Performance Analysis of Hybrid Frequency Hopping/Direct Sequence Spread Spectrum Communication Systems with DPSK Modulation for the Indoor Wireless Environment

E. Walther

DELFT UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering
Telecommunication and Traffic-control Systems Group

Title: Performance analysis of Hybrid Frequency Hopping/Direct Sequence Spread Spectrum Communication Systems with DPSK Modulation for the Indoor Wireless Environment

Author: E. Walther

Type: Graduation Thesis
Number of pages: 150 (139 + xi)
Date: August 19, 1991

Graduate Professor: Prof. dr. J.C. Arnbak
Mentor: Prof. dr. R. Prasad
Period: November 1990 - July 1991

Indoor wireless communication has recently drawn the attention of many researchers because of its significant advantages over conventional cabling: allowing mobility of users, time and cost saving, office arrangements and physical moves can be handled with minimal disruption of work, and temporary services can be provided.
The average bit error probability in a Rician fading channel is evaluated for two types of spread spectrum systems using differential phase shift keying modulation: frequency hopping and hybrid frequency hopping / direct sequence. Also the influence of diversity, as a means to enhance the performance of the system has been investigated.

Indexing Terms: Radiocommunication, Spread-spectrum multiple access, Rayleigh and Rician Fading, Indoor Wireless Communication, Personal Communication Networks.

SUMMARY

A theoretical performance analysis, in terms of the average bit error probability, of a Slow Frequency Hopping (SFH) spread spectrum multiple access system in a Rician fading channel is made in this report. Two types of diversity, namely selection diversity and maximal ratio combining are considered. Also the performance of a hybrid Slow Frequency Hopping/Direct Sequence (SFH/DS) spread spectrum multiple access system with selection diversity in a Rician channel is investigated. Mathematical models are developed for the transmitter, channel and receiver.
The influence on the performance of the number of resolvable paths, the order of diversity, the value of the Rician parameter, the channel bit rate, the length of the user codes (the number of chips per bit) and the number of simultaneously transmitting users are investigated.

For both systems, the performance is found to decrease with increasing number of resolvable paths, and to increase with increasing order of diversity. Also an increase in the value of the Rician parameter enhances the system performance.

The results show that a slow frequency hopping system with maximal ratio combining has enhanced performance compared to slow frequency hopping system with selection diversity. Also it is shown that for random hopping patterns, the probability of a hit dominates the average bit error probability.

The multi-user interference in the hybrid SFH/DS system (due to non-zero cross correlations of the user codes) is approximated by Gaussian noise. The results for this system show that the performance decreases with a higher number of users and with a higher bit rate.

The hybrid SFH/DS system is found to be the best suited for an indoor wireless communication system, mainly because of its capacity to allow multiple users transmit over the same frequencies.

LIST OF SYMBOLS

$\alpha(t)$	attenuation factor
β	path gain
β_{max}	maximal path gain
γ_b	bit signal-to-noise ratio
ξ	decision variable
μ	covariance of the current and previous matched filter outputs
μ_0	variance of the current matched filter output
μ_{-1}	variance of the previous matched filter output
σ^2	power received over specular paths
σ_τ	rms delay spread
σ_n^2	variance of the noise samples
τ	path (propagation) delay
$(\Delta f)_c$	coherence bandwidth
$(\Delta t)_c$	coherence time
$a_k(t)$	direct sequence code waveform of user k
$\hat{b}$	bit estimate
$b_k(t)$	data waveform of user k
b_k^{-1}	current data bit of user k
b_k^0	previous data bit of user k
B_d	Doppler spread
C_{k1}	discrete aperiodic correlation
E_b	bit energy
f_N	rms Doppler bandwidth
G_p	process gain
k	number of hops per bit
K	number of (simultaneously transmitting) users
L	number of resolvable paths
m	number of bits that control the frequency synthesizer
m	mean of the matched filter outputs
M	order of diversity
N	number of chips
N_b	number of bits per hop

N_o	average noise power
p_β	pdf of the path gain
p_{γ_b}	pdf of the bit signal-to-noise ratio
P	received signal power
$\overline{P}$	mean received signal power
P_e	average bit error probability
Q	number of frequency slots
r	distance between transmitter and receiver
R	Rice factor
R_c	channel bitrate
R_{k1}	partial correlation function for user k
s	peak value of the constant path
T_b	data bit duration
T_c	chip duration
T_h	hop duration (dwell time)
T_m	multipath or time delay spread
U_m	matched filter output
V	optimum demodulator output
W	transmission bandwidth

LIST OF ABBREVIATIONS

AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Keying
cdf	cumulative distribution function
CDMA	Code Division Multiple Access
DPSK	Differential Phase Shift Keying
DS	Direct Sequence
FDMA	Frequency Division Multiple Access
FFH	Fast Frequency Hopping
FH	Frequency Hopping
FSK	Frequency Shift Keying
ISI	Intersymbol Interference
IWC	Indoor Wireless Communication
kb	kilobit
Kb	kilobyte
LOS	Line Of Sight
MFSK	M-ary Frequency Shift Keying
MRC	Maximal Ratio Combining
PCN	Personal Communications Network
pdf	probability density function
PN	Pseudo Noise
PSK	Phase Shift Keying
RF	Radio Frequency
rms	root mean square
SAW	Surface Acoustic Wave
SD	Selection Diversity
SFH	Slow Frequency Hopping
SFH/DS	Slow Frequency Hopping/Direct Sequence
SSMA	Spread Spectrum Multiple Access
TDMA	Time Division Multiple Access
TH	Time Hopping

TABLE OF CONTENTS

1. INTRODUCTION

The field of wireless communication has recently regained a lot of interest, not only from the users, but also from the service providers, who quickly recognised the economical possibilities of providing wireless services. The need for mobile or wireless communication in our highly developed society is obvious. The demand for current mobile systems such as cellular vehicular telephony, radio paging systems, is growing every year and also for the indoor environment several systems are under development. The applications of indoor wireless communication can be divided into three groups.

The most common application of indoor wireless communication today is residential use. Cordless telephones serve here a very basic need of being more mobile at home and many millions are in use worldwide. These are simple analogue units which access often only a limited number of channels and provide a simple point-to-point radio connection. In many areas where there is a high user density, there will be problems with quality and lack of security in the system. The group of residential users is one of the largest potential users of indoor wireless communication, but it will not be the first to make use of the new wireless services.

The second area of application for indoor wireless communication is the office environment. Wireless communication appeals here, because it can provide flexibility in the placement of terminals and also because of its enormous potential of increase in efficiency. Office arrangements and physical moves can be handled with minimal disruption of work and temporary services can be provided.
The number of terminals for indoor use is growing rapidly in offices and manufacturing floors and it will soon be the application with the highest density of users. Because price is not the only criterion for these applications, the wireless communication services are expected to be used first in the office environment.

The third area of application for indoor wireless communication is public access. The user can make telephone calls in public places such as shopping areas or airports by making use of a wireless telephone that has access to the public switched telephone network via a base station.

There have been several approaches to realise the concept of indoor wireless communication; of which spread spectrum microwave technology is the most promising.

Spread spectrum systems have been developed since the 50's and initial applications have been in military anti-jamming tactical communications, in guidance systems and in experimental anti-multipath systems. A definition of spread spectrum that more or less reflects the characteristics of this technique is as follows [1]:

"*Spread spectrum is a means of transmission in which the signal occupies a bandwidth in excess of the minimum necessary to send the information; the band spread is accomplished by means of a code which is independent of the data, and a synchronised reception with the code at the receiver is used for despreading and subsequent data recovery.*"

In fact, the advantage of spread spectrum is the spread of the transmitted power over a wide frequency band so that the power per unit bandwidth (watts per Hertz) is very small. At the receiver the signal is compressed into its original narrow band, while leaving the power of other (interfering) signals scattered over the wide transmission band.

The advantages of spread spectrum over other possible techniques to build an indoor wireless communication system are:

- **Resistance to multi-path fading**

  When the transmission bandwidth of a (spread spectrum) signal is much larger than the coherence bandwidth of the channel, the signal splits into several independently fading signals. This is called the inherent diversity property of spread spectrum. These independently fading signals can be used to combat the multi-path fading.
- **Code Division Multiple Access (CDMA) capability**

  The signal transmitted to a given user is "tagged" with a spread spectrum code pattern that only that user's receiver recognises. The receiver knows in advance how the transmitter will spread the spectrum, acquires the signal and continues to track the transmitted pattern.
- **Low spectral power density per user**

  By increasing the transmission bandwidth of the signal, the signal power per Hertz becomes very low. Because of this low spectral power density, the signals are unlikely to interfere with other signals intended for business and consumer use, even ones transmitted on the same frequencies. This special feature of spread spectrum communications opens up crowded frequency spectra to expanded use.

  Tests are now undertaken [2] to show that spread spectrum users can share a frequency band with conventional radio users, without one group interfering with the other, and increase the efficiency with which that band is used.

- **Resistance to interception**

  Only the user's transmitter and its receiver know the spread spectrum code pattern which is used to spread the signal. Spread spectrum transmission is not secure, but it is private. A casual listener would not be able to intercept messages. However, encryption must be used for true security.

- **Resistance to intentional and unintentional interference**

  One feature of spread spectrum communications that is especially attractive to military users is its resistance to "jamming", that is, the intentional interference of the transmitted signal. Again because of the large transmission bandwidth, it is almost impossible to jam all of the signal for most of the time.

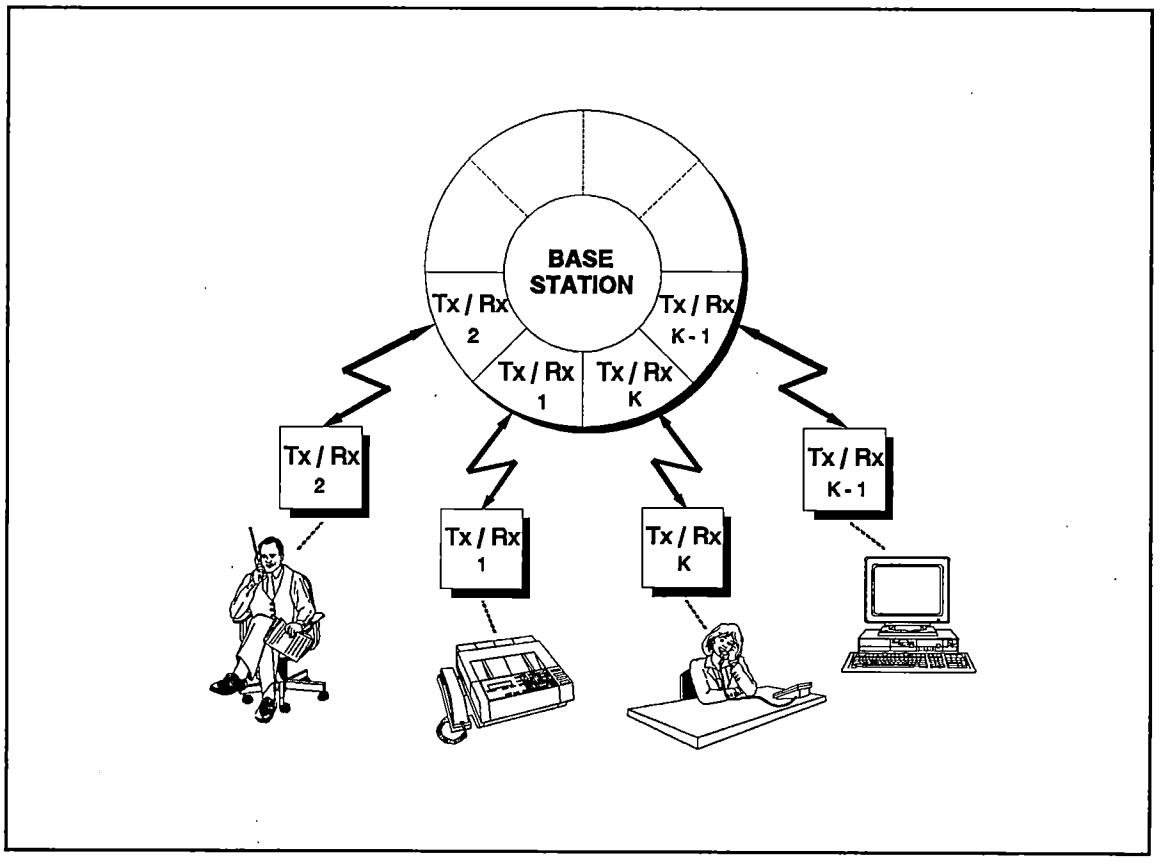

Figure 1.1 Star connected indoor wireless local area network with CDMA

The system we shall use for our performance analysis is a star connected CDMA network with K users, as shown in figure 1.1. The base station consists of a bank of spread spectrum transmitters/receivers, one for each active user. We assume that each user has a unique spread spectrum code sequence and spread spectrum can provide the multiple access capability

because each user has a unique code and the codes have low cross-correlations (are orthogonal).

Spread spectrum cellular systems would not employ spatial frequency reuse (separate bands for adjacent cells), a crucial element in both analog and digital cellular telephony [2]. Instead, frequency diversity (separate bands for transmitting and receiving) would be applied.

Because of the presence of multipath fading on the radio path, which results in a loss of phase coherence in the received signal, it is hard to use coherent detection of the signal. This problem also arises when there are high levels of interference in the channel due to jamming or the presence of a large number of spread spectrum signals in the same frequency band. We avoid this problem by using differentially coherent detection at the receiver. In this report we will use Differential Phase Shift Keying (DPSK) modulation as the modulation technique (for more information see Appendix A).

