Delft University of Technology
Department of Electrical Engineering
Telecommunications and
Traffic Control Systems Group

Graz University of Technology
Faculty of Electrical Engineering
Institute for Communications and
Wave Propagation

Title: **Performance Enhancement of a Dual-Signal Receiver System for Simultaneous Reception of two Co-Channel Signals by Applying Error Correction Coding**

Author: Klaus Witrisal

Type: Graduation Thesis
Size: 98 + ix pages
Date: June 25, 1996

Professor: Prof. Dr. R. Prasad
Mentors: F. van der Wijk, G.J.M. Janssen
Period: January 1996 – June 1996

Abstract:

The Dual-Signal Receiver (DSR) was proposed for simultaneous reception of two narrowband Binary Phase Shift Keying (BPSK) modulated co-channel signals of (slightly) different strength.
In this report improvement of the signal cancellation used in the DSR is obtained with Forward Error Correction (FEC) coding. Rather short linear block codes were evaluated using hard and soft decision decoding techniques.
Especially soft decision decoding shows large coding gain for a channel with (heavy) co-channel interference and Additive White Gaussian Noise (AWGN). The gain in presence of interference is even greater than the gain achieved in a pure AWGN-channel. The absolute performance is still worse.

Indexing Terms:
Signal Cancellation Techniques, Dual-Signal Receiver, Error Correction Coding, Soft Decision Decoding, Linear Block Codes

Summary

The Dual-Signal Receiver (DSR) was proposed for simultaneous reception of two narrowband Binary Phase Shift Keying (BPSK) modulated co-channel signals. The receiver consists of two BPSK-demodulators in succession, its concept is based on cancellation of the large signal. The difference in transmitted signals is used for separation of both signals, i.e., difference in signal strength, carrier phase, etc. The large signal (s_i) is demodulated and the estimated data is used for its cancellation. The small signal (s_c) remains with less interference and can be demodulated. Comparison of the BER-performance of the small signal to the optimum results when only Additive White Gaussian Noise (AWGN) is present, shows that nearly optimum demodulation of s_c is possible when the signal differences are not too small.

The performance of the small signal is almost equivalent to the performance of the large signal, since errors in the estimated data of s_i result in imperfect signal cancellation, and thus in errors in s_c. The idea of this work is to improve the signal cancellation of s_i by applying Forward Error Correction (FEC) coding.

The signal model of the DSR represents a (heavy) co-channel interference, AWGN channel. Performance analysis of the large signal means performance analysis of a coherent BPSK-demodulator in that particular environment. Rather short linear block codes were evaluated using hard and soft decision decoding techniques.

Especially soft decision decoding enables large coding gains in the given channel. The gains in presence of interference are even greater than the gains achieved in a pure AWGN-channel by the same decoding techniques. The absolute performance is worse.

Difference in transmitted signals is essential for the principle of the DSR. In case both Signals have about equal strength, demodulation is still possible when there is sufficient carrier phase shift (ϕ') between our two signals. This carrier phase difference makes the signals partly orthogonal. The dependency of the performance of ϕ' (and of another parameter) was used to define an Outage Rate (OaR), which gives the probability that the receiver performs worse than a certain Bit Error Rate (BER) threshold. The outage rate is an excellent performance measure, especially for the case where both signals have about equal strength. Generally outage rates are good performance measures for digital communications systems, since the user is only interested if a connection is established or not.

A full analysis of the small signal error mechanism shows that coding gives almost no additional gain for s_c compared to s_i. Interference by insufficiently suppressed bits of s_i determines the performance of s_c. Coding does give gain in those cases where noise is the greater impact on s_c than the remainders of s_i after signal suppression. This means large difference in signal strength. The gain is similar to the gain achieved in a AWGN-channel.

Table of Contents

List of Abbreviations

ADC	Analogue Digital Converter
ARQ	Automatic Repeat Request
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhuri-Hocquenghem
BER, BERs	Bit Error Rate, Bit Error Rates
BERi, BERc	BER of signal s_i, BER of s_c
$\overline{BERi}$, $\overline{BERc}$	averaged BERi and BERc
BERD	Bit Error Rate Decoder (Decoder output BER)
BERDi, BERDc	BERD of signals s_i and s_c, respectively
BER_{max}	BER threshold used for OaR computations
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CDMA	Code Division Multiple Access
DC	Direct Current
DPSK	Differentially Phase Shift Keying
DSP	Digital Signal Processor
DSR	Dual Signal Receiver
FEC	Forward Error Correction
GSM	General System for Mobile communications
ITU	International Telecommunications Union of the United Nations
OaR	Outage Rate
pdf	Probability Density Function
PSK	Phase Shift Keying
PTT	Post Telegraph and Telecommunication
QoS	Quality of Service
SFH	Slow Frequency Hopping
SIR	Signal-to-Interference Ratio
SIRi, SIRc	SIR of s_i and s_c, respectively
SNR	Signal-to-Noise Ratio
SNRi, SNRc	SNR of s_i and s_c, respectively
TDMA	Time Division Multiple Access
UMTS	Universal Mobile Telecommunications System
WER	Word Error Rate
WERi, WERc	WER of s_i and s_c, respectively

List of Symbols

$\{0_\varepsilon, 1_\varepsilon\}$	bits {1, 0} transmitted and bits {0, 1} received erroneously
#00, #01, ...	number of the patterns 00, 01, ... of one sequence $\boldsymbol{\nu}$
A_i, A_c	signal amplitude of s_i and s_c, respectively
A_i, A_0	events: the i^{th} and the all-zero code word was transmitted, respectively
B_j, B_j', B_j''	events: received sequences for performance evaluations
c	subscript indicating the small signal
C, C_j'	events: received sequence was not decoded
d	distance between two code words
d_0	distance between the received sequence and the all-zero code word
$d_i(t)$, $d_c(t)$	the asynchronous random data signals of s_i and s_c, respectively
$d_{i,j}$, $d_{c,j}$	transmitted symbols of s_i and s_c, respectively
$d_{i,j}'$	one symbol of Δs_i
$\hat{d}_i(t), \hat{d}_c(t)$	the estimated data of s_i and s_c, respectively
$\hat{d}_{i,j}$	one estimated symbol of s_i
d_l	distance between the received sequence and the l^{th} code word
d_{min}	minimum distance of the code
E_b	average received energy per information bit
E_{bi}, E_{bc}	E_b for s_i and s_c, respectively
E_{bi}/N_0	signal-to-noise ratio per information bit
E_c	average received energy per coded (= transmitted) bit
f_0	carrier frequency
G	generator matrix (dimension k × n)
i	subscript indicating the large signal
$I_c(t)$	the in-phase component of s_c being used for demodulation
if_v, if_v', if_v''	weight of the influencing sequences $\boldsymbol{\nu}$, $\boldsymbol{\nu}'$, $\boldsymbol{\nu}''$
j	sequence length; weight of a code word, etc.
k	number of information bits per code word
m	metric; the values being use by the decoder to calculate the decision variables
m_{li}	metric for the i^{th} symbol of the l^{th} code word
$m_{x\rho}$	metric of the transmitted bit x received as ρ
n	code length
n(t)	white Gaussian noise waveform
nec(x)	number of patterns of influencing class x

N_0	level of the power spectral density of white Gaussian noise
n_j	weight structure of the code
p	crossover-probability of the BSC
$p(\bullet)$	probability of $(\bullet)$; continuos value
pec(x)	influence of class x
$Pr(\bullet)$	probability of $(\bullet)$; discrete value
$Pr(\bullet\|\bullet)$	probability of left side assuming that the right side has occurred
p_x	bit-and-error patterns defining class x
P_ρ	$Pr(\rho\|0)$; the probability that the demodulator outputs the symbol ρ assuming a transmitted symbol 0
Q	number of output levels of a soft quantisation demodulator
$Q(\bullet)$	integral over the Gaussian pdf
r	number of parity bits
R	code rate
r(t)	the composite input signal of the DSR
$rem_I(t)$	in-phase component of r(t) after remodulation
$r_I(t)$, $r_Q(t)$	the composite input signal of the DSR after down conversion to $f_0 = 0$; in-phase and quadrature components, respectively
s_i, s_c	the large signal, and the small signal, respectively
$\mathbf{S}_l$	the l^{th} code word, consisting of n symbols S_{li}
$\mathbf{S}_l''$	an l^{th} code word having weight 2
S_{li}	the i^{th} symbol of the l^{th} code word
t	error correction capability of a code
T	duration of a modulated bit
th_i	the i^{th} threshold level of the quantising demodulator
u	information sequence (row vector of length k)
$\mathbf{v}$, $\mathbf{v}_1$, $\mathbf{v}_2$	code words (row vectors of length n)
$\boldsymbol{\nu}$	a sequence of influencing patterns w
$\boldsymbol{\nu}'$, $\boldsymbol{\nu}''$	modified sequences $\boldsymbol{\nu}$
$\#\nu'$	number of sequences $\boldsymbol{\nu}'$ having a certain influence
V	total number of sequences $\boldsymbol{\nu}$
w	pattern of symbols of s_c influencing on s_i; $w = (d_{c,0}, d_{c,1})$
$w_{i,j}$, $w_{c,j}$	code words of s_i and s_c, respectively
x	one transmitted symbol, $x \in \{0, 1\}$
α, α_j	amplitude factor of A_i for symbols of Δs_i
β	amplitude factor of A_i for correctly suppressed bits in Δs_i

$\Delta_i T$	error in recovered bit timing
$\Delta m'$, $\Delta m''$, $\Delta m^{(n)}$	decision variable for deciding between 1, 2, n bits, respectively
$\Delta m_l''$	decision variable for the l^{th} code word being assumed to have weight 2
$\Delta r_I(t)$, $\Delta r_Q(t)$	in-phase and quadrature components of r(t) after signal cancellation
Δs_i	the remainders of s_i after signal cancellation
ΔT	word timing difference between s_i and s_c
$\Delta' T$	bit timing difference between s_i and s_c normalised to $\Delta' = \lvert\Delta\rvert \leq 0.5$
ε	an error
ε_i, ε_c	bit error in s_i and s_c, respectively
ε_c, ε_n	a correlated, a non-correlated error
ε_w	a word error
ϕ'	carrier phase difference between s_i and s_c
ϕ_i, ϕ_c	carrier phases of s_i and s_c, respectively
γ	signal-to-noise ratio
γ_i, γ_c	signal-to-noise ratio SNR for s_i and s_c, respectively
η	carrier suppression performance
κ	factor expressing the influence of $\Delta' T$
ρ	the (un)quantised demodulator output
$\boldsymbol{\rho}$	received sequence, with (un)quantised soft decision demodulation
ρ_i	the i^{th} of the n symbols of $\boldsymbol{\rho}$
τ	a delay time
ω_0	carrier frequency
ψ_i, ψ_c	signal-to-interference ratio SIR for s_i and s_c, respectively
ψ_c'	SIR for s_c, assuming that s_i was suppressed perfectly
$\binom{n}{j}$	combinations of j elements within a sequence of n
( )	a sequence of symbols, etc.
[ ]	interval of values
{ }	set of elements

1. Introduction

In the last decades mobile communication has become a research subject of increasing importance. At this moment all around the world numerous mobile communications systems are active, a number which is growing tremendously. Mobile telephony is expected to be a valid competitor to fixed telephony systems in the near future. On the one hand, the national PTTs have already lost their monopoly, thus other providers have entered the market which has caused the prices to fall significantly and they continue to do so. On the other hand some important extra features are available that make this kind of communication interesting for more and more people. The most important difference is that mobile communication means communication with a certain person: In traditional fixed telephony a phone call is made to a telephone where the person to be contacted is expected to be. This is not the case with mobile telephony. Here a phone call is made to a person, not a telephone terminal, wherever this person may be. Furthermore, mobile telephony offers a range of other services such as transmitting short data messages, voice mail box, access to information data banks, etc.
The rapidly increasing number of (potential) users, and some requirements like large variety of services, high degree of reliability, flexibility and privacy, or high spectral efficiency make digital mobile communication systems more economical than analogue systems. Only modern microprocessors and signal processors provide the flexibility to implement complex, but efficient modulation types, access protocols and coding schemes that are needed to share the scarcely available bandwidth between the increasing number of subscribers.

In Europe, currently the second generation system called GSM (Global System for Mobile communications) is in use. This is a digital system providing voice and data communications using data rates of up to 10 kbit/s [1]. The next step in the development of mobile radio communications systems is called UMTS (Universal Mobile Telecommunications System), which should provide a much wider range of mobile services to the user via a range of mobile terminals at a data rate of up to 2 Mbit/s. The UMTS is a European project which will be standardised by the European Telecommunications Standard Institute (ETSI).

An important issue in UMTS is that it should have a high spectral efficiency, because it is assigned a limited bandwidth by the International Telecommunications Union of the United Nations (ITU). A way to make the system more efficient is to reuse frequency geographically, i.e., to reuse the same bandwidth in areas which are sufficiently apart in space. This is the basic principle of cellular communication systems.

The capacity can be increased by reducing the so called frequency reuse factor which means reducing the distance between base stations using the same frequencies. Co-channel interference, i.e., interference from cells using the same frequencies, is becoming the main limiting factor of this principle. Further efficiency enhancement is obtained with signal cancellation techniques, as implemented in the receiver system investigated in this work.

In [2], the Dual-Signal Receiver (DSR) concept is proposed for simultaneous reception of two narrowband Binary Phase Shift Keying (BPSK) modulated co-channel signals, of (slightly) different strength. The receiver concept is based on cancellation of the large signal, the small signal remains with less interference and can be demodulated. The DSR was developed in the Telecommunications and Traffic Control Systems Group of Delft University of Technology, its principle will be reviewed in Chapter 2.

In [3], BER and Outage performance of the DSR are investigated for a Rician fading environment. In [4], an alternative scheme is proposed using a differentially coherent detector to demodulate the large signal (DPSK-demodulation).

In [5], a Dual Signal Receiver - Time Division Multiple Access/Slow Frequency Hopping (DSR-TDMA/SFH) protocol was investigated as a candidate for UMTS in the macrocellular environment. In this protocol the Dual Signal Receiver (DSR) is used to enhance the uplink capacity (Mobile Station $\rightarrow$ Base Station) by simultaneously allowing two subscribers to operate on one channel, and the downlink performance (Base Station $\rightarrow$ Mobile Station) by reducing the possibility of receiver outage due to interfering signals from adjacent cells. It has been shown that the spectral efficiency can be improved significantly using the DSR. A comparison to a DS/SFH-CDMA (Direct Sequence/Slow Frequency Hopping - Code Division Multiple Access) protocol showed better spectral efficiency for the DSR-system for the uplink and for the downlink.
A major advantage of this system compared to CDMA-systems is that the same multiple access scheme is used than in GSM, thus compatibility is provided. This fact logically makes the system economically very interesting for current GSM-providers since the existing GSM Base Station-infrastructure can be used to provide nation-wide coverage during an adoption phase.

In Chapter 2, the description of the DSR, we will see that the performance of the DSR is very good if there is some difference in strength between the two signals, but it gets poor when the signals have about same power. Research is being done in order to improve the performance in these specific situations.

In [6], optimum and sub-optimum detection schemes exploiting information about signal parameters like Signal-to-Interference Ratio (SIRi) or carrier phase difference have been evaluated. Those schemes give a maximum Bit Error Rate (BER) improvement of a factor 4 for SIRi < 6dB, which is not sufficient at bad SIRi ~ 0dB.
Another strategy is to apply antenna diversity techniques which are known to be advantageous to enhance receiver performance in fading environments as given with mobile communications. Research in this area is currently done at the Telecommunications and Traffic Control Systems Group of Delft University of Technology [7].

In this work Forward Error Correction (FEC) coding will be investigated to enhance the signal suppression performance of the large signal and therefore enhance the performance of the DSR, especially in the area of bad Signal-to-Interference Ratios SIRi = [0 ... 6dB]. Despite the fact that additional bandwidth is needed, FEC-coding is used in almost any implemented digital communications system, since it allows good performance at the poor conditions of a (mobile) radio channel.

When digital data is transmitted over a noisy channel, there is always a chance that the received data will contain errors. In our particular case of the DSR, the data signal is not only corrupted by noise, but also by (heavy) co-channel interference. One of the goals of this thesis work is to evaluate the benefit of some well-known and frequently used coding schemes in that particular environment. This analysis is given in Chapter 4, the performance evaluation of the large signal using coding. A short introduction to the coding techniques can be found in Chapter 3.
Since we wish to have performance enhancement for both signals, and we cannot determine a priori which signal will be the large one at the receiver, coding has to be applied to both signals. Whether there is additional performance improvement possible for the small signal will be evaluated in Chapter 5.
In Chapter 6, a proposal is given for the implementation of the DSR to the GSM-system. Code-interleaving is used in this system which makes the cancellation-enhancement-scheme evaluated in Chapter 4, less practicable.
Finally, conclusions and recommendations are given in Chapter 7.

2. The Dual-Signal Receiver

In this chapter the Dual-Signal Receiver (DSR) is presented for simultaneous reception of two narrowband Binary Phase Shift Keying (BPSK) modulated co-channel signals. This receiver concept was proposed in [2], a detailed block diagram is shown in Figure 2-1.
The first section of this chapter reviews the signal model of the DSR which defines a (heavy) co-channel interference AWGN channel. The block diagram and an explanation of the principle of the DSR is given in Section 2.2. Computational results of the BER analysis and a qualitative description of the error mechanism are presented in Section 2.3, in order to see the performance limiting factors of the DSR which are to be improved.

2.1 The Signal Model

The composite input signal r(t) of the receiver consists of two BPSK-modulated signals s_i and s_c of different strength, and Additive White Gaussian Noise n(t) (AWGN). The signal s_i is the signal captured by the first BPSK demodulator, we will refer to it as "the large signal" throughout this report. The signal s_c is demodulated by the second BPSK demodulator, it will be called "the small signal".

$$r(t) = A_i d_i(t)\cos(\omega_0 t) + A_c d_c(t)\cos(\omega_0 t + \phi'(t)) + n(t) \tag{2-1}$$

where A denotes signal amplitude, $\omega_0 = 2\pi f_0$ is the carrier frequency and $\phi'(t)$ indicates carrier phase difference. The signal parameters belonging to the large and small signal are indicated by the subscripts i and c, respectively. $d_{i,c}(t)$ are asynchronous random data signals with a timing difference ΔT

$$\begin{aligned} d_i(t) &= \sum_{k=0}^{\infty} \{-1,1\}\Pi(t - kT) \\ d_c(t) &= \sum_{k=0}^{\infty} \{-1,1\}\Pi(t - kT - \Delta T) \end{aligned} \tag{2-2}$$

$\{-1,1\}\Pi(t - kT)$ indicates a rectangular pulse of duration T and the values $\{-1,1\}$ with equal probability of occurrence, the symbols $\{0, 1\}$, respectively, are often used in the text to refer to these values.

For the analysis of the interference between the signals s_i and s_c, it is appropriate to define $\Delta' = |\Delta| \leq 0.5$. If $\Delta' > 0.5$, just another bit constellation in s_i and s_c has equivalent influence, hence resulting in symmetric Bit Error Rate (BER) functions BER(Δ') with respect to $\Delta' = 0$ and $\Delta' = 0.5$.

Further, we define SIRi = $\psi_i = A_i^2/A_c^2$ as the signal-to-interference ratio of signal s_i, and SNRi = $\gamma_i = E_{bi}/N_0$ as the signal-to-noise ratio of s_i. $E_{bi} = A_i^2 T$ [W·s] is the received signal energy per symbol time for BPSK modulation, and $N_0/2$ [W/Hz] is the two sided power spectral density of white Gaussian noise (see e.g. [8, p. 583]).

It was mentioned above that the difference in transmitted signals is used for separation of the two signals. This difference is expressed by the parameters SIRi, the difference of signal power, ϕ', the difference of carrier phase and Δ', the difference of bit timing. These three parameters and of course the signal-to-noise ratio will be taken into account for the calculation of the bit error rates of s_i and s_c. For these calculations, we will assume the parameters to be constant, which is appropriate during the transmission of short data bursts as given by a system using Time Division Multiple Access (TDMA) [1].

2.2 The Receiver Concept

The dual-signal receiver consists of two BPSK demodulators in succession, as shown in Figure 2-1.

[1] To verify the assumption of constant signal parameters, let us consider a mobile phone (GSM) used in a car going with 100km/h towards the Base Station. In GSM, the duration of a data burst is 0.58ms [1]. In this time the car is moving 1.6cm which means a phase shift of 0.1π for a carrier frequency of 900MHz. This value shows that the assumption of a constant phase difference is fairly good for GSM.

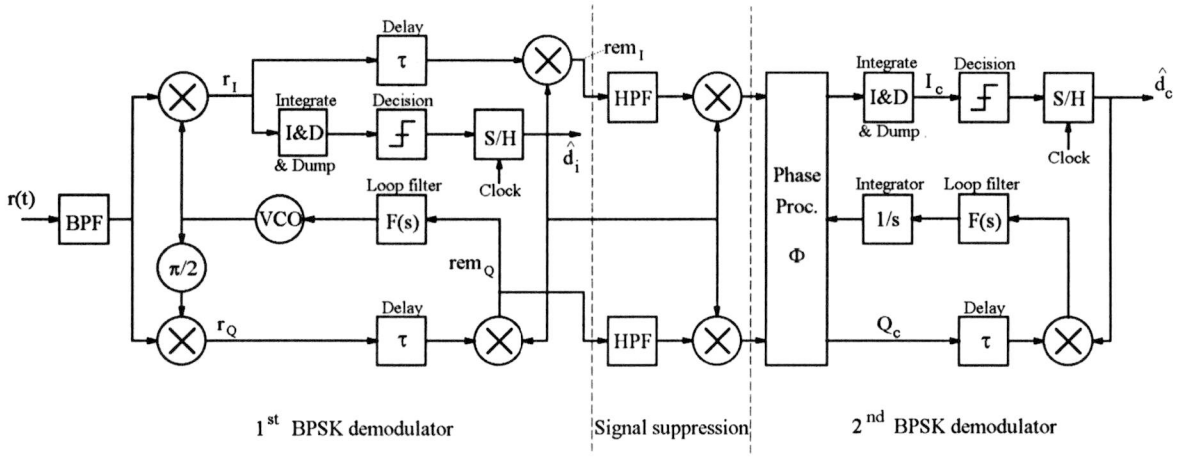

Figure 2-1: Block diagram of the dual-signal receiver

The first demodulator is captured by the large signal s_i, and produces an estimate of the data signal $\hat{d}_i(t)$ consisting of symbols $\hat{d}_{i,j} \in \{-1,+1\}$. Errors in $\hat{d}_i(t)$ are caused by the presence of the small signal s_c and noise. The bit error rate of s_i, BERi, gives the error probability of $\hat{d}_i(t)$; its analysis will be presented in Chapter 4.

Decorrelation of the signal s_i is achieved by remodulation of $r_I(t)$ with the estimated data $\hat{d}_i(t)$. This remodulation is also used in a decision feedback PLL for carrier recovery, it is known to be a robust technique for this purpose [9, p. 347]. In case no error occurred in $\hat{d}_i(t)$, and assuming perfect clock timing recovery, this remodulation results in a concentration of the signal power of s_i in a DC-component, since $\hat{d}_i(t)d_i(t) = +1$. Now, the large signal s_i is removed by means of high-pass filtering. In practice, due to bit errors, s_i will not be completely concentrated in the DC-component. However, even a relatively high BER results in good performance [2]. Before demodulation of the small signal s_c is possible, a second remodulation with $\hat{d}_i(t)$ is required to remove the modulation with $\hat{d}_i(t)$ from s_c. This is done perfectly, since $\hat{d}_i^2(t) = +1$.

The resulting signal after cancellation consists of s_c, noise and Δs_i, the remainders of s_i due to bit errors, timing errors and imperfect cancellation. This signal is demodulated by the second BPSK-demodulator.

The BER-analysis for the small signal will be given in Chapter 5, as well as an analytical description of the signal path of the DSR.

Computational results of the BER-analysis of s_i and s_c are presented in the following section.

2.3 Computational Results of the BER Analysis

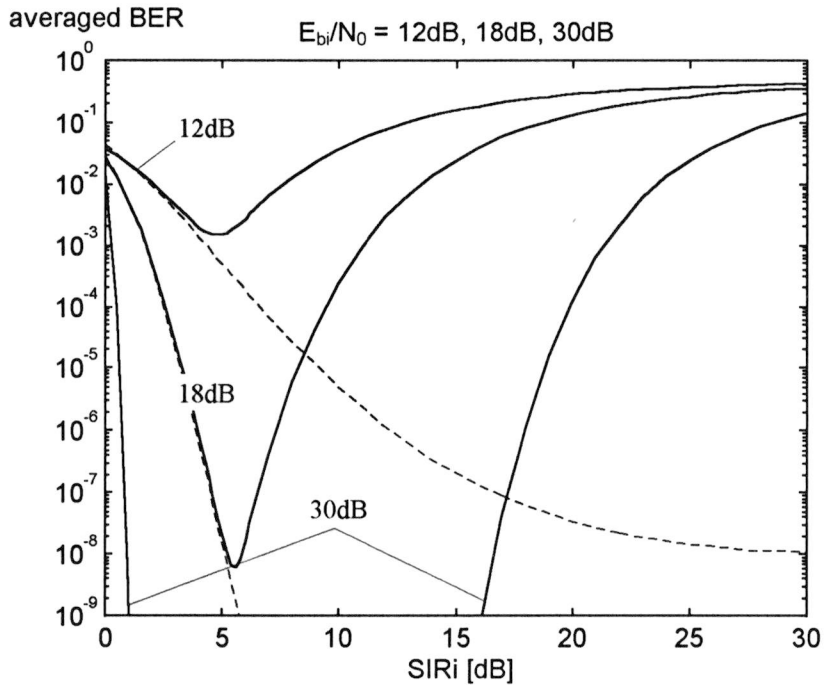

Figure 2-2: Averaged BER results for the large signal s_i (BERi, dashed lines) and for the small signal s_c (BERc, solid lines) against SIRi; SNRi = {12, 18, 30dB}, η^2 = –30dB.

Figure 2-2 shows averaged BER results for the bit error rates BERi and BERc of the large and the small signal, respectively, averaging was performed over $\phi' = [0 \ldots \pi/2]$ and $\Delta' = [0 \ldots 0.5]$ [2]. One may make the following observations: At low signal-to-interference ratios (SIRi = [0 ... 4-5dB]) BERi is high due to strong interference by s_c. A bit error in $\hat{d}_i$ means that the wrong bit value is subtracted from the composite input signal r(t), hence resulting in heavy interference on s_c by Δs_i and thus in a high error rate for s_c as well. BERc is more or less the same as BERi at these values of SIRi. With increasing SIRi both, BERi and BERc, are decreasing since the influence of s_c is decreasing and better signal suppression is possible. With further increasing SIRi, noise is becoming the major limiting factor for the performance of s_c because the signal-to-noise ratio of s_c is decreasing with increasing SIRi (SNRc = SNRi – SIRi [dB]); a minimum for BERc can be found.

---

[2] MATLAB program: *fig6_1*

Figure 2-2 was calculated by the program *fig6_1* using the functions *PeiPecCd* to calculate the BERs, and *igrl* to average them over ϕ' and Δ'.

The analysis of BERi in Chapter 4 will show that we have interfering situations, where a reliable operation of the DSR just is not possible. Unfortunately, these cases are included in the averaged BER results, therefore, they are not very suitable for performance evaluations.
A basic definition of "good" performance for any system for digital communications is to reach a certain BER value, BER_{max}. Due to a large variety of impacts like noise, fading, interference etc., it is generally not possible for a practical system to achieve all the time a performance better than this specified threshold. An Outage Rate (OaR) which gives the probability of the BER being greater than this threshold is a much more suitable measure to evaluate the peformance of such systems:

$$OaR = \frac{\text{cases where BER} > \text{BER}_{max}}{\text{all the cases}} \tag{2-3}$$

A typical BER specification to be met is 10^{-3} for voice communications, an acceptable outage rate is 0,5%.
The OaR concept is applied to the DSR in our co-channel interference, AWGN-channel as follows: Both, BERi and BERc are dependant on the phase difference between the two signals (ϕ') and on their bit timing difference expressed by Δ'. These dependencies are used to destinguish the cases mentioned in (2-3) (see Section 4.2).
The outage rates allow a good comparison of the performance of the DSR, especially at low signal-to-interference ratios SIRi. Table 2-1 lists OaR-results for our signals s_i and s_c for SIRi = 0dB.

Table 2-1: Outage rate results for SIRi = 0dB

SIRi = 0dB	BER_{max}	*SNRi = 12dB*	*SNRi = 18dB*	*SNRi = 30dB*
OaRi, Large Signal:	0.01	0.53	0.37	0.19
	0.001	0.65	0.45	0.24
OaRc, Small Signal:	0.01	0.47	0.35	0.18
	0.001	0.58	0.43	0.23

The results given in Figure 2-2 look very promising, at least, for high signal-to-noise ratios (SNRi) and when some strength-difference is present. However, we wish to keep SNRi as low as possible, and we cannot controll SIRi to have suitable values in practical systems. Therefore, methods have to be found to improve the performance (e.g. References [6 – 7]).
In this work coding will be investigated to enhance the performance of the DSR. We will evaluate the performance at SNRi = 12dB and at several SIRi-values, for several codes.

3. Review of the Linear Block Codes Used in this Work

When digital data is transmitted over a noisy channel, there is always a chance that the received data will contain errors. In the particular case of the DSR, the data signal is not only corrupted by noise, but also by (heavy) co-channel interference. The user generally establishes an error rate above which the received data are not useable. If the received data will not meet the error rate requirement, error-correction coding can often be used to reduce errors to a level at which they can be tolerated. In recent years the use of error-correction coding for solving this type of problems has become widespread.

The principle of error-correction coding is to transmit additional information (redundancy) in order to test if an error has occurred (Automatic Repeat Request (ARQ) systems) or even to correct errors. The latter is called Forward Error Correction (FEC), which will be applied to the DSR within this work.

Redundancy means that if n binary symbols have been transmitted, less than 2^n valid messages are possible. A certain number r of the n bits is used as parity bits, the remaining $k = n - r$ bits are information bits.

For example, one bit error can be detected within a sequence of length n by introducing one parity bit to make the number of ones in each code word even. However, if two errors occur within this sequence, the number of ones is even again, thus we have just another valid sequence and the error detection fails.

If we wish to correct patterns of t or fewer errors, it is both necessary and sufficient for every message sequence to differ from every other sequence in at least $(2 \cdot t + 1)$ positions. For example, to correct all single and double symbol errors it is necessary that all patterns of message sequences differ in at least five symbols. Any received sequence which contains two errors and, therefore, differs from the correct sequence in exactly two places will always differ from all other messages in at least three places, error correction is possible.

The number of places in which two sequences differ from each other is called the Hamming distance (d). The smallest value of d for all pairs of code sequences is called the minimum distance of the code and is designated d_{min}. Error correction is possible up to a maximum number of

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor \tag{3-1}$$

errors, where $\lfloor \rfloor$ denotes the integer part. The code is used to add redundancy (parity bits) to the information sequence in order to ensure a particular minimum distance between any two code words.

The topic of coding is immense, only a short introduction to the coding techniques and to some special properties used in this works can be given in this chapter. Other techniques related to coding, like soft decision decoding, will be introduced later in this report when they are applied to the DSR.
Further introduction and explanation of how codes are generated can be found in any book about coding, for example in the book by P. Sweeney, 1991, [10], which also contains numerous bibliographical notes for further readings. A large part of this work is based on the book by G.C. Clark and J.B. Cain, 1981, [11]. Another important work is the book by S. Lin and D.J. Costello, 1983, [12], which is known to be the most comprehensible and comprehensive in the area of convolutional codes, but also contains excellent coverage of block codes.

In this chapter we will first review some important properties of linear block codes (Section 3.1), performance evaluation and an analysis of the error mechanism is given in the next section, and finally, the codes used throughout this work are described in Section 3.3.

3.1 Linear Cyclic Block Codes: Basics, Properties

The principle of coding is that not all 2^n possible combinations of a sequence consisting of n binary symbols are valid message-sequences, i.e., redundancy is introduced by the coding algorithm. Let us make the following definitions for a (n,k)-block code:

n	block length; each code word consists of n binary symbols,
k	number of information bits, i.e. 2^k valid code words consisting of n bits,
$r = n - k$	number of parity bits.
$R = k/n$	is the code rate giving the ratio between information bits and coded bits.
d; d_{min}	distance between two code words; minimum distance of the code;
w	denominates the weight of a code word which means the number of ones,
n_j	is the weight structure of the code. It gives the number of weight-j code words.

The codes used throughout this work are linear (cyclic) block codes.
The definition of linearity in the case of codes is somehow similar to the definition used for linear systems because the principles of superposition and scaling apply to these codes. With

binary codes only the principle of superposition is important, it will be used several times in this report: If we encode two information sequences and add the resulting code words modulo 2 ([1]) we shall get the same sequence as by adding the two information sequences first and encoding the resulting word. Since every one of the 2^k information sequences is possible, thus the (modulo 2) sum of two sequences is another information sequence, it becomes clear that also the (modulo 2) sum of any two code words is just another code word.

