || !((mxGetM(prhs[0]) == 1) && (mxGetN(prhs[0]) == 24576))) {
   mexErrMsgTxt("The 1st input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[1]) || mxIsComplex(prhs[1])
   || mxIsSparse(prhs[1]) || !mxIsDouble(prhs[1])
   || !((mxGetM(prhs[1]) == 1) && (mxGetN(prhs[1]) == 24576))) {
   mexErrMsgTxt("The 2nd input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[2]) || mxIsComplex(prhs[2])
   || mxIsSparse(prhs[2]) || !mxIsDouble(prhs[2])
   || !((mxGetM(prhs[2]) == 1) && (mxGetN(prhs[2]) == 24576))) {
   mexErrMsgTxt("The 3rd input argument must be a real 1 x 24576
vector!");
}

if (!mxIsNumeric(prhs[3]) || mxIsComplex(prhs[3])
   || mxIsSparse(prhs[3]) || !mxIsDouble(prhs[3])
   || !((mxGetM(prhs[3]) == 1) && (mxGetN(prhs[3]) == 24576))) {
   mexErrMsgTxt("The 4th input argument must be a real 1 x 24576
vector!");
}

if (nrhs>4) {
   if (!mxIsNumeric(prhs[4]) || mxIsComplex(prhs[4])
      || mxIsSparse(prhs[4]) || !mxIsDouble(prhs[4])
      || !((mxGetM(prhs[4]) == 1) && (mxGetN(prhs[4]) == 24576))) {
      mexErrMsgTxt("The 5th input argument must be a real 1 x 24576
vector!");
   }
}

/* Create matrices for output arguments. */
plhs[0]=mxCreateDoubleMatrix(96, 64, mxREAL);

/* Dereference the arguments and call computational routine. */
lcode=NULL; /* since optional */

corr=mxGetPr(plhs[0]);

iframe=mxGetPr(prhs[0]);
qframe=mxGetPr(prhs[1]);
icode=mxGetPr(prhs[2]);
```

```
qcode=mxGetPr(prhs[3]);
if (nrhs>4) lcode=mxGetPr(prhs[4]);

computation(corr,iframe,qframe,icode,qcode,lcode);
}
/* --------------------   END OF GATEWAY   ---------------------- */
```

B.15. df_decode.c - Decision Feedback Decoder

```
/*

  Version 1.0, Copyright 1997, Patrick Volz

  Decision Feedback Decoder
  Uses soft decoding metrics based on the correlation values of an
  IS-95 frame (dual-maxima metric rule) and the Viterbi algorithm.
  The deinterleaving operation is incorporated in the decoder.
  If decision feedback is used, the allowed Walsh codes for each
  Walsh group are updated after each chain-back operation. Then the
  metrics for the Viterbi decoder are updated accordingly if
  necessary.

  The interleaver can be specified by input arguments:
  intlv([0:575]) and dintlv([0:575]) of the desired interleaver need
  to be passed to the decoder on the first call (will be put in
  static variables)!
  Note: In order to use change interleavers the decoder function has
  to be cleared from the MATLAB memory! If no interleaver information
  is supplied the decoder defaults to the IS-95 Uplink interleaver
  for the 9.6 kbit/s data rate.

  nstates is used to specify number of path histories that have to
  match (minus one) in order to make a bit decision. This is used for
  'earlier decisions'. If not specified the value 255 is used which
  corresponds to all path histories.

  Usage:

  [frameout timeout df_count]=
   df_decode(walshcorr,decode_option,df_flag, intlv, dintlv, nstates)

  walshcorr(96,64)   Walsh correlation values
  decode_option      1(default) for signed correlations,
                     2 for correlation magnitudes (not used)
  df_flag            0(default) no decision feedback,
                     1 decision feedback
  intlv              result of intlv([0:575]) of interleaver
  dintlv             result of dintlv([0:575]) of deinterleaver
  nstates            number of matching path histories required for a
                     bit decision (default=255)

  Outputs are the decoded frame, the output time(-steps) of the data
  bits, and the DF count. The DF count indicates how many decoding
  metrics had no prior DF update, 1 DF update, and so on, up to 5 DF
  updates.
```

```
*/

#define TRUE 1
#define FALSE 0

#include <stdio.h>

/* MEX support function declarations and prototypes. */
#include "mex.h"