The object of this report is to investigate the performance of systems that use frequency hopping techniques with DPSK modulation as a means to provide multiple access for use in the indoor wireless environment. In chapter 2 the different types of multiple access techniques and spread spectrum techniques are discussed. The propagation characteristics of the indoor environment are discussed in chapter 3. In chapter 4, a system using fast frequency hopping with DPSK modulation is described. In chapter 5, 6 and 7 the performance analysis of a slow frequency hopping system with DPSK modulation is performed for two types of diversity: selection diversity and maximal ratio combining. In chapter 8 and 9 the performance analysis of a hybrid SFH/DS system is given for selection diversity. And finally conclusions are drawn and recommendations for further research given in chapter 10.

2. MULTIPLE ACCESS TECHNIQUES

The two most common multiple access techniques are Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA). In FDMA, all users transmit simultaneously, but use disjoint frequency bands. In TDMA, all users occupy the same RF bandwidth, but transmit sequentially in time.

When users are allowed to transmit simultaneously in time and occupy the same RF bandwidth as well, some other means of separating the signals at the receiver must be found. CDMA, also called Spread Spectrum Multiple Access (SSMA), provides this necessary capability. The three multiple access techniques are shown in figure 2.1.

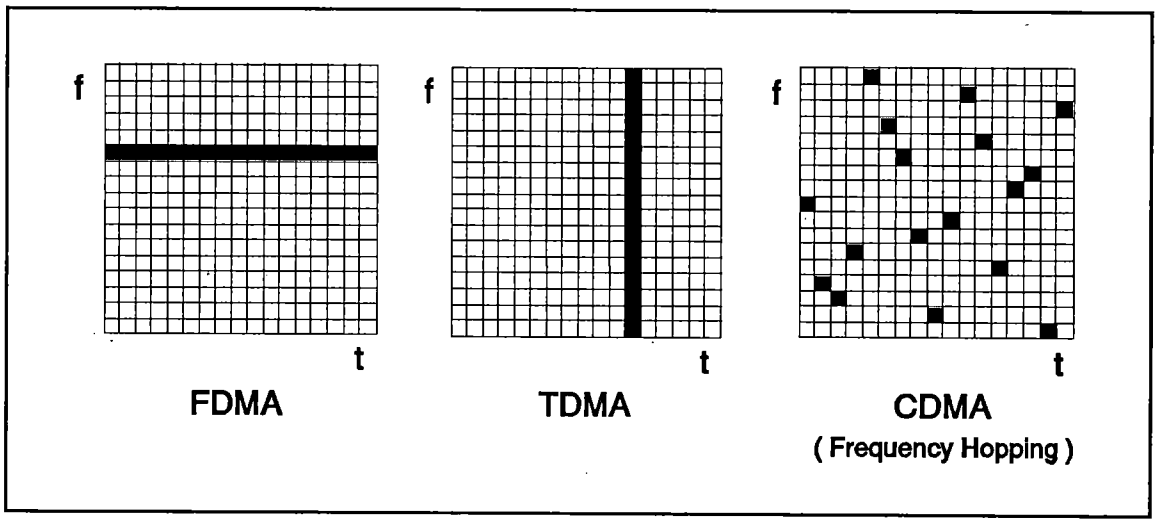

Figure 2.1 Multiple Access Techniques

It is interesting to compare the capacity of a CDMA system with FDMA or TDMA. In a perfectly linear, perfectly synchronous system, the number of orthogonal users for all three systems is the same, since this number only depends upon the dimension of the total signal space. The differences between the three multiple access techniques become clear when various real-world constraints are forced upon the ideal situation described above.

The question remains whether TDMA or CDMA should be used for Personal Communication Networks (PCN) and indoor wireless communications. Many communications companies today are choosing TDMA for their systems, and the companies who are building a nationwide wireless

PCN in Great Britain have chosen TDMA, a relatively mature technology, in order to achieve their goals in the shortest possible time.

CDMA offers several advantages over the other multiple access techniques, in addition to its potentially greater capacity. CDMA is far less susceptible to multi-path fading, an important factor in the indoor environment. CDMA is interference limited, that is, the number of subscribers that can use the same spectrum is determined by the total interference power that all simultaneous user generate in the receiver. Total interference power in turn is affected by such factors as the processing gain, direction relationships, and the proximity of transmitters and receivers.
Another attractive feature of CDMA is that it does not require the overall network synchronisation that TDMA requires, but requires only link synchronisation. It is also easier to add additional users to the system, but the main reason for using CDMA in our system, is the need to be able to reject external interference such as multipath fading or intentional jamming.

There are several ways to implement CDMA: Direct Sequence (DS), Frequency Hopping (FH) and Time Hopping (TH). Also hybrid forms of these methods can be used. In this chapter we describe the two most important techniques of CDMA: direct sequence and frequency hopping.

2.1 Direct Sequence CDMA

Direct sequence spreads the spectrum by modulating the original baseband signal with a very wide-baseband digital signal. The wideband modulating signal's amplitude changes continually between two states, high and low. The speed of alternation is called the *chip* rate, because direct sequence spread spectrum chops up, or chips, bits. With direct sequence CDMA, a multipath signal having a delay larger than 1 chip will be uncorrelated and appears only as a weak additional interference at the receiver.
The sequence of highs and lows is pseudorandom; that is, at equally spaced time intervals, control logic decides whether the wideband modulating signal should be high or low. Thus the sequence is not truly random, a logic chip alternates the amplitudes so that they seem to be random over a long enough period, the numbers of highs and lows are about equal. The pattern, the receiver can recognize, is approximately orthogonal (has a low cross correlation) with the codes of the other users.

The processing gain G_p, a key parameter in spread spectrum systems, is the ratio of the bandwidth of the spread signal to the bandwidth of the unspread signal. In the case of DS we can write for the processing gain:

$$G_p = \frac{T_b}{T_c} \tag{2.1}$$

where T_b is the bit duration and T_c the chip duration.

Direct sequence systems have several advantages and disadvantages over the other CDMA techniques:

Advantages:

Because of the (high-rate) chopping of bits and consequently large transmission bandwidth:

- **Best noise and anti-jam performance**
- **Most difficult to detect**
- **Best discrimination against multipath fading**

Disadvantages:

- **Long acquisition time**
- **Fast code generator needed**

  A limitation in direct sequence spreading is the speed at which circuitry can alternate between high and low levels. Today, the maximum chip rate with silicon CMOS integrated circuits is about 50 million chips per second, which is considered adequate for current applications and those planned for the near future.
- **Near-Far problem**

  Since the N users are geographically separated, a receiver trying to detect the signal of the *k*th transmitter might be much closer physically to the *ith* transmitter rather than to the *kth* transmitter. So, if each user transmits with equal power, the signal from the *ith* transmitter will arrive at the receiver with a larger power than that of the *kth* signal. This problem is often so severe that direct sequence cannot be used. A solution to this problem is to use adaptive power control to reduce power output as a terminal approaches the receiver.

2.2 Frequency Hopping CDMA

In the case of Frequency Shift Keying (FSK) modulation we can describe frequency hopping more accurately as *multiple-frequency, code-selected, frequency shift keying*. It is nothing more than FSK except that the set of frequency choices is greatly expanded [4]. In the case of differential phase shift keying, we have to compare the received bits with a reference bit, which has consequences in designing such a system. A frequency hopping system is much less susceptible to the near-far problem because it is an avoidance system rather than an averaging system. In a frequency hopping signal, the frequency is constant during each hopping interval, but changes from hop to hop. When each user is given a different hopping pattern, and if all hopping patterns are orthogonal, the near-far problem will be solved. The frequency spectrum of a frequency hopping system is shown in figure 2.2.

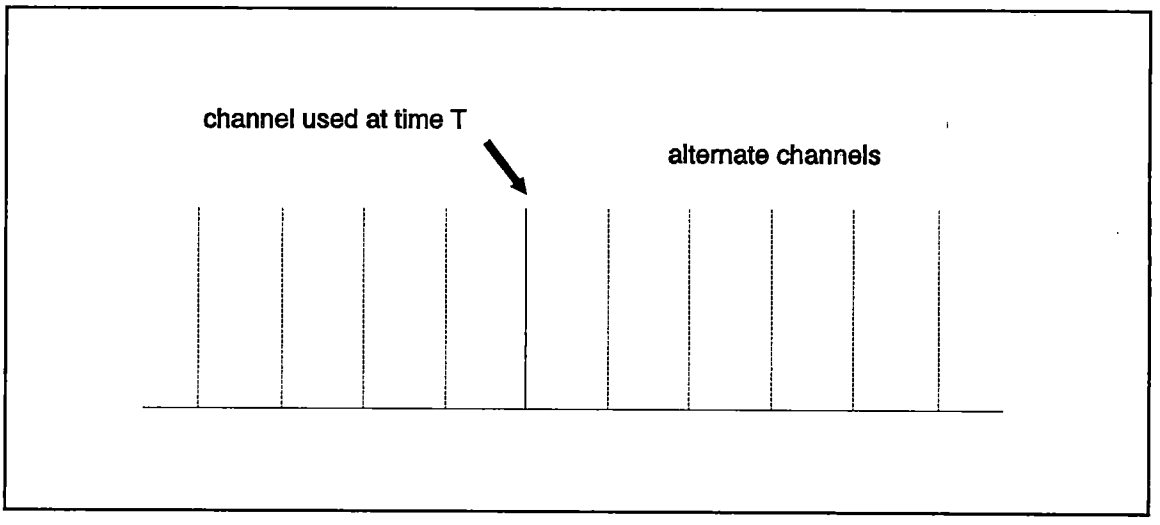

Figure 2.2 Frequency Hopping spectrum

Frequency hopping systems are classified as fast frequency hopping (FFH) and slow frequency hopping (SFH) because there is a considerable difference in performance of these two types of systems. A fast frequency hopping system transmits a bit during multiple hops and a slow frequency hopping system transmits multiple bits per hop. There is also the intermediate case in which the hop rate is equal to the bitrate.

The number of frequencies over which the signal may hop is usually a power of 2, although not all these frequencies are necessarily used in a given system.

The block diagram of a typical frequency hopping transmitter is shown in figure 2.3. Frequency hopping is realised by a digital frequency synthesizer, which is driven by a PN code generator. The frequency synthesizer is controlled by m binary digits and produces one of $Q = 2^m$ frequencies for each combination of these digits. One of the m controlling digits comes from the message and the other m-1 digits come from the PN code generator.

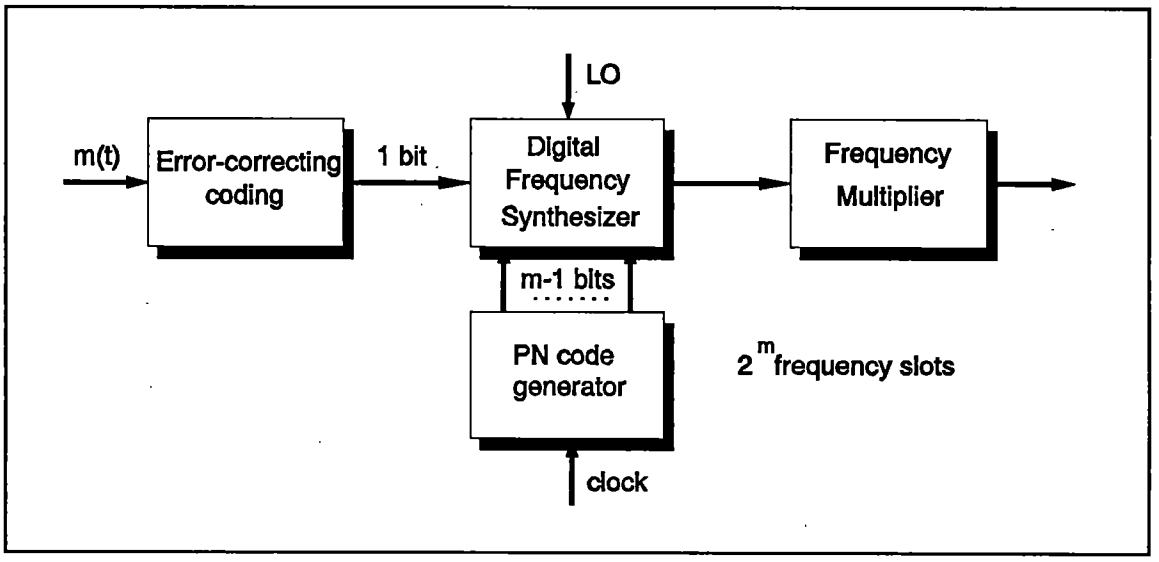

Figure 2.3 Standard Frequency Hopping Transmitter

When the digit from the message produces the smallest frequency change, it produces a binary FSK signal. The m-1 digits from the PN code generator then hop this FSK signal over the range of possible frequencies.

The message will normally have error-correction coding applied to it. If there is interference during a hop, all the bits in that particular hop may be destroyed, and so, it is necessary to be able to reconstruct the message by using error-correction techniques [5].

There is also a frequency multiplier at the output of the system; its purpose is to increase the bandwidth and thereby increase the processing gain that is available. It also changes the shape of the spectrum.

A measure of the processing gain for frequency hopping systems is

$$G_p \triangleq \frac{W}{R_c} \tag{2.2}$$

If the channels are not contiguous, a better measure of the processing gain is

$$G_p = \text{the number of available frequency choices} = Q \tag{2.3}$$

which also holds for non-contiguous channels.

The maximum processing gain follows directly from Shannon's information-rate theorem and gives the gain produced by the spreading of the signal's bandwidth over the spectrum. For example, a frequency hopping system containing 1000 frequency choices could theoretically have 30 dB available processing gain.

The reception of a frequency hopping signal is usually noncoherent. Coherent reception is possible, but it is more difficult to do, because it places some constraints on the nature of the transmitted signal and the transmission medium.