One consequence of linearity is that the distance structure of the code is the same for each code word, particularly, it is the same as the distance structure of the all-zero word. The all-zero word is always a valid word of a linear block code, its distance structure is equal to the weight structure of the code, n_j.

If we have two code words $\mathbf{v}_1$ and $\mathbf{v}_2$ and we add one of these words (e.g. $\mathbf{v}_1$) to both sequences, we will get the sequences $\mathbf{v}_1 + \mathbf{v}_1 = \mathbf{0}$, the all-zero word, and $(\mathbf{v}_1 + \mathbf{v}_2)$ which is another code word. It can be seen easily that the distance between the two code words $\mathbf{v}_1$ and $\mathbf{v}_2$ is the same as the weight of the sequence $(\mathbf{v}_1 + \mathbf{v}_2)$. We can conclude: The minimum distance of a linear block code is equal to the minimum number of non-zero symbols occurring in any code word (excluding the all-zero code word).

Note, that we obtain the total number of information sequences by summing up over the weight distribution of the code: $\sum_{j=0}^{n} n_j = 2^k$.

Encoding can be expressed in matrix form by

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} \tag{3-2}$$

where $\mathbf{v}$ is a length n row vector representing the coded word, $\mathbf{u}$ is a length k row vector containing the information sequence and $\mathbf{G}$ is the $k \times n$ generator matrix. Modulo 2 arithmetic is assumed.

The matrix multiplication $\mathbf{u} \cdot \mathbf{G}$ in modulo 2 arithmetic means that those rows of the generator matrix are added where the corresponding bits of $\mathbf{u}$ are 1. Since the generator matrix consists of k linearly independent code words, it can be shown that this operation results in 2^k different code words.

The cyclic property is for this work only of interest as far as the generation of the generator matrix is concerned. The structure of cyclic codes is such that if any code word is shifted cyclically, the result is another (linearly independent) code word. Exploiting this property, we can obtain a generator matrix consisting of k linearly independent row vectors by simply

---

[1] Modulo 2 arithmetic means: $0 + 0 = 0$, $1 + 0 = 1$ but $1 + 1 = 0$, for summation, which is just the same as the exclusive OR operation, and $0 \cdot 0 = 0$, $0 \cdot 1 = 0$ and $1 \cdot 1 = 1$ for multiplication.

shifting one code word defined by the generator polynomial, (k – 1) times. Almost every book about coding contains a list of generator polynomials [11]. The code words resulting from coding by this generator matrix are not systematic, i.e., information bits and parity bits are not separated within the coded sequence.

3.2 Performance Evaluation for the BSC

We have seen that a block code with a minimum distance d_{min} is able to correct up to $t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$ errors, thus, a word error probability (Word Error Rate, WER) can be calculated by calculating the probability of (t + 1) or more errors occurring within one received code word. Assuming a Binary Symmetric Channel [2] (BSC) with a bit error probability p, we can write

$$WER = \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \tag{3-3}$$

To compare BER results of several codes with uncoded results, we have to use a signal-to-noise ratio per information bit. Coding means introduction of parity bits, i.e., additional bits have to be transmitted which requires additional energy and bandwidth to gain the same channel BER p. A signal-to-noise ratio per coded bit can be calculated from the signal-to-noise ratio per information bit by $E_c/N_0 = E_b \cdot R/N_0$ [9], where R is the code rate. For coherent BPSK demodulation the crossover probability p of an additive white Gaussian noise BSC is given by

$$p = Q\left(\sqrt{2\frac{E_c}{N_0}}\right) = Q\left(\sqrt{2\frac{E_b R}{N_0}}\right) \tag{3-4}$$

At low channel BERs an approximation of the decoder output BER (Bit Error Rated Decoder, BERD) can be given if the WER is known:

---

[2] The binary symmetric channel has an error probability of p for both transmitted symbols {0, 1}, respectively. I.e., one symbol is received correctly with a probability of (1 – p), a sequence of n symbols is received correctly with a probability of $(1 - p)^n$, one error within a sequence of n occurs with probability $p(1 - p)^{n-1}$, etc. The probability p is also called "crossover-probability".

$$BERD \approx \frac{d_{\min}}{n} WER \tag{3-5}$$

In this formula we assume the error event to be the event where exactly (t + 1) bit errors occur, thus the decoder inserts t more errors and $d_{min} \geq 2 \cdot t + 1$ errors appear within the sequence of n bits. This is indeed the most probable error event, thus (3-5) gives a very good approximation at least for low channel BERs.

Note that this approximation gives a lower bound on BER only for perfect codes, where each of the 2^n possible sequences has a distance $d \leq t$ from a code word. However, for non-perfect codes the approximation gives an upper bound since these codes can often correct or at least detect errors of distance $d > t$, thus the approximated results are worse than the exact ones.

In the following sub-section a method is given to calculate the exact BERD. A comparison to the approximation presented above will show that the approximation gives excellent results even at rather high channel BERs. Therefore, we will use it throughout this report to obtain BER results from WER results.

The explanation of the exact error mechanism will also be instructive for the union bound calculations proposed in Section 4.4. Beyond that, we will use the "minimum-error-event" (t + 1 channel errors and t errors introduced by the decoder) in Section 5.3 to simplify the calculation of the correlation between word errors in s_i and s_c. The following comparison shows that this assumption is appropriate.

3.2.1 Decoder Output Bit Error Rate

We have defined above that our code consist of n_j sequences of weight-j. It has been shown that the distance of the transmitted sequence to the rest of the possible code words is also given by the weight structure n_j of the code.

To calculate the exact error probability of a linear block code, we have to calculate the probabilities of the received sequence lying within the t-error correction radius of any one of the erroneous code words. It is advantageous to define the following events:

event A_i: the i^{th} of the 2^k code words was transmitted

event B_j: the received sequence was corrected to the j^{th} code word

event C: the received sequence was outside of the error correction range t of any code word, no correction was tried, an error was detected

$Pr(B_j|A_i)$ is the probability that the i^{th} code word which was transmitted, got corrected to the j^{th} code word, thus an error occurs. $Pr(C|A_i)$ is the probability that no error correction was tried because of the received sequence being outside of the error correction portion of any code

word. $Pr(C|A_i) = 0$ for perfect codes, since each possible received sequence lies within the correction range of a code word.

These events are illustrated in Table 3-1 showing the 4 code words and all 32 possible received sequences of a (5,2)-code with t = 1:

Table 3-1: Decoding table for a four-word code

Nr. i, j:	*1*	*2*	*3*	*4 = 2^k*
Code Words A_i:	00000	11100	00111	11011
Correctable	10000	01100	10111	01011
Sequences B_j:	01000	10100	01111	10011
	00100	11000	00011	11111
	00010	11110	00101	11001
	00001	11101	00110	11010
Not-Correctable	10001	01101	10110	01010
Sequences C:	10010	01110	10101	01001

A received sequence can be corrected if i = j, i.e., the received sequence is one of the five sequences listed below the correct code word. Each of these sequences differs in one position from the correct one. This portion defines the error correction range. The not-correctable sequences have the same distance to more than just one code word. We define not to try error correction in these cases, error detection can be performed.

An undetected word error occurs if the received sequence lies within the error correction range of another code word, i.e., $i \neq j$. The exact error probability can be calculated by calculating the probability of the received sequence lying in B_j, $i \neq j$, or in C.

For the used linear codes we can define the following events to simplify our BER calculations. The influence of the channel on any transmitted code word is assumed to be the same as the influence on the all-zero word. The all-zero word stands for any transmitted code word, thus all possible erroneous received sequences are given by the n_j weight-j code words $j \neq 0$. Now, all code words B_j having the same weight are combined in the event B_j'. The probability of one of those code words occurring due to an error event is only depending on the weight of the code word. Let us define:

event A_0: the all-zero word was transmitted

event B_j': the received sequence was corrected to a code word of weight-j

event C_j': the received sequence contains j errors and is out of the correction range t of any code word, thus no correction was performed; the j bit errors remain in the code word.

$\Pr(B_j'|A_0)$ is the probability that the transmitted all-zero word was corrected to a weight-j code word resulting in the output of j errors for this block of length n. $\Pr(B_j'|A_0)$ can be calculated for the BSC by

$$\Pr(B_j'|A_0) = \sum_{v=0}^{t}\sum_{r=0}^{v}\binom{j}{v-r}\binom{n-j}{r}p^{j-v+2r}(1-p)^{n-j+v-2r} \tag{3-6}$$

summing up the probabilities that a received sequence has a distance of [0 ... t] from one of the weight-j error words B_j'. We can see from Table 3-1 that for example one of the d_{min}-sequences can occur if [t + 1 ... d_{min} + t] symbol errors occur in the received sequence (d_{min} = 3 → [2 ... 4] symbol errors). The decoder output BER (BERD) is

$$BERD = \sum_{j=1}^{n}\frac{j}{n}\left[n_j \Pr(B_j'|A_0) + \Pr(C_j'|A_0)\right] \tag{3-7}$$

The calculation of $\Pr(C_j'|A_0)$, the probability of a j-error sequence lying out of the correction portion of any code word, shall not be demonstrated here, it may be found from the program listings given in the appendix [3]. The weight structures of the codes (n_j) were calculated by [4].

---

[3] MATLAB program: *calber*

Calculates the plots given in the next sub-section and provides the used functions with the weight distribution vector nj and other code parameters. The function *berout(p,n,t,nj)* calculates BERD as shown in (3-7). The probability $\Pr(B_j'|A_0)$ is calculated by the function *pew(p,n,t,j)*, $\Pr(C_j'|A_0)$ by *pne(p,n,t,nj,j)*.

[4] MATLAB program: *cw_struc*

Calculates the weight structure of linear, cyclic block codes. The generator matrix is generated by cyclically shifting a generator polynomial. Each code word is generated by combining the rows of the generator matrix corresponding to the ones in the 2^k possible information sequences. The weight of each code word is determined and accumulated to obtain the weight distribution vector n_j.

3.2.2 Comparison Between Exact and Approximated BERD Results

Figure 3-1 and Figure 3-2 show the comparison between exact and approximated BER results. Figure 3-1 gives results for the Golay(23,12)-code which is known to be a perfect code. We can see that the approximation represents a lower bound on the exact calculations. At higher input (channel) BERs the error events with more than t + 1 channel errors per word become more likely, and, as expected, the approximated results become too low. The approximation gives good results up to an input BER of 0.15 which is sufficient for our calculations.

The graph for the BCH(15,7)-code given in Figure 3-2 shows a different behaviour since this is a non-perfect code. With a non-perfect code, not every sequence which has distance t + 1 to the transmitted code word has a distance t to an erroneous code word, i.e., there are received sequences which are outside of the error correction range t of any code word. In the definition of the error event C_j' from Section 3.2.1 we decided not to try any correction in that cases, hence no additional errors are introduced and the output BER is reduced compared to the approximation. However, the difference is rather small, the approximation gives good results up to input BERs of 0.25.

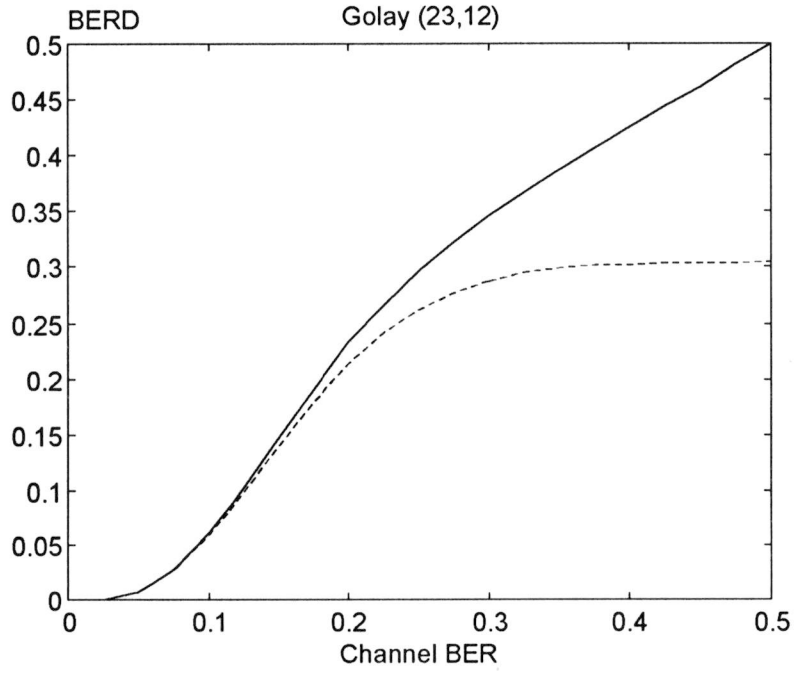

Figure 3-1: Decoder output bit error rates (BERD) against input (channel) bit error rates; solid curve: exact calculation, dashed curve: approximation

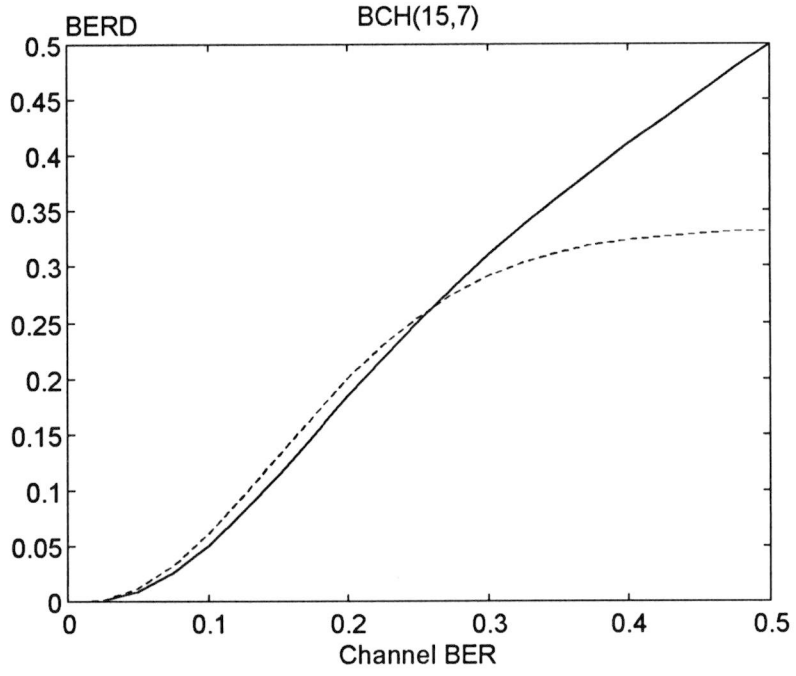

Figure 3-2: BERD against channel BER p; solid curve: exact calculation, dashed curve: approximation

3.3 Overview of the Used Codes

The idea of coding applied to the DSR is to enhance the cancellation performance of the large signal by reducing its bit error rate. As mentioned above, linear cyclic block codes were evaluated in this work for that purpose. Two important reasons for the selection of this class of codes are, first, they are relatively easy to analyse and second, the qualitative results of the analysis of these codes can be transferred to any other class of codes quite well(e.g. convolutional codes). In the recommendations given in Chapter 6 we will examine the DSR implemented to an existing system (the GSM system) and we will see other problems which make the evaluation of the DSR using block codes both appropriate and sufficient.

3.3.1 Description of the Codes

The most powerful codes used in Chapter 4 are the Golay(23,12) and the Golay(24,12) codes. The Golay(23,12)-code has a $d_{min} = 7$, hence an error correction capability of $t = 3$. It is a perfect code, i.e., all 2^{23} possible sequences are lying within the correction portion of a code word. The Golay(24,12)-code is the same as the Golay(23,12)-code but with one additional over-all parity bit which results in better performance using maximum likelihood soft decision decoding. An important property of this code is its rate $R = 1/2$ that makes it interesting for a

comparison with convolutional codes with R = 1/2. Note that this code is non-cyclic due to the addition of the parity bit.

Three BCH codes of block length 15 and error correction capabilities t = {1, 2, 3} are investigated as well as another Hamming code, the very simple (7,4)-code. The BCH(17,9) code has also t = 2 like the BCH(15,7) but a better code rate R. Finally, the shortened Hamming(13,8)-code (shortened from the Hamming (31,26)-code) was taken because there is a soft decision algorithm proposed in [11] for this code which is based on the chase algorithm [11].

Generally, long codes perform better than short codes since the number of channel errors per code word has less variance from the average. Short codes, on the other hand, perform better at high BERs since more errors can be corrected using a comparable code rate. The codes used in this work are rather short block codes. Table 3-2 gives a brief overview.

Table 3-2: Overview of the used codes

Code	*n*	*k*	*t*	*R*	*Generator Polynomial [octal]*
Golay(23,12)	23	12	3	0.52	5343
Golay(24,12)	24	12	3	0.5	– (non-cyclic)
BCH(15,11) (Hamming)	15	11	1	0.73	23
BCH(15,7)	15	7	2	0.47	721
BCH(15,5)	15	5	3	0.33	2467
BCH (7,4) (Hamming)	7	4	1	0.57	13
BCH(17,9)	17	9	2	0.53	727
shortened Hamming(13,8)	13	8	1	0.62	45

In Chapter 5 the BCH(15,7)-code was used as a model for other multiple error correction linear block codes. The complexity of this code is just that high to allow the analysis of the correlation between word errors in s_i and s_c, as presented in Section 5.3. In Section 5.5 an approximation of BERDc, the BER of s_c after decoding, will be derived from this analysis which applies to any code.

4. Signal Cancellation Enhancement by Coding of the Large Signal

Figure 4-1 shows the general idea of implementing coding to the DSR in order to enhance the signal suppression of the large signal. Evaluation of the performance of the large signal means evaluation of the performance of coding for a coherent BPSK-demodulator in a (heavy) co-channel interference, Additive White Gaussian Noise (AWGN) environment. That will be demonstrated in this chapter, a method will be presented to calculate upper bounds on the performance of soft decision decoding.

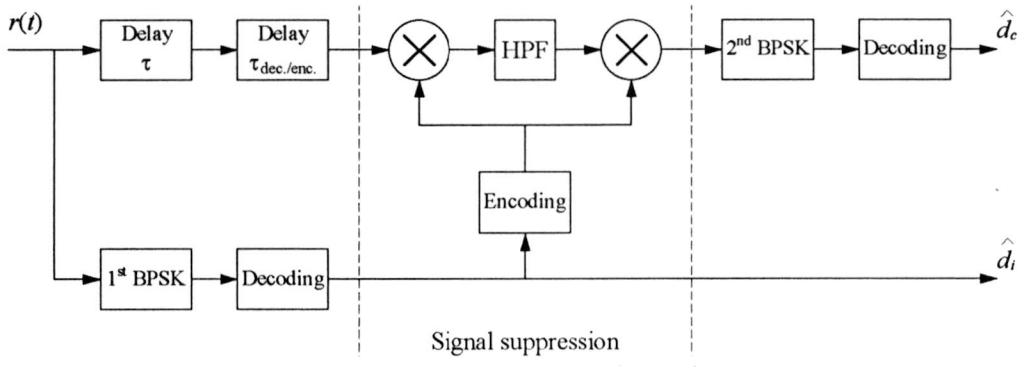

Figure 4-1: Coding implemented to the DSR to enhance signal suppression of the large signal

The estimated data of the large signal s_i is used in the DSR to suppress the dominating signal s_i, thus, each error occurring in s_i has large influence on the small signal s_c. The idea of this work is to use the estimated data of s_i after error correction for this signal suppression. This requires of course to encode the data stream $\hat{d}_i$ again in order to obtain an improved estimate of the transmitted (coded) bit-stream.

Obviously, the composite input signal must be delayed while this decoding/encoding is performed, which is one reason, why short block codes are used. We must not forget that the small signal also has to be decoded after the suppression and demodulation of the large signal, hence introducing additional delay to the small signal. In Chapter 5 we will analyse the performance of the small signal in presence of the remainders of s_i after signal cancellation, and we will see that performance enhancement of the large signal equivalently enhances the performance of the small signal, at low signal-to-interference ratios SIRi.

4.1 BER-Analysis for the Strong Signal without Coding

The BER calculation of the strong signal is repeated in great detail, since it is very instructive for the union bound calculations proposed in Section 4.4.

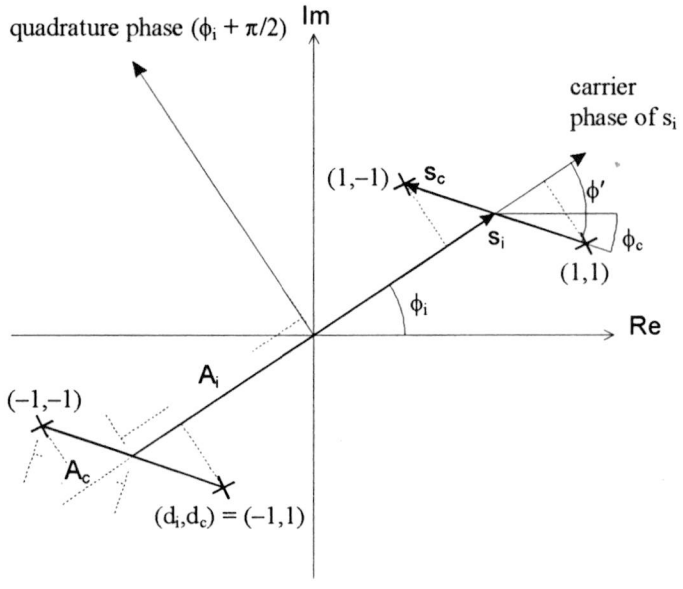

Figure 4-2: Phasor diagram of the composite input signal r(t)

Figure 4-2 shows the phasor diagram of the composite input signal r(t) of the DSR defined in Section 2.1. The DSR consist of two coherent BPSK demodulators. The first demodulator is captured by the large signal s_i and produces an estimate of the data signal $\hat{d}_i(t)$ consisting of symbols $\hat{d}_{i,j} \in \{-1,+1\}$. Errors in $\hat{d}_i(t)$ are caused by the presence of the small signal s_c, which is the interfering one for s_i, and by noise n(t).

To understand the calculation of the large signal BER (abbreviated by: BERi) it is helpful to have a closer look at Figure 4-2. The small signal is added to the large signal with a certain phase difference $\phi'(t)$, noise causes uncertainty of the signal vector positions which are marked by $\times$.

Coherent demodulation means demodulation by multiplication with a recovered carrier having the same phase as the carrier used for modulation. This multiplication of our input signal r(t) results in down conversion of r(t) to baseband, i.e. the carrier frequency $f_0 = 0$. Since we also multiply with the quadrature carrier, the carrier shifted by $\pi/2$, we get two signals $r_I(t)$ and $r_Q(t)$. We will need the quadrature component later, to demodulate the small signal.

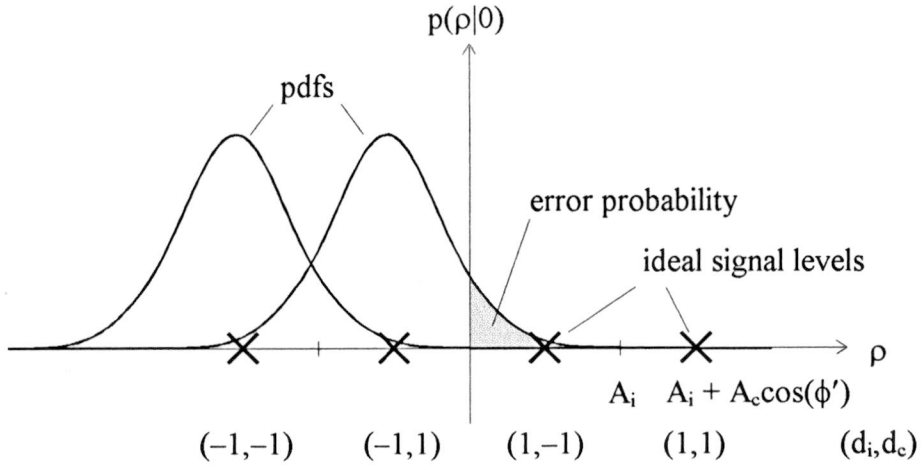

Figure 4-3: Pdfs of the in-phase component $r_I(t)$ of the composite input signal r(t); the transmitted bit was a one corrupted by s_c by a one and a minus one, where $\Delta' = 0$

The in-phase signal $r_I(t)$ consists of the large signal s_i corrupted by the component of s_c being in-phase with s_i, and by noise. This signal, which is simply the projection of the phasors in Figure 4-2 to the carrier phase, is shown in Figure 4-3 with its probability density function (pdf) for additive white Gaussian noise (AWGN). This pdf gives the uncertainty of the signal levels at the receiver. Expressed in algebraic terms we have

$$r_I(t) = A_i d_i(t) + A_c d_c(t)\cos(\phi'(t)) + n_I(t) \tag{4-1}$$

As inferred by Figure 4-3 it is sufficient to calculate BERi for one of the two possible transmitted symbols $d_{i,0} \in \{-1, 1\}$ because for each pattern of symbols $d_{c,0}$ and $d_{c,1}$ interfering on (–1), there is an equivalent one having the same effect on (1). Figure 4-4 shows the two symbols of s_c influencing on $d_{i,0}$.

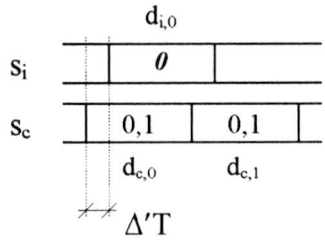

Figure 4-4: Two symbols of s_c interfering on $d_{i,0}$

The interference on the symbol (1), expressed by an "amplitude factor" is

$$r_I'(t) = A_i\left[1+\frac{d_c\kappa}{\sqrt{\psi_i}}\cos(\phi'(t))\right]+n_I(t) \tag{4-2}$$

where ψ_i = SIRi = A_i^2/A_c^2 is the signal-to-interference ratio of signal s_c influencing on s_i, i.e., $A_c = A_i/\sqrt{\psi_i}$. This formula shows clearly how the influence of s_c depends on the factor $\cos(\phi'(t))/\sqrt{\psi_i}$, i.e., even at bad signal-to-interference ratios SIRi = $\psi_i \approx 1$, demodulation of s_i is still possible with good performance if $\cos(\phi')$ is small ($\phi' \to \pi/2$). In this case the two signals s_i and s_c are orthogonal to each other.

On the other hand, Figure 4-3 (and Equation (4-2)) show that for SIRi = 1 and $\cos(\phi') = 1$, the level of the composite signal becomes zero when $d_{i,j} \neq d_{c,j}$, i.e., reliable estimation of the two signals is only (theoretically) possible when $d_{i,j} = d_{c,j}$, hence resulting in equal sequences $\hat{d}_i(t)$ and $\hat{d}_c(t)$, i.e., no information is provided by r(t) any more. This situation causes receiver outage, no matter whether coding is implemented or not. In Section 4.2 a receiver Outage Rate (OaR) will be defined which gives the probability of a receiver BER being greater than a certain threshold value. The dependency of BER of ϕ' will be used to calculate this probability. Since $\cos(\phi'(t))$ is symmetric to 0 and $\pi/2$, it is sufficient to survey this range $\phi' = [0 \ldots \pi/2]$.

The factor $\kappa = \{1,(1-2\Delta')\}$ describes the effect of the timing-difference $\Delta'T$ between the data signals $d_i(t)$ and $d_c(t)$. Due to this timing difference usually two symbols of s_c ($d_{c,0}$ and $d_{c,1}$) have influence on one symbol of s_i ($d_{i,0}$), with the effect that their influence decreases if $d_{c,0}$ and $d_{c,1}$ have opposite signs. Here it is sufficient to survey $\Delta' = |\Delta| \leq 0.5$: It can be easily seen that the influence of a symbol being delayed by $(1-\Delta')T$ is obtained by simply exchanging the bits $d_{c,0}$ and $d_{c,1}$ and calculating the influence of this pattern delayed by $\Delta'T$.

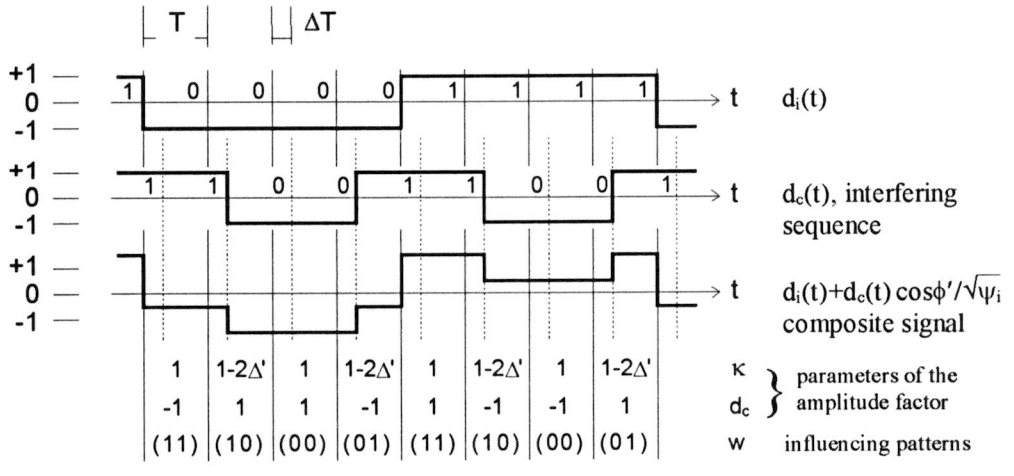

Figure 4-5: Effect of timing-difference Δ'T on the interference

Figure 4-5 shows a time-domain diagram of d_i and d_c and the four possible interference patterns $w = (d_{c,0},d_{c,1}) \in \{(00),(01),(10),(11)\}$, where $\cos(\phi'(t)) / \sqrt{\psi_i} = 0.5$, $A_i = 1$. We can see that the signal amplitude of the composite signal and therefore it's SNR is even raised by the influence of bits $d_{c,j}$ having equal signs than $d_{i,j}$, hence resulting in a lower probability of error (see also the left pdf in Figure 4-3). In the calculation of BERi we have to consider these four cases having different amplitude factors and appearing with equal probabilities.
The error probability p of coherent demodulation of a binary antipodal PSK-modulated signal without interference, is given by

$$p = \mathrm{Q}\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{4-3}$$

where $2E_b/N_0$ is the signal-to-noise ratio (Note: SNR = $\gamma = E_b/N_0$). The Q-function is the integral over the Gaussian probability density function, defined by

$$\mathrm{Q}(z) = \frac{1}{\sqrt{2\pi}} \int_z^\infty e^{-\lambda^2/2} d\lambda \tag{4-4}$$

A conditional error probability (depending on ϕ' and Δ', and SNRi, SIRi) can be calculated in a very similar way to the calculation of the BER in (4-3) by using the amplitude factors defined in (4-2). This error probability is the integral over part of the pdfs being on the wrong side of the decision threshold as shown in Figure 4-3. The amplitude factor shifts the pdf to the appropriate position for each pattern w. The error probability is the average of the error probabilities of the 4 cases w, expressed by d_c and κ [1]

$$P(\varepsilon_i|\phi',\Delta') = BERi(\phi',\Delta') = \frac{1}{4}\sum_{d_c\kappa} \mathrm{Q}\left[\sqrt{2\gamma_i}\left(1+\frac{d_c\kappa}{\sqrt{\psi_i}}\cos\phi'\right)\right] \tag{4-5}$$

Finally, the averaged BER of s_i is calculated by averaging (4-5) over ϕ' and Δ'

$$\overline{BERi} = \frac{1}{\pi}\int_0^{2\pi}\int_0^{0.5} P(\varepsilon_i|\phi',\Delta')\, d\Delta'\, d\phi' \tag{4-6}$$

---

[1] MATLAB function: *PeiPecCd*

Calculates BERi and BERc. A description can be found in Section 5.1.

4.2 Definition of a Receiver Outage Rate

When both components of the DSR input signal, the large signal s_i and the small signal s_c, have about equal strength (E_{bi}/E_{bc} = SIRi $\approx$ 1 = 0dB), equal carrier phase ($\phi' = 0$) and equal bit timing ($\Delta' = 0$), the receiver cannot lock on either signal, and the estimated data will be of no use.

Fortunately, if there is some phase difference ϕ' and/or timing difference Δ' present between the two signals, the influence of the interference will decrease and demodulation with good bit error rates will be possible again. Figure 4-6 shows BERi, the bit error rate of the strong signal s_i, with respect to ϕ' and Δ'. This plot has symmetries at $\phi' = \{0, \pi/2\}$ and $\Delta' = \{0, 0.5\}$, therefore, it is sufficient to consider this part of the ϕ'-Δ'-plane.

Receiver outage is defined by a demodulator BER being worse than a specific threshold BER_{max}. This threshold is a plane having a constant value of BERi in the 3-D plot of Figure 4-6, the intersection of such planes with our plot results in contour lines as we know them from geographical maps.