/* Global variables. */
static int   s_flag=FALSE;     /* static variables are not computed */
static int   E_OUTPUTS[256][8];  /* encoder outputs */
static int   WCN_BITS[64][6]; /* the six bits for each Walsh code  */
static int   CSN_WGN[576];       /* code symbol number to Walsh group
                                    number */
static int   CSN_WBN[576];       /* code symbol number to Walsh bit
                                    number */
static int   BN_CSN[192][3];  /* bit number to code symbol number */
static int   WBN_USEWC1[6][64];  /* Walsh bit number to Walsh code
                                    for bit value 1 */
static int   WBN_USEWC2[6][64];  /* Walsh bit number to Walsh code
                                    for bit value -1 */
static int   WGN_CSN[96][6];/* Walsh group to code symbol numbers */
static int   ICSN_CSN[576]; /* interleaved code symbol number to code
                                 symbol number */
static int   CSN_ICSN[576];   /* code symbol number to interleaved
                                 code symbol number */

int USE_CORR[96][64];   /* for decision feedback, allowed Walsh
                           codes for each Walsh group */
int walshs1[96][6];     /* Walsh code with max. corr., bit value 1 */
int walshs2[96][6];     /* bit value -1 */
double walshsv1[96][6]; /* corresponding correlation values */
double walshsv2[96][6];
double walshc[96][64];  /* copy input correlations to walshc */

/* Metric computation */
double b_metric(int *bitnumber, int *oldstate)
{
   /* Compute branch metric for an '1' input. */
   /* Metric for '-1' is negative of this.    */

   static char i;
   static int wgn, wbn, symbolnumber;
   static double result, bv;

   for (result=0,i=0;i<3;i++) {
      symbolnumber=BN_CSN[*bitnumber][i];
      bv=(double) E_OUTPUTS[*oldstate][i];
      wgn=CSN_WGN[symbolnumber];
      wbn=CSN_WBN[symbolnumber];

      /* Dual-maxima soft-decision decoding metric. */
      result+=bv*(walshsv1[wgn][wbn]-walshsv2[wgn][wbn]);
   }
   return(result);
}

/* ----------------------   COMPUTATION   ---------------------- */
```

```
void   computation(double   frameout[],   double   timeout[],   double
df_count[], double walshcorr[], double *option1, double *option2,
double *option3, double *option4, double *option5)
{
int    i,j,n,m,p,q;                /* Variables for loops. */
unsigned int k,k1,k2,k3,k4;        /* Variables for state loops. */

double S_METRIC1[256],S_METRIC2[256];  /* State metrics. */
double *smetric1, *smetric2, *smetric3; /* Ptrs in state metrics.  */
double *smetric_toggle;     /* Store metric pointer to switch between
                               S_METRIC1 and S_METRIC2. */
double sum1,sum2,sum3,sum4;    /* Used when updating state metrics. */
double B_METRIC1[256],B_METRIC2[256];  /* Branch metrics. */
double *bmetric1,*bmetric2;        /* Pointers into branch metrics. */

int B_HISTORY1[256][192];  /* Store bit history for each state. */
int B_HISTORY2[256][192];  /* Store bit history for each state
                               (need 2 copies). */
int *bh1[256],*bh2[256],*bh_toggle; /* Used to switch between
                                       B_HISTORY1 and B_HISTORY2. */

int PATH[256];           /* Store current path origins. */

int bctr=0;              /* Output bit counter. */
double wcv1,wcv2;
int wc1,wc2;
int decode_option=1;
int df_flag=FALSE;
int e_output[3];         /* Store encoder outputs. */
int e_state=0;     /* Store encoder state, begin in zero state. */

int match=FALSE; /* TRUE if required # of bit histories match. */
int nstates=255; /* Default value of req. matching bit histories. */
int nmatch;      /* To count matching bit histories. */

int DF_CHECK[96][6]; /* To count decision feedback metric updates. */
int dfc1,dfc2;

if (option1!=NULL) decode_option= (int) *option1;
if (option2!=NULL) df_flag= (int) *option2;
if (option5!=NULL) nstates = (int) *option5;

/* ----------   First compute all static variables.   ----------- */
if (s_flag==FALSE) {

   s_flag=TRUE;