One of the relative advantages of frequency hopping is that the PN code generator can run at a considerable slower rate than in a direct sequence system. For example, when T_c equals T_b/k, the PN code generator clock rate is simply $k\times(m-1)$ times the message bit rate, and is independent of the signal bandwidth. The maximum rate at which the code generator can run in a frequency hopping transmitter or receiver is generally determined by the switching speed in the frequency synthesizer, rather than by the code generator itself.
When several frequency hopping signals occupy a common RF channel, interference results. Events of this type are called *hits*, and become more and more of a problem as the number of users hopping over a fixed bandwidth increases.

Frequency hopping systems have several advantages and disadvantages over the other CDMA techniques:

Advantages:

- **Greatest amount of spreading**
- **Programmable to avoid portions of the spectrum**
- **Short acquisition time**
- **Less affected by near-far problem**

Disadvantages:

- **Complex frequency synthesizer**

  The limitation of frequency hopping is the speed at which the frequency synthesizer can change frequency without generating too much noise. If a synthesizer tries to hop too fast, it cannot turn off completely between hops.

- **Error correction required**

  Often error correction coding is used in frequency hopping systems, the reason for this is that the error correcting capability of the code can restore the messages that are damaged by a hit. Error correction is used primarily in slow frequency hopping systems where there are several message bits per hop.

2.3 Data Redundancy

When we send three or more chips for each data bit, the performance of the system will be greatly enhanced [4]. In such a system it is possible to make bit decisions based on more than one chip.

For example, when the receiver's bit decision is based on two out of three being correct, we will pay for the increased performance with a threefold increase in transmission rate. In frequency hopping, the frequency synthesizer must hop three times as fast. Given that the synthesizer can do so, it is almost always advantageous to make the three-for-one trade in speed to reduce errors 3000 to 1 [4].

The possibility of increasing redundancy (and the hop rate) to improve bit error rate depends on the system parameters. It is obvious that, when we send more chips for a bit, the bit error rate will become lower. The hopping rate and the transmission bandwidth increase in direct proportion. If the allocated bandwidth or the frequency synthesizer is limited, then some trade-off must be made.

3. PROPAGATION ASPECTS OF INDOOR RADIO

In order to design an indoor wireless communication system we need to have some knowledge of the indoor radio propagation characteristics. This should include the path loss and the multipath interference, which will be described below.

3.1 Path Loss

In mobile communications the mean received signal power $\overline{P}$ statistically decreases as a function of distance r according to a general attenuation law:

$$\overline{P} = r^{-\alpha} \tag{3.1}$$

In free space $\alpha = 2$, so the mean received power obeys an inverse-square law. When the transmitter and the receiver are in the same corridor in an office building, $\alpha < 2$ is found; a value of 1.8 has been reported in [6]. This gain over the free space environment is possibly due to waveguiding effects in the corridor. Values of about 3 have been measured in rooms which are situated off the corridor containing the transmitter; in rooms off a corridor perpendicular to the transmitter's corridor, values of 4 up to 6 have been measured [7]. Very high values of α ($\alpha = 6$) correspond to buildings with metallised partitions.

3.2 Multipath Interference

Multipath interference is caused by multiple reflections of the transmitted signal from the building structure and surrounding inventory. The resulting received waveform is a sum of time and frequency shifted versions of the original transmission, and, depending on parameters of the signal and the channel, the received signal may be severely distorted.
An example of a faded signal is shown in figure 3.1.

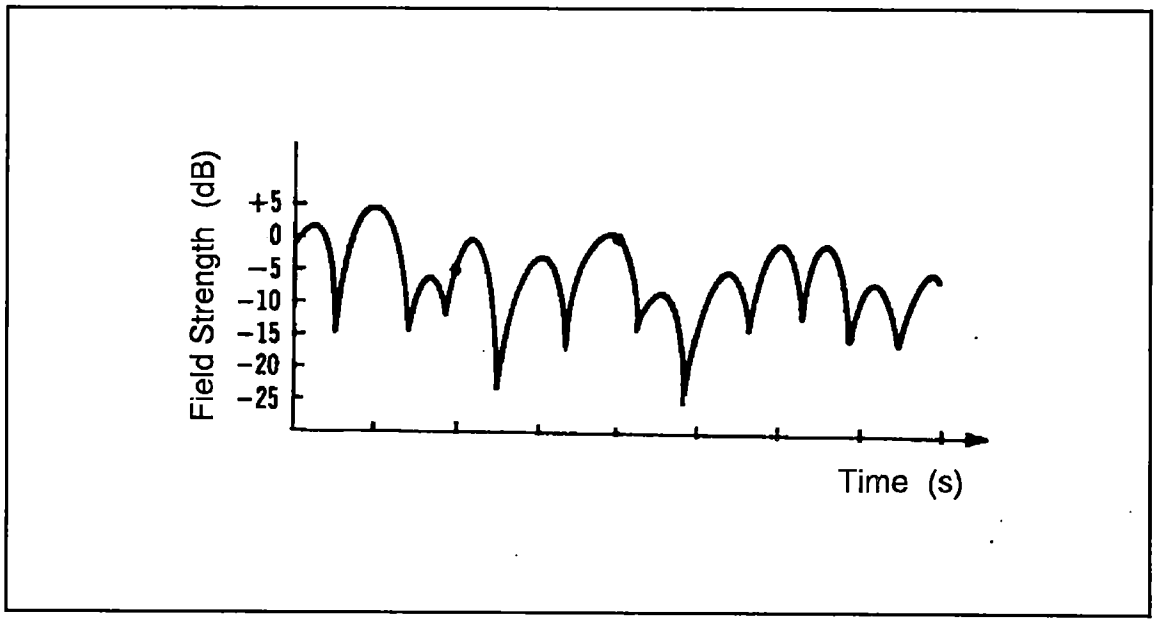

Figure 3.1 Influence of multipath fading on a transmitted signal

The equivalent low-pass channel that contains discrete multipath components is described by the time-variant impulse response

$$c(\tau;t) = \sum_n \alpha_n(t) \cdot e^{-j2\pi f_c \tau_n(t)} \cdot \delta[\tau - \tau_n(t)] \qquad (3.2)$$

Here, $\alpha_n(t)$ is the attenuation factor for the received signal on the n*th* path and $\tau_n(t)$ is the propagation delay for the n*th* path.

In most radio transmission media the attenuation and phase shift of the channel associated with path delay τ_1 is uncorrelated with the attenuation and phase shift associated with path delay τ_2. This is usually called uncorrelated scattering.

3.2.1 Coherence Bandwidth

The range of values of τ over which the average power output of the channel as a function of the time delay τ ($\phi_c(\tau)$) is essentially nonzero is called the *multipath spread* or time delay spread of the channel and is denoted by T_m.

The rms delay spread is defined as [8]:

$$\sigma_\tau \triangleq \sqrt{\overline{\tau^2} - (\overline{\tau})^2} \quad \text{and} \quad \overline{\tau^n} \triangleq \frac{\sum_l \tau_l^n \beta_l^2}{\sum_l \beta_l^2}, \; n = 1,2 \tag{3.3}$$

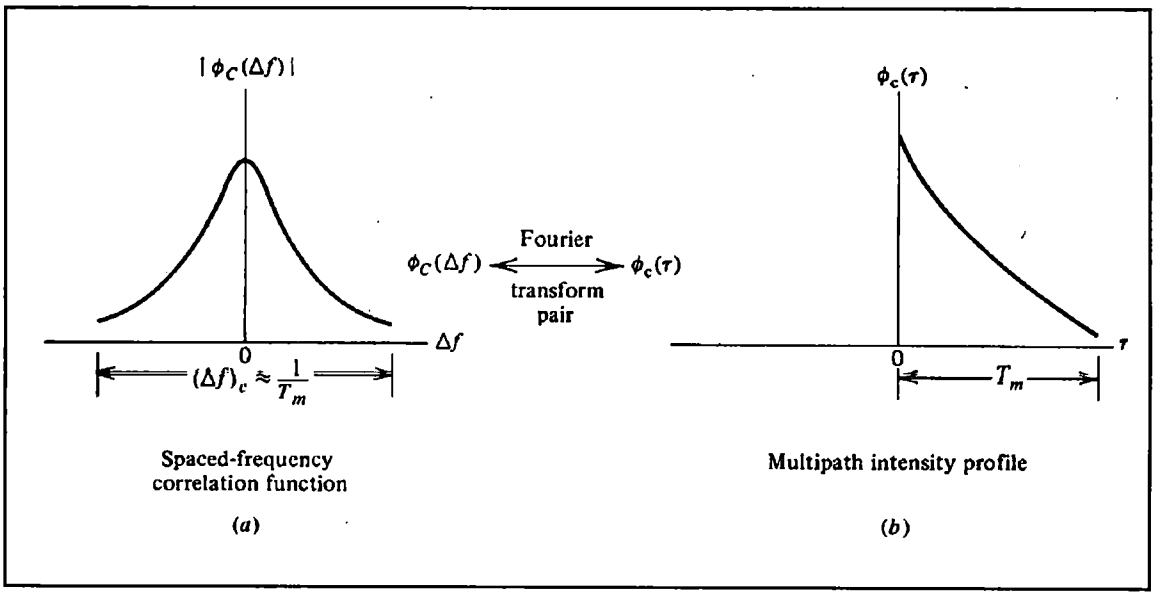

Figure 3.2 Relationship between $\phi_C(\Delta f)$ and $\phi_c(\tau)$

A measure of the frequency coherence of the channel is given by $\phi_C(\Delta f)$, which is an autocorrelation function in the frequency variable. As a result of the Fourier transform relationship between $\phi_C(\Delta f)$ and $\phi_c(\tau)$, the reciprocal of the multipath spread is a measure of the *coherence bandwidth* of the channel. That is

$$(\Delta f)_c \approx \frac{1}{T_m} \tag{3.4}$$

where $(\Delta f)_c$ denotes the coherence bandwidth. Thus two sinusoids with frequency separation greater than $(\Delta f)_c$ are affected differently by the channel. When a signal is transmitted through the channel, and if $(\Delta f)_c$ is small in comparison to the bandwidth of the transmitted signal, the channel is said to be *frequency selective*. In this case the signal is subjected to different gains and phase shifts across the band and is severely distorted by the channel. On the other hand, if $(\Delta f)_c$ is large in comparison to the bandwidth of the transmitted signal the channel is said to

be *frequency nonselective*. All the frequency components in the transmitted signal undergo the same attenuation and phase shift in transmission through the channel.

Summarizing: $W \gg (\Delta f)_c \Rightarrow$ frequency selective channel
$W \ll (\Delta f)_c \Rightarrow$ frequency nonselective channel

There are several papers where measurements of the time delay spread have been reported, in small and large office buildings and factories.
Maximum delay spreads of 300 ns have been measured by Saleh/Valenzuela [8] and maximum rms delay spreads of 270 ns have been reported by Devasirvatham [9] and others for indoor-indoor communication in the frequency bands of 850 MHz - 4.0 GHz. For outdoor-indoor communication values of 420 ns have been reported.

3.2.2 Coherence Time

The signal intensity as a function of the doppler frequency f is given by the function $S_C(f)$, also called the Doppler power spectrum of the channel.

The range of values of f over which $S_C(f)$ is essentially nonzero is called the *Doppler spread* B_d of the channel. Since $S_C(f)$ is related to $\phi_C(\Delta t)$ by the Fourier transform, the reciprocal of B_d is a measure of the *coherence time* of the channel. That is,

$$(\Delta t)_c \approx \frac{1}{B_d} \tag{3.5}$$

where $(\Delta t)_c$ denotes the coherence time.

Clearly a slow changing channel has a large coherence time or, equivalently, a small Doppler spread.
The rapidity of the fading in a frequency nonselective channel is determined either from the correlation function $\phi_C(\Delta t)$ or from the Doppler power spectrum $S_C(f)$. Alternatively, either of the channel parameters $(\Delta t)_c$ or B_d can be used to characterize the rapidity of the fading.
When we select T smaller than the coherence time of the channel, the channel attenuation and phase shift are essentially fixed for the duration of at least one signalling interval. When this condition holds, we call the channel a *slowly fading* channel.

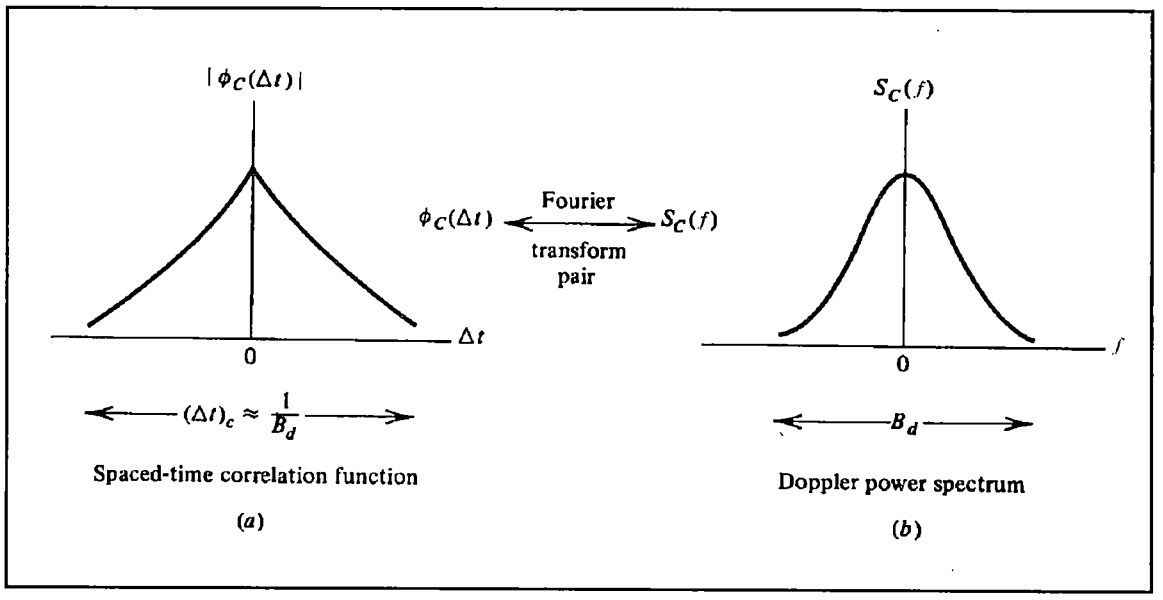

Figure 3.3 Relationship between $\phi_C(\Delta t)$ and $S_C(f)$

The maximum doppler shift can be estimated by

$$B_d \approx f_d = \frac{2vf_c}{c} \tag{3.6}$$

The rms value of B_d is called f_N or the rms Doppler bandwidth, given by

$$f_N = \sqrt{\frac{\int f^2 S_C(f)\, df}{\int S_C(f)\, df}} \tag{3.7}$$

For example, a person walking at 5 km/h (1.389 m/s) will produce a Doppler spread of 8.33 Hz with f_c=900 Mhz and a Doppler spread of 37 Hz with f_c=4 Ghz. Based on these values, we can conclude that for practical values, the fading will be slow. This is also shown by the following example. Even with f_c=100 Ghz, and a data/signalling rate of 16 kbit/s, the speed of the person has to be higher than 24 m/s (86.4 km/h) in order for the fading to be fast.