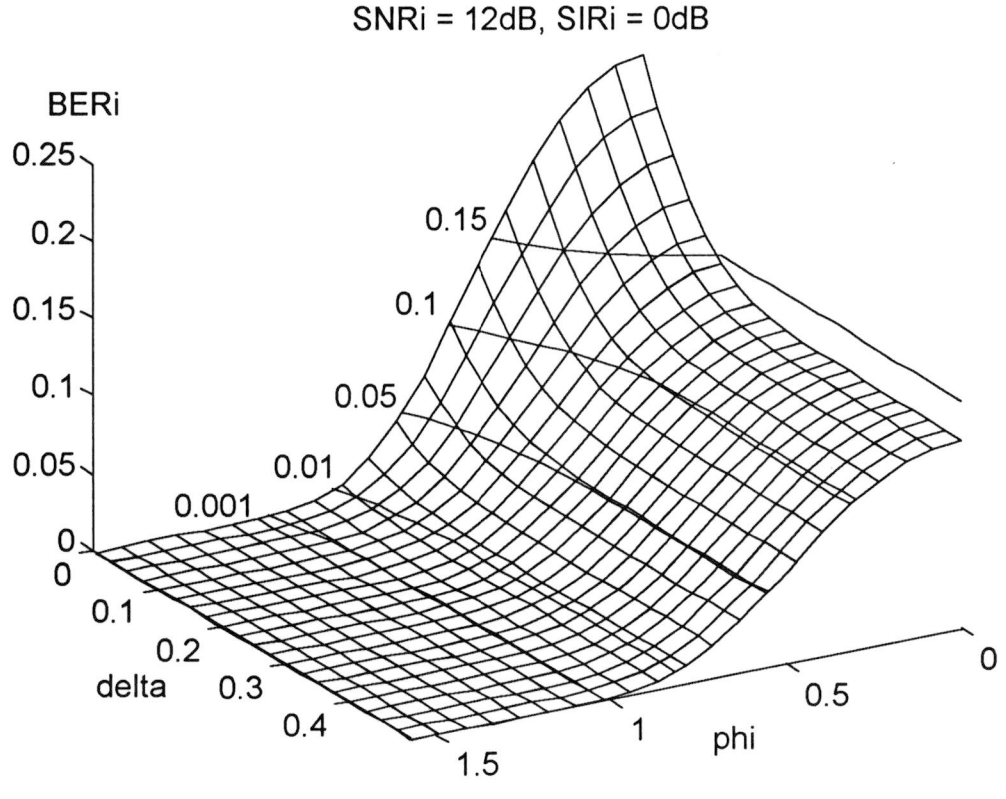

Figure 4-6: Large signal bit error rate of the DSR against ϕ' and Δ'; contour lines used to define outage areas for a certain threshold error rates

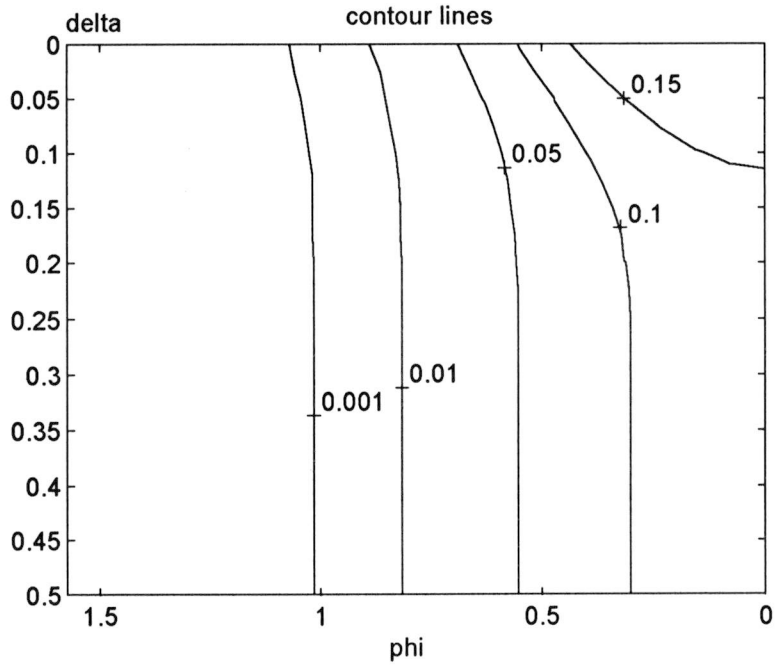

Figure 4-7: Top view of Figure 4-6 (BERi); the outage rate is defined by the fraction of the area to the right of any contour line to the area of the whole ϕ'-Δ'-plane ($0 \le \Delta' \le 0.5$ and $0 \le \phi' \le \pi/2$)

In a bird's eye view (Figure 4-7) one can see easily that the ϕ-Δ-plane is split in two parts by any contour line. This results in one area where BER is greater than BER_{max}, and in another where BER is smaller.

Assuming random phase and timing differences (ϕ' and Δ'), we define the receiver outage rate OaRi,c by the fraction of the area where BER is worse than BER_{max}, to all the possible cases, i.e. the fraction of the area to the right of any contour line, to the area of the whole ϕ'-Δ'-plane ($0 \le \Delta' \le 0.5$ and $0 \le \phi' \le \pi/2$) [2]:

$$OaR = \frac{(\phi', \Delta') \text{ where } BER(\phi', \Delta') > BER_{max}}{\text{all } (\phi', \Delta')} \tag{4-7}$$

In a similar fashion, this definition of a receiver outage rate can be used as a performance measure for coding algorithms applied to the DSR, where the threshold level BER_{max} is the maximum acceptable decoder output BER (BERD). It will be used later in this chapter to compare the union bound calculations of BERD for several codes utilising hard and soft decisions at SIRi = 0dB (Section 4.5.1). Even the best decoding scheme cannot improve the

---

[2] MATLAB function: *[outage,opoly] = OutPoly(bermat,bermax)*

Calculates the intersections of the columns of *bermat* with *bermax*. This intersections are returned by the vector *opoly* (normalised to [0 ... 1]), the outage rate (*outage*) is obtained by averaging over this vector.

performance of the DSR at some special cases, e.g., SIRi = 1, $\phi = 0$ and $\Delta = 0$, since no information is provided by r(t) any more (see 4.1), but it can be seen as a goal to reduce the probability of these cases to a minimum, i.e., to reduce the receiver outage rate to a minimum. Results are given in Table 4-1.

Table 4-1: Outage rates in [%] for the BER threshold levels shown in the plots:

BER threshold:	*0.001*	*0.01*	*0.05*	*0.1*	*0.15*
outage rate OaRi [%]:	65.1	52.5	36.6	22.0	4.0

4.3 Hard Decision Decoding

The principles of the performance evaluation of hard decision decoding were already described in Section 3.2. Equation (3-3) is used in this chapter to calculate exact word error rates, taking $p = \Pr(\varepsilon_i|\phi',\Delta')$, which is the error probability of our co-channel interference AWGN-channel (Equation (4-5)). This probability is dependent of the parameters SNRi, SIRi, ϕ' and Δ'.
The loss of signal-to-noise ratio due to the transmission of redundancy, was applied to this calculation by substituting for $\gamma_i \rightarrow \gamma_i \cdot R$, hence calculating a signal-to-noise ratio per coded bit and keeping the signal-to-noise ratio per information bit constant. This enables comparison between different codes and also comparison to the uncoded case.
Finally, the decoder output BER is calculated using the approximation defined by Equation (3-5).
Computational results will be given in Section 4.5. We will refer to the results obtained by the method described in this section as "exact" results for the BSC.

4.4 Union Bound Computations of BER, and Soft Decision Decoding

The basic idea of a union bound calculation is to express an event as the union of several subevents, where the probability of that event occurring is always less than or equal to the sum of the probabilities of all subevents. This sum is an upper bound since we count the probabilities of overlapping subevents more than once. Therefore, this sum can get even greater than 1.
Applying linear block codes, an error rate can be computed by considering the effect on the *all-zero code word transmitted*. In case of hard decisions, a decoding error is made when more than t errors (ones) are added by the channel (t is the maximum number of correctable errors per code word and satisfies the condition $t \leq (d_{min}-1)/2$, d_{min} being the minimum Hamming

distance of the applied code). According to the definition of the event B_j' made in Section 3.2.1 we can also say that an error will be made if the received sequence falls within the correction portion of any code word of weight-j $\neq 0$, resulting in the output of j errors, i.e. the distance of the received sequence to a weight-j code word was less or equal t.

The subevents used to calculate the union bound are defined by the received sequence having a smaller distance to a weight-j code word than to the transmitted all-zero word, i.e. a decoding error is made, when more than half of the bits used to distinguish between two code words are in error [11].

For example, assume we have a code with $d_{min} = 3$ and $t = 1$, i.e. one error can be corrected, and we wish to determine the probability that the transmitted all-zero word gets corrected to a weight-5 word. The actual error event occurs if the received sequence has a distance $d \leq t$ to this weight-5 sequence, i.e., 4 – 6 errors have to occur within the received code sequence of length n. The union bound subevent will occur if the received sequence has a smaller distance to the weight-5 sequence than to the all-zero sequence, thus we will assume an error if 3, 4, or 5 errors occur within the sequence of length 5 which is separating the two sequences.

At this point it can be seen easily that we are dealing with an overbound, since the received sequence being corrected to weight-j might have a distance $j/2$ greater than t to the weight-j code word.

Let us define the following events:

event A_0: the all-zero code word was transmitted

event B_j'': the distance between the received sequence and a code word of weight-j was smaller than the distance between the received sequence and the all-zero code word. This is the subevent used to calculate the union bound.

Using these definitions the decoder output bit error probability is upper bounded by [3]

$$BERD \leq \sum_{j=1}^{n} \frac{j}{n} n_j \Pr(B_j'' | A_0) \tag{4-8}$$

This expression looks very similar to Equation (3-7) with the difference that $\Pr(B_j'|A_0)$ is computed by summing over all patterns that are t errors or less from the given code word and $\Pr(B_j''|A_0)$ is computed by summing over all patterns that differ in j/2 or fewer of the j nonzero

[3] MATLAB function: *uBound*

Performs this calculation for unquantised soft decision decoding, the calculation of $\Pr(B_j''|A_0)$ is carried out by the function *uquant1*.

digits of the given code word. For the BSC (hard decisions, see 4.1) this probability is calculated by

$$\Pr(B_j''|A_0) = \begin{cases} \sum_{i=(j+1)/2}^{j} \binom{j}{i} p^i (1-p)^{j-i} & , j \text{ odd} \\ \frac{1}{2}\binom{j}{j/2} p^{j/2}(1-p)^{j/2} + \sum_{i=j/2+1}^{j} \binom{j}{i} p^i (1-p)^{j-i} & , j \text{ even} \end{cases} \tag{4-9}$$

where p is the crossover-probability of the BSC.

At low error rates p the contribution of error terms of weight greater than t + 1 becomes very small and thus our overbound is a good approximation of the exact computation using (3-7).

The advantage of the union bound is that it is much simpler to calculate $\Pr(B_j''|A_0)$ than to calculate $\Pr(B_j'|A_0)$, particularly for the case of soft decisions, assuming maximum likelihood decoding. This will be demonstrated in the following sub-sections.

4.4.1 Soft Decision Decoding

In this section the principal advantage of soft decision decoding is explained.

In the case of hard decisions made by the demodulator, there is no information provided whether the received bit was reliably detected or almost equal to the decision threshold. This channel model including modulator and demodulator is know as Binary Symmetric Channel (BSC), having a certain error probability p for each transmitted bit.

In the case of soft decisions the receiver gives a "confidence number" that specifies how close the received symbol actually was to the hard decision threshold. We get a binary input analogue output channel if no quantisation is performed, and a binary input Q-ary output channel if quantisation to Q levels is performed by the demodulator.

To evaluate the benefit of soft decision decoding we have to define a suitable function to calculate the distance between the received sequence ρ and any possible code word $\mathbf{S}_l$. A maximum likelihood decoder selects the code word $\mathbf{S}_l$ which minimises the distance to the received sequence. In case of quantised demodulator outputs the difference of the output value to one of the possible code symbols {0,1} is called metric, i.e., each received symbol has two metric values, where the smaller one indicates the code symbol that was transmitted with a higher probability.

To figure out the advantage of soft decision decoding it is instructive to consider the following example introducing a binary input, 8-ary output channel with linearly spaced quantisation levels and metrics {0,1,..7}. Let the minimum Hamming distance between two code words be

3, then the minimum distance using those metrics is 21 (ideal distance between two bits 0 and 1 is 7; 3·7 = 21). Therefore, any received sequence which differs from the transmitted sequence by a total metric of 10 or less is uniquely decodeable, i.e., we can correct two errors having metrics of 4 or 5 in stead of one error in the hard decision case. Considering Figure 4-8 we can see that the probability of these errors, being close to the decision threshold, is much higher than the probability of errors having metrics of 6 or 7. Thus, the soft decision decoder allows correction of errors which could not have been corrected using a hard decision decoder.
An appropriate distance function for the case of analogue voltages provided by the demodulator is obtained by the calculation of Euclidean distances between the received sequence ρ consisting of n symbols ρ_i and any code word $\mathbf{S}_l$ consisting of n Symbols S_{li}, i.e., correctly decodeable sequences are lying within regions in an n-dimensional space around the code words [11]:

$$d_l^2 = \sum_{i=1}^{n} (\rho_i - S_{li})^2 \tag{4-10}$$

In practical communications systems we will rather use a relatively coarse quantisation scheme than analogue voltages. Arguments for this statement are the number of bits required to represent each quantised symbol ρ_i, the time required for the calculation of the decision variables, and last but not least, the implementation using digital signal processors, which implies quantisation of the demodulator output, anyway. The argument to use a simple quantisation scheme in order to minimise the cost of the ADC is obsolete, because an ADC providing a fine resolution is needed to implement the principle of the DSR with a DSP.
Nevertheless, we will see that an 8-level scheme already shows near-to-optimum performance compared to unquantised case. The fine quantisation provided by the ADC of the DSP allows a different method of optimisation, which will be proposed in Section 4.6.
The 8-level scheme with linear spaced decision thresholds and linear metrics, applied to the union bound computations, will be discussed in the following section.

4.4.2 Union Bound on Quantised Soft Decision Decoding

To characterise a binary input, Q-ary output channel, we have to calculate a set of transition probabilities $\Pr(\rho|x)$. $\Pr(\rho|x)$ is the probability that the output voltage falls within the quantisation range ρ given that the symbol $x \in \{0,1\}$ was transmitted. Since we are dealing with a symmetric channel, we can write $\Pr(\rho|0) = \Pr(Q - 1 - \rho|1)$ abbreviated by P_ρ. Q is the number of quantisation levels. Figure 4-8 shows the pdf of the demodulator output and the

quantisation levels used in the definition of the binary input, Q-ary output channel, where Q = 8. Transmitted signal was a (0), corrupted by $d_c = (0)$ and $d_c = (1)$, where $\Delta' = 0$.

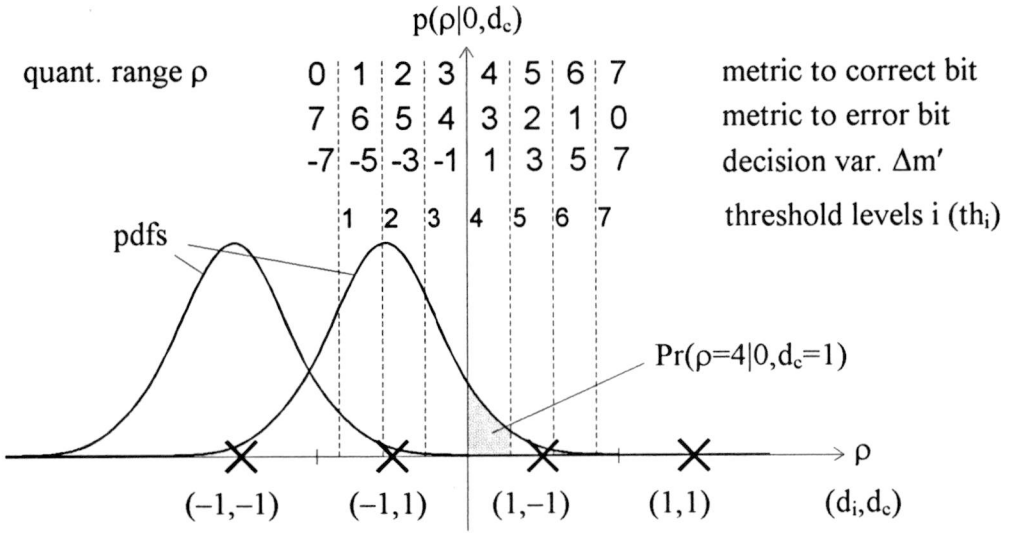

Figure 4-8: Pdf of the demodulator output voltage with linearly spaced quantisation thresholds to obtain a binary input, 8-ary output channel

The transition probabilities are computed by integrating the pdf over each quantisation range ρ. Similar to equation (4-5) we have to consider the four cases of interference by the small signal s_c. The (Q – 1) threshold levels th_i are applied to this equation by shifting the pdfs to the left or to the right again (see equation (4-5), Section 4.1). The transition probabilities are

$$
\begin{aligned}
P_{Q-1} &= \Pr(\rho = 7|0,\phi',\Delta') = \frac{1}{4}\sum_{d_c\kappa} Q\left[\sqrt{2\gamma_i}\left(1+\frac{d_c\kappa}{\sqrt{\psi_i}}\cos\phi' + th_{Q-1}\right)\right] \\
P_{\rho} &= \Pr(\rho|0,\phi',\Delta') = \frac{1}{4}\sum_{d_c\kappa} Q\left[\sqrt{2\gamma_i}\left(1+\frac{d_c\kappa}{\sqrt{\psi_i}}\cos\phi' + th_{\rho}\right)\right] \\
&\quad - \frac{1}{4}\sum_{d_c\kappa} Q\left[\sqrt{2\gamma_i}\left(1+\frac{d_c\kappa}{\sqrt{\psi_i}}\cos\phi' + th_{\rho+1}\right)\right] \qquad \rho = 1,2,..,Q-2 \\
P_0 &= \Pr(\rho = 0|0,\phi',\Delta') = 1 - \frac{1}{4}\sum_{d_c\kappa} Q\left[\sqrt{2\gamma_i}\left(1+\frac{d_c\kappa}{\sqrt{\psi_i}}\cos\phi' + th_1\right)\right]
\end{aligned}
\tag{4-11}
$$

In the model used in this section, the number of the quantisation range is simply assigned to the metric, i.e., $m = \rho \in \{0,1,2,..7\}$. The computation of optimum metrics will be demonstrated in Section 4.6.

To apply the union bound on the quantised channel specified by the probabilities P_ρ, we have to calculate $Pr(B_j''|A_0)$, the probability that the distance between the received sequence and a code word of weight-j is smaller than the distance between the received sequence and the all-zero code word.

First, assume the symbol (0) was transmitted and the symbol ρ was received. We have two metrics $m = \rho$ and $m = (7 - \rho)$ assigned to the possible transmitted symbols (0) and (1), respectively. By calculating the difference between these two metrics $\Delta m' = \rho - (7 - \rho)$ we get a decision variable $\Delta m'$ with $Pr(\Delta m') = Pr(\rho|0) = P_\rho$. $\Delta m'$ is in the range of $\{-7..7\}_{(\text{odd values})}$, where a positive value indicates that the symbol (1) was rather transmitted than the symbol (0), i.e., a positive value of $\Delta m'$ is indicating an error. Regarding the decision variable $\Delta m'$ as a random variable we can plot a discrete pdf with the probabilities $Pr(\rho|0) = P_\rho$ as shown in Figure 4-9a). The total probability of an error is simply the sum over the positive half of this pdf.

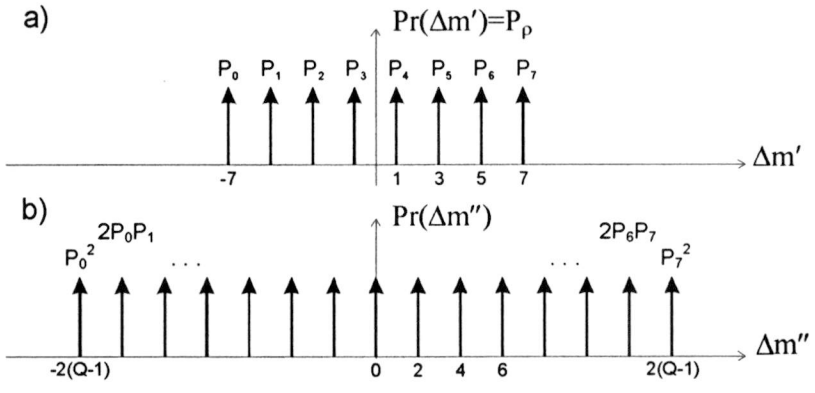

Figure 4-9: a) Discrete probability density function for the metric differences for the 8-level discrete channel; b) Pdf of the decision variable Δm'', calculated by the convolution of Pr(Δm') with itself

In a second step, let us consider two sequences of length n having a weight distance of 2. The all-zero word was transmitted and the sequence ρ consisting of n symbols ρ_i was received. To decide between the correct (all-zero) word and any other code word $\mathbf{S}_l$, the maximum likelihood decoder computes distances to the received sequence by summing the metrics.

$$d_l = \sum_{i=1}^{n} m_{li} \qquad m_{li} = \begin{cases} \rho_i & S_{li} = (0) \\ 7 - \rho_i & S_{li} = (1) \end{cases} \tag{4-12}$$

An error will be made if the distance to any weight two sequence $\mathbf{S}_l''$ is smaller than the distance to the all-zero word, which is computed by $d_0 = \sum_{i=1}^{n} \rho_i$. We can calculate a decision variable $\Delta m''$ in a similar fashion to $\Delta m'$ by calculating the difference between two distances

$$\begin{aligned} \Delta m_l'' = d_0 - d_l &= \sum_{i=1}^{n} (\rho_i - m_{li}) = \sum_{i=1}^{n} \begin{cases} 0 & S_{li}'' = (0) \\ \rho_i - (7 - \rho_i) & S_{li}'' = (1) \end{cases} \\ &= \sum_{S_{li}''=(1)} [\rho_i - (7 - \rho_i)] = \sum_{S_{li}''=(1)} \Delta m_i' \end{aligned} \tag{4-13}$$

Since the (n – 2) zeros occurring in both words have the same contribution on the distances computed by (4-12), the probability of error is simply the probability of error in deciding between two possible transmitted sequences, either 2 zeros or 2 ones. Let us define $S_{l0}'' = (1)$ and $S_{l1}'' = (1)$ being the ones in the code word $\mathbf{S}_l$, then $\Delta m_l'' = \Delta m_0' + \Delta m_1' \in \{-2(Q-1)...2(Q-1)\}_{(\text{even values})}$, i.e., the decision variable is simply the sum of two decision variables of single symbols, occurring with probabilities: $\Pr(\Delta m_0') = \Pr(\rho_0|0)$ and $\Pr(\Delta m_1') = \Pr(\rho_1|0)$. Assuming statistical independence between ρ_0 and ρ_1, we have $\Pr(\Delta m_0', \Delta m_1') = \Pr(\rho_0|0)\Pr(\rho_1|0)$. Since we can usually find more than one combination of ρ_0 and ρ_1 (with $\Delta m_0'$ and $\Delta m_1'$) resulting in one certain decision variable $\Delta m_l''$, we have to sum up these probabilities to obtain $\Pr(\Delta m_l'')$.

$$\Pr(\Delta m_l'') = \sum_{\Delta m_0', \Delta m_1'} \Pr(\Delta m_0', \Delta m_1') \Big|_{\Delta m_0' + \Delta m_1' = \Delta m_l''} \tag{4-14}$$

For example, to compute $\Pr(\Delta m'' = 2)$ we have to sum up the following product terms: $\Delta m_0' + \Delta m_1' = 2$; $2\rho_0 + 2\rho_1 - 14 = 2$; $\rho_0 = 8 - \rho_1$; i.e., $(\rho_0,\rho_1) = \{(1,7),(2,6),(3,5), \ldots ,(7,1)\}$, hence, $\Pr(\Delta m'' = 2) = P_4^2 + 2(P_1P_7 + P_2P_6 + P_3P_4)$. It can be shown easily that the convolution of the discrete pdf of $\Delta m'$ with itself exactly results in the pdf of $\Delta m''$ (see Figure 4-9b)). The probability of a sequence error is, again, the sum over the positive half. This time, the probability $\Pr(\Delta m'' = 0)$ is not equal to zero and it has to be added after division by two, since a decision variable of zero ($\Delta m'' = 0$) means that no decision for either sequence was possible, thus resulting in an error with a chance of 50 %.

This method generalises to weight-j sequences by performing the (j – 1)-fold convolution of the pdf of $\Delta m'$ with itself. $\Pr(B_j''|A_0)$ being the sum over the positive half of $\Delta m^{(j)}$ [4]

[4] MATLAB function: *quant1*

$$\Pr(B_j''|A_0) = \sum \Pr(\Delta m^{(j)} > 0) + 0.5\Pr(\Delta m^{(j)} = 0) \quad (4\text{-}15)$$

Computational results are shown in Section 4.5.

4.4.3 Union Bound on Unquantised Soft Decision Decoding

To apply the union bound to soft decision decoding utilising unquantised demodulator outputs, we must, again, calculate $\Pr(B_j''|A_0)$, the probability that the distance between the received sequence and a weight j code word is less than the distance between the received sequence and the transmitted all-zero code word. In case of unquantised demodulator outputs an appropriate distance function was proposed by Equation (4-10), the Euclidean distance between two points in an n-dimensional space.

Observing two code words, the transmitted all-zero word (00...0) and the weight-j word (11..1 00...0) consisting of j ones and (n – j) zeros, and their distances to the received sequence calculated by (4-10), we can see that the contribution of the (n – j) zeros to the distances is the same for both words. Since we are only interested in the difference between the distances to get a decision variable, these (n – j) zero-digits can be neglected and $\Pr(B_j''|A_0)$ is the probability of error in deciding between two possible transmitted words, either j zeros or j ones.

Without any interference the error probability would simply be the probability that a transmitted sequence consisting of j zeros was received as a sequence of j ones. This probability can be calculated by equation (4-3) but with a factor of j more energy [11, p. 29]

$$\Pr(B_j''|A_0, SIRi = \infty) = Q\left(\sqrt{2j\,E_{bi}R/N_0}\right) \quad (4\text{-}16)$$

Considering the effects of interference, we can see that our sequence is affected by s_c in a similar fashion as described above for the computation of the conditional error probability, Equation (4-5).

Figure 4-4 illustrates that each symbol of our sequence is influenced by two symbols of s_c, i.e., each symbol is influenced by one of four patterns $w \in \{(00),(01),(10),(11)\}$. That means, as we are dealing with a sequence of j-symbols, that one of 4^j different pattern sequences $\boldsymbol{v}$ =

Calculates the union bound on BERD for quantised soft decision decoding as given in Equation (4-8). The transition probabilities defined by (4-11) are calculated, $\Pr(B_j''|A_0)$ is obtained by performing the convolution of

($w_1w_2w_3$ ... w_j) is affecting the probability calculated by (4-16). Statistical independence between any two patterns (w) was assumed, which is allowed in mean over many calculations. The influence of these pattern sequences (ν) is expressed by a weight factor if_ν in the range [–1 ... 1], where (–1) stands for the all-zero sequence and (1) for the all-one sequence, for constructive and destructive interference, respectively. Referring to the calculation of (4-5) these pattern sequences can be seen as amplitude factors that can be applied to (4-16) easily

$$\Pr(B_j''|A_0,\phi',\Delta') = \frac{1}{4^j}\sum_{\nu=1}^{4^j} Q\left[\sqrt{2j\gamma_i R}\left(1+\frac{if_\nu}{\sqrt{\psi_i}}\cos\phi'\right)\right] \tag{4-17}$$

Fortunately, many of the 4^j patterns (ν) have the same weight if_ν and, therefore, we can combine them as shown in the Appendix, Section 4.8. The number of 4^j patterns reduces to V patterns of weight if_ν', each one occurring with a probability Pr(v'). We have

$$\Pr(B_j''|A_0,\phi',\Delta') = \sum_{\nu=1}^{V}\Pr(\nu')\cdot Q\left[\sqrt{2j\gamma_i R}\left(1+\frac{if_\nu'}{\sqrt{\psi_i}}\cos\phi'\right)\right] \tag{4-18}$$

For a weight-7 sequence, $4^j = 16384$ reduces to V = 120, hence, speeding up the computation tremendously. Further simplification was achieved by taking a fixed set of interference weights $if_\nu'' \in [-1 \ldots 1]$ and calculating the probabilities Pr(v''). This can be done without loss of accuracy, since the increments of if_ν'' can be selected to be the same than the increments of if_ν' appearing due to a certain value of Δ' [5].

4.5 Computational Results

In this section computational results are given for union bound computations for several channel quantisation schemes. These schemes are defined by the number of quantisation levels Q and by the metrics assigned to the quantised demodulator outputs. Channel models with linear spaced metrics and the following quantisation thresholds are evaluated:

---

these transition probabilities.

[5] MATLAB function: *uquant1*

This function generates the pattern sequences ν', quantifies and accumulates them in order to obtain Pr(v''). The Q-function is evaluated for the influencing weights if_ν'' and $\Pr(B_j''|A_0)$ is calculated.

Table 4-2: Channel definitions

Nr.	*Channel Name*	*Nr. of Quant. Levels (Q)*	*Normalised Thresholds* $(th_i)A_i$
a)	BSC	2	{0}
b)	Erasure Ch.	3	{-0.15,0.15}
c)	4-Level Ch.	4	{-4/7,0,4/7}
d)	8-Level Ch.	8	{-6/7,-4/7,-2/7,0,2/7,4/7,6/7}
e)	Unquantised Ch.	∞	–

To compare the performance of different codes, a signal-to-noise ratio per coded bit was calculated by multiplying SNRi with the code rate R = k/n ($E_{ci}/N_0 = E_{bi} \cdot R/N_0$, see Section 3.2).

4.5.1 Performance at SIRi = 1 = 0dB

One of the goals of this graduation work was to evaluate the performance gain of the coding-improved DSR at SIRi = 0dB, i.e., both signals having about equal strength.
There are two facts that make the outage rates defined in Section 4.2 ideal to evaluate the performance of soft decision techniques in that situation:

- no demodulation is possible at $\phi' \approx 0$ and $\Delta \approx 0$; coding cannot improve this situation (see Section 4.1).
- union bounds provide a good approximation of the decoder output BER for small values of input BER. At high input BERs the output error rates calculated by union bound techniques are getting even bigger than 1 (see Section 4.4). Using outage rate calculations, only small input BERs (resulting in output BERs, BERDi < BER_{max}) are included in the results.

The last statement has to be revised: Linear interpolation was used to calculate the intersection of BERDi(ϕ') with BER_{max}, i.e., one value of BERDi being greater than BER_{max} was used. Since this value tends to be greater than the exact one, the outage rates given in Table 4-3 are rather too high [6].

[6] MATLAB program: *CalUbnd1*

Calculates OaR-results for union bound calculations and several quantisation schemes. The program also provides the functions described above with the required code-parameters like n_j. Several codes are used.

Table 4-3: OaR-Results of the large signal for SIRi = 0dB, applying several linear block codes and decoding techniques.

Parameters			Coding:		Union Bound Computations (OaRs)				
SNRib [dB]	BERmax	OaR w/o coding	Code	BSC "exact"	a) BSC	b) Q = 3	c) Q = 4	d) Q = 8	e) unquant.
6	0.01	0.79	BCH(15,7)	0.76	0.83	0.75	0.64	0.59	0.56
	0.001	1.00		1.00	1.00	1.00	0.81	0.76	0.74
12	0.01	0.53		0.47	0.51	0.44	0.34	0.28	0.23
	0.001	0.65		0.59	0.60	0.54	0.48	0.44	0.42
18	0.01	0.37		0.32	0.35	0.26	0.22	0.09	0.07
	0.001	0.45		0.40	0.41	0.36	0.33	0.29	0.27
12	**0.01**	**0.53**	Golay(24,12)	*0.44*	*0.51*	0.43	0.35	**0.26**	**0.19**
	0.001	0.65	R = 1/2	*0.54*	*0.58*	0.50	0.45	0.39	0.36
6	0.01	0.79	Golay(23,12)	0.68	0.81	0.74	0.63	0.59	0.55
	0.001	1.00		0.86	1.00	0.87	0.75	0.71	0.68
12	0.01	0.53		0.43	0.51	0.44	0.36	0.29	0.25
	0.001	0.65		0.53	0.58	0.51	0.45	0.41	0.38
18	0.01	0.37		0.29	0.35	0.27	0.24	0.11	0.08
	0.001	0.45		0.37	0.40	0.34	0.32	0.27	0.24
12	0.01	0.53	BCH(15,5)	0.42	0.44	0.37	0.16	0.10	0.07
	0.001	0.65		0.55	0.56	0.50	0.39	0.34	0.30
12	0.01	0.53	BCH(15,11)	0.49	0.58	0.52	0.50	0.45	0.44
	0.001	0.65	Hamming	0.60	0.65	0.60	0.59	0.55	0.52
12	0.01	0.53	BCH(7,4)	0.48	0.54	0.49	0.43	0.39	0.38
	0.001	0.65	Hamming	0.61	0.65	0.61	0.57	0.53	0.52
12	0.01	0.53	BCH(17,9)	0.46	0.52	0.45	0.38	0.32	0.29
	0.001	0.65	nonprimitive	0.57	0.60	0.54	0.49	0.45	0.43
12	0.01	0.53	shortened	0.51	0.55	0.50	0.45	0.41	0.39
	0.001	0.65	Ham.(13,8)	0.62	0.65	0.60	0.56	0.52	0.51

The parameters used in the computations are: SIRi = 1 = 0dB, SNRi = 12dB for all codes, SNRi = {6, 12, 18} dB for the BCH(15,7) and the Golay(23,12) code. Outage thresholds $BER_{max} = \{0.01, 0.001\}$. These parameters are listed in the first two columns of Table 4-3.