/*
  First compute the possible encoder outputs for this code.
  The states of the encoder are numbered from 0 to 255. The state
  number corresponds to the 8 leftmost bits in the encoder shift
  register. The 9th bit is discarded when a new input bit is
  shifted in:

  shift register, after new bit is shifted in:

  | current input | b7 | b6 | b5 | b4 | b3 | b2 | b1 | ->   b0:
discarded

  The old state number in binary is :  b7   b6 b5 b4 b3 b2 b1 b0
```

The new state number in binary is : input b7 b6 b5 b4 b3 b2 b1

Thus, if the new input is '1' (corresponds to binary '0'), the new state number is the old state divided by two, and if the new input is '-1' the new state number is the old state number divided by two plus 128.

The encoder output for the '1' input is computed according to the generator polynomials of the convolutional code. Since the input bit is used in the computation of each output symbol, the output for the '-1' input is simply the negative of that. The encoder outputs are stored in positions 0, 1, and 2.

The new state for an '1' and '-1' input are stored at positions 3 and 4 of E_OUTPUTS so that they can be looked up when needed.

The expected input bit and the two originating states for each state are stored in positions 5, 6, and 7.

```
*/

  /* for all states (numbered 0 ... 255) calculate elements of
     E_OUTPUTS */
  for (k=0;k<=255;k++) {

     /* initialize output to all ones */
     for (i=0;i<=2;i++) E_OUTPUTS[k][i]=1;

     /* use state and '1' (one) to determine actual outputs */
     /* first output: generator mask 1 0 1 1 0 1 1 1 1 */
     /* check bits 1,2,3,4,6,7 and toggle output if set */
     if (k & 0x01) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
     if (k & 0x02) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
     if (k & 0x04) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
     if (k & 0x08) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
     if (k & 0x20) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];
     if (k & 0x40) E_OUTPUTS[k][0]= -E_OUTPUTS[k][0];

     /* second output: generator mask 1 1 0 1 1 0 0 1 1 */
     /* check bits 1,2,5,6,8 and toggle output if set */
     if (k & 0x01) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
     if (k & 0x02) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
     if (k & 0x10) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
     if (k & 0x20) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];
     if (k & 0x80) E_OUTPUTS[k][1]= -E_OUTPUTS[k][1];

     /* second output: generator mask 1 1 1 0 0 1 0 0 1 */
     /* check bits 1,4,7,8 and toggle output if set */
     if (k & 0x01) E_OUTPUTS[k][2]= -E_OUTPUTS[k][2];
     if (k & 0x08) E_OUTPUTS[k][2]= -E_OUTPUTS[k][2];
     if (k & 0x40) E_OUTPUTS[k][2]= -E_OUTPUTS[k][2];
     if (k & 0x80) E_OUTPUTS[k][2]= -E_OUTPUTS[k][2];

     /* Add the next state for a '1' and '-1' input in position 3
        and 4. For a '1' input the next state will be the current
        state shifted one bit to the right and for a '-1' input the
        next state will be the current state shifted one bit to the
        right + 128. */
     k1=(k >> 1);       /* k1 = k/2        */
     k2=(k1 | 0x80);    /* k1 = k/2 + 128 */
```

```
   E_OUTPUTS[k][3]=k1;
   E_OUTPUTS[k][4]=k2;

   /* Add the expected input bit in order to reach this state in
      position 5. */
   if (k<128) E_OUTPUTS[k][5]=1;
   else E_OUTPUTS[k][5]=-1;

   /* Add the two originating states at positions 6 and 7. */
   k1=(k << 1);    /* k1 = 2*k   */
   k2=k1+1;        /* k2 = 2*k+1 */
   k3=(k | 0x80); /* k3 = k+128 */
   if (k<128) {
      E_OUTPUTS[k][6]=k1;
      E_OUTPUTS[k3][6]=k1;
      E_OUTPUTS[k][7]=k2;
      E_OUTPUTS[k3][7]=k2;
   }
}     /* E_OUTPUTS is now computed. */

/* Compute WCN_BITS to lookup the 6 bits corresponding to a Walsh
   code. */
for (k=0;k<64;k++) {
   for (i=0;i<6;i++) WCN_BITS[k][i]=1;
   if (k & 0x01) WCN_BITS[k][0]=-1;
   if (k & 0x02) WCN_BITS[k][1]=-1;
   if (k & 0x04) WCN_BITS[k][2]=-1;
   if (k & 0x08) WCN_BITS[k][3]=-1;
   if (k & 0x10) WCN_BITS[k][4]=-1;
   if (k & 0x20) WCN_BITS[k][5]=-1;
}