Summarizing: $T \gg (\Delta t)_c \Rightarrow$ fast fading channel ($B_d \gg 1/T$)
$T \ll (\Delta t)_c \Rightarrow$ slow fading channel

The values of the Doppler spread B_d tend to be quite low; in the order of several (tens of) Hertz [10]. Doppler spread measurements (narrowband) reported by Howard and Pahlavan [11] for the indoor radio environment show a maximum Doppler spread of 6.1 Hz, and a maximum rms Doppler bandwidth of 0.9 Hz. Because 1/T is in many cases significantly larger than the measured values of B_d, the results imply slow fading.

3.3 Channel Models

The indoor wireless channel is considered to be quasi-static because of the large number of non-moving scatterers, or very slowly time varying (related to people's movements).
It is apparent from the last sections that the time-variant channel characterised by the transfer function distorts the transmitted signal.
Additional distortion is caused by the time variations in the transfer function. This type of distortion is seen as variation in the received signal strength and has been termed *fading*.
It should be emphasized that the frequency selectivity and fading are viewed as two *different* types of distortion. The former depends on the coherence bandwidth of the channel (on the multipath spread) relative to the transmitted signal bandwidth W. The latter depends on the time variations of the channel, which are characterised by the coherence time (or Doppler spread).

The effect of the channel on the transmitted signal is a function of the choice of signal bandwidth and signal duration.
For example, if we select the signalling interval T to satisfy the condition $T \gg T_m$, the channel introduces a negligible amount of intersymbol interference. So in this report we shall assume the data bit duration to be larger than the rms delay spread.
Furthermore, when W=1/T, the conditions that the channel is frequency nonselective and slowly fading imply that the product of T_m and B_d must satisfy the condition $T_m B_d < 1$.
The product $T_m B_d$ is called the spread factor of the channel. If $T_m B_d < 1$, the channel is said to be *underspread*; otherwise it is *overspread*.

In the indoor environment the transmitted signal will be reflected from different walls and therefore arrive at different times with different phases at the receiver. These reflections are grouped in clusters, each with a difference of less than T_c between the first and last arrival time of the rays in the cluster. We assume these resolvable clusters of paths to be independent and Rician fading [6].

The spatial distribution of the signal strength is also Rician distributed, given a static environment. The Rician distribution is characterised by a parameter R, which is the ratio of the peak power and the power received over specular paths.
Fading was found to be bursty [6], during tens of seconds, in an office environment with separate rooms. This bursty fading was caused by moving personnel. In another office, which had an open plan construction, the fading was more continuous.

CHANNEL MODELS	$T \ll (\Delta t)_c$	$T \gg (\Delta t)_c$	
$W \ll (\Delta f)_c$	• non-dispersive • flat-flat fading	• frequency-dispersive • frequency-flat fading	frequency non-selective channel
$W \gg (\Delta f)_c$	• time-dispersive • time-flat fading	• doubly (time and frequency) dispersive	frequency selective channel .
	time-nonselective or fast fading channel	time-selective or slow fading channel	

Table 3.1 Terminology

Several other names for the multipath interference found in literature were: flat fading, temporal fading, small scale (=fast) fading, (quasi) wide sense stationary fading, continuous / bursty fading. These names can all be categorised in table 3.1, but depend more or less on the individual propagation assumptions of the authors.

4. FAST FREQUENCY HOPPING WITH DPSK

The fast frequency hopping system described in this chapter is the only one suitable for DPSK modulation found in literature. It has been described extensively in several papers [12-18] during the period 1977-1982, but there have not been any new developments reported since then. The system can be used for true frequency hopping (one bit per hop) and for fast frequency hopping.

4.1 System Description

It is possible to use a tapped delay line to build a matched filter receiver for a frequency hopped signal. To illustrate this, we consider the specific frequency-hopped signal shown in figure 4.1.

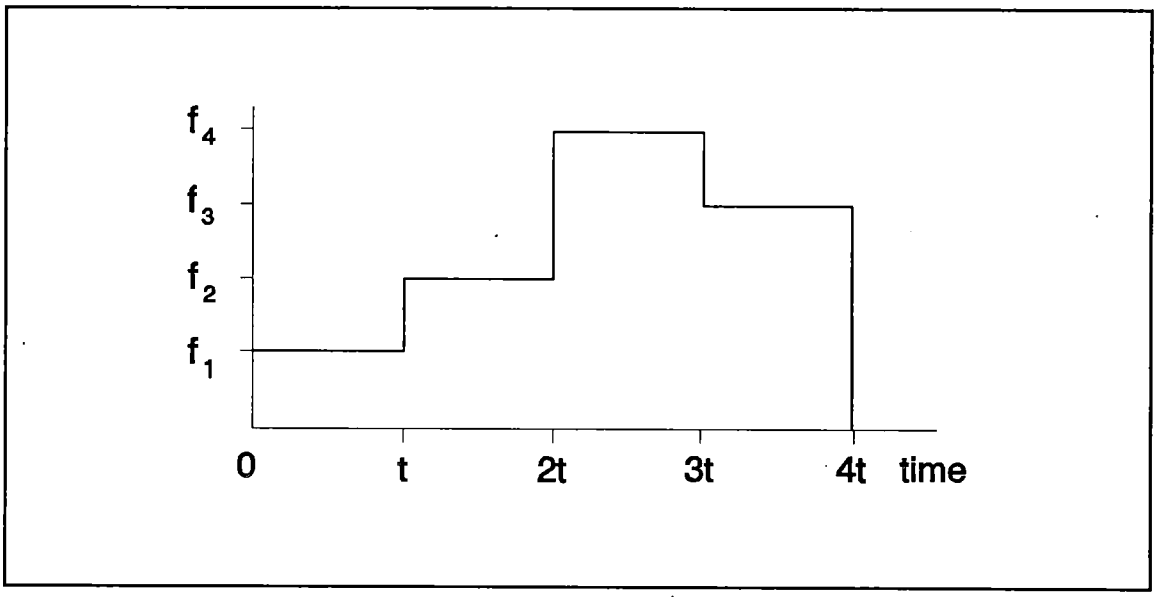

Figure 4.1 A frequency hopped signal for Q=4

The tapped delay line in the receiver has a bandpass filter connected to each tap and each bandpass filter is associated with a particular frequency component. This is shown in figure 4.2. Note that the sequence in which the frequencies appear in the matched filter is opposite to the sequence in which the frequencies arise in the incoming signal. The first part of the signal must

be delayed the most.

One of the difficulties with the matched filter shown in figure 4.2 is that the tapped delay line must have both a large bandwidth and a large time delay. For this reason, it may not be possible to build a matched filter using this particular configuration.

The maximum processing gain that can be achieved in matched filters of this type is determined by the available time-bandwidth product of the delay devices. Surface Acoustic Wave devices would be used when the time delay required is small and the bandwidth large. Processing gains in excess of 500 to 2000 are not readily obtained with any practical delay devices [5].

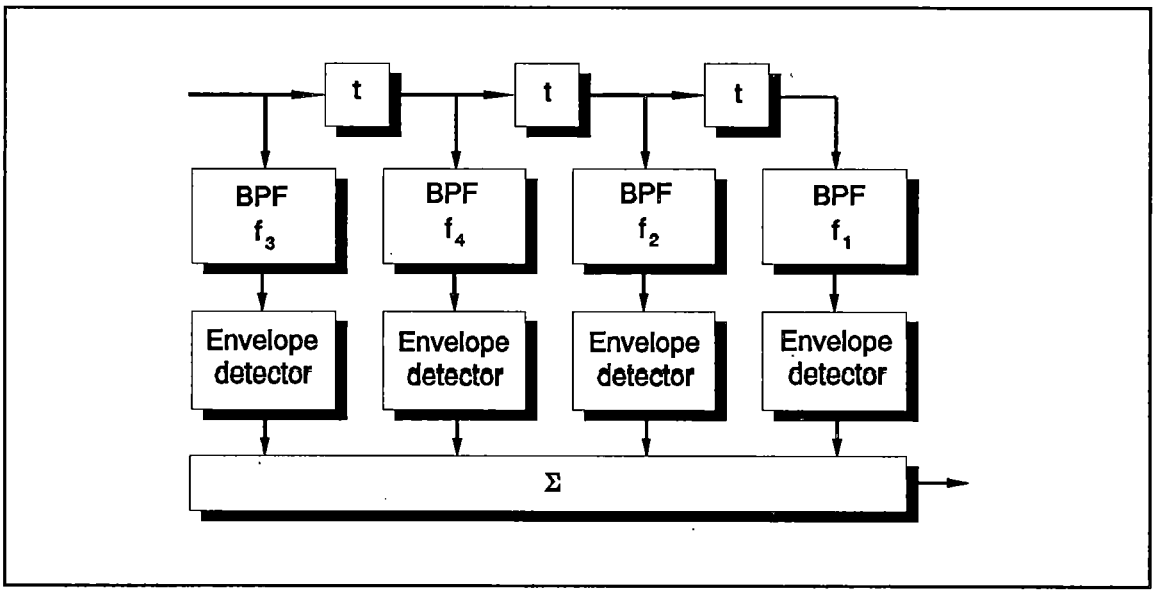

Figure 4.2 Tapped-delay-line matched filter for the Q=4 frequency hopped signal

4.2 FFH/DPSK Transmitter

A schematic diagram of a fast frequency hopping DPSK transmitter is given in figure 4.3. There are two parts in the modulation process in the transmitter: addressing and encoding. Addressing is achieved by the MFSK generator, which repeats with period T a specific sequence of N (N is an integer power of 2) different tones or chips, each of duration t (t = T/N).

Signal information is encoded onto the MFSK address sequence in the form of binary differential phase shift keying (DPSK). If a binary 0 is to be transmitted in the *lth* chip of the address

sequence, the phase of that chip is changed by π radians relative the phase in the *lth* chip of the *previous* sequence. For a binary 1, no phase change takes place.

In order to increase the resistance of this type of modulation to interference, the words are selected from a set of N orthogonal words: the user's data is coded using a user-specific K×N matrix followed by a N×N Hadamard matrix with ±1 elements. The block encoder accepts K = ^{2}log N bits of data and outputs an N bit word, so its code rate is r = ^{2}logN/N. By making use of the Hadamard matrix all resulting code words are orthogonal.

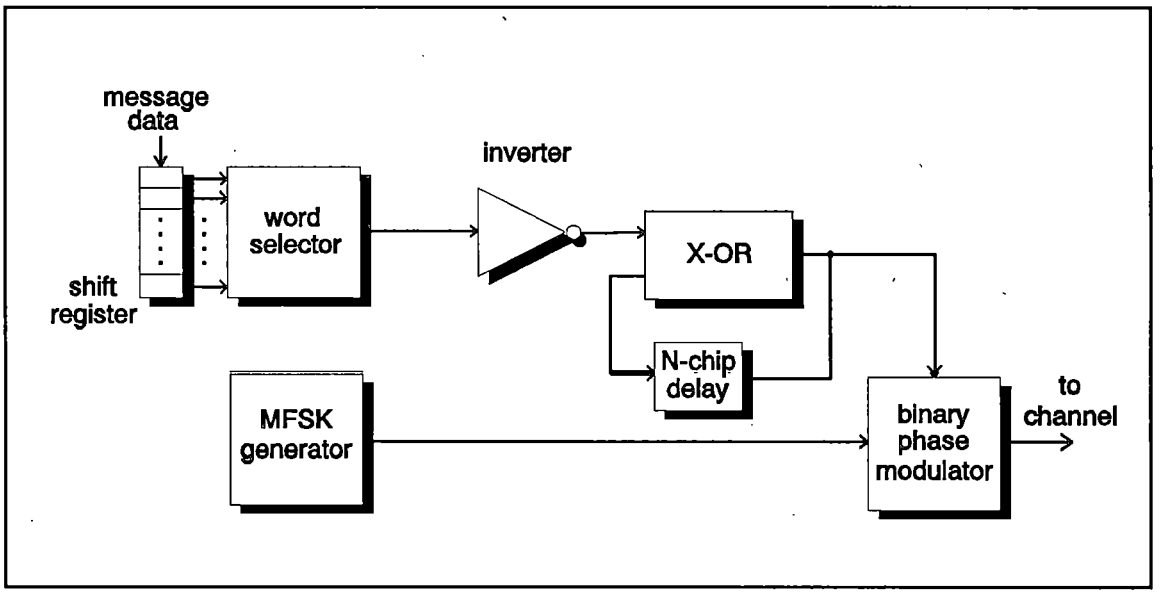

Figure 4.3 FFH/DPSK Transmitter

4.3 FFH/DPSK Receiver

At the receiver, shown in figure 4.4, the array of t-second delay lines and bandpass filters selects the desired address waveform out of the incoming signal. Each filter is matched to a rectangular chip of duration t, and therefore has a noise bandwidth of 1/t. All N chips pass through the filters at the same time and their phases (relative to the previous word) are detected using T-second delay lines and product detectors. After low-pass filtering to remove the second-harmonic product terms, the detector outputs are processed in a combiner circuit.
By sampling the combiner outputs at the appropriate instant, one can determine which word was transmitted.