The next column gives the large signal outage rates (OaR) of the DSR without any coding, computed in the way described in 4.2. At SNRi = 12dB we have outage rates of $53\%|_{BERmax=0.01}$ and $65\%|_{BERmax=0.001}$. This is the values that are to be improved.

The column BSC "exact" gives OaRs for the BSC as defined by (4-5). A word error rate was computed using (3-3), the decoder output BER was obtained by the approximation (3-5). These results clearly show that some OaR-enhancement is gained by almost any coding scheme.

In the next four columns, named a), b), c) and d), results of union bound computations are listed, utilising maximum likelihood decoding with several quantised channel models, Q is the number of quantisation levels. The union bound computations are described in 4.4 and 4.4.2.

Finally, column e) gives outage rates for the unquantised channel model applied to the union bound computations as explained in 4.4.3.

The set of codes is described in Section 3.3: rather short, linear block codes were used.

The Golay(24,12) code will be used to explain the results, since it has a rate R = 1/2 and, therefore, can be compared to rate R = 1/2 convolutional coding. The union bound computations show that large improvement of outage rate (OaR) is possible by increasing the number of quantisation levels (OaR = 44% with hard decision decoding ("exact"); OaR = 26% 8-level scheme; OaR = 19% unquantised channel, where $BER_{max} = 0.01$). It should be possible to reduce the gap between the 8-level scheme and the unquantised scheme by optimisation of the quantisation levels as proposed in Section 4.6. Comparing the pairs of results for the BSC (BSC "exact" and a)), we can see that they match quite well. The union bound OaR is (slightly) higher because of the union bound being an overbound.

Comparing the OaR-results to the results obtained without coding (Table 2-1), we can estimate a coding gain which is the factor of energy that can be saved by using coding in order to achieve the same performance. For example, the OaR for Golay(24,12)-coding utilising unquantised soft decision decoding is 19% at SNRi = 12dB. From Table 2-1 we can see that SNRi = 30dB is required to obtain the same OaR without coding, thus the coding gain is 18dB. This is a fantastic value, however, it won't be possible to achieve such gains for all situations in practice. The gains for $BER_{max} = 0.001$ are ~ 10dB for this coding scheme, which is still a very good value.

4.5.2 Averaged BERD-Results for Varying Signal-to-Interference Ratio

Figure 4-10 and Figure 4-11 [7] show BERD-results averaged over ϕ' and Δ'. Curves a) – e)): union bound computations with a varying number of soft decision quantisation levels; curve f): BERD derived from the exact calculation of WER by using approximation (3-5); and curve g): Results for the BSC without coding to evaluate the gain.

The points marked on these curves by small circles 'o' and crosses '+' indicate the SIRi and corresponding averaged BER-values where the BER for the worst case ($\phi' = 0$ and $\Delta' = 0$) equals a certain threshold BER_{max}. According to our definition of outage rate given in Section 4.2, the outage rate OaR = 0 for SIRi values greater than the indicated ones. BER_{max} was taken 0.01 and 0.001 for the positions marked by 'o' and '+', respectively.

These points also mark the ranges where union bound computations provides good results. This can be seen easily if we compare curves a) and f), the union bound- and "exact" results for the BSC. Especially the results given for the BCH(15,7)-code (Figure 4-10) match very well at the right of these marks, i.e., at higher signal-to-interference ratios.

---

[7] MATLAB program: *CalUbnd6*

Calculates averaged BERD-results against SIRi as shown in Figure 4-10 and Figure 4-11.

Comparing those two curves of the Golay(24,12)-code (Figure 4-11) we see that the union bound results are in general worse than the "exact" results. The reason for this discrepancy is that the union bound calculations give an upper bound on the error rate due to the definition of the sub-events given in Section 4.4. The curves match better for the BCH(15,7)-code because the approximation (3-5) used to obtain the BER from the exact WER also gives an upper bound on BER for non-perfect codes like this BCH code. The Golay(24,12)-code is a non-perfect code as well, but it is very similar to the perfect Golay(23,12)-code. Therefore, the approximation gives better results; the union bound results are too high.

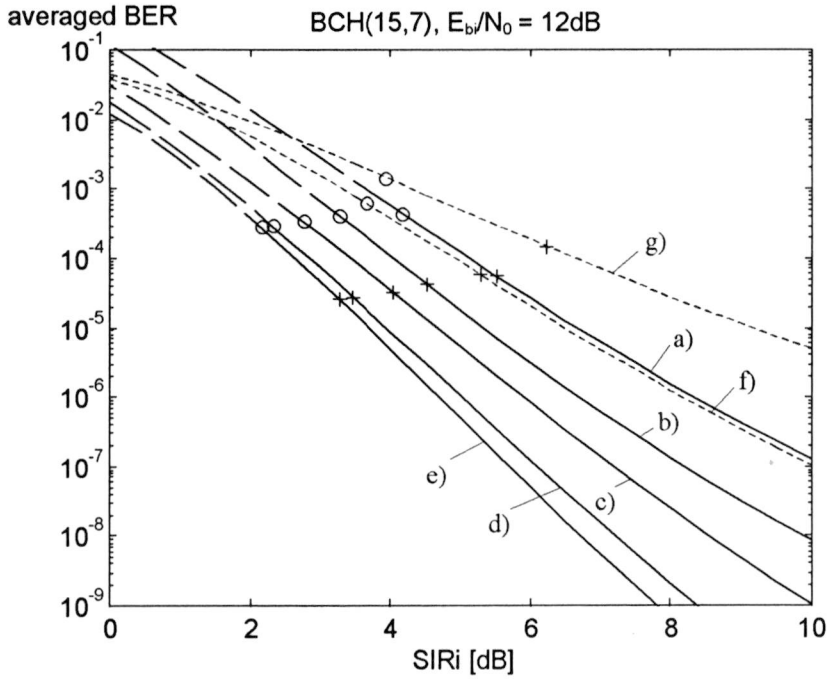

Figure 4-10: Averaged BERD-results against SIRi for the BCH(15,7)-code; curves a) – e): Union bound computations of BERD for a varying number of soft decision quantisation levels (see Table 4-2); f) BERD for the BSC calculated from the exact WER; g) BER without coding

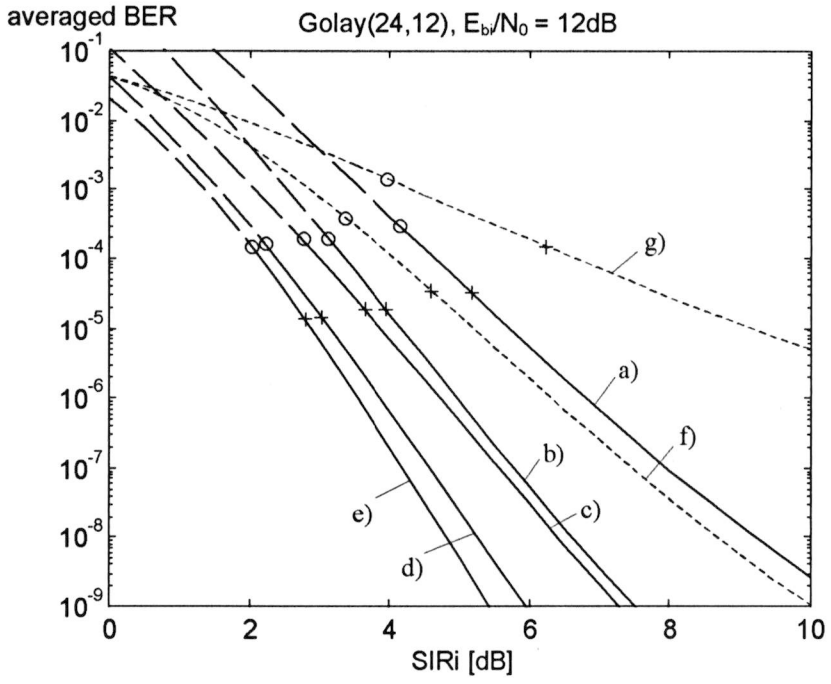

Figure 4-11: Averaged BERD-results against SIRi for the Golay(24,12)-code; see previous figure for the description of the curves.

Comparing curve f), the "exact" calculation of BER for the BSC, to the performance curves of the uncoded case given in Figure 2-2, we can estimate a coding gain of 2 – 3dB for the Golay(24,12)-code and hard decision decoding at E_{bi}/N_0 = 12dB.

Comparing the union bound results given by curves a) – e) for an increasing number of soft decision quantisation thresholds (see Table 4-2), we can see increasing improvement, as expected. Especially, it is interesting to compare the results of the 8-level scheme to the results of the unquantised case (curves d) and e)). We see that the difference between those curves is quite small, even without doing any optimisation of the quantisation levels. Therefore, we will only consider the results of the uncoded case, the hard decision case ("exact" method) and the case of unquantised soft decision decoding in the following comparisons of averaged BER. The union bound results for the other channels lie in-between those curves as shown in the figures.

The coding gain for unquantised soft decision decoding estimated like explained above is 5 – 7dB for the Golay(24,12)-code. In the next section averaged BER results against SNRi will be given which allow the evaluation of the coding gains much better.

4.5.3 Averaged BERD-Results for Varying Signal-to-Noise Ratio

Figure 4-12 and Figure 4-13 [8] show averaged BER results against SNRi. The BCH(15,7)-code and the Golay(24,12)-code, respectively, were taken again; SIRi was taken 3, 6 and 30dB. The curves for 30dB stand for the case without co-channel interference, i.e., for a pure AWGN-channel. A comparison at SIRi = 0dB is not possible in that way, since the worst case represented by $\phi' = 0$ and $\Delta' = 0$ cannot be improved by increasing SNR, and thus will always cause a certain outage probability > 0. The 'o' and '+' marks again indicate the areas of OaR = 0 (see Section 4.5.2).

In these graphs, results are only given for the cases no coding, hard decision decoding "exact"-computation and unquantised soft decision decoding. It can be seen from the figures given in the previous section, how these results are related to the results for other quantisation schemes.

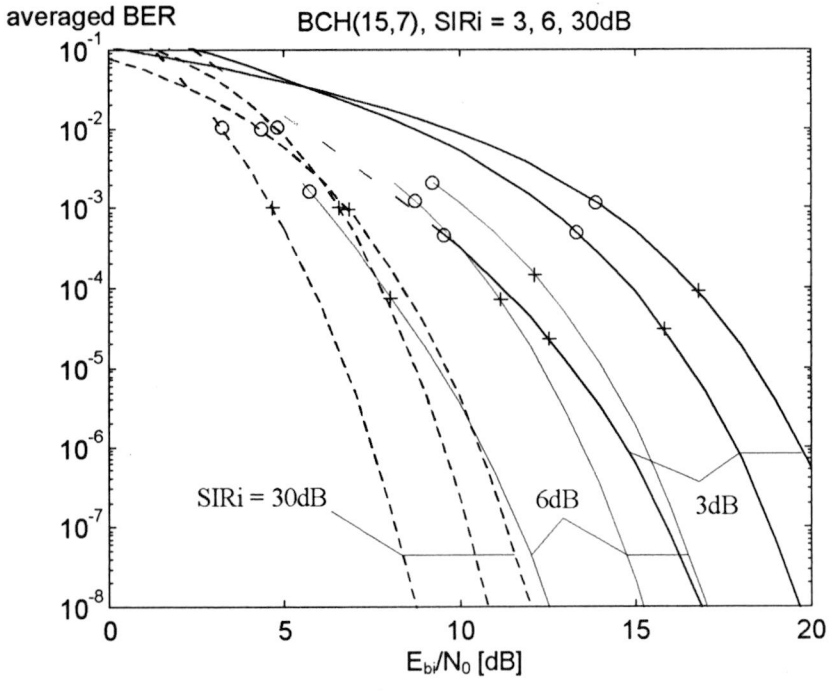

Figure 4-12: Averaged BER-results against SNRi for the BCH(15,7)-code. Dashed lines: SIRi = 30dB; thin lines: SIRi = 6dB and thick lines: SIRi = 3dB. The three lines given for each SIRi stand for (from left to right): Unquantised soft decision decoding; hard dec. decoding ("exact") and no coding.

[8] MATLAB program: *CalUbnd5*

Calculates averaged BERD-results against SNRi as shown in Figure 4-12 and Figure 4-13.

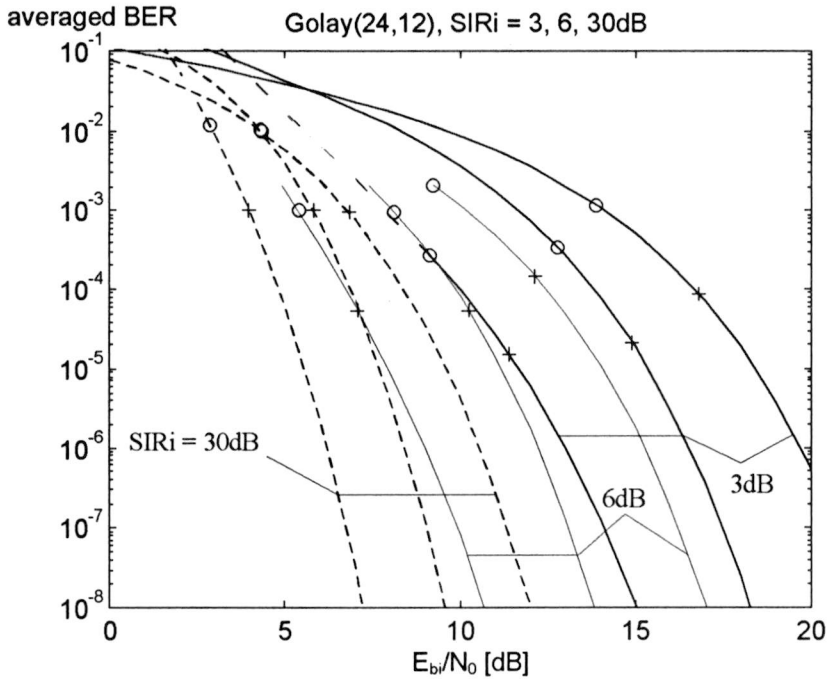

Figure 4-13: Averaged BER-results against SNRi for the Golay(24,12)-code. From left to right: Unquantised soft decision decoding; hard dec. decoding ("exact") and no coding.

From Figure 4-12 and Figure 4-13, it is very easy to determine coding gains for various interference situations, BERs and coding schemes. Results are given in Table 4-4.

Table 4-4: Coding gains for various interference situations SIRi

BER	*SIRi*	*BCH(15,7):*		*Golay(24,12):*	
		Hard Dec. [dB]	*Soft Dec. [dB]*	*Hard Dec. [dB]*	*Soft Dec. [dB]*
10^{-4}	3dB	1.7	5.5	2.9	6.6
	6dB	1.5	4.6	2.6	5.7
	30dB	0.6	2.6	1.5	3.5
10^{-8}	3dB	2.1	4.8	3.4	6.8
	6dB	1.8	4.5	3.2	6.3
	30dB	1.2	3.3	2.5	4.8

These large coding gains show that there is large performance enhancement possible in the given heavy co-channel interference AWGN environment, especially by applying soft decision decoding to powerful codes like the Golay(24,12). The coding gains for the interference channel are much higher than the gains achieved for a pure AWGN-channel using the same coding techniques. This can be explained by the fact that the considered signal-to-interference

ratios are rather high, i.e., the uncertainty of the signal states is small, errors are mainly caused by interference which fortunately affects only a part of the transmitted bits.
Soft decision decoding turned out to be very efficient in controlling the bits affected by interference.

4.6 Optimisation of Quantised Soft Decision Decoding using Optimum Metrics

In Section 4.4.1 a distance function for soft decision decoding was proposed to find the most likely transmitted code word from the received sequence. This distance was obtained by calculating the Euclidean distance between the received sequence ρ consisting of n symbols ρ_i and any code word $\mathbf{S}_l$ consisting of n symbols S_{li} (4-10). The code word having the smallest distance to the received sequence is the most likely transmitted code word. A true maximum likelihood decoder calculates those values for all 2^k code words resulting in a set of decision variables, and selects the most probable one.
The distances between a single received bit and the two possible transmitted bits $\{-1, 1\}$ are called metrics, the task of the decoder is to calculate the decision variables from the metric values which are provided by the demodulator. To simplify these calculations we quantised the demodulator output using linearly spaced quantisation thresholds, the quantised output was assigned to linearly spaced metrics (e.g., demodulator output = 3 $\rightarrow$ metric = 3). The decision variables were simply obtained by summing up the resulting linear metrics as explained in Section 4.4.2. The computational results of Sections 4.5.1 and 4.5.2 have shown that this simplification is a fairly good method of quantised soft decision decoding. However, considering for instance the OaR results for Golay(24,12)-coding, there is still quite a lot of improvement from the quantised to the unquantised results (OaR = 26% with 8-level quantisation; OaR = 19% for the unquantised scheme; $BER_{max} = 0.01$).
One method to improve this situation is to determine optimised metrics and to calculate the decision variables from them. The drawback of that method is that these calculations become much more complex and thus will take more time. How to determine optimised metrics will be shown in the following section.
The nature of the composite input signal of the DSR requires sampling of the down converted RF-signal with a rather high resolution in order to perform the signal processing shown in Figure 2-1. This sampling provides a much finer quantisation of the demodulator output, therefore, we can try to obtain optimised metrics by adjusting the threshold levels rather than increasing the complexity of the decoder. In Section 4.6.2 we will try to improve the result for the Golay(24,12)-code presented above by performing this optimisation.

4.6.1 Calculation of Optimum Metrics

Suppose we receive a sequence ρ consisting of n symbols ρ_i and we wish to find the most likely transmitted sequence $\mathbf{S}_l$ consisting of n symbols S_{li}. To perform this we need to find the maximum value of $\Pr(\mathbf{S}_l|\rho)$, the probability that $\mathbf{S}_l$ was the transmitted sequence given that ρ was received. Invoking Bayes' rule, we can write

$$\Pr(\mathbf{S}_l|\rho) = \frac{p(\mathbf{S}_l,\rho)}{p(\rho)} = p(\rho|\mathbf{S}_l)\frac{\Pr(\mathbf{S}_l)}{p(\rho)} \tag{4-19}$$

where $p(\rho,\mathbf{S}_l)$ is the probability that $\mathbf{S}_l$ was transmitted and ρ was received; note that no quantisation of ρ is assumed by now.

If the transmitted sequences are equally likely, then maximising $\Pr(\mathbf{S}_l|\rho)$ is equivalent to maximising $p(\rho|\mathbf{S}_l)$. Assuming the n received symbols to be statistical independent, which is given for a Gaussian noise channel but is a simplification for our interference affected environment, we may write $p(\rho|\mathbf{S}_l)$ as follows

$$\begin{aligned} p(\rho|\mathbf{S}_l) &= \prod_{i=1}^{n} p(\rho_i|S_{li}) \\ \log p(\rho|\mathbf{S}_l) &= \sum_{i=1}^{n} \log p(\rho_i|S_{li}) \end{aligned} \tag{4-20}$$

The logarithmic probability is negative and $p(\rho|\mathbf{S}_l)$ will thus be maximised, whenever the sum on the right-hand-side is minimised. Let us now define a metric for any symbol transmitted as x and received as ρ, by

$$m_{x\rho} = -A - B\log\Pr(\rho|x) \tag{4-21}$$

$\Pr(\rho|x)$ was defined in Section 4.4.2 for quantised channels, the constants A and B can be chosen to obtain convenient metric values. Over the BSC, for example, we can make the metric be 0 for $x = \rho$ and 1 otherwise. If we now search over all possible transmitted sequences (code words) for the minimum total metric value, we shall find the most likely transmitted sequence.

In [11] an optimum metric evaluation is demonstrated for an AWGN channel at $E_c/N_0 = 0$dB. Metric values of 0, 1, 2, ..., 6, 8.67 were obtained which is very little difference to the linear metrics 0, 1, ..., 7 used in Section 4.4.2. This linear metric assignment works perfectly well in a wide range of practical cases.

4.6.2 Optimisation by Adjusting the Soft Decision Thresholds

From Equation (4-11), the calculation of the transition probabilities $Pr(\rho|x)$, we see that a lot of parameters influence them. In this Section we will try to optimise the OaR-performance of Golay(24,12)-coding for the cases where BERDi = BER_{max} = 0.01, taking SNRi = 12dB, SIRi = 0dB. This optimisation shall be done by optimising the thresholds in order to obtain linear metrics.

Unfortunately, this performance specification can be met by a large variety of ϕ' and Δ' – values. The pairs of ϕ' and Δ' – values were determined and used to calculate transition probabilities and metrics as explained above. It turned out, that the optimisation is not possible in a straight-forward way, since there are too many parameters influencing the transition probabilities.

Finally, using a trial-and-error method, we have found threshold levels which significantly improve the OaR-performance of the DSR using 8-level quantisation soft decision decoding. Table 4-5 lists OaR-results for BER_{max} = {0.01, 0.001} and various quantisation thresholds. Linearly spaced thresholds and the performance of the unquantised scheme are repeated in the first and second row, respectively, to enable a comparison.

Table 4-5: Outage ratios for Golay(24,12)-coding using several quantisation level assignments; SNRi = 12dB, SIRi = 0dB. Optimization was tried for BER_{max} = 0.01

Quantisation Thresholds	*BER_{max} = 0.01*	*BER_{max} = 0.001*	*Comments*
Unquantised Scheme	19.02	36.36	Optimum scheme
[0 2 4 6]/7	26.33	38.87	Linearly spaced scheme
[0 2 4 5]/7	26.0	38.74	
[0 2.5 5 6]/7	24.57	39.25	
[0 3.5 5 6]/7	22.51	40.49	
:	:	:	
[0 3 5 6]/7	21.87	39.67	
[0 3 4.5 6]/7	21.77	39.5	
[0 3 4.5 5.5]/7	21.77 - best trial	39.45	for BER_{max} = 0.01

An ideal receiver could use the signal parameters provided by the DSR to adapt the threshold levels to the actual channel condition. Linear metrics and linear thresholds result in a good compromise for all situations.

4.7 Conclusions and Recommendations

In this chapter the performance of several well-known linear block codes was evaluated for a heavy co-channel interference, AWGN channel, as given by the signal model of the DSR. The BER calculation of the large signal was demonstrated in Section 4.1 to understand the error mechanism of this particular channel. The large signal is demodulated by a conventional coherent BPSK demodulator, i.e., the results given in this chapter are always valid for this kind of demodulators, the DSR concept does not affect the function of the first demodulator.
The outage situation of this receiver was described in Section 4.2. If both signals have about equal strength and if also other signal parameters like the carrier phases of both signals are about equal, the receiver cannot lock on either signal and the estimated data will be of no use. On the other hand, if the signals differ for instance in their carrier phases, demodulation with good performance is possible again. An Outage Ratio (OaR) for a certain SIRi, SNRi and coding scheme was defined, based on these signal parameters. The OaR gives the probability of the receiver BER being greater than a threshold value BER_{max}. It has shown to be suitable for performance evaluations, especially at SIRi = 0dB, where both signals have equal strength. Evaluation of performance enhancement for this situation was one of the goals of this work. The calculation of BERDi was explained in Section 4.3 for hard decision decoding, and in Section 4.4 for soft decision decoding. Union bound calculations were proposed for the analysis of soft decision decoding for several quantised and unquantised channel models.

The OaR results given for SIRi = 0dB show only very little improvement for hard decision decoding. The improvement utilising demodulator soft decisions was significantly better, but OaR results of about 0.5%, which are necessary for reliable communications, are still impossible under any circumstances. Nevertheless, we have estimated enormous coding gains from the OaR-results that are very promising (e.g., Golay(24,12)-coding: Coding gain derived from OaR-results: 10 – 18dB at SNRi = 12dB, SIRi = 0dB).
From averaged BERD-results, we have obtained coding gains of 3.5 – 7dB for ideal unquantised detection using the Golay(24,12)-code. 3.5 – 5dB at high SIRi-values, i.e., few interference, and 5 – 7dB at low signal-to-interference ratios (large interference).
The gains evaluated for our interference-channel are much higher than the gains achieved by the same coding technique in a pure AWGN-environment.
A combination of soft decision decoding with code interleaving and antenna diversity techniques can be advantageous to meet the specification of OaR = 0.5%, especially in a fading environment. A system being suitable for these techniques will be proposed in Chapter 6.
Unquantised soft decision decoding is a rather idealistic model of an implementable decoder, although we have seen that a rather simple 8-level quantised, linear metric scheme provides

almost the same performance. In Section 4.6 optimisation of this scheme was investigated which reduced the performance gap to the optimum unquantised scheme significantly. Optimisation was obtained by adjusting the threshold levels in order to obtain linear metrics. A trade-off is that these optimum threshold levels are strongly dependant from the signal parameters and thus, can be achieved in practice only by an adaptive algorithm.
Linear metrics and linear decision thresholds result in a good compromise for all situations [11].

4.8 Appendix

Influence of s_c on a Received Sequence ρ, Utilising Unquantised Soft Decisions

As stated in 4.4.3, each symbol of our sequence ρ is usually influenced by two symbols of s_c, i.e., each symbol of s_i is influenced by s_c by one of four patterns $w \in \{(00),(01),(10),(11)\}$. Therefore, assuming statistical independence, a sequence of j-symbols will be influenced by one of 4^j different pattern sequences $\boldsymbol{v}$ consisting of j patterns w. The computation of 4^j error probabilities is not practicable, except for very small values of j. It will be demonstrated in this section that many patterns have the same influence on our sequence error rate, hence, we can combine them resulting in V different pattern sequences $\boldsymbol{v}'$ of weight if_v' to speed up the computation

Table 4-6: Influencing sequences for the 3-zero sequence

Pattern Distribution *#00,#01,#10,#11*	*Pattern Sequences* $\boldsymbol{v}'$	*Number of Sequences* $\boldsymbol{v}'$ $\#v'$	*Weight of Influence* if_v'
3,0,0,0	(00)(00)(00)	1	1
2,0,0,1	(11)(00)(00)	3	1/3
1,1,1,0	(01)(10)(00)	6	1/3
1,0,0,2	(11)(11)(00)	3	−1/3
:	:	:	:

In Table 4-6 some of the sequences $\boldsymbol{v}'$ influencing the (j = 3)-zero sequence (0)(0)(0) are listed. #00, #01, #10 and #11 are the numbers of the patterns (00), (01), (10), (11), respectively, giving the pattern distribution of our influencing sequence $\boldsymbol{v}'$. The number of sequences having a certain pattern distribution is listed, as well, as the weight of influence $if_v' \in [-1 \ldots 1]$ that a pattern $\boldsymbol{v}'$ has on the j-zero sequence, where positive values have positive influence. The factor

if_v' multiplied by $\cos(\phi'(t))/\sqrt{\psi_i}$ gives the amplitude factor used in the calculation of $\Pr(B_j''|A_0)$ by (4-18).

The weight of influence is calculated from the pattern distribution by:

$$if_v' = \frac{2}{j}\left[\#11+\#01\Delta'+\#10(1-\Delta')\right]-1 \tag{4-22}$$

To calculate Pr(v'), the probability of an influencing sequence $\boldsymbol{v}'$ with weight if_v' occurring, we have to calculate the number of these sequences #v' (see Table 4-6). This can be done in a similar fashion to the calculation of the combinations of j symbols out of two within a sequence of length n, expressed by $\binom{n}{j}$. Extending the set of symbols to four (our patterns w), we can write

$$\#v' = \binom{j}{\#01,\#10,\#11} = \binom{j}{\#01}\binom{j-\#01}{\#10}\binom{j-\#01-\#10}{\#11} = \frac{j!}{\#01!\#10!\#11!\#00!} \tag{4-23}$$

Pr(v') is simply calculated by dividing this number by the total number of sequences 4^j

$$\Pr(v') = \frac{\#v'}{4^j} \tag{4-24}$$

This concept reduces the number of calculations to get $\Pr(B_j''|A_0)$ from $4^j = 16384$ to V = 120 for a weight-j = 7 sequence. The pattern sequences $\boldsymbol{v}'$ are generated by the following algorithm

```
for #01 = 0:j
   for #10 = 0:j-#01
      for #11 = 0:j-#01-#10
         #00 = j-#01-#10-#11;
         ifv' = ... ;          % calculation of the sequence-weight
         Pr(v') = ... ;        % calculation of the probability of occurrence
```

Further simplification was obtained by quantising if_v' to a fixed number of values $if_v'' \in [-1 \ldots 1]$ and calculating the probabilities Pr(v''). This can be done without loss of accuracy, since the increments of if_v'' can be selected to be the same than the increments of if_v' appearing due to a certain value of Δ'.

5. Coding of the Small Signal

Right from the beginning, we predicted heavy correlation between bit errors in the large signal s_i and the small signal s_c. Demodulation of the small signal can be performed after cancellation of the large signal. To yield this cancellation, the estimated data of s_i is used to suppress the signal s_i. Thus, we have to expect an error in s_c if an error occurred in s_i, because a wrong bit was used for this suppression (= subtraction). The averaged BER-results given in Figure 2-2 show this correlation in form of almost equal bit-error rates of s_i and s_c at low signal-to-interference ratios. Section 5.1 will explain the calculation of BERc, the BER of s_c, the degree of correlation between *bit errors* will be analysed in Section 5.2.
The idea of this work was to enhance the signal suppression performance of s_i by reducing its BER, using coding. Better signal suppression means less influence for s_c, and thus better performance of s_c as well. Section 5.3 concentrates on the correlation between *word errors* in s_i and s_c, in order to evaluate if additional gain can be achieved for s_c, since coding is applied to the small signal as well. BER-results for s_c with coding will be given in Section 5.4. The results of these two sections show that we can make a simple approximation of the BER of s_c with coding. We will propose this simplified model in Section 5.5 and use it to evaluate the benefits of soft decision decoding for s_c.
Finally, conclusions and recommendations are given in Section 5.6.

5.1 BER-Analysis for the Small Signal without Coding

To examine the small signal BER-performance, we need to evaluate the influence of the remainders of the large signal after signal cancellation (large signal: s_i, remainders after signal cancellation: Δs_i) on the small signal s_c. This is obtained by an algebraical approach to the signal processing performed by the DSR, which will give the influencing term Δs_i in a very straight-forward way. The block diagram of the DSR was shown in Figure 2-1. We start our analysis with Equation (4-1) describing $r_I(t)$, the baseband input signal in phase with $s_{i,I}$, which is used by the first BPSK demodulator to obtain the estimated data $\hat{d}_i(t)$. In the last chapter we used $r_I(t)$ to calculate BERi, the bit-error-rate of the large signal. Signal decorrelation is achieved by remodulating this signal using the estimated data $\hat{d}_i(t)$. That is

$$rem_I(t) = \hat{d}_i(t - \Delta_i T - \tau) r_I(t - \tau) \tag{5-1}$$

where $\Delta_i T$ is the timing error between $d_i(t)$ and $\hat{d}_i(t)$, and τ is the delay time needed to perform the demodulation of s_i. Assuming perfect timing recovery ($\Delta_i = 0$) and substituting $t \rightarrow t - \tau$ we can write

$$rem_I(t) = A_i \hat{d}_i(t) d_i(t) + A_c \hat{d}_i(t) d_c(t) \cos\phi'(t) + \hat{d}_i(t) n_I(t) \tag{5-2}$$

If no errors occur in the estimated data, the product term $\hat{d}_i(t) d_i(t)$ will be +1, hence concentrating the signal power of s_i in a DC component where it can be removed easily by means of high pass filtering. In practice bit and timing errors are present and the product term will show short jumps to –1 resulting in imperfect suppression of s_i. It is shown in [2] that these impacts can be expressed by a factor β, defining the DC-component as $A_i(1 - \beta)$.

$$\beta = 1 - (1 - 2\Pr(\varepsilon_i | \phi', \Delta'))(1 - \eta) \tag{5-3}$$

where $\Pr(\varepsilon_i|\phi',\Delta')$ is the error probability of the large signal without coding and η^2 is the carrier suppression performance of the filter given by

$$\eta^2 = \frac{\text{Remaining Carrier Power}}{\text{Input Carrier Power}} \tag{5-4}$$

Throughout this report $\eta^2 = -30$dB was taken in all the calculations concerning the BER of s_c. After suppression of the DC-component by high-pass filtering we have

$$\begin{aligned} rem_I'(t) &= rem_I(t) - (1-\beta) A_i \\ &= A_i \begin{cases} \beta & \notin \text{ in } \hat{d}_{i,j} \\ -(2-\beta) & \varepsilon \text{ in } \hat{d}_{i,j} \end{cases} + A_c \hat{d}_i(t) d_c(t) \cos\phi'(t) + \hat{d}_i(t) n_I(t) \end{aligned} \tag{5-5}$$

where ε means an error and $\notin$ means no error. A second remodulation is carried out to remove $\hat{d}_i(t)$ from the component of s_c being in phase with s_i. This can be done perfectly because $\hat{d}_i^{\,2}(t) = +1$. $\alpha_j \in \{\beta, (2 - \beta)\}$ was defined for the ease of notation, $\alpha_j = \beta$ in case of a correctly estimated bit $\hat{d}_{i,j} = d_{i,j}$, and $\alpha_j = (2 - \beta)$ in case of an error. β can be assumed to be very small compared to 1, since $\eta^2 << 1$ and $\Pr(\varepsilon_i|\phi',\Delta') << 1$, in cases of reliable performance.

$$\Delta r_I(t) = A_i d_i(t) \alpha_j + A_c d_c(t) \cos\phi'(t) + n_I(t) \tag{5-6}$$

This is the signal remaining after cancellation of s_i, the in-phase component of s_c corrupted by Δs_i and noise. Δs_i consists of symbols $d_{i,j}' = \alpha_j d_{i,j}$; i.e., the symbol magnitude $A_i d_{i,j}'$ is of two values, depending on whether an error occurred or not.