/* Check if (de)interleaver specs supplied. If so, use them. */
if (option3!=NULL) {
   for (k=0;k<576;k++) {
      ICSN_CSN[k]= (int) option3[k];
   }
   printf("Using specified deinterleaving vector!\n");
}

if (option4!=NULL) {
   for (k=0;k<576;k++) {
      CSN_ICSN[k]= (int) option4[k];
   }
   printf("Using specified interleaving vector!\n");
}

/* Compute code symbol # to Walsh group and Walsh bit lookup. */
for (k=0;k<576;k++) {

   if (option3!=0) { /* Use supplied interleaving information. */

      CSN_WGN[k]=CSN_ICSN[k]/6;
      CSN_WBN[k]=CSN_ICSN[k]-CSN_WGN[k]*6;

   } else {          /* Use IS-95 uplink 9600 interleaver. */

      k1=k/32;          /* This is the interleaving operation. */
      k2=k-32*k1;       /*                                     */
      k2=k2*18+k1;      /*                                     */
```

```
            CSN_WGN[k]=k2/6;
            CSN_WBN[k]=k2-CSN_WGN[k]*6;
         }
      }

      /* Compute bit number to code symbol number lookup. */
      for (i=0;i<192;i++) {
         j=3*i;
         BN_CSN[i][0]=j;
         BN_CSN[i][1]=j+1;
         BN_CSN[i][2]=j+2;
      }

      /* Calculate Walsh bit number to poss. Walsh code lookups. */
      for (i=0;i<6;i++) {
         for (j=0;j<64;j++) {
            if (WCN_BITS[j][i]==1) {
               WBN_USEWC1[i][j]=1;
               WBN_USEWC2[i][j]=0;
            } else {
               WBN_USEWC1[i][j]=0;
               WBN_USEWC2[i][j]=1;
            }
         }
      }

      /* Calculate Walsh group number to code symbol number lookup. */
      for (i=0;i<96;i++) {
         k1=i*6;
         for (j=0;j<6;j++) {

            if (option4!=0) { /* Use interleaving information. */

               k2=k1+j; /* interleaved CSN */
               WGN_CSN[i][j]=ICSN_CSN[k2];

            } else {    /* Use IS-95 uplink 9600 interleaver. */

               k2=k1+j;     /* Interleaved CSN   */
               k3=k2/18;       /* Deinterleaving */
               k4=k2-18*k3;    /*                */
               k4=k4*32+k3;    /*                */
               WGN_CSN[i][j]=k4;
            }
         }
      }
   }

   /* ---------------------   Decoding Part   ----------------------- */

   /* Copy walshcorr to walshc in order not to modify MATLAB workspace.
   */
   /* If decode_option is 2, use magnitudes.                          */
   for (j=0;j<64;j++) {
      for (i=0;i<96;i++) {
         walshc[i][j]=*walshcorr++;
         if ((walshc[i][j]<0) && (decode_option==2)) walshc[i][j]*=-1;
      }
   }
```

```
/* For all Walsh groups and Walsh bits find the Walsh code with the
 max. correlation value for bit values 1 and -1, and store in
 walshs1, walshs2. */
for (i=0;i<96;i++) {        /* Walsh group number */
   for (j=0;j<6;j++) {      /* Walsh bit number */

      DF_CHECK[i][j]=0;     /* Initialize DF_CHECK[][] to zero. */

      wcv1=wcv2=-100000.0;
      for (k=0;k<64;k++) { /* Walsh code number */
         if (WBN_USEWC1[j][k]==1) {
            /* Walsh code k is possible for bitvalue 1. */
            if (walshc[i][k]>wcv1) {
               wcv1=walshc[i][k];
               wc1=k;
            }
         } else {
            /* Walsh code k is possible for bitvalue -1. */
            if (walshc[i][k]>wcv2) {
               wcv2=walshc[i][k];
               wc2=k;
            }
         }
      }
      walshs1[i][j]=wc1;
      walshs2[i][j]=wc2;
      walshsv1[i][j]=wcv1;
      walshsv2[i][j]=wcv2;
   }
}

/* Initialize USE_CORR if we use decision feedback. */
if (df_flag==TRUE) {
   for (i=0;i<96;i++) {
      for (j=0;j<64;j++) {
         USE_CORR[i][j]=1;
      }
   }
}