The performance of this system in the Rayleigh fading environment was analyzed in [14].
Digitized speech (Rc = 31.25 kb/s) was transmitted over the mobile radio channel (bandwidth W = 20 Mhz) using orthogonal coding with rate r = 5/32, which means sending 5 bits of data with 32 chips. The maximum number of users of interference-free operation is then K = r×(W/R) = 100.
The frame-asynchronous performance of the differentially coherent detector in a two-way radio system for 100 users was found to be interference limited at $P_e \simeq 10^{-1}$. The system could accommodate only 26 users if a bit error rate of 10^{-3} was desired, which corresponded to 770 kHz per user.

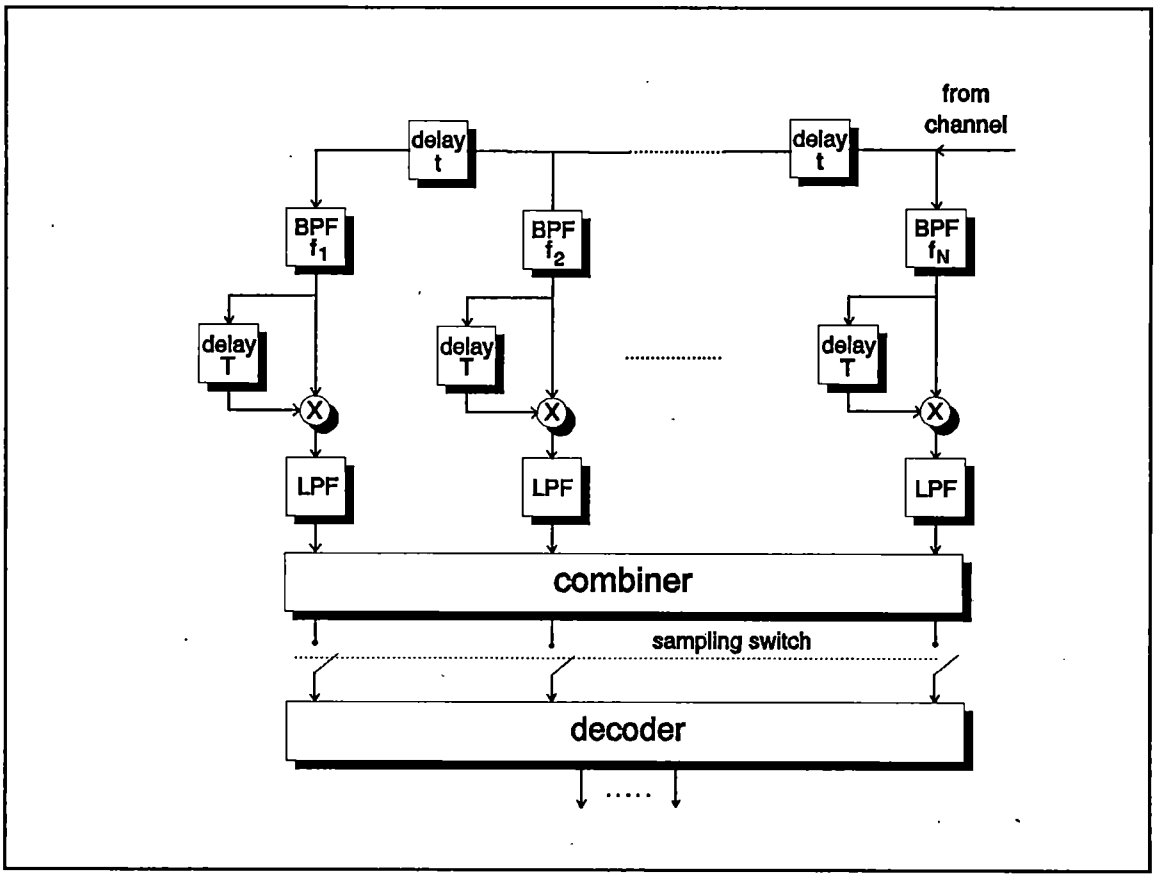

Figure 4.4 FFH/DPSK Receiver

When we look at this system more closely, we can see that it lacks the flexibility needed to build a multiple access system. Because of the fixed bandpass filters in the receiver each receiver has a build-in spread spectrum code in hardware. Therefore we will not investigate the performance of a fast frequency hopping system in this report.

5. SLOW FREQUENCY HOPPING WITH DPSK MODULATION

In a slow frequency hopping system with DPSK modulation, a sequence of bits is transmitted during a hop and because phase coherency is not maintained from hop to hop, a reference bit is used as a startup bit for *each* hop. Except for the first chip, the demodulation reference is derived directly from the received signal [19-21].

A set of N phase changes is sent by transmitting a reference tone simultaneously with N other tones. The relationship between the phases of any two adjacent tones corresponds to the phase change required for the transmission of a DPSK symbol.

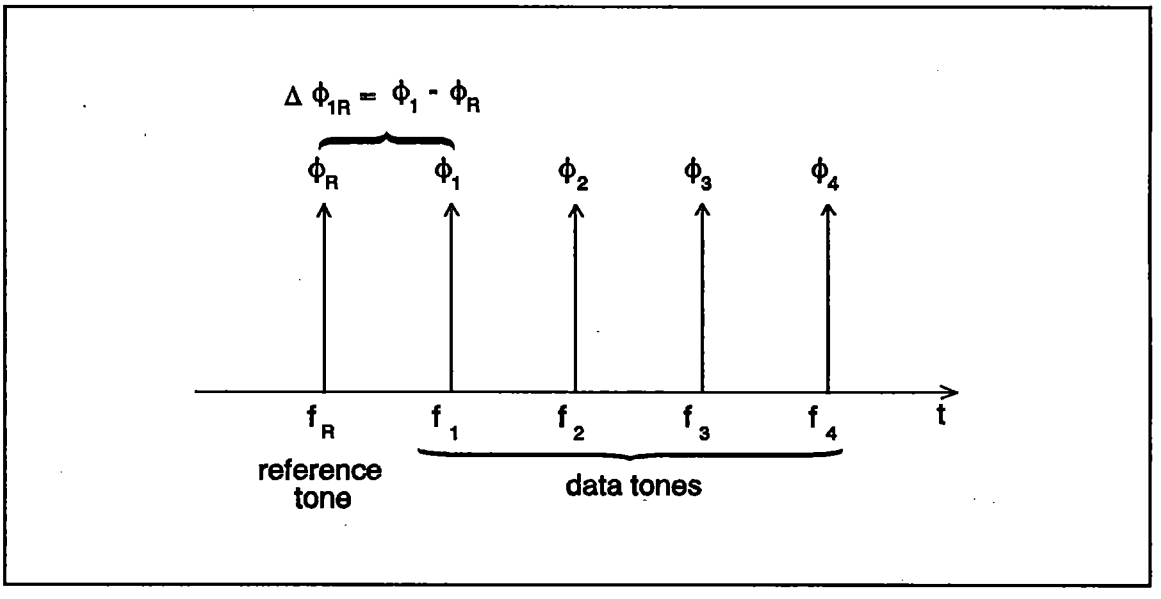

Figure 5.1 Five-tone Example

For example, suppose we have one reference tone and four data tones. Then we might have the frequency plan shown above. Using ϕ_R (the phase corresponding to the reference tone at frequency f_R) as the reference phase, the data bit corresponding to tone f_1, depends on the phase difference $\Delta\phi_{1R} = \phi_1 - \phi_R$.

This multi-tone set is transmitted once every T_h seconds and corresponds to one frame of data. Therefore, in this example every T_h seconds, this system can transmit four bits of DPSK.

In this chapter we define the transmitter, the channel and the receiver models respectively.

5.1 Transmitter Model

The transmitter for the slow frequency hopped spread spectrum signal with DPSK modulation, as shown in figure 5.2, is modelled as in [22]. There are K such transmitters in the spread spectrum multiple access system. The data waveform is written as:

$$b_k(t) = \sum_j b_k^j P_{T_b}(t - jT_b) \qquad b_k^j \in \{0,1\} \tag{5.1}$$

where b_k^j is the *jth* data bit of user k and P_{T_b} is a rectangular pulse of unit height and duration T_b.

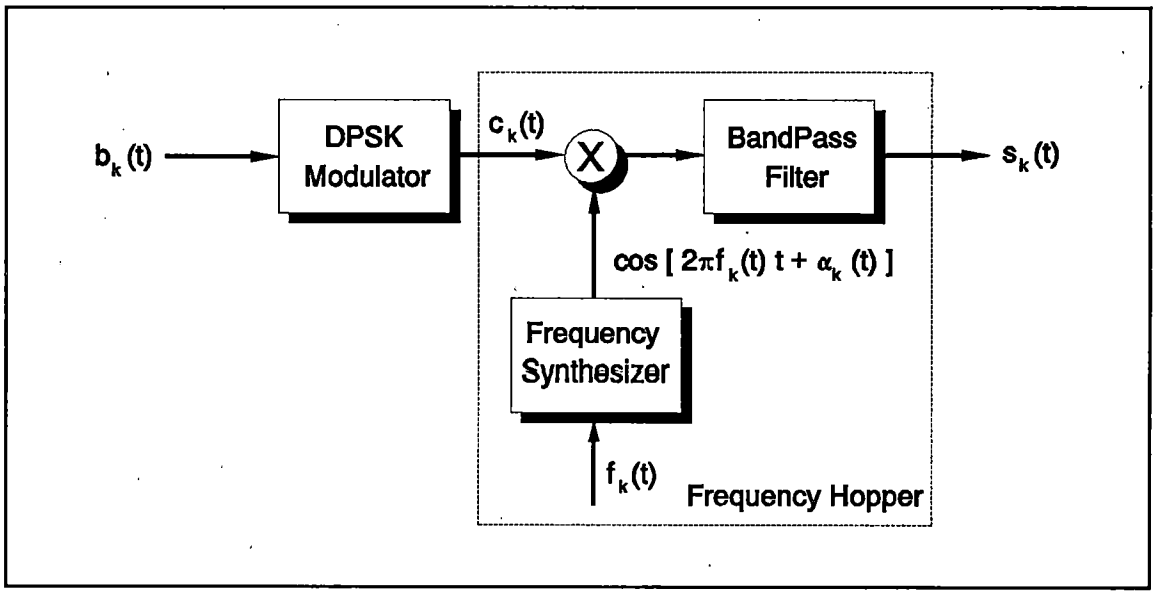

Figure 5.2 Transmitter Model

DPSK is based on normal PSK modulation with differential encoding and phase comparison decoding in the demodulator. PSK modulation of equation 5.1 results in

$$c_k(t) = \sum_j \cos(2\pi f_c t + \theta_k + \phi_k^j)\, P_{T_b}(t - jT_b) \qquad \phi \in \{0, \pi\} \tag{5.2}$$

where f_c is the carrier frequency, which is assumed the same for all the users and $\theta_k(t)$ is the phase angle introduced by the *kth* DPSK modulator.

Because the phase shift of π in DPSK just reverses the sign of the cosine, equation 5.2 can be written as:

$$c_k(t) = b_k(t)\cos(2\pi f_c t + \theta_k) \qquad b_k(t) \in \{-1, 1\} \tag{5.3}$$

The DPSK signal is then frequency hopped according to the k*th* hopping pattern $f_k(t)$.

$$c_k^*(t) = b_k(t)\cos(2\pi f_c t + \theta_k)\cos(2\pi f_k(t)t + \alpha_k(t)) \tag{5.4}$$

The bandpass filter shown in figure 5.2 removes unwanted frequency components present at the output of the multiplier. The signal at the output of the filter is

$$s_k(t) = \sqrt{2P}\, b_k(t)\cos\{2\pi[f_c + f_k(t)]t + \theta_k + \alpha_k(t)\} \tag{5.5}$$

where P is the power of the *kth* signal at the receiver. The signal $\alpha_k(t)$ represents the phase shift introduced by the frequency hopper as it switches from one frequency to another. We assume $\alpha_k(t)$ to be constant during the time intervals that $f_k(t)$ is constant.

5.2 Channel Model

We model the channel like in [23] and [24] as a discrete multipath slow fading channel, this means that there are L independently fading multiple signal path links between each user and the receiver antenna at the base station, as shown in figure 5.3. The multipath model is used here in order to investigate the influence of multiple resolvable paths on the system performance and we want to exploit it in the form of diversity.

The criterion for the existence of multiple resolvable paths is that the signal bandwidth W, is much larger than the coherence bandwidth of the channel ($W \gg (\Delta f)_c$). Because the slow frequency hopping signal has a transmission bandwidth equal to the base bandwidth, we cannot assume that there are multiple resolvable paths (and inherent diversity) in reality. So the channel model we assume is not a realistic one for SFH, but we wish to investigate how the performance of a SFH system responds to this channel model, in order to compare it with DS.

We assume that the transmitted signal power is equally distributed over the L specular paths. Each path has a gain denoted by β, a phase denoted by γ, and a time delay denoted by τ. The lowpass equivalent impulse response of the bandpass channel for the link between the k*th* user and the base station can be written as:

$$h_k(t) = \sum_{l=1}^{L} \beta_{kl}\, \delta(t-\tau_{kl})\, \exp(j\gamma_{kl}) \tag{5.6}$$

where the index kl refers to the l*th* path of the k*th* user.