Equation (5-6) gives the in-phase component $\Delta r_I(t)$ as described above. A similar expression can be found for $\Delta r_Q(t)$, the quadrature signal, consisting of the $\sin(\phi'(t))$-component of s_c corrupted mainly by noise. These two components of s_c are representing a phasor lying in the complex signal space under the angle ϕ' to the real-axis. The phase processing, being the next stage in the signal path of the DSR, rotates this phasor into the real-axis to perform the demodulation of s_c. Δs_i is also rotated by this angle which is expressed by the cos-factor in the following equation. Therefore, Δs_i looses its influence if $\phi' \to \pi/2$.

$$I_c(t) = A_i d_i(t)\alpha_j \cos\phi'(t) + A_c d_c(t) + n_I'(t) \tag{5-7}$$

This is the signal being used to obtain the estimated data $\hat{d}_c(t)$ of s_c. Similar to the expression $r_I(t)$, which was used to calculate BERi in Section 4.1, we have this time the small signal s_c corrupted by the remainders of the large signal Δs_i and noise. To calculate the bit-error-probability we will express the interfering part of $I_c(t)$ by an "amplitude factor" again. Allowing bit timing difference $\Delta'T$ between s_i and s_c, as defined in Section 4.1, we can write

$$I_c'(t) = A_c\left[1 + \frac{\alpha_0\Delta' d_{i,0} + \alpha_1(1-\Delta')d_{i,1}}{\sqrt{\psi_c}}\cos\phi'(t)\right] + n_I'(t) \tag{5-8}$$

where $\psi_c = \mathrm{SIRc} = A_c^2/A_i^2 = 1/\psi_i$ the signal-to-interference ratio of s_i interfering on s_c. This equation shows that there are 16 different situations of interference for the symbol $d_{c,0}$ now, that are $\alpha_j \in \{\beta, (2-\beta)\}$ and $d_{i,j} \in \{-1, 1\}$, where $j \in \{0, 1\}$. $d_{c,0}$ has to be (+1) to obtain the positive signs given in (5-8), it is sufficient to calculate the error probability for one of the symbols $d_{c,0} \in \{-1, 1\}$, because the simple inversion of all the bits in the two sequences results exactly in an equivalent situation for the inverted bit $d_{c,0}$ having the same error probability.

Figure 5-1 shows these 16 cases using the symbols $\{0, 1, 0_\varepsilon, 1_\varepsilon\}$ for the symbols $d_{i,j}'$ of Δs_i. $\{0_\varepsilon, 1_\varepsilon\}$ are assigned to the error events: bit $\{0, 1\}$ received, bit $\{1, 0\}$ transmitted, i.e. $\{0_\varepsilon, 1_\varepsilon\}$ stand for $d_{i,j}' = \{1, -1\}\cdot(2-\beta)$; $\{0, 1\}$ stand for $d_{i,j}' = \{-1, 1\}\cdot\beta$.

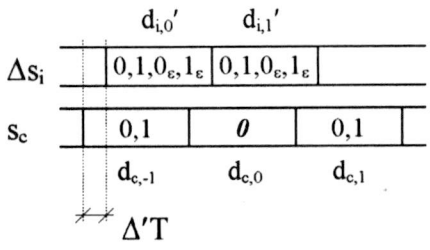

Figure 5-1: The complex interference situation of the DSR

If no error occurs in both symbols $d_{i,0}'$ and $d_{i,1}'$ we have a situation very similar to the case of s_i. With $\alpha_j = \beta$ we can define a signal-to-interference ratio $\psi_c' = A_c^2/\beta^2 A_i^2$ and use the equations of Section 4.1 to calculate the BER. The benefit is that also the effects of coding can be evaluated using the same methods as described in Chapter 4. Note that the dependency of ϕ' and Δ' will be different, since β is a function of ϕ' and Δ' itself.

In case of errors in the estimated data $\hat{d}_i(t)$, the wrong bit is subtracted from s_i and $\alpha_j = (2 - \beta)$. If we insert this value into Equation (5-8) it seems, if $d_{i,j} = +1$, that the influence of the wrong subtracted signal can even improve the error probability of s_c (symbol 1_ε in Figure 5-1). That is true, but the probability of this error 1_ε occurring in Δs_i is not significant. Figure 5-2 gives state diagrams which show the situations for "constructive" errors as explained above (graph c)), and "destructive" errors, e.g., the error 0_ε influencing on the symbol 0 (graph b)).

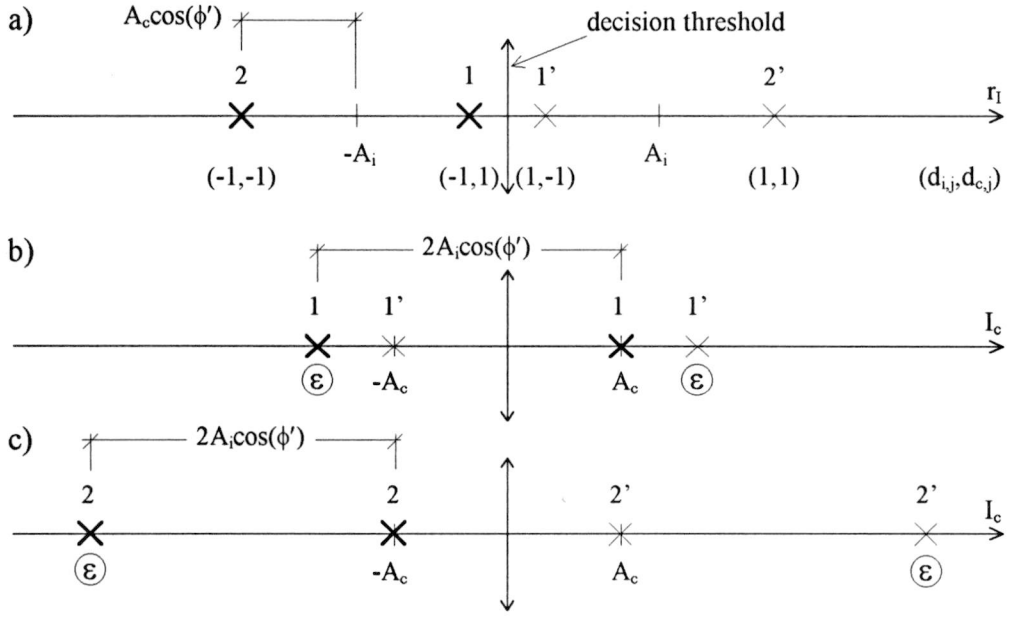

Figure 5-2: Signal-state diagrams for $\Delta' = 0$, $\beta = 0$, various combinations of $d_{i,j}$ and $d_{c,j}$, and various error events of $d_{i,j}$. (a) 1st BPSK demodulator input; (b), (c) 2nd BPSK demodulator input. (b) shows the probable and (c) the non-probable error events.

These state diagrams show the idealised amplitudes (no noise) of the input signals of both BPSK demodulators. The amplitudes are discrete values, since our signals consist of the combination of two binary signals; noise causes an uncertainty of these positions. (See state diagrams in Section 4.1.) The following assumptions were made for that figure: $\Delta' = 0$, i.e., only one symbol of s_i ($d_{i,j}$) and one symbol of s_c ($d_{c,j}$) are interfering on each other; $\beta = 0$, i.e., perfect cancellation of s_i.

Figure 5-2a) shows the composite input signal $r_I(t)$ which is used by the first BPSK demodulator to obtain the estimated data $\hat{d}_i(t)$. We can see that there are two different signal constellations indicated by 1 and 2, having different error rates. In the signal states 1 and 1', the two symbols $d_{i,j}$ and $d_{c,j}$ have opposite signs, their interference is destructive which can be seen in the small distance between the signal states and the decision threshold, hence resulting in a rather high error probability of $d_{i,j}$. In the cases 2 and 2' ($d_{i,j}$ and $d_{c,j}$ having equal signs) the interference is constructive and results in low error probabilities (large distance between the signal states and the decision threshold).

Figure 5-2b) gives the input of the second demodulator, the signal s_c corrupted by Δs_i, for the signal constellations 1 and 1'. In case of an error, we can see the signal states being on the wrong side of the decision threshold with a large distance, hence causing an error in s_c with a very high probability. Unfortunately, this is the situation occurring with a high probability.

Finally, Figure 5-2c) shows situation 2, errors in s_i are occurring with very low probabilities in that case. We can see from the signal states of these errors (lying with a large distance on the same side of the decision threshold) that the error probability of s_c is even smaller than in the non-error case.

The error probability of $d_{c,0}$, assuming a certain bit-and-error pattern ($d_{i,0}'$, $d_{i,1}'$) in Δs_i, with the parameters $d_{i,j}$ and α_j, $j \in \{0, 1\}$ is given by

$$\Pr(\varepsilon_c | d_{i,0}', d_{i,1}') = Q\left[\sqrt{2\gamma_c}\left(1 + \frac{\alpha_0 \Delta' d_{i,0} + \alpha_1 (1-\Delta') d_{i,1}}{\sqrt{\psi_c}} \cos\phi'(t)\right)\right] \tag{5-9}$$

$\gamma_c = \mathrm{SNRc} = E_{bc}/N_0$, the signal-to-noise ratio of s_c ($E_{bc} = A_c^2 T/2$); SNRc = SNRi – SIRi [dB].

To calculate the total conditional bit-error probability $\Pr(\varepsilon_c|\phi',\Delta')$ of the small signal for particular values of ϕ' and Δ', we will multiply (5-9) with $\Pr(d_{i,0}', d_{i,1}')$, the probability of the pattern ($d_{i,0}'$, $d_{i,1}'$) occurring, and sum up over all possible patterns [1].

---

[1] MATLAB function: *[Peicd,Peccd] = PeiPecCd(SNRi,SIRi,eta,phi,delta)*

This function calculates the probabilities $\Pr(\varepsilon_c|\phi',\Delta')$ and $\Pr(\varepsilon_i|\phi',\Delta')$. Vectors can be used as input elements for phi and delta, averaging can be performed by *igrl(igrl(Peicd)')*.

$$\begin{aligned}\Pr(\varepsilon_c|\phi',\Delta') &= BERi(\phi',\Delta') = \sum_{d_{i,0}'}\sum_{d_{i,1}'}\Pr(\varepsilon_c, d_{i,0}', d_{i,1}') \\ &= \sum_{d_{i,0}'}\sum_{d_{i,1}'}\Pr(\varepsilon_c|d_{i,0}', d_{i,1}')\Pr(d_{i,0}', d_{i,1}')\end{aligned} \tag{5-10}$$

Figure 5-1 shows how each symbol $d_{i,j}'$ is correlated to two bits $d_{c,j}$ and $d_{c,j-1}$. We have 16 different patterns ($d_{i,0}'$, $d_{i,1}'$). Since the bit $d_{c,0}$ was assumed to be (0), there are 4 combinations of $d_{c,-1}$ and $d_{c,1}$ left, which are influencing on $d_{i,0}'$ and $d_{i,1}'$, respectively. To calculate the total error probability of $d_{c,0}$, these 16·4 = 64 combinations must be taken into account in the calculation of $\Pr(d_{i,0}', d_{i,1}')$. The symbols $d_{c,-1}$ and $d_{c,1}$ are statistically independent from each other. Therefore, $\Pr(d_{i,0}', d_{i,1}')$ can be determined as follows

$$\begin{aligned}\Pr(d_{i,0}', d_{i,1}') = \Pr(d_{i,0}')\Pr(d_{i,1}') &= \sum_{d_{c,-1}}\Pr(d_{i,0}', d_{c,-1})\sum_{d_{c,1}}\Pr(d_{i,0}', d_{c,1}) \\ &= \frac{1}{4}\sum_{d_{c,-1}}\Pr(d_{i,0}'|d_{c,-1})\sum_{d_{c,1}}\Pr(d_{i,0}'|d_{c,1})\end{aligned} \tag{5-11}$$

An averaged BER can be calculated, once again, by averaging $\Pr(\varepsilon_c|\phi',\Delta')$ over ϕ' and Δ'

$$\overline{BERc} = \frac{1}{\pi}\int_0^{2\pi}\int_0^{0.5}\Pr(\varepsilon_c|\phi',\Delta')d\phi' d\Delta' \tag{5-12}$$

Averaged BER results are shown as function of SIRi in Chapter 2, the description of the DSR, for SNRi = {12, 18, 30dB}.

5.2 Correlated and Non-Correlated Errors

In the previous section, we calculated the bit error probability of one symbol of s_c. It turned out that this probability is usually dependant on two symbols $d_{i,j}'$ of Δs_i which are dependant on three symbols of s_c themselves. Δs_i is the remainders of the large signal s_i after decorrelation. The state diagrams shown in Figure 5-2 for $\Delta' = 0$ made clear that the more probable errors in s_i (the errors where s_i and s_c had opposite signs) are causing errors in s_c again with probabilities of almost 1 (Figure 5-2b). This event will be called a "correlated error", the error in s_i is correlated to s_c and causes, with a very high probability, an error in s_c again.

Figure 5-2c showed that there are other errors in s_i having even a positive influence on the signal s_c. Unfortunately, these errors are usually occurring with very low probabilities, the

signal states corresponding to these situations are having large distances from the decision threshold. Nevertheless, if a coding algorithm is applied and a word error appears in the sequence s_i, additional bit-errors are introduced at random positions. Thus, errors having this positive influence on s_c are just as probable as correlated errors, in that case. We call this error a "non-correlated error", which has (in mean) about half the influence on s_c as a correlated error.

This definition was based on Figure 5-2 showing the state diagrams for $\Delta' = 0$ and one particular value of SIRi and ϕ', which is, of course, not the only situation that is possible. In this section we will survey the effect of ϕ' and Δ' on the error correlation, and we will see that errors in s_i loose their influence on s_c under certain conditions of ϕ' and Δ'.

5.2.1 Correlation of a Bit Error in s_c to an Error in s_i against ϕ' and Δ'

In order to calculate the probability giving the correlation between errors in both signals, let us assume symbol $d_{i,1}' = 0_\varepsilon$ being a correlated error, since we also take $d_{c,0} = 0$. The error event 0_ε means that a one (1) was transmitted but, due to interference by $d_{c,0}$, a zero (0) was received, i.e., $d_{c,0}$ and $d_{i,1}$, the transmitted symbols, had opposite signs. Figure 5-3 shows this situation.

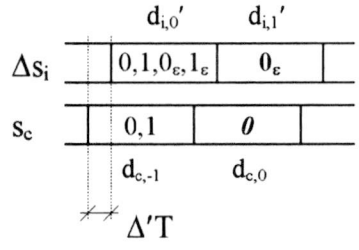

Figure 5-3: The signal constellation used to calculate the correlation between errors in s_i and s_c

The degree of correlation is the probability of $d_{c,0}$ being in error, given $d_{i,0}$ is in error. It can be calculated by taking $d_{i,1}' = 0_\varepsilon$ in Equations (5-9) – (5-11). We have [2]

$$\begin{aligned} \mathrm{PeCorr} = \Pr(\varepsilon_c'|\phi',\Delta') &= \sum_{d_{i,0}'} \Pr(\varepsilon_c|d_{i,0}', d_{i,1}'=0_\varepsilon)\Pr(d_{i,0}') \\ &= \sum_{d_{i,0}'} \Pr(\varepsilon_c|d_{i,0}', d_{i,1}'=0_\varepsilon)\left[\frac{1}{2}\sum_{d_{c,-1}} \Pr(d_{i,0}'|d_{c,-1})\right] \end{aligned} \tag{5-13}$$

[2] MATLAB program: *fig4_4*

Plots the figures given in 5.2.2 using the function *CorrErr1* to calculate PeCorr.

5.2.2 Computational Results and Conclusions

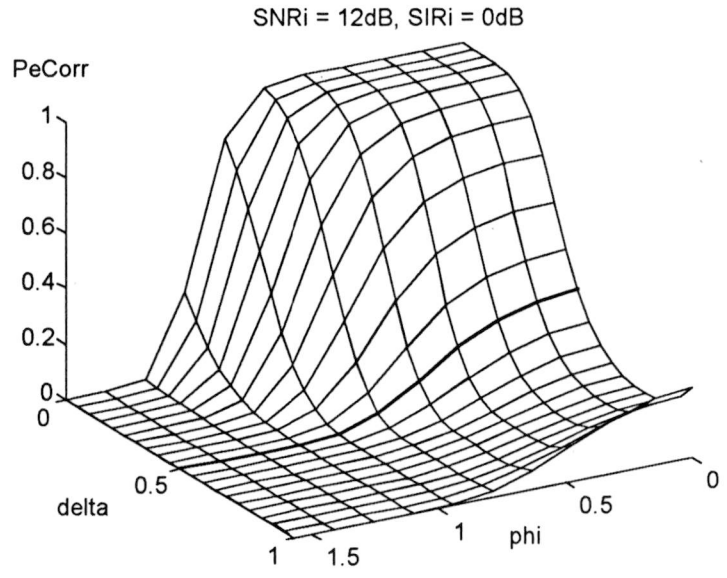

Figure 5-4: The conditional error probability of $d_{c,0}$, which describes the correlation between errors in s_i and s_c (PeCorr), against ϕ' and Δ', where SNRi = 12dB and SIRi = 0dB

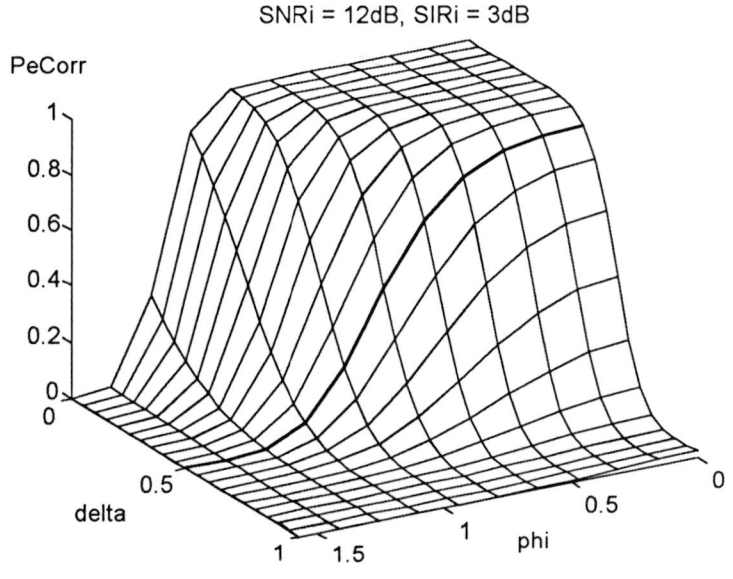

Figure 5-5: PeCorr against ϕ' and Δ', where SNRi = 12dB and SIRi = 3dB

Figure 5-4 and Figure 5-5 give the conditional error probability of $d_{c,0}$ which describes the correlation between errors in s_i and s_c ($\Delta' = [0 \ldots 1]$). In both plots, the correlation decreases with increasing ϕ' and disappears completely for $\phi' = \pi/2$, because the two signals become orthogonal to each other for that phase difference. Unfortunately, this decreasing correlation between the two signals does not give significant performance enhancement of s_c, since BERi

is already low in these situations resulting in low error rates for s_c. Loss of influence means for the state diagram given in Figure 5-2b) that the errors are moving to the other (correct) side of the decision threshold due to the $\cos(\phi')$ term getting smaller. For higher signal-to-noise ratios (SNRi) the uncertainty of the signal states is smaller and the slope dividing the areas of high influence and low influence will be steeper.

Let us consider the effect of a bit timing difference $\Delta'T$, e.g., $\Delta' = 0.5$. In this situation one error in $d_{i,1}'$ influences two symbols $d_{c,j}$ resulting in two different possibilities: Either the error has not enough influence on any bit of s_c any more to cause an error, hence reducing the error rate, or the error causes errors in both bits now, hence enlarging the error rate. Figure 5-4 (SIRi = 0dB) shows decreasing correlation for $\Delta' = 0.5$, thus indicating a loss of influence. This fact indeed gives enhancement of BERc compared to BERi, the outage rates of s_c are better than those of s_i (see Chapter 2). Due to our definition of the symbols $d_{c,0}$ and $d_{i,1}'$, PeCorr is going up again for $\Delta' = 1$. In this case ($d_{c,0}$ is completely overlapping $d_{i,0}$) $d_{c,0} = 0$ is causing errors in $d_{i,0}$ with Pr = 0.5, which cause errors in $d_{c,0}$ with Pr = 0.5 again, i.e. PeCorr = 0.25.

At SIRi = 3dB the correlation is still very high for $\Delta' = 0.5$ and small ϕ' (see Figure 5-5). In that case, one error in s_i is very likely causing two errors in s_c, i.e. performance of s_c is worse. The correlation between errors is higher since s_i has a greater amplitude than s_c, but BERc is lower than before since BERi is lower as well.

The high correlation between errors in s_i and s_c producing the majority of errors in s_c, allows us to make the simplifications described in the next section to calculate the correlation between two code-words of s_i and s_c in reasonable time. The simplifications are based on the definition of correlated and non-correlated errors made in this section. Anyway, note that we will use a different definition of the conditional error probabilities to calculate the influence of an error word in s_i on s_c in the following section.

5.3 BCH(15,7)-Coding Applied to s_i and s_c

In this section we will consider a short, multiple error correction, linear block code, the BCH(15,7)-code, applied to both, the strong and the weak signal, s_i and s_c, respectively. The BCH(15,7)-code was selected, because an equivalent analysis for the better Golay codes seemed to be too complex to be carried out on a PC in reasonable time. The more simple Hamming codes, on the other hand, might represent a too simple model for the multiple error correction capabilities of powerful block codes. The basic properties of such codes were reviewed in Chapter 3.

The most important property used in this section is the block-error mechanism. If a block- (or word-) error appears in s_i, at least d_{min} (the minimum Hamming distance of the code) errors are

present within the n-bit sequence of one code word. This large number of errors influencing on the small signal s_c makes another block error in that signal highly probable, at least in the case of synchronous sequences. However, if the code sequences in s_i and s_c are not synchronous, one erroneous code word will affect two code words of s_c. Now, there are two possibilities again: Either the influence on each word is not strong enough to cause an error in any of these words, hence reducing the number of errors, or errors are caused in both words, hence enlarging the number of errors. In this section we will develop a simplified but still precise model to evaluate which of these situations is present in case of sequence timing difference between s_i and s_c.

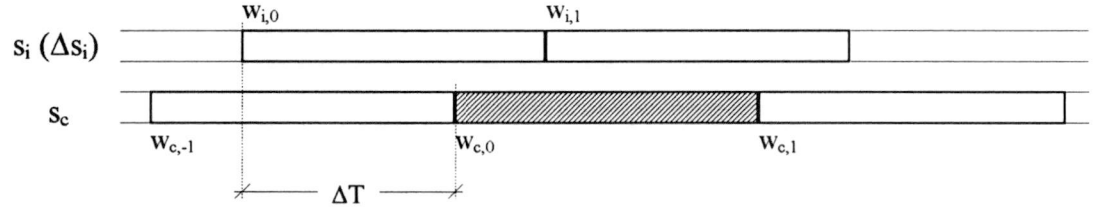

Figure 5-6: Code words $w_{i,j}$ and $w_{c,k}$, influencing on each other

The calculation of BERc without coding was explained in Section 5.1. It turned out that we have to consider 64 different cases for the calculation of the error probability of one single bit. Applying BCH(15,7) coding to our signals s_i and s_c means that there are $2^7 = 128$ different code words of length $n = 15$ bit, which are influencing on each other now. Our goal is to calculate a block error probability for the word $w_{c,0}$ being influenced by two words $w_{i,0}$ and $w_{i,1}$. Each of these two words, which are taken out of the set of 128 again, is influenced by two words of s_c itself. In case of a block error in $w_{i,j}$, there are 127 possible error patterns for each code word of $w_{i,j}$. Due to the linear property of the used code, we can assume the word $w_{c,0}$ to be the all-zero code word, which is similar to the bit constellation considered in Section 5.1. But even with this simplification we still have 128 correct sequences and 128·127 error sequences in both words $w_{i,j}$ influencing on $w_{c,0}$. That gives $128^4 = 268.435.456$ cases, without considering the influence of $w_{c,-1}$ and $w_{c,1}$ on $w_{i,0}$ and $w_{i,1}$, respectively, and without calculating the actual influence on the 15 bits of $w_{c,0}$.

Further simplifications, based on the significantly differing influence of correlated and non-correlated errors, will be described in this section.

5.3.1 Calculation of a Conditional WER Assuming a Word Error in s_i

To simplify the calculations described above, let us consider a typical error event in s_i. Referring to the linear property of the used codes, it becomes clear that the error symbols occurring in case of a block error will have the same pattern as the ones of a code word [3]. That means for the BCH(15,7)-code that 127 different error patterns are possible in any word $w_{i,j}$. To reduce this number we will concentrate on the most probable errors. The BCH(15,7)-code can correct up to t = 2 errors. The most probable error event is the one where t + 1 = 3 (correlated) errors occur in $w_{i,j}$. This error pattern can only be corrected into a code word having the minimum Hamming distance d_{min} = 5 to the word $w_{i,j}$, i.e., t = 2 non-correlated errors are added by the decoder. Figure 5-7 shows this event with t + 1 = 3 correlated errors (ε_c) and t = 2 non-correlated errors (ε_n). Correlated and non-correlated errors were defined in Section 5.2. The numbers in the sequence s_c stand for different symbol patterns in Δs_i, the "influencing classes". Each number indicates influence by Δs_i with a particular strength. The empty squares in s_c stand for class 9, i.e., two correctly suppressed symbols in Δs_i are overlapping that symbol of s_c. The classes will be defined in the next section.

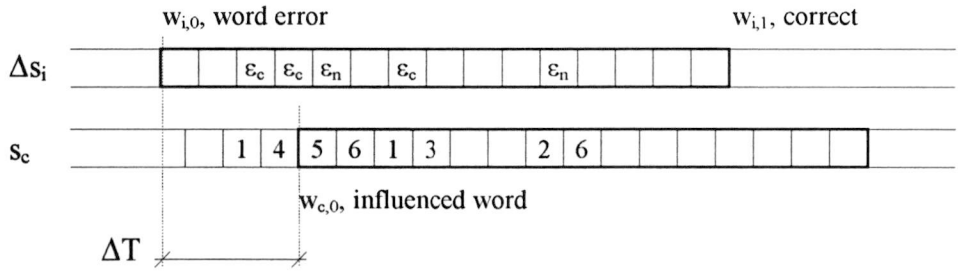

Figure 5-7: Correlated and non-correlated errors in the error word $w_{i,0}$

A close analysis of the weight-structure of the code showed that there are $n_{j=5}$ = 18 code words having this minimum weight of d_{min} = 5, i.e. we have 18 error-distribution patterns in stead of 128, which is a much more suitable number. The positions of the 2 non-correlated

[3] Assume, the code word A_i, the i-th of the 2^k code words was transmitted and the received sequence was corrected to the j-th code word B_j. Performing modulo 2 addition (exclusive OR operation) of the word B_j with the correct word A_i, we will get either the all-zero sequence in case of a correct word B_j (j = i), or just another code word due to the linear property in case of an error. Therefore, the ones are representing the positions of the error symbols.

errors are assumed to be random within the 5 errors, i.e., there are $\binom{5}{2} = 10$ different positions these two errors can have. Taking these 10 positions into account, we have to consider 18·10 = 180 different error sequences influencing on our word $w_{c,0}$.

The approximation of the error events by this "minimum-distance-event" was already proposed in Chapter 3 to calculate the decoder output BER. The analysis of this approximation in that chapter, showed that it is a strict lower bound for the Golay(23,12)-code for input BERs up to 0.15. For the BCH(15,7)-code it turned out to be a very good approximation as well, but this time it is an upper bound up to input BERs of 0.25. The reason is, that this code is a non-perfect one, thus, there are many distance d = 3 sequences out of the correction range (t = 2) of any code word. In these cases, no error correction will be performed, and no additional errors will be introduced.

The error word $w_{i,0}$ in Figure 5-7 consists of three different symbols, each of these symbols influencing on s_c with significantly varying strength. We will call these symbols "classes of influence", which of these classes we have to distinguish will be defined in the next section.

We also have to determine the number of these influencing symbols overlapping with our word $w_{c,0}$, particularly in the case of asynchronous sequences. This is done by a computer program [4], which is generating the error patterns (code words of weight 5), introducing the non-correlated errors in the 10 different positions and counting the number of symbols for each class, considering ΔT, the timing shift between the two sequences. The result of this program is a coded list nec(x) for the 180 error sequences and the 30 different overlapping situations ($\Delta \in \{-15, -14, -13, \ldots, 14\}$), $x \in \{1, 2, \ldots, 9\}$.

---

[4] MATLAB program: *c_struc*

This program generates the code words of the BCH(15,7)-code by using the generator matrix G. If a code word has weight $d_{min} = 5$, the non-correlated errors are introduced and nec(x) is determined for each influencing class and for different degrees of overlap Δ. That gives 9 values for 180 error sequences and 30 overlap-situations. The 9 values are coded into one number, using 2 bits for each value. This is sufficient for the BCH(15,7)-code, because t + 1 = 3 is the maximum number occurring for one class. The list is saved in the File 'nec.mat'.

5.3.2 Classes of Influencing Symbols

We have seen in the previous section that we can distinguish three influencing classes for each symbol of Δs_i, each class having significantly varying influence on s_i. There are two classes for the correlated and the non-correlated errors described in Section 5.2 (abbreviated by ε_c and ε_n, respectively), and a third class for correct symbols, abbreviated by 0. Considering a bit-timing difference $\Delta' \neq 0$ [5], we have symbol pairs of Δs_i influencing on each symbol of s_c. Since both symbols are out of the three classes defined above, there are 9 different patterns influencing on one bit of s_c now. Each of these influencing patterns (we will call them influencing classes, again) consists of several bit-and-error patterns ($d_{i,0}'$, $d_{i,1}'$). Table 5-1 shows which of these patterns are combined to calculate the conditional error probabilities pec(x) of s_c [6]. The number x is the number of the influencing class, p_x is the set of patterns ($d_{i,0}'$, $d_{i,1}'$) combined to the influencing class x.

Table 5-1: Influencing patterns consisting of several bit-and-error patterns

Nr. x	*Influencing Classes*	*Bit-and-Error Patterns* $(d_{i,0}',d_{i,1}')_x \in p_x$	*Weighed Averaging*	*Arithmetical Averaging*
1	$0,\varepsilon_c$	$0,0_\varepsilon$ $1,0_\varepsilon$ $0,1_\varepsilon$ $1,1_\varepsilon$	X	
2	$0,\varepsilon_n$	$0,0_\varepsilon$ $1,0_\varepsilon$ $0,1_\varepsilon$ $1,1_\varepsilon$		X
3	$\varepsilon_c,0$	$0_\varepsilon,0$ $0_\varepsilon,1$ $1_\varepsilon,0$ $1_\varepsilon,1$	X	
4	$\varepsilon_c,\varepsilon_c$	$0_\varepsilon,0_\varepsilon$ $0_\varepsilon,1_\varepsilon$ $1_\varepsilon,0_\varepsilon$	X	
5	$\varepsilon_c,\varepsilon_n$	$0_\varepsilon,0_\varepsilon$ $0_\varepsilon,1_\varepsilon$		X
6	$\varepsilon_n,0$	$0_\varepsilon,0$ $0_\varepsilon,1$ $1_\varepsilon,0$ $1_\varepsilon,1$		X
7	$\varepsilon_n,\varepsilon_c$	$0_\varepsilon,0_\varepsilon$ $1_\varepsilon,0_\varepsilon$		X
8	$\varepsilon_n,\varepsilon_n$	$0_\varepsilon,0_\varepsilon$ $0_\varepsilon,1_\varepsilon$ $1_\varepsilon,0_\varepsilon$ $1_\varepsilon,1_\varepsilon$		X
9	0,0	0,0 0,1 1,0 1,1		X

If a non-correlated error appears in one of the symbols $d_{i,j}'$, simply the arithmetic mean of the conditional probabilities $\Pr(\varepsilon_c|d_{i,0}',d_{i,1}')$ is calculated and assigned to the conditional probability

---

[5] In the case of a sequence timing difference ΔT between s_i and s_c, Δ' is defined by $\Delta' = \Delta - \text{round}(\Delta)$, i.e., $\Delta' \in [-0.5 \ldots 0.5]$, the corresponding bit timing shift.