/* Initialize pointers to state metrics. */
smetric1=S_METRIC1;
smetric2=S_METRIC2;

/* First time we use S_METRIC1 to read and S_METRIC2 to write. */
smetric_toggle=S_METRIC1;

/* Initialize S_METRIC1 in zero state. */
for (i=1, *smetric1++=0;i<256;i++, smetric1++) *smetric1=-100000;
smetric1=S_METRIC1;

/* Initialize pointers to BIT_HISTORY and C_HISTORY. */
for (i=0;i<256;i++) {
   bh1[i]=B_HISTORY1[i];
   bh2[i]=B_HISTORY2[i];
}

/* First time we use B_HISTORY1 to read and B_HISTORY2 to write. */
bh_toggle=B_HISTORY1;
```

```
/* For all groups of 3 input symbols - do decoding step. */
for (n=0;n<192;n++) {

   /* Calculate all the 256 branch metrics for '1' and '-1' input. */
   for (k=0,bmetric1=B_METRIC1,bmetric2=B_METRIC2;k<256;k++) {
      *bmetric1=b_metric(&n,&k);
      *bmetric2++ = -*bmetric1++;
   }

   /* Now we have all the branch metrics for the current channel
      output. The next step is the add, compare, select operation
      that implements the Viterbi algorithm. */

   /* Set smetric3 128 positions into smetric2 to store results for
      -1 input. */
   smetric3=smetric2+128;

   /* Set pointers to B_METRIC1 and B_METRIC2. */
   bmetric1=B_METRIC1;
   bmetric2=B_METRIC2;

   for (k=0;k<=127;k++) {

      /* Variables for repeated use in this loop. */
      k1=(k << 1);    /* k1 = 2*k    */
      k2=k1+1;        /* k2 = 2*k+1 */
      k3=(k | 0x80); /* k3 = k+128 */

      /* "1" input: compare state 2*k and 2*k+1, transition to k. */
      /* "-1" input: compare state 2*k and 2*k+1, transition to   */
      /* k+128.                                                   */

      sum1=*smetric1+*bmetric1;
      sum3=*smetric1+*bmetric2;
      smetric1++;
      bmetric1++;
      bmetric2++;
      sum2=*smetric1+*bmetric1;
      sum4=*smetric1+*bmetric2;
      smetric1++;
      bmetric1++;
      bmetric2++;

      if (sum1>sum2) {

         /* This path wins. If equal, we arbitrarily let the other
            path win. */
         PATH[k]=k1;
         *smetric2++=sum1;

      } else {

         /* Other path. */
         PATH[k]=k2;
         *smetric2++=sum2;

      }

      bh2[k][n]=1; /* Set bit history. */
```

```
      if (sum3>sum4) {

         /* This path wins. If equal, we arbitrarily let the other
            path win. */
         PATH[k3]=k1;
         *smetric3++=sum3;

      } else {

         /* Other path. */
         PATH[k3]=k2;
         *smetric3++=sum4;

      }

      bh2[k3][n]=-1; /* Set bit history */

   }  /* Finished state metric updates. */

   /* Setup metric pointers for next round. */
   if (smetric_toggle == S_METRIC1) {
      smetric_toggle=S_METRIC2;
      smetric1=S_METRIC2;
      smetric2=S_METRIC1;
   } else {
      smetric_toggle=S_METRIC1;
      smetric1=S_METRIC1;
      smetric2=S_METRIC2;
   }

   /* Re-arrange bit histories. */
   for (j=0;j<256;j++) {
      for (k=bctr;k<n;k++) {
         bh2[j][k]=bh1[PATH[j]][k];
      }
   }

   /* Setup bit history pointers for next round. */
   if (bh_toggle == B_HISTORY1) {
      bh_toggle=B_HISTORY2;
      for (j=0;j<256;j++) {
         bh1[j]=B_HISTORY2[j];
         bh2[j]=B_HISTORY1[j];
      }
   } else {
      bh_toggle=B_HISTORY1;
      for (j=0;j<256;j++) {
         bh1[j]=B_HISTORY1[j];
         bh2[j]=B_HISTORY2[j];
      }
   }

   /* Check if we can output any bits at this time. */

   for (k=bctr;k<n;k++) {
      match=FALSE;
      nmatch=0;

      /* Look for maximum state metric. Current state metric is
         smetric1. Use smetric3 as temporary pointer. */
```