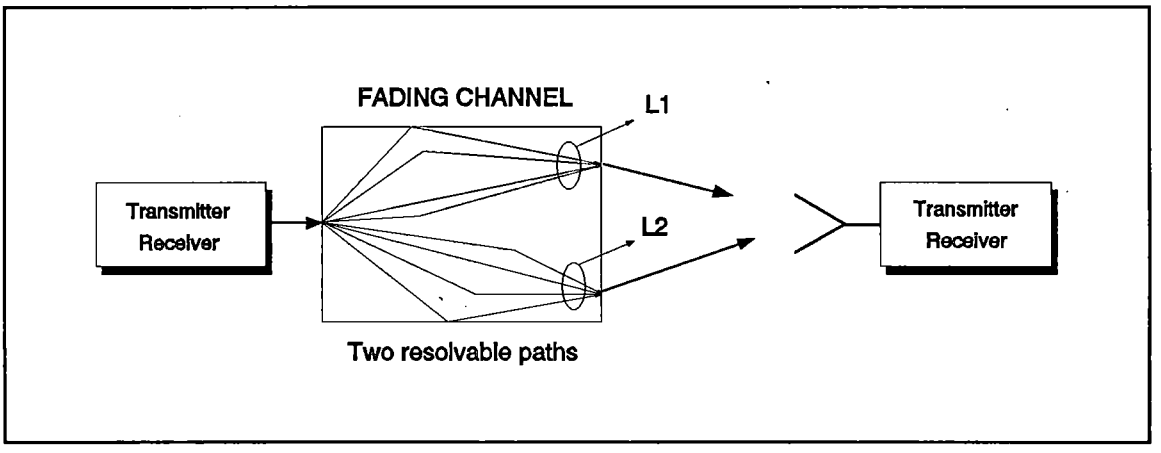

Figure 5.3 Channel Model with multiple resolvable paths

We assume that the path phase γ of the signal at the receiver is an independent random variable uniformly distributed in $[0, 2\pi]$. The path delay τ is also an independent random variable and is assumed uniform over $[0, T_b]$, so we restrict our attention to channels for which the multipath time delay is less than the bit duration T_b. The intersymbol interference here is due to partial correlation and involves only the previous bit. For most indoor channels, the delay spread (one microsecond or less) is considerably less than the bit intervals of practical interest, but as it turns out, this assumption will not have a negative influence on our results, as shown in appendix B.

We use the path gain β_{kl} in this report to incorporate the different statistical models for the multipath fading. The fading rate in an indoor radio environment is slow, as shown in chapter 3, so that the random parameters which describe the channel do not vary significantly over two bits.

5.2.1 Rayleigh Distribution

The Rayleigh pdf describes the envelope of the sum of two statistically independent zero-mean Gaussian random variables, with equal variance σ^2. We can assume that β_{kl} is an independent Rayleigh random variable. This would be in agreement with measurements done with received signals that have zero mean (There are no fixed scatterers and/or line-of-sight [LOS] component).

The Rayleigh probability density function is defined as:

$$p_\beta(r) = \frac{r}{\sigma^2} \exp\left(- \frac{r^2}{2\sigma^2} \right) \qquad r \geq 0 \tag{5.7}$$

We are interested in the Rayleigh fading environment, because of the large number of publications on it [e.g., 26]. They will be used in this report to compare our results.

5.2.2 Rician Distribution

The Rician pdf describes the envelope of the sum of two statistically independent non zero-mean Gaussian random variables with identical variance σ^2, so it is also possible to assume that β_{kl} is an independent Rician random variable. This would be in agreement with measurements done in office [6] and factory [7] buildings, so for Indoor Wireless Communications the Rician model would be more appropriate. The Rician distribution is applicable when a significant part of the received signal envelope is due to a constant path (such as a line-of-sight component or fixed signal reflectors).

The Rician probability density function is defined as:

$$p_\beta(r) = \frac{r}{\sigma^2} \exp\left(- \frac{r^2+s^2}{2\sigma^2} \right) I_0\left(\frac{s\,r}{\sigma^2} \right) \qquad r \geq 0,\ s \geq 0 \tag{5.8}$$

where $I_0(\ )$ is the modified Bessel function of the first kind and zero order. The noncentrality parameter, s, is the peak value of the specular radio signal due to the superposition of the dominant LOS signal and the time invariant scattered signals reflected from walls, ceiling and stationary inventory, and is defined as:

$$s^2 = m_1^2 + m_2^2 \tag{5.9}$$

where m_1 and m_2 are the mean values of the Gaussian random variables

The Rician distribution is characterised by the parameter R, which is the ratio of the peak power and the power received over specular paths:

$$R = \frac{s^2}{2\sigma^2} \tag{5.10}$$

When R = 0, there is typically no dominant path and the distribution becomes Rayleigh. From [6] we know that R = 6.8 dB corresponds to a 30-year-old brick building with reinforced concrete and plaster, as well as some ceramic block interior partitions. R = 11 dB corresponds to a building that has the same construction, but has an open-office interior floor plan, and non-metallic ceiling tiles.

5.3 Receiver Model

The receiver model we describe in this section consists of two parts: the frequency dehopper and the DPSK demodulator. We now choose user 1 as the reference user and assume that the receiver is capable of acquiring the frequency hopping pattern, and time synchronisation with the signal of user 1.

The received signal at the antenna for a certain user is the time convolution of equation 5.5 and 5.6 of the transmitted signal and the channel impulse response:

$$r(t) = \sqrt{2P} \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl}\, b_k(t-\tau_{kl}) \cos\{2\pi[f_c + f_k(t-\tau_{kl})]t + \phi_{kl}\} + n(t)$$
$$\text{with } \phi_{kl} = \theta_k + \alpha(t-\tau_{kl}) - 2\pi[f_c + f_k(t-\tau_{kl})]\tau_{kl} - \gamma_{kl} \tag{5.11}$$

where n(t) is the white Gaussian noise process with two-sided power spectral density $\frac{1}{2}N_0$ W/Hz.

The received signal is the input to a bandpass filter. This filter is followed by a dehopper, as shown in figure 5.4, which introduces a phase waveform $\delta_1(t)$ analogous to that introduced by the frequency hopper and is assumed constant during the hopping interval. The dehopper is

followed by a bandpass filter which removes the high frequency components. The output of the filter is given by

$$r_d(t) = \sqrt{\tfrac{1}{2}P} \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl}\, \delta[f_k(t-\tau_{kl}), f_1(t-\tau_{kl})]\, b_k(t-\tau_{kl}) \cos\{2\pi f_c t + \Phi_{kl}(t)\} + \hat{n}(t) \qquad \text{with } \Phi_{kl}(t) = \phi_{kl}(t) + \delta_1(t) \tag{5.12}$$

where $\hat{n}(t)$ is the white Gaussian noise process with two-sided power spectral density $\frac{1}{2}N_0$ W/Hz and we consider $\delta(u,v)=0$ for $u \neq v$ and $\delta(u,v)=1$ for $u=v$ (both u and v are real).
For simplicity reasons we assume that Φ_{kl} is an independent random variable uniformly distributed in $[0, 2\pi]$.

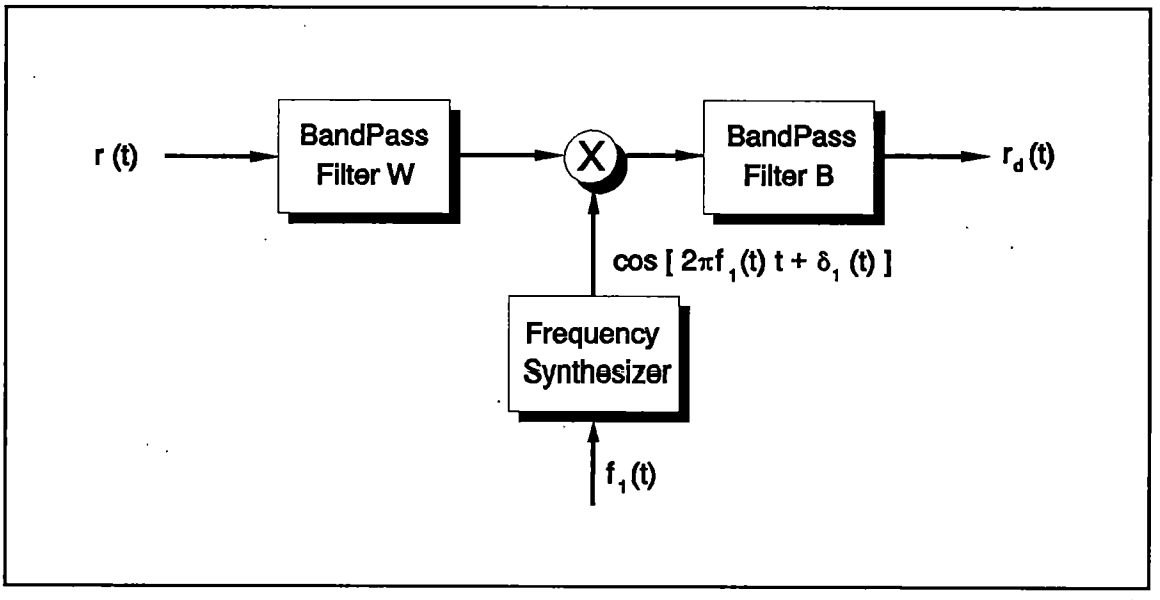

Figure 5.4 Frequency Dehopper for user 1

Equation 5.12 can be written in the lowpass equivalent form:

$$r_d(t) = x(t)\cdot\cos(2\pi f_c t) + y(t)\cdot\sin(2\pi f_c t) + \hat{n}(t) \tag{5.13}$$

with

$$\hat{n}(t) = n_c(t)\cdot\cos(2\pi f_c t) + n_s(t)\cdot\sin(2\pi f_c t) \tag{5.14}$$

$$x(t) = \sqrt{\tfrac{1}{2}P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot b_k(t-\tau_{kl}) \cdot \cos(\Phi_{kl}) + n_c(t) \tag{5.15}$$

$$y(t) = \sqrt{\tfrac{1}{2}P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot b_k(t-\tau_{kl}) \cdot \sin(\Phi_{kl}) + n_s(t) \tag{5.16}$$

where x(t) and $n_c(t)$ are the in-phase components and y(t) and $n_s(t)$ the quadrature components.

The standard DPSK demodulator is shown in figure 5.5. This type of demodulator is called a matched filter demodulator and is an optimum demodulator [25], because the probability of a decision error is minimised, when the outputs of the matched filters are used for the decision. When the signals are equally likely, the output of a matched filter in a standard optimum demodulator is given by

$$U_m = \mathrm{Re}\left[e^{j\psi} \int_0^T r(t) u_m^*(t)\, dt \right] \tag{5.17}$$

where ψ is the carrier phase. Each of the m matched filters is matched to a particular waveform $u_m(t)$. The decision is then made by selecting the largest of the m outputs.

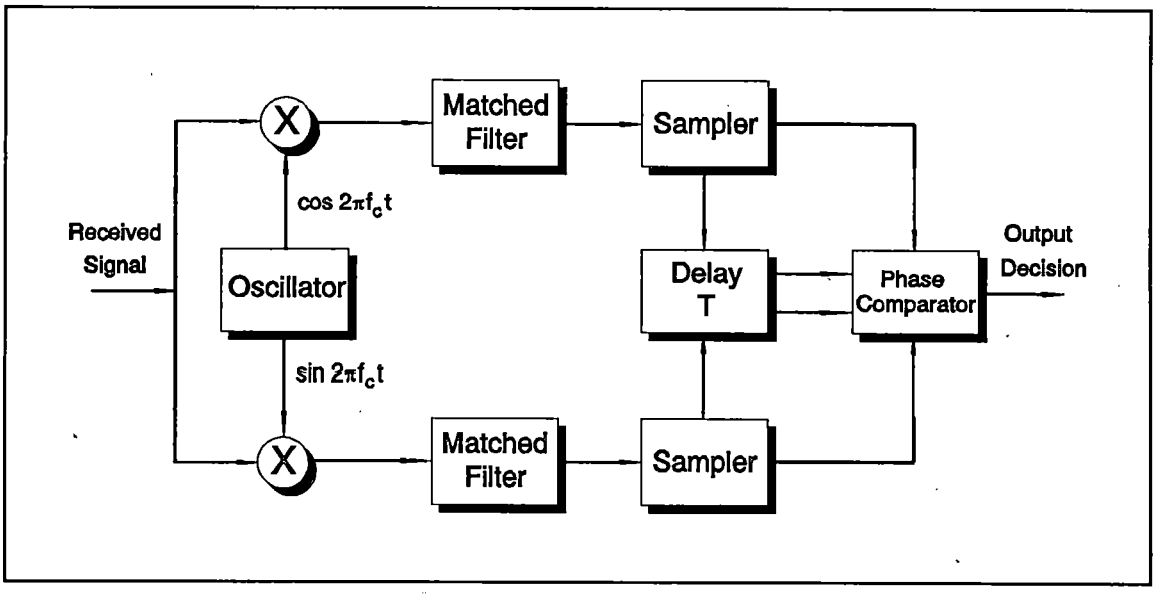

Figure 5.5 Standard DPSK Demodulator [25]

In our DPSK demodulator the optimum demodulator is implemented as a single filter [25] which computes the vector

$$V = e^{j\Phi} \int_0^T r(t)\, u^*(t)\, dt \tag{5.18}$$

where the superscripted asterisk * represents the complex conjugate.
In DPSK the decision variable is the phase difference between the vector of the present signalling interval V_0 and the vector of the previous signalling interval V_{-1}.