[6] MATLAB function: *cal_pec*

Calculates the conditional error probabilities pec(x) as explained in the text. This function is called by the program *wer_sml* which calculates the total conditional word error probability of $w_{i,0}$.

pec(x) of that influencing class. That is allowed, since the symbols 0_ε and 1_ε are inserted randomly by the decoder.

$$pec(x) = \frac{1}{\#p_x} \sum_{(d_{i,0}',d_{i,1}')\in p_x} \Pr(\varepsilon_c | d_{i,0}', d_{i,1}') \tag{5-14}$$

where $\#p_x$ is the number of patterns included in the summation.

To calculate the influence of the correlated errors we take the probability of occurrence of the symbols $d_{i,0}'$ and $d_{i,1}'$ into account, i.e., we calculate a weighed average over the probabilities $\Pr(\varepsilon_c | d_{i,0}', d_{i,1}')$. That is

$$pec(x) = \frac{\sum_{(d_{i,0}',d_{i,1}')\in p_x} \Pr(\varepsilon_c | d_{i,0}', d_{i,1}') \Pr(d_{i,0}', d_{i,1}')}{\sum_{(d_{i,0}',d_{i,1}')\in p_x} \Pr(d_{i,0}', d_{i,1}')} \tag{5-15}$$

Examples:

Table 5-2 gives numerical results of the error probabilities of s_c conditional on the patterns ($d_{i,0}'$, $d_{i,1}'$) (1st row) and the probabilities of the occurrence of these patterns (2nd row). The patterns ($d_{i,0}'$, $d_{i,1}'$) were taken from the influencing classes 1 and 2. Each two rows belong to one parameter assignment. We can see the significant difference between the values of the probable (0_ε) and the non-probable (1_ε) errors which we predicted from Figure 5-2 showing the state diagrams. The weighed and the arithmetic average over these values is shown in the next two columns, standing for the correlated and the non-correlated errors, respectively. In the last column the influence of the non-error pattern is given. If we compare these three values we see that there is indeed a big difference in influence between the three basic influencing classes as we defined them above.

Some special points were taken for the parameters: The first group of results is rather theoretical, since the DSR can't work if SIRi = 0, $\phi' = 0$ and $\Delta' = 0$. SNRi = 12dB and $\eta^2 = -30$dB in all calculations. The next group shows how s_i looses its influence due to increasing phase difference ($\phi' = \pi/3$). The last two groups allow a comparison of the influence of one error in s_i on two bits of s_c ($\Delta' = 0.5$) between SIRi = 0dB and SIRi = 3dB (see Section 5.2.2).

Table 5-2: Numerical examples to show the difference between the influencing classes 1, 2 and 9

Parameters		$0,0_\varepsilon$ $1,0_\varepsilon$	$0,1_\varepsilon$	$1,1_\varepsilon$	*Weighed*	*Arithmetical*	*no errors*
SNRi	*SIRi*	$Pr(\varepsilon_c \mid d_{i,0}', d_{i,1}')$			*Averaging*	*Averaging*	*in* $d_{i,j}'$
ϕ'	Δ'	$Pr(d_{i,0}', d_{i,1}')$			*pec(1)*	*pec(2)*	*pec(9)*
12dB	0dB	0.996 0.996	2.6e-44	2.6e-44	0.996	0.498	1.88e-3
0	0	0.094 0.094	2.4e-30	2.4e-30			
12dB	0dB	0.465 0.465	3.4e-29	3.4e-29	0.465	0.232	1.07e-8
π/3	0	6.2e-4 6.2e-4	4.2e-18	4.2e-18			
12dB	0dB	0.112 0.5	1.3e-29	5.0e-24	0.278	0.153	1.34e-6
0	0.5	0.063 0.047	1.2e-9	8.9e-10			
12dB	3dB	0.929 0.950	4.1e-22	2.3e-21	0.939	0.470	4.02e-5
0	0.5	6.2e-3 6.1e-3	1.2e-9	1.2e-9			

5.3.3 Calculation of the Word Error Probability

The last two sections described how the number of bits of s_c influenced by a certain influencing class (nec(x)) and the conditional error probability of these bits (pec(x)) were determined. To calculate a word error probability, we have to calculate the probability that more than t = 2 errors occur in $w_{c,0}$, taking into account the various influencing symbols and probabilities. This is done by calculating the probability of the complementary event (the probability that 0, 1 or 2 bits are in error) and subtracting this value from 1, which is much faster and much more easy to implement.

The probability of no errors is calculated by

$$\Pr(no\ errors) = \prod_{x=1}^{9} (1 - pec(x))^{nec(x)} \tag{5-16}$$

To get the probability of one error, we have to consider the number of positions this error can have within the sequence of nec(x) symbols influencing in the same way

$$\Pr(one\ error) = \sum_{x=1}^{9} nec(x) pec(x) (1 - pec(x))^{nec(x)-1} \prod_{y=1, y\neq x}^{9} (1 - pec(y))^{nec(y)} \tag{5-17}$$

In a similar fashion an expression can be found for two errors. The word error probability for one of the 180 influencing error patterns in $w_{i,0}$ is calculated by [7]

$$\Pr(\varepsilon_w|\phi',\Delta,w_{i,0}) = 1 - \Pr(\textit{no errors}) - \Pr(\textit{one error}) - \Pr(\textit{two errors}) \tag{5-18}$$

Finally, the total conditional word error probability is obtained by averaging over the 180 error sequences [8]

$$\Pr(\varepsilon_w|\phi',\Delta) = \frac{1}{180}\sum_{w_{i,0}} \Pr(\varepsilon_w|\phi',\Delta,w_{i,0}) \tag{5-19}$$

5.3.4 Results and Conclusions

We see from Figure 5-8 and Figure 5-9 that in case of synchronous sequences ($\Delta = 0$), a block error in s_i will cause a block error in s_c again with a very high probability. However, if we introduce some bit timing difference, e.g. $\Delta' = 0.5$, we have totally different situations for different signal-to-interference ratios. As explained in Section 5.2.2 the error bit looses part of its influence if SIRi = 0dB, hence reducing the bit error probability. This is also of benefit if we apply coding: Figure 5-8 shows improvement for all Δ values, where $\Delta' = 0.5$. For higher signal-to-interference ratios (e.g. SIRi = 3dB was taken for Figure 5-9), the effect of this bit timing difference is just the opposite: The correlation is going up for $\Delta' = 0.5$, indicating that one error bit in s_i causes even two bit errors in s_c now, with a rather high probability.
For asynchronous sequences the correlation is decreasing in general; the figures are symmetric to $\Delta = 0$, only one side was displayed. Unfortunately, we can't expect large enhancement in means of BER or WER for s_c, since there are two words $w_{i,j}$ influencing on $w_{c,0}$, each one being in error with a certain probability. This will be analysed in the next section.

---

[7] MATLAB function: *err_prob* called by *wer_sml*

This function calculates the conditional word error probabilities for one error pattern in $w_{i,0}$ and all the overlapping situations Δ. The probabilities pec(x) are handed to this function via the global variables pec_pow and pec_c_pow, which are containing all the needed powers of pec(x) and (1 – pec(x)).

[8] MATLAB program: *wer_sml*

Loads the error pattern distribution from the file 'nec.mat', calculates the probabilities pec(x) of the influencing classes 'x' by using the function *cal_pec*; calculates the powers of these probabilities and stores them in global variables; invokes the function *err_prob* and averages over the 180 sequences to gain the total word error probability. The results are displayed and saved to the file 'wer_dat.mat', to be used by the program *ber_sml*.

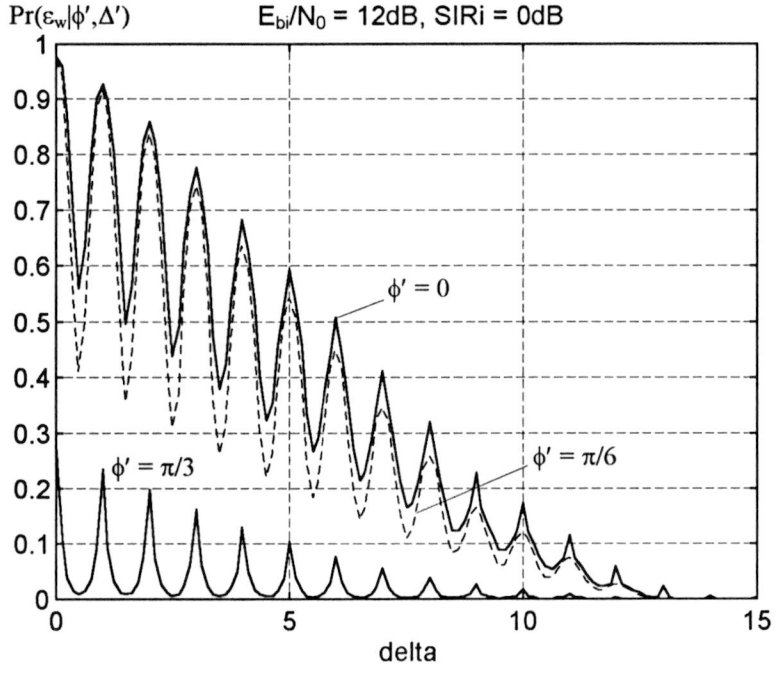

Figure 5-8: The total conditional word error probability of $w_{c,0}$ assuming a word error in $w_{i,0}$, against delta, the sequence timing difference between $w_{i,j}$ and $w_{c,j}$. E_{bi}/N_0 = 12dB, SIRi = 0dB, BCH(15,7)-coding

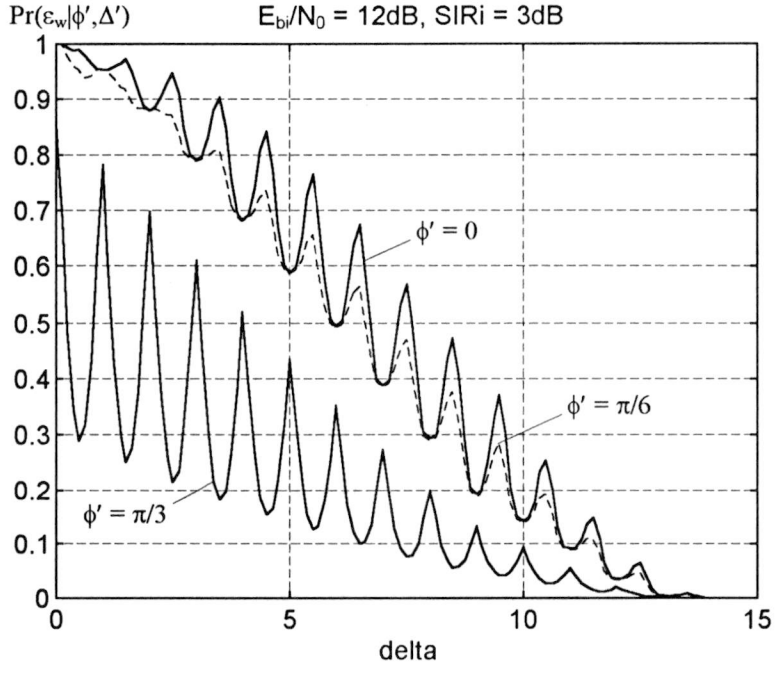

Figure 5-9: The conditional word error probability for SIRi = 3dB. (E_{bi}/N_0 = 12dB, BCH(15,7)-coding)

To simplify the calculations described in this section, we have defined a special error event which is in practice not the only one, but the most probable one. For a perfect code this event

would be the one having the least influence on s_c, hence resulting in a lower bound for the error probability of s_c. However, that is not true for the used non-perfect BCH(15,7)-code, since there are many received sequences (~ 50 %) with distance t + 1 = 3 from the correct word, lying out of the correction range of any possible code word. Those words can be detected but not decoded, thus no additional errors are introduced by the decoder. Fortunately, the influence of these introduced (non-correlated) errors is much smaller than the influence of the t + 1 = 3 correlated errors anyway, and our algorithm gives a good approximation. Moreover, the assumption of this minimum error event is much better, if we use better codes, e.g., the Golay(24,12)-code. The BCH(15,7)-code as it was used here can be seen as a simplified computational model for those codes, since it is impossible to do the calculations described in this section for the 759 weight-8 sequences of this Golay-code (in stead of 18 weight-5 sequences for the BCH(15,7)).

If we consider soft decision maximum likelihood decoding, the mechanism introducing additional errors is not the same any more. The soft decision decoder selects the code word having the minimum metric distance to the received sequence. In case of a word error, the selected word will be one with all error bits having small metric values, i.e. small distances to the decision threshold. Figure 5-2b showed that exactly these errors are the ones we called correlated errors, having strong influence on s_c. Soft decision reduces the error rate of s_i, but introduces more correlated errors, thus a word error in s_i has more influence on s_c. However, the dependency of that error function on ϕ', Δ' and Δ (decreasing correlation for increasing ΔT; improvement for SIRi = 0dB and $\Delta' = 0.5$) will keep the same as in the analysis explained in this section.

5.4 Small Signal BER with BCH(15,7)-Coding

In the last section a model was proposed to calculate a conditional word error probability for s_c, assuming a word error in Δs_i. With this probability it is quite simple to calculate the total word error rate (WERc) and the bit error rate (BERDc) for s_c. This will be demonstrated in the following section.

5.4.1 Calculation of WERc and BERc

Without implementing some kind of sequence synchronisation, we have to expect a sequence timing difference ΔT between code words of the sequences s_i and s_c, respectively. Thus, two words of s_i are influencing on each word of s_c. This situation is shown in Figure 5-10.

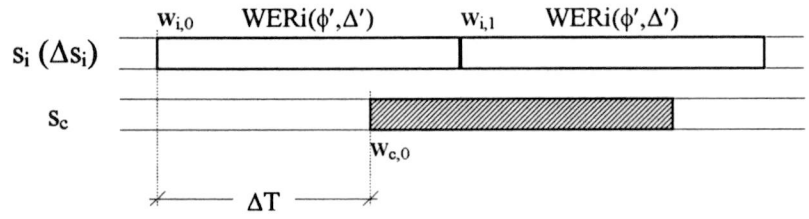

Figure 5-10: Two words of s_c influencing on each word of s_i

We have to distinguish four cases: case (a) both words $w_{i,j}$ are correct; cases (b) and (c) one of the words is in error; and case (d) both words are in error. We assume the same word error probability (WERi) for each word $w_{i,j}$. WERi(ϕ',Δ') is calculated by Equation (3-3).

The WER of s_c (WERc) is obtained by calculating the probabilities of these four cases, multiplying them with the conditional word error probabilities for s_c, and summing up.

The probability of case (a) occurring is the probability of both words being correct: $P_0 = (1\text{-}\mathrm{WERi}(\phi',\Delta'))^2$. In this case each bit of $w_{c,0}$ is influenced by class-9, two correctly suppressed bits of s_i in Δs_i. The conditional word error probability for this case ($\Pr(\varepsilon_w'|\phi',\Delta')$) is calculated by Equation (3-3), defining p = pec(9).

Cases (b) and (c) are occurring with $P_1 = (1 - \mathrm{WERi}(\phi',\Delta'))\cdot\mathrm{WERi}(\phi',\Delta')$, the conditional word error probabilities are $\Pr(\varepsilon_w|\phi',\Delta)$ and $\Pr(\varepsilon_w|\phi',(15-\Delta))$ as calculated in the previous section.

Finally, case (d) occurs with $P_2 = \mathrm{WERi}^2(\phi',\Delta')$, causing an error in s_c with $\Pr(\varepsilon_w|\phi',\Delta')$. That case was assumed to cause an error with the same probability as a synchronous word causes an error, i.e., $\Delta' = (\Delta - \mathrm{round}(\Delta)) \in [-0.5 \ldots 0.5]$. The total expression is

$$WERc(\phi',\Delta) = P_0 \Pr(\varepsilon_w'|\phi',\Delta') + P_1\left[\Pr(\varepsilon_w|\phi',\Delta) + \Pr(\varepsilon_w|\phi',(15-\Delta))\right] + P_2 \Pr(\varepsilon_w|\phi',\Delta') \tag{5-20}$$

The decoder output bit error rate BERDc was calculated from WERc by using the approximation described in Chapter 3, in which any word error is assumed to output $d_{min} = 5$ bit errors [9], i.e., BERDc = 5/15·WERc.

[9] MATLAB program: *ber_sml*

Reads the file 'wer_dat.mat' which was generated by the program *wer_sml* from disc, calculates WERc and BERDc using the probabilities from the file, and plots the results against Δ.

5.4.2 Results and Conclusions

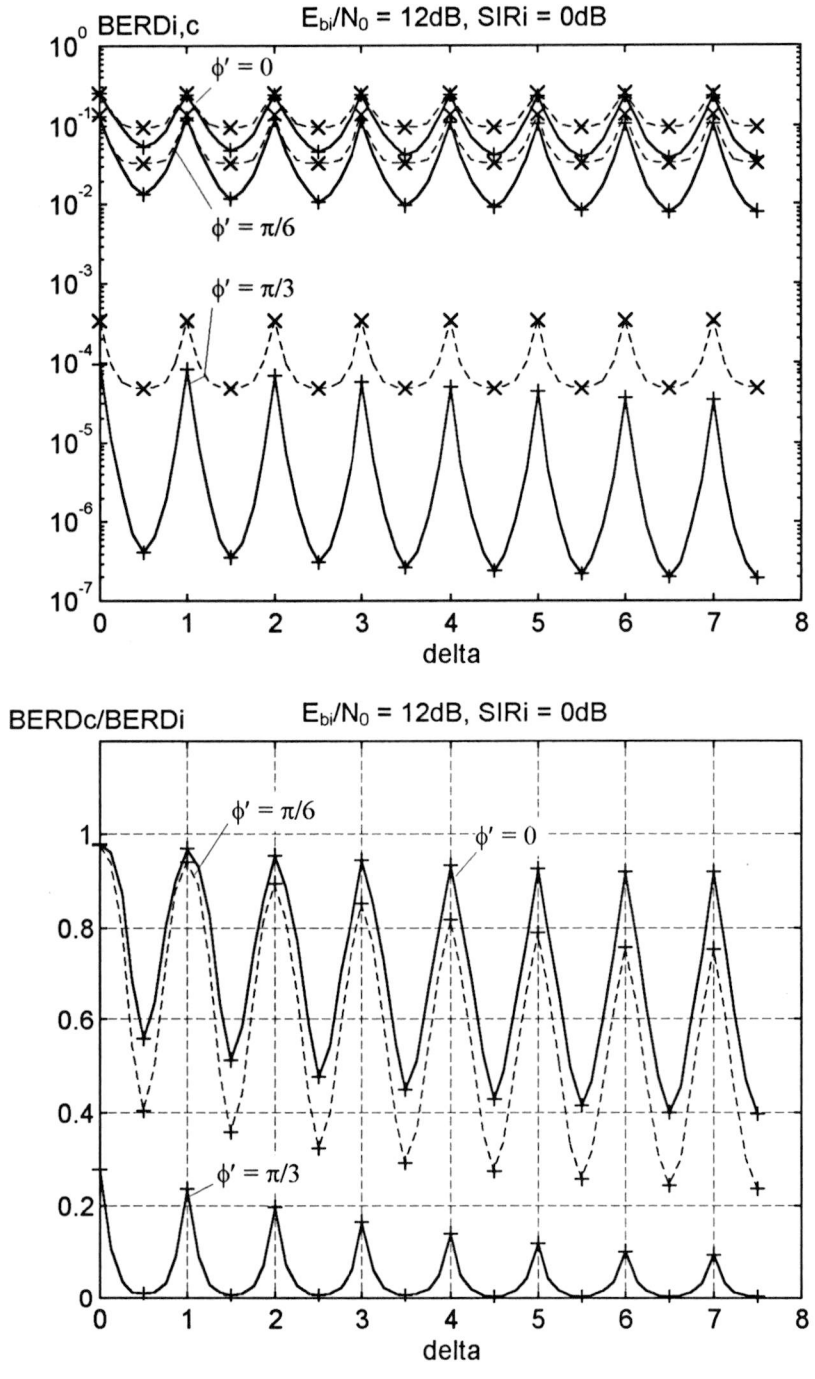

Figure 5-11: Upper plot: BERDi (dashed graphs, '– – x – –') and BERDc (solid graphs, '—+—'); Lower plot: BERDc/BERDi, for BCH(15,7)-coding against Δ with $\phi' \in \{0, \pi/6, \pi/3\}$ as parameter. E_{bi}/N_0 = 12dB, SIRi = 0dB.

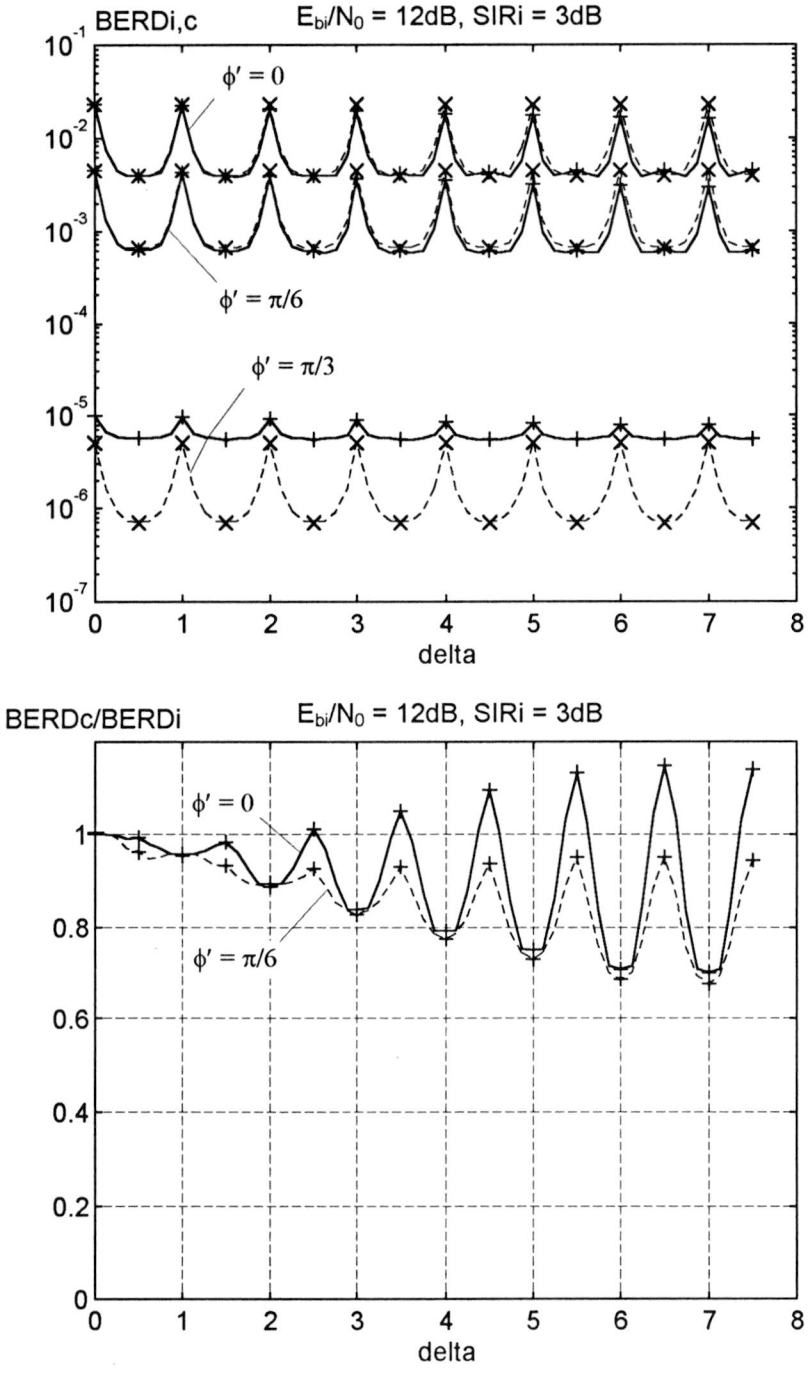

Figure 5-12: BERDi (dashed graphs, '– – x – –') and BERDc (solid graphs, '—+—'); and BERDc/BERDi (lower plot) for SIRi = 3dB. The curve for $\phi' = \pi/3$ in the lower plot has values between 2 and 10 due to the noise becoming the more important impact than interference.

We see from Figure 5-11 and Figure 5-12 that there is only a small performance gain for s_c compared to s_i in general. The decreasing influence of errors in s_i for asynchronous sequences $\Delta \rightarrow 7.5$ is almost compensated by the fact that there are two words of s_i influencing on s_c now, and each of them is in error with a certain probability.

For SIRi = 0dB, we get some performance enhancement for bit-asynchronous sequences ($\Delta' \neq 0$) as predicted in Section 5.3, and further enhancement for $\phi' > \pi/4$ because of the loss of influence for $\phi' \rightarrow \pi/2$ (see Section 5.2). Unfortunately, these benefits vanish for higher values of SIRi, e.g. SIRi = 3dB. In that case the influence of each error in s_i on s_c is much stronger, because of the signal amplitude of s_i being greater than the amplitude of s_c. For $\phi' = 0$ and $\Delta' = 0.5$, and some sequence timing difference ($\Delta \rightarrow 7.5$), BERDc is even a little bit greater than BERDi. For $\phi' \rightarrow \pi/2$ the two signals become orthogonal and the correlation between the two signals disappears. In this case BERDc will be greater than BERDi, since its signal-to-noise ratio (SNRc) is smaller than SNRi (SNRi = SIRi + SNRc [dB]). OaR and averaged-BER results are only a little bit better for s_c in that case.
Table 5-3 shows outage rate and averaged-BER results for SIRi = 0dB and SIRi = 3dB.

Table 5-3: Outage rate and averaged-BER results for BERDi and BERDc, E_{bi}/N_0 = 12dB

SIRi:	*0dB*		*averaged*	*3dB*	*OaRs*	*averaged*
BER_{max}:	*0.01*	*0.001*	*BER*	*0.01*	*0.001*	*BER*
BERDi (large signal):	0.468	0.585	0.0387	0.020	0.330	0.0015
BERDc (small signal):	0.408	0.523	0.0270	0.018	0.318	0.0013

In general we can see that the BER of the strong signal represents an upper bound on the BER of the small signal at low signal-to-interference ratios. This bound will be used in the next section to analyse the benefits of soft decision decoding applied to s_c.

5.5 BER-Analysis Against SIRi, Using a Coarse Approximation

Let us introduce a very simple model to evaluate the averaged BER of s_c for soft decision decoding. This model is

$$BERD_c = BERD_i + BERD_c\Big|_{\text{no errors in } s_i} \tag{5-21}$$

simply calculating the sum of BERi after decoding (BERDi) and BERDc, assuming for BERDc that no errors influencing on s_c occur in Δs_i [10]. The factor β described in Section 5.1 was

[10] MATLAB program: *cal_ub_s*

calculated taking BERDi well into account, in order to consider the impact of non-perfect signal suppression, i.e., Δs_i is interfering on s_c with a SIR: $\psi_c' = A_c^2 / \beta^2 A_i^2$.

The last section has shown that BERDi represents an upper bound on BERDc, for SIRi = [0dB ... 3dB]. At these SIRs s_i and s_c are highly correlated and even soft decision decoding is not likely to give additional gain. Correlated errors in s_c are usually having large distances from the decision threshold (see Figure 5-2b), thus, large metric values which makes them appear to the soft decision decoder like reliably detected bits. (The soft decision decoder was described in Section 4.4.) Correlated errors are also more probable in the case of soft decision decoding, which was explained in Section 5.3.4. These facts make good coding of the large signal essential for the performance of the DSR, at least at small values of SIRi.

The correlation between s_i and s_c does not disappear if SIRi increases, but since SNRc decreases in the same way (SNRc = SNRi – SIRi [dB]), noise becomes the more important limiting factor for s_c. In that case, the error rate of s_i causing the correlated errors is very small, and soft decision decoding will enhance the performance of the noise-limited channel.

5.5.1 Results and Conclusions

Figure 5-13 gives the results of the approximation explained above for the BCH(15,7)-code which was considered throughout this chapter. Figure 5-14 gives results for the Golay(24,12)-code which was selected because of its special property of a code rate of R = 1/2 (besides of its excellent performance). This code rate makes this code interesting for a comparison with rate R = 1/2 convolutional codes.

Three different cases are given in each figure: The BERs without any coding to see the enhancement, the BERs for hard decision decoding, and the union bound calculation of the BERs for unquantised maximum likelihood soft decision decoding. The values were calculated by averaging the conditional BERs $\Pr(\varepsilon|\phi',\Delta')$ over ϕ' and Δ'. The dashed lines stand for s_i, the large signal, the solid lines for s_c. The curves for s_i and s_c are, of course, the same for low SIRi values, due to the approximation (Equation (5-21)) defined above.

For the uncoded case, two lines are given for s_c: The solid line stands for the exact BERc results obtained by the calculations proposed in Section 5.1, the dotted line shows the approximation for that case. If we compare these two plots, we can see that they are very

---

Calculates the approximation shown in the graphs of the following section. *PeiPecCd* was used to calculate the uncoded case; *cod_appr* was used for the approximation of the uncoded case and for hard decision decoding; the function *ub_sml*, calling *uq_sml*, calculates the union bound on unquantised soft decision decoding for s_c.

similar for low SIRi-values (the interference limited case) and almost the same for higher values of SIRi, for the noise limited case.

Another comparison was made for the BCH(15,7)-code (Figure 5-13). There are points 'x' which mark results of the exact BER-analysis explained in sections 5.3 and 5.4. The results were obtained by averaging over ϕ' and $\Delta = [0 \ldots 7.5]$. We can see that they are very similar again, the approximation gives an upper bound on the BER, since the small benefits of coding for s_c are not taken into account there.

There are also some points marked by small circles 'o' and by crosses '+' on the curves. Those points are indicating ranges of SIRi within them the BER is smaller than a certain threshold value for any combination of ϕ' and Δ'. I.e., the BER for $\phi' = 0$ and $\Delta' = 0$, which is known to be the worst case, equals the thresholds there. According to the definition of the outage rate in Section 4.2 we have OaR = 0 between those marks. The threshold value for the points marked by 'o' is 0.01, for the points '+' it is 10^{-3}.

For the interference limited case (left marks), the averaged BER-values have much smaller values than the threshold value, because BERi is varying with ϕ' and Δ' a lot; for the noise limited case they are almost equal, indicating less influence of ϕ' and Δ'. These points also mark the range, where we can expect the union bound calculations to give good results. For high input BERs the union bound results are far too high, which makes a comparison, especially at SIRi = 0dB, impossible. (See Section 4.5 for the analysis at SIRi = 0dB.)

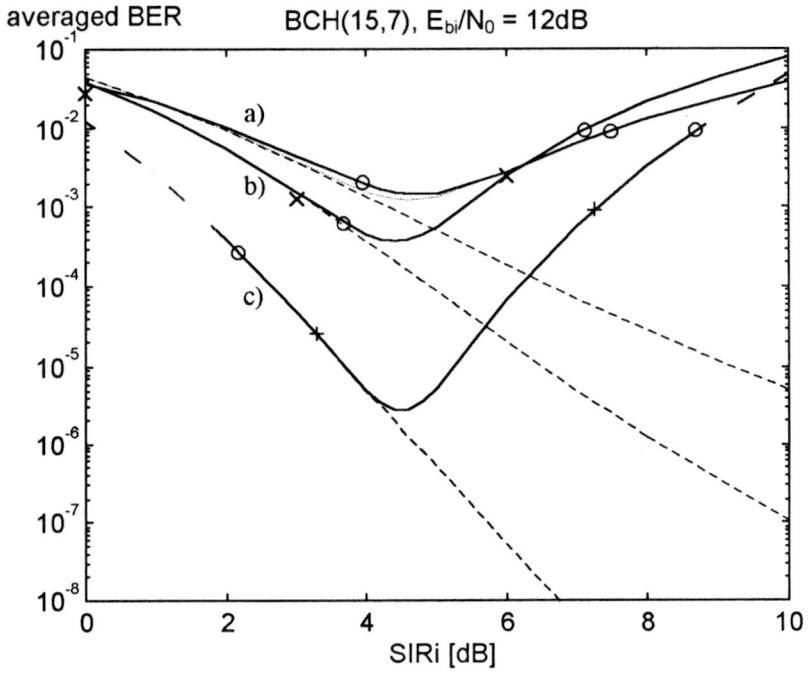

Figure 5-13: Averaged bit-error-rates against SIRi; curves (a) no coding (for comparison), curves (b) BCH(15,7)-coding with hard decisions, curves (c) BCH(15,7)-coding with maximum-likelihood soft decision decoding; dashed curves: $BERD_i$, solid curves: $BERD_c$.