```
      for  (smetric3=smetric1,sum1=-100000,  j=0;j<256;j++,smetric3++)
{
         if (*smetric3>sum1) {
            sum1=*smetric3;
            k1=j;
         }
      }
      /* Max. state metric = sum1. */

      p = bh1[k1][k];
      for (j=0;j<256;j++) {
         if (p == bh1[j][k]) {
            nmatch++;
            if (nmatch>nstates) {
               match=TRUE;
               break;
            }
         }
      }
      if (match==FALSE) {
         break;
      }
      else {
         frameout[k]=p;
         bctr++;
         if (timeout!=NULL) timeout[k]=n;

         /* In case of decision feedback, update USE_CORR. */
         if (df_flag==TRUE) {

         /* DF effectiveness check patched into following section */
         /* Goal: determine how many metrics during frame decoding */
         /*       have no prior DF update, 1 DF update, ..., 5 DF  */
         /*       updates.                                         */

         /* Here we check for the first Walsh bit in the current   */
         /* Walsh group for that the corresponding convolutionally */
         /* coded bit number is greater or equal than the one to   */
         /* be considered during the next decoding step. Once      */
         /* this Walsh bit is identified the DF_CHECK[][] for this */
         /* Walsh group is incremented for that Walsh bit and all  */
         /* remaining Walsh bits.                                  */

         /* We just finished decoding step n.                      */
         /* We just decided on bit number k.                       */

            /* Get the encoder outputs for the bit decision. */
            if (frameout[k]==1) {
               for (m=0;m<3;m++) {
                  e_output[m]=E_OUTPUTS[e_state][m];
               }
               e_state=E_OUTPUTS[e_state][3];
            } else {
               for (m=0;m<3;m++) {
                  e_output[m]=-E_OUTPUTS[e_state][m];
               }
               e_state=E_OUTPUTS[e_state][4];
            }
```

```
for (m=0;m<3;m++) { /* for each of the 3 code symbols */
   k1=CSN_WGN[BN_CSN[k][m]]; /* Walsh group number */
   k2=CSN_WBN[BN_CSN[k][m]]; /* Walsh bit number */

   if (df_count!=NULL) {
      dfc1=k2+1; /* Next Walsh bit in Walsh group. */
      while(dfc1<6) {
      /* WGN_CSN[k1][dfc1] is the next CSN in this WG. */
         if (WGN_CSN[k1][dfc1]>=(3*(n+1))) {
         /* DF effective for this Walsh bit and all   */
         /* the following=> for all remaining Walsh   */
         /* bits in this Walsh group:                 */
            for(dfc2=dfc1;dfc2<6;dfc2++)
               DF_CHECK[k1][dfc2]++;
            break;
         } else {
         dfc1++;
         }
      }
   }

   if (e_output[m]==1) {
      for (p=0;p<64;p++) {
         if (WBN_USEWC1[k2][p]==0) {
            USE_CORR[k1][p]=0;
         }
      }
   } else   {
      for (p=0;p<64;p++) {
         if (WBN_USEWC2[k2][p]==0) {
            USE_CORR[k1][p]=0;
         }
      }
   }

   /* Update walshs1, walshs2, walshsv1, and walshsv2
      for this Walsh group. */
   for (p=0;p<6;p++) {
      wcv1=wcv2=-100000.0;
      for (q=0;q<64;q++) { /* Walsh code number */
         if (USE_CORR[k1][q]==1) {
            if (WBN_USEWC1[p][q]==1) {
            /* Walsh code k is possible for bitvalue 1 */
               if (walshc[k1][q]>wcv1) {
                  wcv1=walshc[k1][q];
                  wc1=q;
               }
            } else {
            /* Walsh code k possible for bitvalue -1 */
               if (walshc[k1][q]>wcv2) {
                  wcv2=walshc[k1][q];
                  wc2=q;
               }
            }
         }
      }
      walshs1[k1][p]=wc1;
      walshs2[k1][p]=wc2;
      walshsv1[k1][p]=wcv1;
      walshsv2[k1][p]=wcv2;
```

```
                }
            }
        }
      /* Finished decision feedback. */
      }
   }
}

/* Now there are still some undetermined output bits. */
/* Output the ones that end in the zero state. */
for (k=bctr;k<184;k++) { /* For all remaining output bits. */
   frameout[k]=bh1[0][k];
   if (timeout!=NULL) timeout[k]=191;
}