The decision variable can now be found by projecting V_0 onto V_{-1} and use the phase of the resulting complex number:

$$U_0 = V_0\, V_{-1}^* \tag{5.19}$$

In binary DPSK, the two possible transmitted phase differences are zero and π rad. As a consequence, only the real part of the complex number U_0 is needed for recovering the information.

$$\begin{aligned} \text{Re}\,[U_0] &= \tfrac{1}{2}\,(V_0 V_{-1}^* + V_0^* V_{-1}) \\ V_0 = X_{V_0} + jY_{V_0} \quad &\text{and} \quad V_{-1} = X_{V_{-1}} + jY_{V_{-1}} \end{aligned} \tag{5.20}$$

The real part of U_0 is called the decision variable ξ and is expressed as

$$\xi = X_{V_0} X_{V_{-1}} + Y_{V_0} Y_{V_{-1}} \tag{5.21}$$

We replace the DPSK demodulator of figure 5.5 by the demodulator shown in figure 5.6.
Because the data waveform is a series of rectangular pulses, we may replace the matched filters by integrate-and-dump filters. The de-hopped output $r_d(t)$ is multiplied by either cos $(2\pi f_c t)$ or sin $(2\pi f_c t)$ and the product is integrated and sampled periodically at the end of each T-second signalling interval. The output of the integrator is reset to zero after sampling.
The integrator acts as a lowpass filter, thus rejecting the double frequency components resulting from the multiplication (The timing signal for sampling the output of the integrator is derived from the received signal).

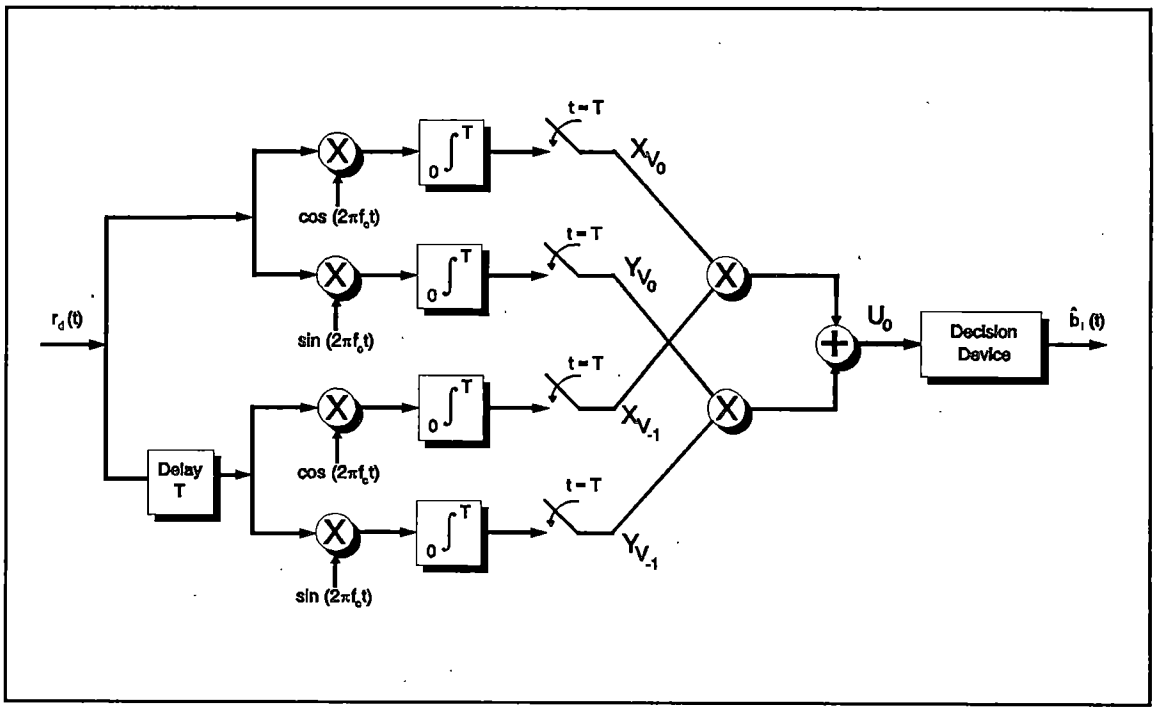

Figure 5.6 DPSK Demodulator for SFH

At time t we can write for the outputs of the integrate-and-dump filters of figure 5.6:

$$X_{V_0} = \sqrt{1/8P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot \cos(\Phi_{kl}) \cdot \int_0^t b_k(s-\tau_{kl})\, ds + \int_0^t n_c(s) ds \qquad (5.22)$$

and

$$Y_{V_0} = \sqrt{1/8P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot \sin(\Phi_{kl}) \cdot \int_0^t b_k(s-\tau_{kl})\, ds + \int_0^t n_s(s) ds \qquad (5.23)$$

5.3.1 Interference due to Multiple Resolvable Paths

We assume that the receiver can synchronise to an arbitrary path at t=0, then at $t=T=T_b$ the filter output is sampled.

The *jth*, the first and the last path signals of the reference user are shown in figure 5.7 during a hopping interval. The L-1 other path signals are at the same frequency as the *jth* path, so each bit of the *jth* path signal is fully hit L-1 times.

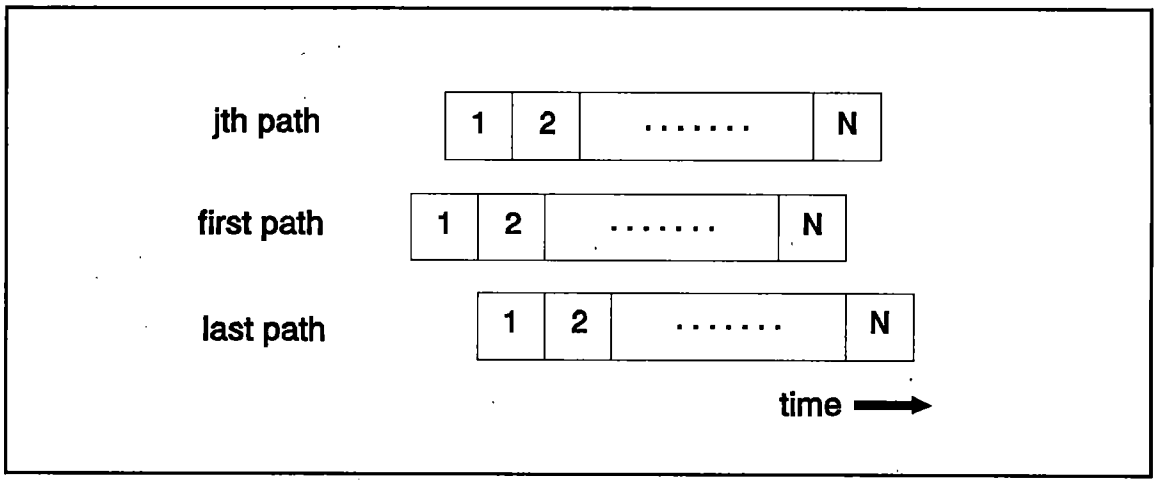

Figure 5.7 Interference due to multiple resolvable paths during a hop interval

The contribution from the *ith* path signal, assuming that $\tau_{1i} > \tau_{1j}$, is incorporated into the expression for the signals at the sampling points $T=T_b$:

$$X_{V_0} = \sqrt{\tfrac{1}{8}P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot \cos(\Phi_{kl}) \cdot \left[\int_0^{\tau_{kl}} b_k^{-1}(t-\tau_{kl})\, dt + \int_{\tau_{kl}}^{T} b_k^{0}(t-\tau_{kl})\, dt \right] + \eta \quad (5.24)$$

and

$$Y_{V_0} = \sqrt{\tfrac{1}{8}P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot \sin(\Phi_{kl}) \cdot \left[\int_0^{\tau_{kl}} b_k^{-1}(t-\tau_{kl})\, dt + \int_{\tau_{kl}}^{T} b_k^{0}(t-\tau_{kl})\, dt \right] + \nu \quad (5.25)$$

where b_k^{-1} and b_k^0 are the previous and current data bit respectively, and

$$\eta = \int_0^T n_c(s)ds \quad , \quad \nu = \int_0^T n_s(s)ds \quad (5.26)$$

The noise samples η and ν are independent zero-mean Gaussian random variables with identical variance $\sigma_n^2 = \frac{1}{16}N_0T$ [22].

We now assume that the *jth* path between the transmitter of user 1 and the corresponding receiver is the reference path and all other paths constitute interference, so that without loss of generality, $\tau_{1j}=0$ and $\Phi_{1j}=0$.

Since the fading is slow, we may also assume that the delayed vector V_{-1} differs from V_0 only in the data bits involved, and in the additive Gaussian noise samples.

$$\begin{aligned} V_0 &= X_{V_0} + jY_{V_0} \\ &= \sqrt{\tfrac{1}{8}P}\cdot\beta_{1j}\cdot T\cdot b_1^0 + \sqrt{\tfrac{1}{8}P}\cdot\left[\sum_{k=1}^{K}\sum_{\substack{l=1\\ l\neq j}}^{L} X_{kl} + j\,Y_{kl}\right] + (\eta_1 + j\nu_1) \end{aligned} \tag{5.27}$$

where

$$X_{1l} = \left[\int_0^{\tau_{1l}} b_1^{-1}(t-\tau_{1l})\,dt + \int_{\tau_{1l}}^{T} b_1^0\,(t-\tau_{1l})\,dt\right]\cdot\beta_{1l}\cdot\cos(\Phi_{1l})$$

$$Y_{1l} = \left[\int_0^{\tau_{1l}} b_1^{-1}(t-\tau_{1l})\,dt + \int_{\tau_{1l}}^{T} b_1^0\,(t-\tau_{1l})\,dt\right]\cdot\beta_{1l}\cdot\sin(\Phi_{1l}) \tag{5.28}$$

and for $k \geq 2$

$$X_{kl} = \left[\int_0^{\tau_{kl}} b_k^{-1}(t-\tau_{kl})\,dt + \int_{\tau_{kl}}^{T} b_k^0\,(t-\tau_{kl})\,dt\right]\cdot\beta_{kl}\cdot\cos(\Phi_{kl})$$

$$Y_{kl} = \left[\int_0^{\tau_{kl}} b_k^{-1}(t-\tau_{kl})\,dt + \int_{\tau_{kl}}^{T} b_k^0\,(t-\tau_{kl})\,dt\right]\cdot\beta_{kl}\cdot\sin(\Phi_{kl}) \tag{5.29}$$

6. SLOW FREQUENCY HOPPING WITH SELECTION DIVERSITY

The major problem in the reception of a radio signal is when the transmitted signal is in a deep fade, i.e., when the channel attenuation is large, the received signal will have several errors.
A solution of this problem is to supply replicas of the transmitted signal to the receiver over independently fading channels. This technique is called *diversity*.
We know from chapter three that spread spectrum provides us with several independently fading multiple signal paths when the transmission bandwidth is larger than the coherence bandwidth of the channel. This is called *inherent* (spread spectrum) diversity.

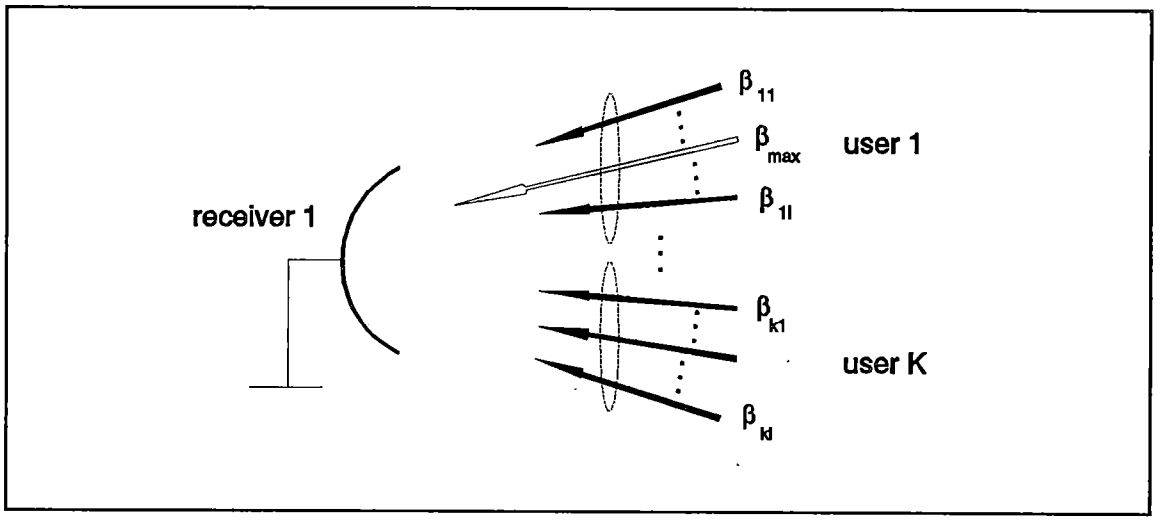

Figure 6.1 Selection Diversity

Selection diversity is based on selecting the largest of a group of signals carrying the same information. The multiple resolvable paths can be used for selection diversity by selecting the path with the largest output of the matched filter. With selection diversity of order M, the decision variable is

$$\xi_{max}^{M} = \max_{l=1,\ldots,M} (\xi_l) \tag{6.1}$$

where $\{\xi_i, i=1,\ldots,M\}$ are identically distributed independent stochastic variables.

The order of diversity that can be achieved with selection diversity is defined as:

$$M = L \times \text{Number of Antennas} \tag{6.2}$$

Another technique used to achieve diversity is to use multiple receiving antennas. These must be spaced sufficiently far apart so that the multipath components in the signal have significantly different propagation delays at the antennas. Usually a separation of at least 10 wavelengths is required [25] between two antennas in order to obtain signals that fade independently.