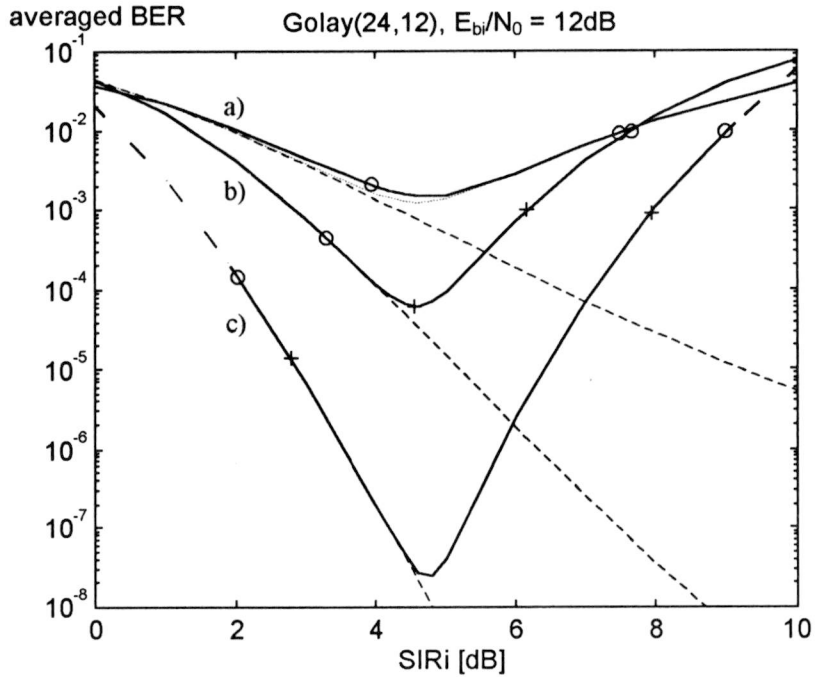

Figure 5-14: Averaged BERs against SIRi for Golay(24,12)-coding; curves (a) no coding, curves (b) hard decision decoding, curves (c) maximum-likelihood soft decision decoding; dashed curves: $BERD_i$, solid curves: $BERD_c$.

We can conclude from Figure 5-13 and Figure 5-14 that there is a rather wide range of SIRi-values within which we can expect reliable communications particularly with soft decision decoding. (Reliable communications means: BER < 10^{-3} which is the Quality of Service (QoS) requirement for speech communications.) For the Golay(24,12)-code with soft decision decoding, we have BERc < 10^{-3} for SIRi = [3dB ... 8dB] at SNRi = 12dB. By comparison to results without coding we can estimate a coding gain of 3 – 5dB for the noise limited case (depending on BER) and even 6 – 7dB for the interference limited case (see BERi in Chapter 4). Note that OaR = 0 for this QoS requirement was never reached without coding at SNRi = 12dB.

Soft decision decoding also gives large improvement compared to hard decision decoding. However, the results for an implemented soft decision decoder will not reach the bound given in this section. We can neither implement unquantised soft decisions, nor a real maximum likelihood algorithm. On the other hand, these results give an upper bound on the BER, therefore, we can expect at least the QoS-range to be very similar to the presented one.

5.6 Conclusions and Recommendations

From Sections 5.1 and 5.2, the analysis of the BER-mechanism of s_c, we have seen that *bit errors* in s_c are very highly correlated to errors in s_i, especially in those cases which are causing most of the errors, i.e., ϕ' is small. Averaged BER and OaR results were given in Chapter 2 for several values of SNRi and varying SIRi.
It is interesting to consider different signal-to-interference ratios SIRi, let us start with SIRi = 0dB: The correlation analysis from Section 5.2 showed that the conditional bit error probability representing this correlation decreases significantly for $\Delta' = 0.5$. In this situation some gain of performance is possible for s_c, we obtained slightly better values of averaged BER and OaR (Chapter 2). For SIRi = 3dB (e.g.) this gain disappears completely: If $\Delta' = 0.5$ now, one error in s_i is causing even two errors in s_c with a rather high probability, hence BERc > BERi. This increasing influence is logical, since SIRi = 3dB means that $A_i > A_c$, the large signal having greater amplitude than the small signal. In these particular cases, where SIRi is rather small, interference causes the majority of the errors in s_c, we call this the "interference limited case".
Increasing SIRi also means decreasing SNRc because SNRi is fixed, i.e., noise becomes the more important limiting factor for s_c since the error rate of s_i is already very low in these cases. We can see that BERc becomes much worse than BERi for SIRi > 4-5dB. That is the "noise limited case".

Sections 5.3 and 5.4 gave BER results for the DSR with BCH(15,7)-coding applied to both signals. We have seen that even with coding the correlation between s_i and s_c hardly reduces. The results for the interference limited case showed to be slightly better for s_c, but we must not forget that a rather "minimum-influencing" error event was assumed for those evaluations.

Sections 5.1 to 5.4 showed in general that in the interference limited case (SIRi = [0 ... 4-5dB]) BERc can be approximated rather closely by BERi. For higher values of SIRi noise becomes the dominating impact. In Section 5.5, we used these facts to define a simplified model, determining BERDc by simply adding BERDi (dominating for SIRi < 4-5dB) and BERDc (for the noise limited case). A comparison with exact results shows that the approximation is very good.
We conclude for further investigations (e.g. Convolutional codes) that it is sufficient to consider the performance of BERi for low SIRi values (the interference limited case) and the performance of BERc in the noise limited case. In Section 5.5 we used this model to evaluate the benefit of soft decision decoding. Applying soft decision Golay(24,12) decoding, it turned out that there is a wide range of SIRi where the outage rate is 0, i.e., BERc < $BER_{max} = 10^{-3}$

(SNRi = 12dB). Without coding OaR = 0 was never reached for these parameters. Comparing the averaged BER results of that coding scheme (Figure 5-14) to the uncoded results given in Figure 2-2, we can estimate a coding gain of 3 – 5dB for the noise limited case and even 6 – 7dB for the interference limited case.

A method to yield additional improvement for s_c by soft decision decoding could be to determine the probable error events in s_i from their low metric values, and assign low metric values to the corresponding (influenced) symbols of s_c as well. An appropriate method to do this assignment has to use the metric values of both signals and the signal parameters provided by the DSR (SIRi, ϕ' and Δ').

6. A Proposal for GSM

One of the characteristics of a fading environment is that the signal strength at the receiver changes significantly due to multipath interference. Models of fading channels are much more suitable for the investigation of mobile communications systems than the simple noise corrupted channel model used in this work [e.g. 13, 14].
In this work we have assumed the signal parameters to be constant during the transmission of the digital signal. Due to fading these signal parameters will change and we have to expect outage situations, where whole error bursts occur at the receiver output (for continuous transmission). Fortunately, we can also expect reliable communications in the cases of good parameters. Code-interleaving is a robust method to correct error bursts, using well-known coding techniques like the block codes described in this work [9, pp. 468 - 470].
In the following section we will consider the DSR applied to the GSM-System, the currently used digital mobile phone system in Europe. TDMA and code-interleaving are implemented in this system.

6.1 The DSR Applied to the GSM-System

The GSM-System is a potential candidate for the use of the DSR [5]. One speciality of the radio interface of the GSM is the use of Time Division Multiple Access (TDMA), transmitting the coded data in short bursts of 116 bits [1]. Referring to the original specifications, the TDMA scheme consists of 8 time slots of 0.58ms duration, i.e., the next time slot of one user appears after 7 other time slots, the frame duration of 4.615ms. For such small time intervals we can indeed assume our channel conditions (thus the parameters) to be constant, i.e. the coding gains evaluated in this work can be achieved for single data bursts. On the other hand, we have to expect outage situations, where a whole data burst gets lost. Interleaving of coded bits between several bursts is implemented in the GSM to improve the performance of error correction coding in such a bursty environment. Interleaving means mixing up the bits of several code words, so that the bits which are close to another in the modulated signal are spread over several code words so that errors within a code word appear to be independent and not bursty any more.
Code interleaving applied to the coding enhanced DSR, as it is proposed in this work, means that the input signal has to be delayed for several bursts, i.e. for several frame times, until decoding and cancellation of the strong signal can be performed. That is not practicable.

Soft decision decoding is an excellent method to improve the performance of the large signal. Unfortunately, it is not possible to apply soft decision decoding to the small signal with equivalent performance, since bits suppressed with an erroneous estimate of the large signal appear to the soft decision decoder like reliably detected bits (Section 5.1, Figure 5-2b). The chase algorithm for soft decision decoding for instance, tries to find the most likely transmitted code word by changing only the less reliably detected bits and, thus, will fail badly for the small signal [11].

Low metrics in detected bits of the large signal give some information about the true reliability of the corresponding bits of s_c, and thus can be used to assign low metric values to these bits as well. The problem of this idea is that low metrics do also appear in cases of high signal-to-noise ratios, where signal suppression is performed properly.

This "metric modification" can be improved by providing additional information about the reliability of the estimated data used for signal cancellation. A suitable coding scheme to obtain this reliability information could be to calculate some additional parity bits for the coded data streams after interleaving and ciphering [1]. It has to be done after ciphering, since the receiver must be able to evaluate the reliability of both, its own and the interfering data streams, and the receiver cannot decipher a foreign signal. The parity bits can be derived from the coded bits of one or one half signal burst, we can expect the data of the whole burst to be either correct or destroyed.

In Figure 6-1 a possible scheme of a Mobile Station is proposed which provides interleaving and soft decision decoding for the desired one of the two signals. The metric modification takes the soft quantised outputs of both decoders, the reliability information from the parity decoding and also receiver parameters into account and decides whether to modify the metrics of s_c or not. If a situation of poor signal suppression (a destroyed burst) is detected, a zero could be assigned to those metrics of s_c where the interference between s_i and s_c was destructive (low metric in s_i). Fortunately, the interference is constructive in half of the cases (assuming $\Delta' = 0$) and the demodulated data is still reliably there, thus no modification needs to be done in these cases. The zeros will be spread over several code words by performing de-interleaving, we can expect good performance of the final soft decision decoding.

[1] Ciphering means protection of the data against usage by a third party which is an important feature for the user. It is achieved by performing an exclusive OR operation between a pseudo-random sequence and the coded bits of a data burst. The pseudo-random sequence is only known by the Base Station and the Mobile Station.

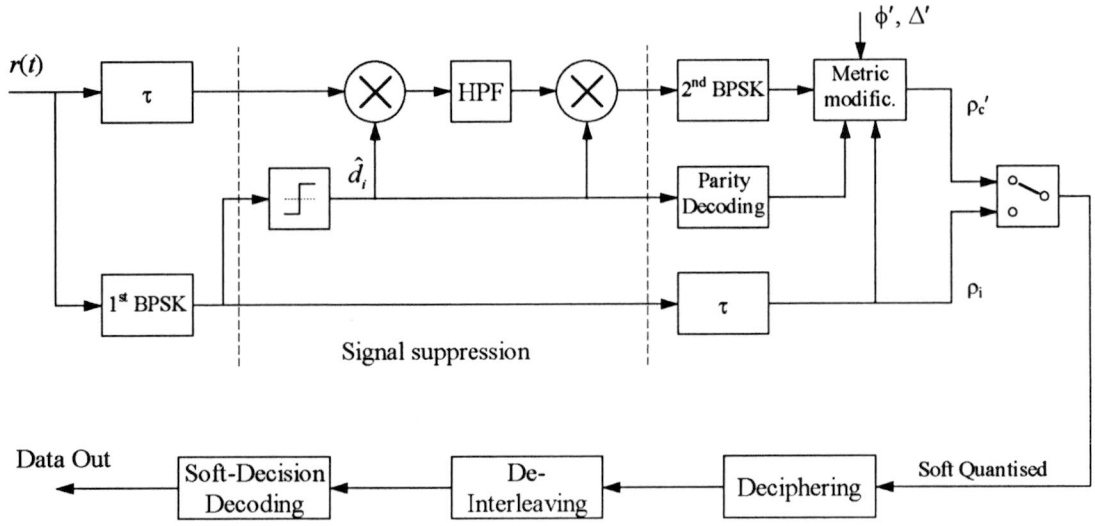

Figure 6-1: A cancellation scheme using the DSR for a GSM Mobile Station. Code-interleaving and soft decision decoding is provided for the desired of both signals.

Evaluation of this system could be done by evaluating how many zeros can be tolerated to achieve a certain BER performance with the applied de-interleaving and decoding, and then evaluating the probability of the reception of non-reliable data bursts.

Antenna diversity techniques should additionally be applied at the Base Station [7]. Unfortunately, these techniques are not suitable for the Mobile Station, since a number of geographically seperated antennas are required for their implementation. That is not practicable for a mobile phone.

7. Conclusions and Recommendations

The DSR-System investigated in this report enables simultaneous reception of two BPSK-modulated co-channel signals. The difference in transmitted signals is used for separation of both signals, i.e., difference in signal strength, carrier phase, data, and bit-timing. Unfortunately, the performance of this receiver becomes bad if these differences are small or missing. In this work FEC-coding is investigated to overcome or at least improve this problem.

7.1 Coding of the Large Signal

The performance analysis of coding applied to the large signal is equivalent to the analysis of coding for a coherent BPSK-demodulator in a (heavy) co-channel interference, additive white Gaussian noise environment.

Several rather short ($n \leq 24$) linear block codes were investigated, union bound computations were proposed to evaluate soft decision decoding.

Especially at low signal-to-interference ratios SIRi ~ 0dB, i.e., both signals have about equal strength, the performance enhancements using hard decision decoding are rather poor compared to the additional bandwidth that is needed.

Soft quantisation of the demodulator output provides information about the reliability of every received coded bit. A soft decision decoder which is processing this additional information to find the most likely transmitted code word enhances the performance significantly.

A fairly simple 8-level quantisation method using linear metrics has almost the performance of the optimum unquantised decoding scheme. Optimisation of the decision thresholds of the 8-level method in order to obtain optimum metrics is possible but rather complex, since many parameters are defining the channel. Thus, an adaptive algorithm is required for the implementation of the optimised scheme.

The achieved coding gains for Golay(24,12) soft decision decoding are: 5 – 7dB at low interference ratios, i.e., strong interference by s_c, and 3 – 5dB for the pure noise limited case.

We see that soft decision decoding gives even more gain for the given co-channel interference environment than for a pure AWGN-channel. This can be explained by the fact that only a part of the transmitted symbols is influenced by destructive interference, the rest of the symbols is reliably detected. The soft decision decoder can separate these bits and only the non-reliable part has to be improved.

Excellent performance with BERs better than 10^{-3} for all situations (OaR = 0, due to our definition) can be achieved for signal-to-interference ratios SIRi > 3dB (all results at SNRi = 12dB).

7.1.1 Performance at SIRi = 0dB

The evaluation of the performance of the DSR at SIRi = 0dB was one of the main goals of this graduation work, since gain at this situation is needed the most, and a good coding scheme for this situation will be advantageous for any situation.
Unfortunately, the union bound computations are only suitable for quite low channel error probabilities, but the error rate of the worst case (SIRi = 0dB, $\phi' = 0$, $\Delta' = 0$) will be 0.25 for all values of SNRi. Therefore, averaging of union bound BER results is not appropriate to evaluate the performance enhancement at very low signal-to-interference ratios. To overcome this problem an outage rate calculation was proposed in Section 4.2. The outage rate OaR gives the probability of the BER being greater than a certain threshold BER_{max}.
Comparing OaR results at SIRi = 0dB with and without coding, we estimated coding gains greater than 10dB for Golay(24,12) soft decision decoding. Although the coding gains look very promising, the actual outage rates are still far away from a value of 0.5% which is required for reliable communications. Thus, other methods have to be considered, one of them is the use of antenna diversity techniques which is currently investigated in the Telecommunications and Traffic Control Systems Group of Delft University of Technology [7].

7.2 Coding of the Small Signal

In the coding-enhanced DSR proposed in this work, an improved estimation of the data of s_i was used to enhance the signal suppression performance, and thus the performance of the small signal. Whether there is additional improvement possible by decoding of the small signal was evaluated in Chapter 5.
An elaborate analysis of the BER-mechanism of the small signal s_c has shown that an error event in the large signal causes an error in s_c with a very high probability as well. Coding does not improve this situation. Therefore, the coding gains obtained for s_c at low signal-to-interference ratios are more less the same as the gains obtained for the large signal. Increasing SIRi means decreasing BERi and BERc first, but decreasing signal-to-noise ratio for s_c (SNRc). Thus, BERc has a minimum and increases again on further increasing SIRi. Noise

becomes the major limiting factor for the performance of s_c. The coding gain for this noise limited case equals 3 – 5dB for Golay(24,12) soft decision decoding. The outage rate analysis for this coding scheme shows excellent results over a wide range of signal-to-interference ratios: The BER performance is always better than $BER_{max} = 10^{-3}$ for SIRi = [3 ... 8dB] (all results at SNRi = 12dB).

Reliable communication with a DSR being used for simultaneous reception of two co-channel signals can be achieved by using a multiple access protocol which ensures this signal-to-interference ratio between the two signals at the receiver. This can be done easily for the AWGN channel assumed in this work, but it will be a matter of gambling in a fading environment as it is usually present with mobile communications.

References

[1] M. Mouly, M.B. Pautet, "The GSM System for Mobile Communications", 1992

[2] G.J.M. Janssen, "Dual-Signal Receiver Structures for Simultaneous Reception of two BPSK Modulated Co-Channel Signals Using Signal Cancellation", Wireless Personal Communications 1: pp. 43–59, Kluwer Academic Publishers, 1994

[3] G.J.M. Janssen, "BER and Outage Performance of a Dual-Signal Receiver for Narrowband BPSK Modulated Co-Channel Signals in a Rician Fading Channel", Proc. PIMRC'94, Vol. 2, pp. 601–606, The Hague, September 1994

[4] G.J.M. Janssen, "A Dual-Signal Receiver for Narrowband DPSK Modulation with Co-Channel Interference Cancellation", Proc. IEEE Third Symposium on Comm. and Veh. Techn. in the Benelux, pp. 115 – 121, Eindhoven, October 1995

[5] A.F. Nollen, G.J.M. Janssen, R. Prasad, "Performance analysis of a Dual Signal Receiver - TDMA/SFH system in a Macro Cellular Environment", IEEE PIMRC'96, Taiwan 1996

[6] M.J. Krapels, G.J.M. Janssen, "Comparison of BER Performance of Different Detectors for a Narrowband BPSK Dual-Signal Receiver with Co-Channel Interference Cancellation", Proc. of the 16th Symposium on Information Theory in the Benelux, pp. 143–150, The Netherlands, May 1995

[7] R.H. Lagarde, "Performance Analysis of a Dual-Signal Receiver with Antenna Diversity Techniques in a Narrowband Rayleigh and Rician Fading Channel", Graduation Thesis, Delft 1996

[8] L.W. Couch, "Digital and Analog Communications Systems", 4th edition, Macmillan, New York, 1990

[9] J.G. Proakis, "Digital Communications", 3rd edition, McGraw Hill, 1995

[10] P. Sweeney, "Error Control Coding: an Introduction", Pretience Hall, 1991

[11] G.C. Clark, J.B. Cain, "Error-Correction Coding for Digital Communications", Plenum Press, New York, 1981

[12] S. Lin, D.J. Costello, "Error Control Coding: Fundamentals and Applications", Pretience Hall, Englewood Cliffs, 1983

[13] F. van der Wijk, A. Kegel, R. Prasad, "Assessment of a Pico-Cellular System Using Propagation Measurements at 1.9 Ghz for Indoor Wireless Communications", IEEE Transactions on Vehicular Technology, Vol. 44, pp. 155–162, February 1995

[14] R.J.C. Bultitude, D.K. Bedal, "Propagation Characteristics on Microcellular Urban Mobile Radio Channels at 910 MHz", IEEE Journal on Selected Areas in Communications, Vol.7, pp. 31 – 39, January 1989.

Appendix: MATLAB programs

The MATLAB programs are listed in order of appearance in the text. Short explanations are given in the programs themselves and in the text as well.

Table of Programs:

fig6_1

```
% fig6_1
%
% averaged BER-results over SIRi without coding

phi = (0:10)*pi/20;
delta = (0:6)/12;
sirdb = [(0:0.5:4) (4.2:.2:6.4) (7:1:30)];
sir = 10.^(sirdb/10);
eta = 10^(-1.5);

aberi = [];
aberc = [];
for siri = sir;
    [Pei12,Pec12] = PeiPecCd(10^1.2,siri,eta,phi,delta);
    [Pei18,Pec18] = PeiPecCd(10^1.8,siri,eta,phi,delta);
    [Pei30,Pec30] = PeiPecCd(10^3.0,siri,eta,phi,delta);

    aberi = [aberi; igrl(igrl(Pei12)'), igrl(igrl(Pei18)'),
          igrl(igrl(Pei30)')];
    aberc = [aberc; igrl(igrl(Pec12)'), igrl(igrl(Pec18)'),
          igrl(igrl(Pec30)')];
end

hold off;
semilogy(sirdb,aberi);
hold;
semilogy(sirdb,aberc);
title(['Ebi/N0 = 12dB, 18dB, 30dB']);
xlabel('SIRi [dB]');
ylabel('averaged BER');
axis([0,30,1e-9,1e-0]);
```

PeiPecCd

```
function [Peicd,Peccd] = PeiPecCd(SNRi,SIRi,eta,phi,deltav)
% function [Peicd,Peccd] = PeiPecCd(SNRi,SIRi,eta,phi,deltav)
%
% calculates the cond. error probability for the large signal (BERi)
% and for the small signal (BERc)
%
% phi-vectors and delta-vectors allowed

Peicd=[]; Peccd=[];

for delta = deltav;      % allow delta - vectors

    Peicond = 0;
    cphi = cos(phi)';
    dck = [-1,1,-(1-2*delta),(1-2*delta)]*sqrt(SNRi/SIRi);
    ifer = cphi*dck;

    peimat = 0.5*erfc(sqrt(SNRi)+ifer);          % Equation (4-5)
    Peicond = sum(peimat')*0.25;

    % ^^^^^^ error probability of si, peimat: depending on dc,0 and dc,1

    pad12 = [2-peimat(:,2)-peimat(:,4),2-peimat(:,1)-
          peimat(:,3),peimat(:,1)+peimat(:,3),peimat(:,2)+peimat(:,4)];
    pad13 = [2-peimat(:,2)-peimat(:,3),2-peimat(:,1)-
          peimat(:,4),peimat(:,1)+peimat(:,4),peimat(:,2)+peimat(:,3)];

    pa01 = [pad13.*(pad12(:,1)*[1 1 1 1]) pad13.*(pad12(:,2)*[1 1 1 1])
          pad13.*(pad12(:,3)*[1 1 1 1]) pad13.*(pad12(:,4)*[1 1 1 1])];

    % ^^^^^^ probabilities of error/non-error patterns di,0' and di,1' (Eq. (5-
          11))

    % error probabilities in dc,0 according to the patterns di,0' and di,1'

    beta = 1-(1-2*Peicond')*(1-eta);

    adi = [beta,-beta,beta-2,2-beta].*(cphi*([1 1 1 1]*sqrt(SNRi)));

    adi0 = adi*delta;
    adi1 = adi*(1-delta);

    addc = [adi0+(adi1(:,1)*[1 1 1 1]) adi0+(adi1(:,2)*[1 1 1 1]) ...
          adi0+(adi1(:,3)*[1 1 1 1]) adi0+(adi1(:,4)*[1 1 1 1])];

    % ^^^^^^ interference matrix

    Pea01 = 0.5*erfc(sqrt(SNRi/SIRi) + addc);    % Equation (5-9)

%pa01*0.0625            % the probabilities of the patterns (di,0',di,1')

    Peccond = sum((pa01 .* Pea01)') *0.0625;     % Equation (5-10)

    Peicd = [Peicd; Peicond];    % results of BERi and BERc for delta-vectors
    Peccd = [Peccd; Peccond];

end
```

igrl

```
function intgrl = igrl(vec)
% function intgrl = igrl(vec)
%
% mean over the rows of vec
% multiply by integration range for integral

[d,n] = size(vec);
if n<=1
intgrl = vec;
return; end;

% Simpson Rule (if possible, i.e., nr. of columns is odd):

if n>=3 & rem(n,2)
gew = ones(n,1)*2;
gew(1)=1;
gew(n)=1;
for i=2:2:n-1;
          gew(i)=4;
end;
gew=gew/(3*(n-1));
intgrl = vec*gew;
return;
end;

% Trapetzoidal approximation, if number of columns is even:

gew = ones(n,1);
gew(1)=0.5;
gew(n)=0.5;
gew=gew/(n-1);
intgrl = vec*gew;
```

calber

```
% program: calber
%
% calculates output bit error rate as a function of
% input block error rate for a specific code
%
% nj: weight-structure of the code
%
% BCH(15,7)

nj = [1,0,0,0,0,18,30,15,15,30,18,0,0,0,0,1]
n = 15;
t = 2;

% BCH(15,5)

%nj = [1,0,0,0,0,0,0,15,15,0,0,0,0,0,0,1]
%n = 15;
%t = 3;

% BCH(15,11) (Hamming)

%nj = [1,0,0,35,105,168,280,435,435,280,168,105,35,0,0,1]
%n = 15;
%t = 1;

% Golay(23,12)

%nj = [1,0,0,0,0,0,0,253,506,0,0,1288,1288,0,0,506,253,0,0,0,0,0,0,1];
%n = 23;
%t = 3;

p = (0:20)*0.5/20;

for i=1:21
    b(i)=berout(p(i),n,t,nj);            % exact calculation

    bb(i) = 0;
    for e=t:-1:0;                 % coding for si
          comb = prod((n-e+1):n)/prod(1:e);
          per = comb*p(i).^e.*(1-p(i)).^(n-e);
          bb(i)=bb(i)+per;        % complement of WER
    end

end

ab = (2*t+1)/n*(1-bb);            % approximation of BER

%plot(bb,b,bb,ab);                % plots BERD against WER
%xlabel('WER');
%ylabel('BERD');

plot(p,b,p,ab);                   % plots BERD against p
xlabel('Channel BER');
ylabel('BERD');

title('BCH(15,7)');
```

berout

```
function pb = berout(p,n,t,nj)
```

```
%function pb = berout(p,n,t,nj)
% calculates the exact output bit error rate of a decoder, applying
% a lenght n code correcting up to t errors
%
% nj: wight distribution vector of lenght n

pb = 0;          % Equation(3-7)

for j=1:n;
    pb = pb + j*(nj(j+1)*pew(p,n,t,j)+pne(p,n,t,nj,j));
end;
pb = pb/n;
```

pew

```
function Pew = pew(p,n,t,j)
%function Pew = pew(p,n,t,j)
%
% calculates the probability that a received bit pattern falls
% in the error correction portion of a weight-j code word

% calculation of the required powers of the channel bit error rate;
% number of errors has to be between j-t and j+t

i = (j-t:j+t);
prob = (p.^i).*((1-p).^(n-i));

Pew = 0;  % Equation (3-6)
for v=0:t;
    for r=0:v;
          combs = prod(j-v+r+1:j)*prod(n-j-r+1:n-j)/(prod(1:v-r)*prod(1:r));
          Pew = Pew + combs*prob(2*r-v+t+1);
    end;
end;
```

pne

```
function Pne = pne(p,n,t,nj,i)
%function Pne = pne(p,n,t,nj,i)
%
% calculates the probability that a received bit pattern
% with i errors is
% not correctable by the code specified by n, t and nj
% nj: weight distribution vector

% number of not correctable patterns:

nrcorr = 0;

for m = -t:t;
    sum = 0;
    j = i-m;
    if (j >= 0) & (j <= n)
          for v = 0:floor((t-abs(m))/2);
              mm = (abs(m)-m)/2;
              mp = (abs(m)+m)/2;
              sum=sum+prod(j-mm-v+1:j)*prod(n-j-mp-v+1:n-
          j)/(prod(1:mm+v)*prod(1:mp+v));
          end;
          nrcorr=nrcorr+nj(j+1)*sum;
    end;
end;

%nrcorr
comb = prod(n-i+1:n)/prod(1:i);

Pne = (comb-nrcorr)*(p.^i).*((1-p).^(n-i));
```

cw_struc

```
% weight-structure analysis of a linear cyclic block code
%
% BCH(15,7)

% generator polynom and -matrix

gen = [1 1,1,0,1 0,0,0,1];
k = 7; n = 15;

G = zeros (k,n);

for i=1:k
    G(i,i:(i+n-k)) = gen;
end

nj=zeros(1,n+1); % weight distribution

tv = pow2(6:-1:0);       % bit-test-vector

% generation of the 2^7 = 128 code words

for wi = 0:(2^k-1);      % word i
    cw = zeros(1,n);
    for bi=1:7           % bit i
          if (wi >= tv(bi))
              cw = xor(cw,G(bi,:));      % gen. of the code word
              wi = wi - tv(bi);
```

```
        end
    end
    weight = sum(cw);
    nj(weight+1) = nj(weight+1) + 1;     % comp. of the weight structure
end
```

OutPoly

```
function [outage,opoly] = outpoly(bermat,bermax)
% function [outage,opoly] = outpoly(bermat,bermax)
%
% calculates a percentage from 0..100, related to the area
% where bermat > bermax
%
% opoly: a row vector containing an index for every colomn of bermax [0..1]
% outage: mean (integral) over these indices

[n,m] = size(bermat);

opoly = [];

for i = 1:m;
    j = 1;
    while j<n & bermat(j,i)>bermax j=j+1; end;
    if j==1 opoly(i) = 1;
    elseif bermat(j,i) > bermax opoly(i) = n;
    else opoly(i) = j-(bermax-bermat(j,i))/(bermat(j-1,i)-bermat(j,i));
    end;
end;

opoly = (opoly-1)/(n-1);

outage = igrl(opoly);
```

uBound

```
function ber = ubound (snr,sir,phi,delta,nj)
% function ber = ubound (snr,sir,phi,delta,nj)
%
% calculation of a union bound on BER for unquantised
% soft decision decoding
%
% nj: wight disribution of the code to check

n = max(size(nj));              % code lenght

ber = 0;

for j = 2:n;
    if nj(j)
        ber = ber + (j-1)/(n-1)*nj(j)*uquant1(snr,sir,phi,delta,j-1);
    end;
end;
```

quant1

```
function pe = quant1(SNRi,SIRi,phi,delta,qt,nj)
% function pe = quant1(SNRi,SIRi,phi,delta,qt,nj)
%
% union bound computation of a code specified by nj
% using quantised soft decision decoding
%
% qt: vector of decoder threshold levels normalized to Ebi

ntl = max(size(qt));
n = max(size(nj))-1;
sp=max(size(phi));
sd=max(size(delta));
pe = zeros(sp,sd);

for ip = 1:sp;          % calculation of pe for phi- and delta-vectors
for id = 1:sd;

dck = [-1,1,-(1-2*delta(id)),(1-2*delta(id))]'*sqrt(SNRi/SIRi);
ifer = dck*cos(phi(ip));

p = [];
pv = [1];

for i = 1:ntl           % cal. of the transition probs.  (Equation (4-10))
    pv(i+1) = sum(0.125*erfc(sqrt(SNRi)*(1+qt(i))+ifer));
    pv(i)=pv(i)-pv(i+1);
    p = [p pv(i) 0];
end;

p = [p pv(ntl+1)];      % calculation of the pdf shown in Figure 4-9
pc = p;


for j = 2:n;            % accumulation as schown in Eq. (4-7)
    pc=conv(pc,p);      % convolution
    if nj(j+1)
        l=max(size(pc));
        m = ceil(l/2);          % middle element of the pc vec

        pe(ip,id) = pe(ip,id) + j/n*nj(j+1)*(0.5*pc(m) + sum(pc(m+1:l))); %
        Eq. (4-14)
```

```
%         [j,j/n,nj(j+1),cnt]
    end;
end;

end;
end;
```

uquant1

```
function pe = uquant1(SNRi,SIRi,phi,delta,j)
% function pe = uquant1(SNRi,SIRi,phi,delta,j)
%
% calculates the error prob of a wight j sequence in case
% of unquantised demodulator outputs
% interfering patterns quantized to (-1:1)

sd=max(size(delta));
pe = [];

res = 6*2;
quants = j*res;          % nr. of influencing patterns
infl = (0:quants)'*(2/quants)-1; % influencing weights ifv"

niff=zeros(max(size(delta)),quants+1);
iff = [];

for i1 = 0:j;                              % interfering patterns v'
for ix1 = 0:j-i1;
for ix2 = 0:j-ix1-i1;
    comb = prod(j-i1-ix1-ix2+1:j)/(prod(1:i1)*prod(1:ix1)*prod(1:ix2));
    iff = round((i1+(1-delta')*ix1+delta'*ix2)*res)+1;    % quantisation
    for id = 1:sd;
        niff(id,iff(id)) = niff(id,iff(id)) + comb;     % #v"
    end;
end; end; end;
piff = niff/pow2(2*j);                     % Pr(v")

dck = infl*sqrt(j*SNRi/SIRi);
ifer = dck*cos(phi);                       % interference terms

pf = 0.5*erfc(sqrt(j*SNRi)+ifer);          % error probs; Eq. (4-17)


for id = 1:sd;           % calculation of pe for delta-vectors

pee = piff(id,:)*pf;     % multiply by Pr(v")
pe = [pe pee'];
```

```
end;
```