/* The last eight frame bits are known to be zero. */
for (k=184;k<192;k++) {
   frameout[k]=1;
   if (timeout!=NULL) timeout[k]=191;
}

/* Go through DF_CHECK, add up how many metrics had no prior DF    */
/* update, 1 DF update, ..., 5 DF updates.                         */
if (df_count!=NULL) {
   for (i=0;i<6;i++) df_count[i]=0; /* initialize df_count[] */
   for (i=0;i<96;i++) {
      for (j=0;j<6;j++) {
         df_count[DF_CHECK[i][j]]++;
      }
   }
}

}
/* ---------------------  END OF COMPUTATION  -------------------- */

/* ---------------   MEX FUNCTION GATEWAY ROUTINE   -------------- */

void mexFunction(
   int nlhs, mxArray *plhs[],     /* # of, ptr on input arguments */
   int nrhs, const mxArray *prhs[]) /* # of, ptr on output args */
{
/* Declare function arguments (i.e. choose names for them here). */
double *walshcorr;       /* input arguments, Walsh correlations */
double *decode_option;   /* decode option */
double *df_flag;         /* decision feedback flag */
double *intlv;           /* deinterleaver vector */
double *dintlv;          /* interleaver vector */
double *nstates;         /* number of path histories that must agree
                            for bit decision (minus 1) */

double *frameout;     /* output arguments, decoded frame */
double *timeout;      /* data bit output decoding steps */
double *df_count;     /* decision feedback count */

/* Check number of arguments. */
if (nrhs > 6) {
   mexErrMsgTxt("Maximum of 6 input arguments is allowed!");
} else if (nlhs > 3) {
   mexErrMsgTxt("Only 3 output arguments allowed!");
}
```

```
/* Check input arguments. */

if (!mxIsNumeric(prhs[0]) || mxIsComplex(prhs[0])
   || mxIsSparse(prhs[0]) || !mxIsDouble(prhs[0])
   || !((mxGetM(prhs[0]) == 96) && (mxGetN(prhs[0]) == 64))) {
   mexErrMsgTxt("The first input argument must be a real 96 x 64
matrix!");
}

if (nrhs>1) {
   if (!mxIsNumeric(prhs[1]) || mxIsComplex(prhs[1])
      || mxIsSparse(prhs[1]) || !mxIsDouble(prhs[1])
      || !((mxGetM(prhs[1]) == 1) && (mxGetN(prhs[1]) == 1))) {
      mexErrMsgTxt("The second input argument must be a real
scalar!");
   }
}

if (nrhs>2) {
   if (!mxIsNumeric(prhs[2]) || mxIsComplex(prhs[2])
      || mxIsSparse(prhs[2]) || !mxIsDouble(prhs[2])
      || !((mxGetM(prhs[2]) == 1) && (mxGetN(prhs[2]) == 1))) {
      mexErrMsgTxt("The third input argument must be a real
scalar!");
   }
}

/* Remaining inputs are not checked. */

/* Create matrices for output arguments. */
plhs[0]=mxCreateDoubleMatrix(1, 192, mxREAL);
if (nlhs>1) plhs[1]=mxCreateDoubleMatrix(1, 192, mxREAL);
if (nlhs>2) plhs[2]=mxCreateDoubleMatrix(1, 6, mxREAL);

/* Dereference the arguments and call computational routine. */
decode_option=NULL;
df_flag=NULL;
timeout=NULL;
intlv=NULL;
dintlv=NULL;
df_count=NULL;
nstates=NULL;

frameout=mxGetPr(plhs[0]);
if (nlhs>1) timeout=mxGetPr(plhs[1]);
if (nlhs>2) df_count=mxGetPr(plhs[2]);

walshcorr=mxGetPr(prhs[0]);

if (nrhs>1) decode_option=mxGetPr(prhs[1]);
if (nrhs>2) df_flag=mxGetPr(prhs[2]);
if (nrhs>3) intlv=mxGetPr(prhs[3]);
if (nrhs>4) dintlv=mxGetPr(prhs[4]);
if (nrhs>5) nstates=mxGetPr(prhs[5]);

computation(frameout,timeout,df_count,walshcorr,decode_option,df_flag
, intlv,dintlv,nstates);
}
/* --------------------   END OF GATEWAY   ----------------------- */
```