CalUbnd1

```
% program: calubnd
%
% calculates union bounds of ber for the dsr and
% codes specified by nj
%
% nj: wight distribution vector
% R:  code rate R=k/n
% qt: quantization threshold vector

%disp('shortened Hamming(13,8)')

%nj = [1,0,0,11,22,36,60,58,33,20,12,3,0,0]
%R = 8/13;

%disp('nonprimitive BCH(17,9)');

%nj = [1,0,0,0,0,34,68,68,85,85,68,68,34,0,0,0,0,1]
%R = 9/17;

%disp('Hamming (7,4)')

%nj = [1,0,0,7,7,0,0,1]
%R = 4/7;

%disp('BCH(15,11) (Hamming)')

%nj = [1,0,0,35,105,168,280,435,435,280,168,105,35,0,0,1]
%R = 11/15;

%disp('BCH(15,5)')

%nj = [1,0,0,0,0,0,0,15,15,0,0,0,0,0,0,1];
%R=5/15;

disp('Golay(24,12)');

nj = [1,0,0,0,0,0,0,0,759,0,0,0,2576,0,0,0,759,0,0,0,0,0,0,0,1];
R = 12/24;

%disp('Golay(23,12)');

%nj = [1,0,0,0,0,0,0,253,506,0,0,1288,1288,0,0,506,253,0,0,0,0,0,0,1];
%R = 12/23;

%disp('BCH(15,7)')
```

```
%nj = [1,0,0,0,0,18,30,15,15,30,18,0,0,0,0,1];
%R = 7/15;

qt8 = (-6:2:6)/7;        % 8-level uniformly spaced quantization
qt4 = (-4:4:4)/7;        % 4-level uniformly spaced quantization
qt3 = [-1,1]*0.15;       % 3-level (erasure zone) quantization
qt2 = [0];               % binary quantization

phi = (0:20)*pi/40;
delta = (0:6)/12;
snr = 15.8*R;
sir = 1;

% calc. of the BERs for the phi and delta - vectors given above

ber2 = quant1 (snr,sir,phi,delta,qt2,nj)
ber3 = quant1 (snr,sir,phi,delta,qt3,nj)
ber4 = quant1 (snr,sir,phi,delta,qt4,nj)
ber8 = quant1 (snr,sir,phi,delta,qt8,nj)
ber0 = ubound (snr,sir,phi,delta,nj)

% calc. of the outage ratios using linear and logarithmic interpolation

[outpoly(log10(ber2),-2), outpoly(ber2,0.01), outpoly(log10(ber2),-3),
         outpoly(ber2,0.001)]
[outpoly(log10(ber3),-2), outpoly(ber3,0.01), outpoly(log10(ber3),-3),
         outpoly(ber3,0.001)]
[outpoly(log10(ber4),-2), outpoly(ber4,0.01), outpoly(log10(ber4),-3),
         outpoly(ber4,0.001)]
[outpoly(log10(ber8),-2), outpoly(ber8,0.01), outpoly(log10(ber8),-3),
         outpoly(ber8,0.001)]
[outpoly(log10(ber0),-2), outpoly(ber0,0.01), outpoly(log10(ber0),-3),
         outpoly(ber0,0.001)]
```

CalUbnd6

```
% program: calubnd6
%
% averaged BERs against SNRi

% codes specified by nj
%
% nj: wight distribution vector
% R:  code rate R=k/n
% qt: quantization threshold vector

%tit = 'Golay(24,12)';

%nj = [1,0,0,0,0,0,0,0,759,0,0,0,2576,0,0,0,759,0,0,0,0,0,0,0,1];
%R = 12/24; n = 24; t = 3; dmin = 8;

%disp('Golay(23,12)');

%nj = [1,0,0,0,0,0,0,253,506,0,0,1288,1288,0,0,506,253,0,0,0,0,0,0,1];
%R = 12/23;

tit = 'BCH(15,7)';

nj = [1,0,0,0,0,18,30,15,15,30,18,0,0,0,0,1];
R = 7/15; n = 15; t = 2; dmin=5;

disp(tit);

qt8 = (-6:2:6)/7;        % 8-level uniformly spaced quantization
qt4 = (-4:4:4)/7;        % 4-level uniformly spaced quantization
qt3 = [-1,1]*0.15;       % 3-level (erasure zone) quantization
qt2 = [0];               % binary quantization

phi = (0:10)*pi/20;
delta = (0:6)/12;
snr = 15.8;
sir = logspace(0,1,21);
eta = 0;

aber1 = [];
aber2 = [];
abers = [];
for siri = sir;
    ber2 = quant1 (snr*R,siri,phi,delta,qt2,nj)
    ber3 = quant1 (snr*R,siri,phi,delta,qt3,nj)
    ber4 = quant1 (snr*R,siri,phi,delta,qt4,nj)
    ber8 = quant1 (snr*R,siri,phi,delta,qt8,nj)
    ber0 = ubound (snr*R,siri,phi,delta,nj)
    [Peicd,Peccd] = PeiPecCd(snr,siri,eta,phi,delta);
    [rpbei,Peccd] = PeiPecCd(snr*R,siri,eta,phi,delta);

    cpbei = 0;
    for e=t:-1:0;                          % coding for si
        comb = prod((n-e+1):n)/prod(1:e);
        per = comb*rpbei.^e.*(1-rpbei).^(n-e);
        cpbei=cpbei+per;
    end

    cpbei = (1 - cpbei)*(dmin/n);          % calculate approx. BER

    aber = [igrl(igrl(ber2)'), igrl(igrl(ber3)'), igrl(igrl(ber4)'),
        igrl(igrl(ber8)'), igrl(igrl(ber0)')];
```

```
    aber1 = [aber1; igrl(igrl(Peicd)')];
    aber2 = [aber2; igrl(igrl(cpbei)')];
    abers = [abers; aber];
end;

hold off;
semilogy(0:.5:10,aber1);
hold;
semilogy(0:.5:10,aber2);
semilogy(0:.5:10,abers);
title([tit ', Ebi/N0 = 12dB']);
xlabel('SIRi [dB]');
ylabel('averaged BER');
axis([0,10,1e-8,1e-1]);
```

CalUbnd5

```
% program: calubnd5
%
% calculates union bounds on BER against SNRi for
% codes specified by nj
%
% nj: wight distribution vector
% R:  code rate R=k/n
% qt: quantization threshold vector

%tit = 'Golay(24,12)';

%nj = [1,0,0,0,0,0,0,0,759,0,0,0,2576,0,0,0,759,0,0,0,0,0,0,0,1];
%R = 12/24; n = 24; t = 3; dmin = 8;

%disp('Golay(23,12)');

%nj = [1,0,0,0,0,0,0,253,506,0,0,1288,1288,0,0,506,253,0,0,0,0,0,0,1];
%R = 12/23;

tit = 'BCH(15,7)';

nj = [1,0,0,0,0,18,30,15,15,30,18,0,0,0,0,1];
R = 7/15; n = 15; t = 2; dmin = 5;

disp(tit);

qt8 = (-6:2:6)/7;       % 8-level uniformly spaced quantization
qt4 = (-4:4:4)/7;       % 4-level uniformly spaced quantization
qt3 = [-1,1]*0.15;      % 3-level (erasure zone) quantization
qt2 = [0];              % binary quantization

phi = (0:10)*pi/20;
delta = (0:6)/12;
snr = logspace(0,2,21);
sir = [2,4,1000];
eta = 0;

aber1 = [];
aber2 = [];
aber3 = [];
for snri = snr;
ber1 = []; ber2 = []; ber3 = [];
for siri = sir;
    ber0 = ubound (snri*R,siri,phi,delta,nj,999)
    [Peicd,Peccd] = PeiPecCd(snri,siri,eta,phi,delta);
    [rpbei,Peccd] = PeiPecCd(snri*R,siri,eta,phi,delta);

    cpbei = 0;
    for e=t:-1:0;                          % coding for si
         comb = prod((n-e+1):n)/prod(1:e);
         per = comb*rpbei.^e.*(1-rpbei).^(n-e);
         cpbei=cpbei+per;
    end

    cpbei = (1 - cpbei)*(dmin/n);          % calculate approx. BER

    ber1 = [ber1, igrl(igrl(Peicd)')];
    ber2 = [ber2, igrl(igrl(cpbei)')];
    ber3 = [ber3, igrl(igrl(ber0)')];
end;
aber1 = [aber1; ber1];
aber2 = [aber2; ber2];
aber3 = [aber3; ber3];
end

hold off;
semilogy(0:20,aber1);
hold;
semilogy(0:20,aber2);
semilogy(0:20,aber3);
title([tit ', SIRi = 3, 6, 30dB']);
xlabel('SNRi [dB]');
ylabel('averaged BER');
axis([0,20,1e-8,1e-1]);
```

Fig4_4

```
% fig4_4
%
% correlation between errors if si and sc
```

```
snr = 15.8;
sir = 10^.0;
eta = 1e-3;

delta = (0:20)/20;
phi = (0:10)*pi/20;

pecorr=zeros(max(size(delta)),max(size(phi)));

for i=1:max(size(delta))
    for j = 1:max(size(phi))
          pe = correrr1(snr,sir,eta,phi(j),delta(i));
          pecorr(i,j) = pe(1);
    end;
end;

mesh(phi,delta,pecorr)

view(150,30)
axis([0,pi/2,0,1,0,1])
title('SNRi = 12dB, SIRi = 0dB')
xlabel('phi')
ylabel('delta')
zlabel('PeCorr')
```

CorrErr1

```
function [pecorr,pei0] = CorrErr(SNRi,SIRi,eta,phi,delta)
% CorrErr(SNRi,SIRi,eta,phi,delta)
%
% pei0 are the probabilities of an error '0' in si (bit 1 was sent)
%      where (dc0,dc1) = [(00),(11),(01),(10)]
% pecorr are the probs. for errornous bits dc0 and dc1, respectively
%      [dc0=0,dc0=1,dc1=0,dc1=1]

Peicond = 0;
cphi = cos(phi)';
dck = [-1,1,-(1-2*delta),(1-2*delta)]*sqrt(SNRi/SIRi);
ifer = cphi*dck;

peimat = 0.5*erfc(sqrt(SNRi)+ifer);
Peicond = sum(peimat')*0.25;
pei0 = peimat;

% ^^^^^^ error probability of di0, peimat: depending on dc0 and dc1

pdi_1 = [(peimat(:,1)+peimat(:,4))*0.25,(peimat(:,2)+peimat(:,3))*0.25];
pdi_1 = [.5-fliplr(pdi_1) pdi_1 .5-pdi_1 fliplr(pdi_1)];

% ^^^^^^ probs of di-1 depending on dc0

pdi1 = [(peimat(:,1)+peimat(:,3))*0.25,(peimat(:,2)+peimat(:,4))*0.25];
pdi1 = [.5-fliplr(pdi1) pdi1 .5-pdi1 fliplr(pdi1)];

% ^^^^^^ probs of di1 depending on dc1

% error probabilities in dc according to the symbols di1 and di-1 of di

beta = 1-(1-2*Peicond')*(1-eta);

adi = [beta,-beta,beta-2,2-beta].*(cphi*([1 1 1 1]*sqrt(SNRi)));

adi0 = adi*delta;
adi1 = adi*(1-delta);

addc = [adi0+(adi1(:,1)*[1 1 1 1]) adi0+(adi1(:,2)*[1 1 1 1])
        adi0+(adi1(:,3)*[1 1 1 1]) adi0+(adi1(:,4)*[1 1 1 1])];

% ^^^^^^ interference matrix

Pea01 = 0.5*erfc(sqrt(SNRi/SIRi) + addc);

pec_1 = [Pea01(:,9:12),Pea01(:,14),Pea01(:,13),Pea01(:,16),Pea01(:,15)];
pec1 =
         [Pea01(:,3),Pea01(:,7),Pea01(:,11),Pea01(:,15),Pea01(:,8),Pea01(:,4),
         Pea01(:,16),Pea01(:,12)];

pecorr = sum(reshape([pdi_1.*pec_1 pdi1.*pec1],4,4));
```

cstruc

```
% codestructure analysis
%
% BCH(15,7)

% generator polynom and -matrix

gen = [1 1,1,0,1 0,0,0,1];
k = 7; n = 15;

G = zeros (k,n);

for i=1:k
    G(i,i:(i+n-k)) = gen;
end

dmin = 5; % minimum Hamming distance
```

```
ndmin=18*10;     % number of code words having this wight and
                 % combinations of two uncorrelated errors

nj=zeros(1,n+1); % weight distribution

nec=zeros(ndmin,2*n);    % matrix to count the various influencing symbols
cod = pow2((0:2:14)')*ones(1,30);        % used to code necl into nec
struc_i = 1;

tv = pow2(6:-1:0);       % bit-test-vector

% generation of the 2^7 = 128 code words

for wi = 0:(2^k-1);      % word i
    cw = zeros(1,n);
    for bi=1:7           % bit i
        if (wi >= tv(bi))
            cw = xor(cw,G(bi,:));      % gen. of the code word
            wi = wi - tv(bi);
        end
    end
    weight = sum(cw);
    nj(weight+1) = nj(weight+1) + 1;     % comp. of the weight structure

%
% ? is the code word a word of weight dmin ?
%

    if (weight == dmin)          % analysis of code structure

        bone = zeros(1,5);       % contains the positions of the ones
        j = 1;
        for i = 1:n
            if cw(i) bone(j) = i; j=j+1; end
        end

        cw_buf = cw;             % insert two uncorrelated errors:
        for uc1 = 1:4; for uc2 = uc1+1:5;
            cw = cw_buf;
            cw(bone(uc1)) = 2;
            cw(bone(uc2)) = 2;

            testw = [cw 0] + 3*[0 cw]; % calc. test word

            necl = zeros(8,2*n);       % local array to count the symbols
            for i = 1:n+1              % count symbols
              if testw(i)
              necl(testw(i),(i:i+n-1)) = necl(testw(i),(i:i+n-1)) + 1;
              end;
            end

            nec(struc_i,:) = sum(necl.*cod);    % coding of necl into nec
            struc_i = struc_i + 1;

        end; end;            % end cw with uncorr. errors

    end                      % end anal. of the c. struc

end                          % end of cw - generation

save 'nec.mat' nec
```

wer_sml

```
% wer_sml
%
% calculates the word error rate for the small signal
% with BCH(15,7) hard decision decoding

snr = 15.8;
sir = 2;
eta = 10^(-3/2);
phi = (0:2)*pi/6;
phi = (0:20)*pi/40;
delta = (0:7)/8;
phis = max(size(phi));
deltas = max(size(delta));

n = 15;          % code parameters. It's not possible to change them simply
k = 7;
t = 2;
R = k/n;
load 'nec'

global n k t;

% calculate the error probabilities of dc (pec)

[pec,pb] = cal_pec(snr*R,sir,eta,phi,delta);              % calculation of
          error probs
pecc = 1-pec;

global pec_pow pec_c_pow;

pec_pow = [pec(1,:);pec(1,:).*pec(1,:); pec(2,:);pec(2,:).*pec(2,:); ...
           pec(3,:);pec(3,:).*pec(3,:); pec(4,:);pec(4,:).*pec(4,:); ...
           pec(5,:);pec(5,:).*pec(5,:); pec(6,:);pec(6,:).*pec(6,:); ...
           pec(7,:);pec(7,:).*pec(7,:); pec(8,:);pec(8,:).*pec(8,:); ...
```

```
        pec(9,:);pec(9,:).*pec(9,:)];

pec_c_pow = [];
for i = 1:9;
    pec_c_pow = [pec_c_pow;pecc(i,:); pecc(i,:).*pecc(i,:); pecc(i,:).^3];
end
for i = 4:15;
    pec_c_pow = [pec_c_pow;pecc(9,:).^i];
end

%pe = zeros(phis*deltas,30);            % calc. error probs
pe = zeros(phis*deltas,16);             % calc. error probs
for i = 1:180;
    i
    pe = pe + err_prob(nec(i,:));
end;

pe = pe/180;

pee = [];       % rearrange the pe matrix
for i = 1:16            % 30 !!
for j = 1:deltas
    pee = [pee, pe((j-1)*phis+(1:phis),i)];
end;
end;

pee = fliplr(pee(:,1:121));

plot((0:0.125:15),pee');
xlabel('delta');
ylabel('pe');
title('Eb/N0 = 12dB, SIRi = 0dB');
grid on;

pb = reshape(pb,phis,deltas);    % wi block error probs

pbi = zeros(size(pec(9,:)));     % error prob of the small signal sc
                                 % when no error occured in si

for e=t:-1:0;
        comb = prod((n-e+1):n)/prod(1:e);
        pbi=pbi+comb*pec(9,:).^e.*(1-pec(9,:)).^(n-e);
end

pbi = 1-pbi;

pbi = reshape(pbi,phis,deltas);

save 'wer_dat' pee pb pbi phi delta
```

cal_pec

```
function [pec,pb] = cal_pec(SNRi,SIRi,eta,phi,deltav)
% function [pec,pb] = cal_pec(SNRi,SIRi,eta,phi,deltav)
%
% calculates the correlated and noncorrelated conditional error
% probabilities of dc for the influencing classes 1-9
%
% pec: conditional error probs.
% pb:  word error prob of si

global n k t;

Peicond = 0;
cphi = cos(phi)';

Pea01 = [];
pa01 = [];

for delta = deltav;

dck = [-1,1,-(1-2*delta),(1-2*delta)]*sqrt(SNRi/SIRi);
ifer = cphi*dck;

peimat = 0.5*erfc(sqrt(SNRi)+ifer);
Peicond = sum(peimat')*0.25;

% ^^^^^^ error probability of di, peimat: depending on dc,0 and dc,1

pad12 = [2-peimat(:,2)-peimat(:,4),2-peimat(:,1)-
         peimat(:,3),peimat(:,1)+peimat(:,3),peimat(:,2)+peimat(:,4)];
pad13 = [2-peimat(:,2)-peimat(:,3),2-peimat(:,1)-
         peimat(:,4),peimat(:,1)+peimat(:,4),peimat(:,2)+peimat(:,3)];

papa = [pad13.*(pad12(:,1)*[1 1 1 1]) pad13.*(pad12(:,2)*[1 1 1 1])
        pad13.*(pad12(:,3)*[1 1 1 1]) pad13.*(pad12(:,4)*[1 1 1 1])];
pa01 = [pa01;papa];

% ^^^^^^ probabilities of error/non-error patterns A0, A1

% Block error probability using the code specified by the global parameters
        n,k,t

pbi = zeros(size(Peicond));

for e=t:-1:0;
        comb = prod((n-e+1):n)/prod(1:e);
        pbi=pbi+comb*Peicond.^e.*(1-Peicond).^(n-e);
end
```

```
pbi = 1-pbi;
pb = [pb,pbi];

% error probabilities in dc according to the patterns A0, A1

beta = 1-(1-(2*(2*t+1)/n)*pbi')*(1-eta); % signal suppression with BCH coding
%beta = 1-(1-2*Peicond')*(1-eta);
%[Peicond;pbi*k/n;beta']


adi = [beta,-beta,beta-2,2-beta].*(cphi*([1 1 1 1]*sqrt(SNRi)));

adi0 = adi*delta;
adi1 = adi*(1-delta);

addc = [adi0+(adi1(:,1)*[1 1 1 1]) adi0+(adi1(:,2)*[1 1 1 1])
        adi0+(adi1(:,3)*[1 1 1 1]) adi0+(adi1(:,4)*[1 1 1 1])];

% ^^^^^^ interference matrix

Pea01 = [Pea01;0.5*erfc(sqrt(SNRi/SIRi) + addc)];

end;      % end delta loop

%Pea01,pa01*0.0625       % error probs for di,0';di,1' - patterns

pp = sum((Pea01(:,[3 4 7 8]).*pa01(:,[3 4 7 8]))')./sum(pa01(:,[3 4 7 8])');
pec = [pp; sum(Pea01(:,[3 4 7 8])')*0.25];
pp = sum((Pea01(:,[9 10 13 14]).*pa01(:,[9 10 13 14]))')./sum(pa01(:,[9 10 13
        14])');
pec = [pec;pp];
pp = sum((Pea01(:,[11 12 15]).*pa01(:,[11 12 15]))')./sum(pa01(:,[11 12 15])');
pec = [pec;pp;sum(Pea01(:,[11 12])')*0.5;sum(Pea01(:,[9 10 13 14])')*0.25; ...
        sum(Pea01(:,[11 15])')*0.5;sum(Pea01(:,[11 12 15 16])')*0.25];
pec = [pec;sum(Pea01(:,[1 2 5 6])')*0.25];
```

err_prob

```
function pe = err_prob(nec)
% function pe = err_prob(nec)
%
% calculates the block error probability of wc, where
% the pattern encoded in nec influences the signal sc
%
% nec: encoded code pattern
% global ber_pow, ber_c_pow: contains the required powers
%                        of the bit error probs

global pec_pow pec_c_pow;

% decoding of nec:
s = max(size(nec));
necl = zeros(8,s);              % uncoded pattern structure

b_test = pow2(0:15);

for i = 8:-1:1                  % decoding of the code structure
    m2 = (nec>=b_test(2*i));
    nec = nec - m2*b_test(2*i);
    m1 = (nec>=b_test(2*i-1));
    nec = nec - m1*b_test(2*i-1);
    necl(i,:) = m1 + 2*m2;
end;
necl = [necl; 15-sum(necl)];

pe = [];

for col = 1:floor(s/2)+1;       % take one column of necl
                                % the comp. is carried out for delta <= 0
    mul = [];           % no errors
    for row = 1:9;              % build matrix mul containing the powers of
        the probs
        if necl(row,col)
            mul = [mul; pec_c_pow((row-1)*3+necl(row,col),:)];
        end;
    end;
    if min(size(mul))>1 mulv = prod(mul); else mulv = mul; end;
    ec = mulv;          % matrix ec: sum up products

    for err = 1:9;      % ONE error
    if necl(err,col) >= 1
        mul = necl(err,col)*pec_pow(err*2-1,:); % prob for the error;
        combinations
        necl(err,col) = necl(err,col) - 1;
        for row = 1:9;                          % probs for the correct
        symbols
            if necl(row,col)
              mul = [mul; pec_c_pow((row-1)*3+necl(row,col),:)];
            end;
        end;
        if min(size(mul))>1 mulv = prod(mul); else mulv = mul; end;
        ec = [ec;mulv];
        necl(err,col) = necl(err,col) + 1;
    end;
    end;

    for err = 1:9;      % TWO errors of one prob.
```

```
    if necl(err,col) >= 2
                    % prob for the error; combinations
        mul = (necl(err,col)*(necl(err,col)-1)*0.5)*pec_pow(err*2,:);
        necl(err,col) = necl(err,col) - 2;
        for row = 1:9;                          % probs for the correct
        symbols
            if necl(row,col)
              mul = [mul; pec_c_pow((row-1)*3+necl(row,col),:)];
            end;
        end;
        if min(size(mul))>1 mulv = prod(mul); else mulv = mul; end;
        ec = [ec;mulv];
        necl(err,col) = necl(err,col) + 2;
    end;
    end;

    for err1 = 1:8;      % TWO errors of two probs
    for err2 = err1+1:9;
    if necl(err1,col) >= 1 & necl(err2,col) >= 1
                    % prob for the errors; combinations
        mul = (necl(err1,col)*necl(err2,col))*(pec_pow(err1*2-
        1,:).*pec_pow(err2*2-1,:));
        necl(err1,col) = necl(err1,col) - 1;
        necl(err2,col) = necl(err2,col) - 1;
        for row = 1:9;                          % probs for the correct
        symbols
            if necl(row,col)
              mul = [mul; pec_c_pow((row-1)*3+necl(row,col),:)];
            end;
        end;
        if min(size(mul))>1 mulv = prod(mul); else mulv = mul; end;
        ec = [ec;mulv];
        necl(err1,col) = necl(err1,col) + 1;
        necl(err2,col) = necl(err2,col) + 1;
    end;
    end;
    end;

    pe = [pe, (sum(ec))'];
end;

pe = 1-pe;
```

ber_sml

```
% ber_sml
%
% calculates bit error rates of the small signal
% using the data calculated by wer_sml

load 'wer_dat'  % pee, pb, pbi, delta, phi

% wer_dat0 : SIRi = 0dB; wer_dat3 : SIRi = 3dB

phis = max(size(phi));
deltas = max(size(delta));

p_one_e = pee + fliplr(pee);

pbc = 1-pb;

% no block errors in wi0 and wi1:

p_block_e = pbc.*pbc.*pbi        % regardless of m

% two block errors

p_block_e = p_block_e + pb.*pb.*pee(:,(1:deltas))

% one block error in wi0 or wi1:

pbe = [];
pbb = [];

for delta_m = 0:10;      % sequence delay; integer part
    pbe = [pbe, p_block_e + pbc.*pb.*p_one_e(:,delta_m*deltas+(1:deltas))];
    pbb = [pbb, pb];
end

pbe = pbe * 5/15;
pbb = pbb * 5/15;

figure(1);
hold off;
plot((0:.125:7.5),pbe(:,1:61)','-',(0:.125:7.5),pbb(:,1:61)','-');
hold
plot((0:.5:7.5),pbe(:,1:4:61)','+',(0:.5:7.5),pbb(:,1:4:61)','x');
xlabel('delta');
ylabel('BERi,c');
title('Eb/N0 = 12dB, SIRi = 0dB');
grid on;

figure(2);
hold off;
semilogy((0:.125:7.5),pbe(:,1:61)','-',(0:.125:7.5),pbb(:,1:61)','-');
hold
semilogy((0:.5:7.5),pbe(:,1:4:61)','+',(0:.5:7.5),pbb(:,1:4:61)','x');
xlabel('delta');
```

```
ylabel('BERi,c');
title('Eb/N0 = 12dB, SIRi = 0dB');

figure(3);
hold off;
plot((0:.125:7.5),pbe(:,1:61)'./pbb(:,1:61)','-');
hold
plot((0:.5:7.5),pbe(:,1:4:61)'./pbb(:,1:4:61)','+');
xlabel('delta');
ylabel('BERc/BERi');
title('Eb/N0 = 12dB, SIRi = 0dB');
grid
```

cal_ub_s

```
% program: cal_ub_s
%
% calculates averaged BERD results as defined in Section 5.5
%
% codes specified by:
%
% nj: wight distribution vector
% R:  code rate R=k/n
% qt: quantization threshold vector


%tit='Golay(24,12)';

%nj = [1,0,0,0,0,0,0,0,759,0,0,0,2576,0,0,0,759,0,0,0,0,0,0,0,1];
%R = 12/24; n=24; t=3; dmin = 8;

tit='BCH(15,7)';

nj = [1,0,0,0,0,18,30,15,15,30,18,0,0,0,0,1];
R = 7/15; n=15; t=2; dmin = 5;

disp (tit);

qt8 = (-6:2:6)/7;        % 8-level uniformly spaced quantization
qt4 = (-4:4:4)/7;        % 4-level uniformly spaced quantization
qt3 = [-1,1]*0.15;       % 3-level (erasure zone) quantization
qt2 = [0];               % binary quantization

phi = (0:10)*pi/20;
delta = (0:6)/12;
snr = 15.8;
%sir = logspace(0,1,11);
sirdb = [(0:4) (4.2:.2:4.8) (5:10)];
sir = 10.^(sirdb/10);
eta = 10^(-1.5);

bers = [];
aberi = [];
aberc = [];
for siri = sir;
    [beri0,berc0] = ub_sml (snr*R,siri,phi,delta,eta,nj)          % ubond, soft
    [pbei,pbec,cpbei,cpbec]=cod_appr(snr,siri,phi,delta,eta,n,t,dmin,R)   %
          approx. hard
    [Peicd,Peccd] = PeiPecCd(snr,siri,eta,phi,delta)                      % no-
          coding

    aberi = [aberi; igrl(igrl(beri0)'), igrl(igrl(pbei)'), igrl(igrl(cpbei)'),
          igrl(igrl(Peicd)')];
    aberc = [aberc; igrl(igrl(berc0)'), igrl(igrl(pbec)'), igrl(igrl(cpbec)'),
          igrl(igrl(Peccd)')];
end

hold off;
semilogy(sirdb,aberi);
hold;
semilogy(sirdb,aberc);
title([tit ', Ebi/N0 = 12dB']);
xlabel('SIRi [dB]');
ylabel('averaged BER');
axis([0,10,1e-8,1e-1]);
```

cod_appr

```
function [pbei,pbec,cpbei,cpbec] =
          cod_appr(SNRi,SIRi,phi,deltav,eta,n,t,dmin,R)
% function [pbei,pbec,cpbei,cpbec] =
          cod_appr(SNRi,SIRi,phi,deltav,eta,n,t,dmin,R)
%
% calculates the cond. err. prob. for the large signal si; pbei
% for the small signal after cancellation of si, assuming no errors: pbec
% and the same values with hard decision coding; parameters: n,t
%
% attention: SNRi is not changed to Eb/N0

pbei=[];pbec=[];cpbei = 0;cpbec = 0;
deltas = max(size(deltav));

for delta = deltav                % BERs for si
    dck = [-1,1,-(1-2*delta),(1-2*delta)]'/sqrt(SIRi);
    ifer = dck*cos(phi);
    pe = sum(0.125*erfc(sqrt(SNRi)*(1+ifer)));
    pbei = [pbei; pe];
end;
```

```
beta = 1-(1-2*pbei)*(1-eta);     % BERs for sc, assuming no errors in si
SNRc = SNRi/SIRi;
SIRc = 1/SIRi;

for id = 1:deltas
    dck = [-1,1,-(1-2*deltav(id)),(1-2*deltav(id))]'/sqrt(SIRc);
    ifer = (dck*cos(phi)).*([1 1 1 1]'*beta(id,:));
    pe = sum(0.125*erfc(sqrt(SNRc)*(1+ifer)));
    pbec = [pbec; pe];
end;

pbec = pbec + pbei;              % do the approximation

for delta = deltav               % BERs for si with Eb/N0 = SNRi * k/n
    dck = [-1,1,-(1-2*delta),(1-2*delta)]'/sqrt(SIRi);
    ifer = dck*cos(phi);
    pe = sum(0.125*erfc(sqrt(SNRi*R)*(1+ifer)));
    rpbei = [rpbei; pe];
end;

beta = 1-(1-2*rpbei)*(1-eta);    % BERs for sc, assuming no errors in si


for e=t:-1:0;                    % coding for si
    comb = prod((n-e+1):n)/prod(1:e);
    per = comb*rpbei.^e.*(1-rpbei).^(n-e);
    cpbei=cpbei+per;
end

cpbei = (1 - cpbei)*(dmin/n);    % calculate approx. BER

beta = 1-(1-2*cpbei)*(1-eta);    % BERs for sc with coding,
                                 % assuming no errors in si
ppbec = [];
SNRc = SNRi*R/SIRi;

for id = 1:deltas
    dck = [-1,1,-(1-2*deltav(id)),(1-2*deltav(id))]'/sqrt(SIRc);
    ifer = (dck*cos(phi)).*([1 1 1 1]'*beta(id,:));
    pe = sum(0.125*erfc(sqrt(SNRc)*(1+ifer)));
    ppbec = [ppbec; pe];
end;

for e=t:-1:0;                    % coding for sc
    comb = prod((n-e+1):n)/prod(1:e);
    per = comb*ppbec.^e.*(1-ppbec).^(n-e);
    cpbec=cpbec+per;
end

cpbec = (1 - cpbec)*(dmin/n);    % calculate approx. BER

cpbec = cpbec + cpbei;
```

ub_sml

```
function [beri,berc] = ub_sml (snr,sir,phi,delta,eta,nj,qt)
% function [beri,berc] = ub_sml (snr,sir,phi,delta,eta,nj,qt)
%
% calculation of a ber union bound
%
% nj: wight disribution of the code to check
% qt: vector of quantization threshold levels

s = size(nj);
n = s(2);        % code lenght

beri = 0; berc = 0;

    for j = 2:n;
        if nj(j)
            [bi,bc] = uq_sml(snr,sir,phi,delta,eta,j-1);
            beri = beri + (j-1)/(n-1)*nj(j)*bi;
            berc = berc + (j-1)/(n-1)*nj(j)*bc;
        end;
    end;

berc = berc + beri;
```

uq_sml

```
function [pei,pec] = uq_sml(SNRi,SIRi,phi,delta,eta,j)
% function [pei,pec] = uq_sml(SNRi,SIRi,phi,delta,eta,j)
%
% calculates the error prob of a wight j sequence in case
% of unquantised demodulator outputs
% interfering patterns quantized to (-1:1)

sd=max(size(delta));
pei = []; pec =[];

res = 6*2;
quants = j*res;          % nr. of influencing patterns
infl = (0:quants)'*(2/quants)-1; % influencing patterns

niff=zeros(max(size(delta)),quants+1);
iff = [];
```

```
for i1 = 0:j;                           % interfering patterns
for ix1 = 0:j-i1;
for ix2 = 0:j-ix1-i1;
    comb = prod(j-i1-ix1-ix2+1:j)/(prod(1:i1)*prod(1:ix1)*prod(1:ix2));
    iff = round((i1+(1-delta')*ix1+delta'*ix2)*res)+1;
    for id = 1:sd;
          niff(id,iff(id)) = niff(id,iff(id)) + comb;
    end;
end; end; end;
piff = niff/pow2(2*j);

dck = infl*sqrt(j*SNRi/SIRi);
ifer = dck*cos(phi);                    % interference terms

pf = 0.5*erfc(sqrt(j*SNRi)+ifer);       % error probs


for id = 1:sd;          % calculation of pe for delta-vectors

pee = piff(id,:)*pf;
pei = [pei pee'];

end;

beta = 1-(1-2*pei)*(1-eta);     % the same for sc
SNRc = SNRi/SIRi;
SIRc = 1/SIRi;

dck = infl*sqrt(j*SNRc/SIRc);
ifer = dck*cos(phi);                    % interference terms

for id = 1:sd;          % calculation of pe for delta-vectors
    betai = ones(quants+1,1)*beta(:,id)';

    pf = 0.5*erfc(sqrt(j*SNRc)+ifer.*betai);     % error probs

    pee = piff(id,:)*pf;
    pec = [pec pee'];

end;
```