In this report, we shall evaluate the system performance with the average probability of error, which is the error probability averaged over the ensemble of channels obtained from the assumed model. This average error probability would be observed in reality if the channel between a stationary terminal and the base station varied with time (due to movements in the medium). Alternatively, it would also be observed if the terminals were mobile in the building where the channel between any two points is relatively static.

We define the (average) bit error probability for slow frequency hopping system with selection diversity as:

$$\begin{aligned} P_e &\triangleq \Pr\{\text{bit error} \mid \text{no hit}\} \cdot \Pr\{\text{no hit}\} + \Pr\{\text{bit error} \mid \text{hit}\} \cdot \Pr\{\text{hit}\} \\ &= \Pr\left\{ \begin{array}{l} \text{bit error due to interference} \\ \text{from multiple resolvable paths} \end{array} \middle| \text{no hit} \right\} \cdot \Pr\{\text{no hit}\} + \Pr\{\text{hit}\} \end{aligned} \tag{6.3}$$

where we have assumed that all bits are destroyed when a hit occurs.

6.1 Probability of a Hit

A hit occurs when two (or more) frequency hopping signals occupy the same frequency "slot" and interfere. We say that a hit from the *ith* signal occurs during a bit period if

$$f_i(t-\tau_i) = f_1(t) \tag{6.4}$$

The frequency hopping transmitter/receiver hops between the available frequencies according to a hopping pattern, which can be user specific. There are several ways to generate the hopping pattern for a particular user:

- **Random hopping patterns**

  The frequency to which the transmitter/receiver hops is chosen at random. The probability of a hit here depends on the number of users in the system and on the number of available frequencies.

  When the random hopping patterns are mutually independent and identically distributed, the probability of one or more hits from the K-1 signals is given by [22] as:

$$\Pr\{\text{ hit }\} = 1 - \left(1 - \frac{1}{Q}\left(1 + \frac{1}{N_b}\right)\right)^{K-1} \tag{6.5}$$

  where $N_b = T_h/T_b$ is the number of bits per hop and Q is the number of frequencies.

  In [22] it is shown that, for large Q, these memoryless random hopping patterns give approximately the same probability of a hit as first order Markov patterns.

  When random hopping patterns are used, it is very easy to add new users to the system.

  The price for this flexibility is a deterioration of the performance (in terms of average bit error probability) for every user in the system. This effect can be seen in figure 6.2, where the probability of a hit increases with increasing number of simultaneously transmitting users.

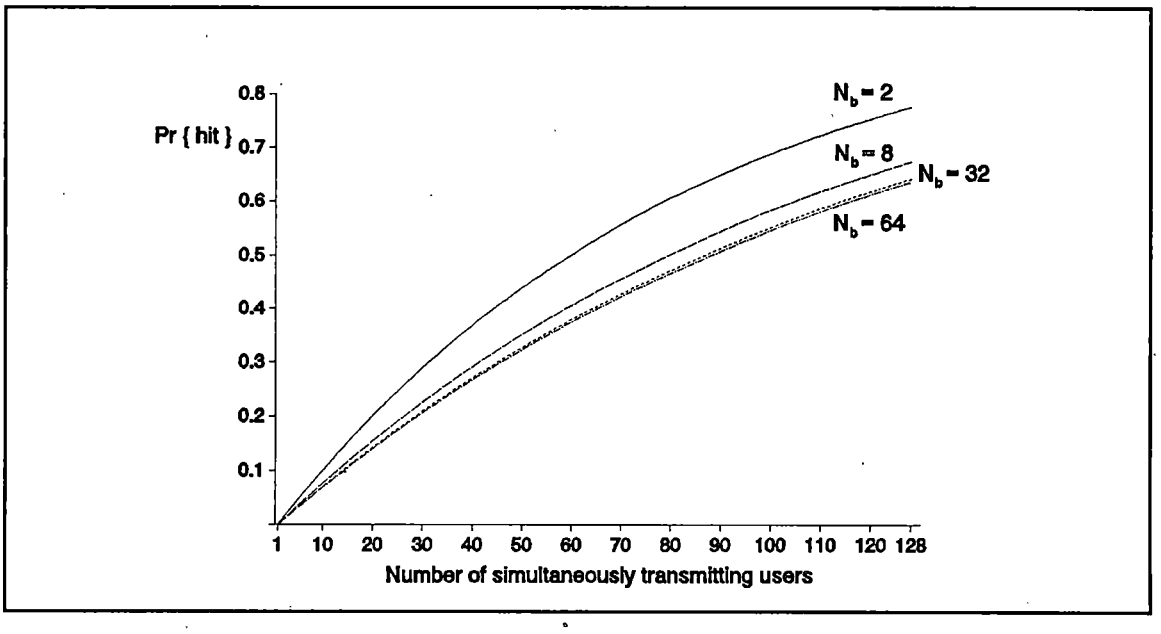

Figure 6.2 Probability of a hit as a function of the number of simultaneously transmitting users for several values of N_b with the number of frequencies Q taken 128

- **Hopping patterns generated by a Pseudo-Noise (PN) generator**

  The frequency to which the transmitter/receiver hops is chosen by means of a user-specific PN-code, such as a Gold-code. The probability of a hit depends on the orthogonality of the chosen codes. A major problem with this method is that the system does not use its full capacity, because not all the frequencies can be used, when the probability of a hit has to be kept low.

- **Deterministic hopping patterns**

  Instead of giving each user a unique PN-code we can give each transmitter/receiver in the system its own unique hopping pattern. When we design the system so that it is not possible for two users to occupy the same frequency at the same time, the maximum number of users for a system will be Q. The hopping pattern of a certain user is a permutation of the set of available frequencies {1, 2, 3, .., Q}. Since there are Q! such permutations, it is necessary to use an algorithm to select Q orthogonal hopping patterns. An example of such an algorithm is described in [12].

  A system using deterministic hopping patterns does not have the flexibility of a system using random hopping patterns, because it does not make use of the advantages of CDMA. The advantage of this technique over random hopping patterns is a much lower average bit error probability.

  A hit can theoretically occur only in the first and last bit of a hop, when $\tau_{il} > \tau_{1j}$ for the first bit (the user that used the frequency before user 1 causes interference) or $\tau_{il} < \tau_{1l}$ for the last bit (the user that uses the frequency after the user 1 causes interference).

  For the system considered in this report we assume that a certain amount of time, larger than T_b, is needed for the acquisition of the signal by the receiver, so we assume the probability of a hit to be zero. The smallest acquisition time needed using a Phase Lock Loop synthesizer has been reported by [30] as 4 μs, but more practical values are around 4 ms.

6.2 Bit Error Probability due to Interference from Multiple Resolvable Paths

When no hit occurs, and because the data bits are equiprobable, we can write the bit error probability due to multi-path interference as:

$$P_e = \tfrac{1}{2} \Pr\{ \xi_{max}^{M} < 0 \mid b_1^0 b_1^{-1} = 1 \} + \tfrac{1}{2} \Pr\{ \xi_{max}^{M} > 0 \mid b_1^0 b_1^{-1} = -1 \}$$
$$= \Pr\{ \xi_{max}^{M} < 0 \mid b_1^0 b_1^{-1} = 1 \} = \Pr\{ \xi_{max}^{M} > 0 \mid b_1^0 b_1^{-1} = -1 \} \quad (6.6)$$

We derive this probability using the formula for the bit error probability for binary DPSK modulated signals, with multichannel reception in a time-invariant Rician fading channel, given in appendix 7A.4 and derived in appendix 4B of [25]. Because with selection diversity we only use the strongest path and all other signals are seen as interference, we may use this equation in the case of L=1, i.e., for a single path:

$$P_e(\gamma_b) = Q(a,b) - \tfrac{1}{2}\left[1 + \frac{\mu}{\sqrt{\mu_0 \mu_{-1}}} \right] \cdot \exp\left(- \frac{a^2 + b^2}{2} \right) \cdot I_0(ab) \quad (6.7)$$

where $I_0(ab)$ is the modified Bessel function of the first kind and zero order which is defined as:

$$I_0(ab) = \frac{1}{2\pi} \int_0^{2\pi} \exp(ab \cos\theta)\, d\theta \quad (6.8)$$

and Q(a,b) is the Marcum Q-function which is defined as:

$$Q(a,b) = \int_b^{\infty} x \cdot \exp\left(- \frac{a^2 + x^2}{2} \right) \cdot I_0(ax)\, dx \quad (6.9)$$

where a and b are:

$$a = \frac{m}{\sqrt{2}} \left| \frac{1}{\sqrt{\mu_0}} - \frac{1}{\sqrt{\mu_{-1}}} \right| , \quad b = \frac{m}{\sqrt{2}} \left[\frac{1}{\sqrt{\mu_0}} + \frac{1}{\sqrt{\mu_{-1}}} \right] \quad (6.10)$$

and

$$m = E[V_0 \mid \beta_{max}, b_1^0] = E[V_{-1} \mid \beta_{max}, b_1^{-1}] \quad (6.11)$$

$$\mu_0 = \mathrm{var}(V_0 \mid L)$$
$$\mu_{-1} = \mathrm{var}(V_{-1} \mid L) \quad (6.12)$$
$$\mu = E[(V_0 - m)(V_{-1} - m)^* \mid L]$$

The statistical moments m, μ_0, μ_{-1} and μ, defined above are calculated in Appendix B. The results are given below, so for a more detailed derivation refer to Appendix B.

$$m = \sqrt{\tfrac{1}{8}P}\cdot\beta_{max}\cdot T_b\cdot b_1^0 = \sqrt{\tfrac{1}{8}P}\cdot\beta_{max}\cdot T_b\cdot b_1^{-1} \tag{6.13}$$

$$\mu_0 = \tfrac{1}{4}P\cdot T_b^2\cdot\left((L-1)\cdot(\sigma^2+\tfrac{1}{2}s^2) + (L-1)(L-2)\cdot\tfrac{1}{2}s^2\right) + \tfrac{1}{8}N_0\cdot T_b \tag{6.14}$$

$$\mu_{-1} = \tfrac{1}{8}P\cdot T_b^2\cdot\left((L-1)\cdot(\sigma^2+\tfrac{1}{2}s^2) + (L-1)(L-2)\cdot\tfrac{1}{2}s^2\right) + \tfrac{1}{8}N_0\cdot T_b \tag{6.15}$$

$$\mu_0 = \tfrac{1}{8}P\cdot T_b^2\cdot\left((L-1)\cdot(\sigma^2+\tfrac{1}{2}s^2) + (L-1)(L-2)\cdot\tfrac{1}{2}s^2\right) \tag{6.16}$$

Selection diversity can be accomplished by letting the reference path β_{1j} be the strongest path, i.e., $\beta_{1j} = \beta_{max}$, and since the fading is slow this path will also be the strongest path for the next bit, so we have maximised the decision variable in equation 6.1.

The bit error probability of equation 6.7 is a function of the bit signal-to-noise ratio, which is defined as:

$$\gamma_b = \frac{E_b}{N_o}\beta^2 \tag{6.17}$$

In order to obtain the average bit error probability we must average the conditional error probability given in equation 6.7 over the Rician fading channel statistics. The pdf of the bit signal-to-noise ratio with correctly selected β_{max}, will be derived in the following section.

6.2.1 The Probability Density Function of the Bit Signal-to-Noise Ratio

Selection diversity is based on selecting the strongest signal from several statistically independent signals carrying the same data. In our case these signals are usually the multiple resolvable paths due the inherent diversity of spread spectrum, but these signals can also arrive from different antennas, in order to increase the order of diversity.

Suppose we have M identically distributed random variables $\{ \beta_1, ..., \beta_M \}$. For the largest random variable, β_{max}, the following inequality holds:

$$\beta_{max} > \beta_i \qquad i \in \{1,2,...,M\} \tag{6.18}$$

The probability that $\beta_{max} < y$, i.e., $P_{\beta_{max}}(y)$ is now given by:

$$\begin{aligned} P_{\beta_{max}}(y) &= Pr\{\beta_1 < y\} \cdot Pr\{\beta_2 < y\} \cdot .... \cdot Pr\{\beta_M < y\} \\ &= P_{\beta_1}(y) \cdot P_{\beta_2}(y) \cdot .... \cdot P_{\beta_M}(y) \end{aligned} \tag{6.19}$$

This means that the cdf of β_{max} is the product of the cdfs of the M path gains.

Since β is Rician distributed, β^2 has a non-central chi-square probability distribution with 2 degrees of freedom [25 eq. 1.1.115]. Consequently, the distribution for γ_b is given by the non-central chi-square pdf with 2 degrees of freedom and by making use of equation B.15 we can write the pdf of γ_b as:

$$p_{\gamma_b}(\gamma_b) = \frac{1}{2\sigma^2 E_b/N_o} \exp\left(- \frac{s^2 + \frac{\gamma_b}{E_b/N_o}}{2\sigma^2} \right) I_0\left(\sqrt{\frac{\gamma_b}{E_b/N_o}} \frac{s}{\sigma^2} \right) \tag{6.20}$$

The cdf of a non-central chi-square probability distribution with 2 degrees of freedom is then:

$$P_{\gamma_b}(\gamma_b) = \int_0^{\gamma_b} \frac{1}{2\sigma^2 E_b/N_o} \exp\left(- \frac{s^2 + \frac{z}{E_b/N_o}}{2\sigma^2} \right) I_0\left(\sqrt{\frac{z}{E_b/N_o}} \frac{s}{\sigma^2} \right) dz \tag{6.21}$$

The cdf of γ_b for β^2_{max} is now given by:

$$P_{\gamma_b|\beta_{max}}(\gamma_b) = \left[P_{\gamma_b}(\gamma_b) \right]^M \tag{6.22}$$

This pdf can be obtained by calculating the derivative with respect to γ_b:

$$p_{\gamma_b|\beta_{max}}(\gamma_b) = M \left[P_{\gamma_b}(\gamma_b) \right]^{M-1} \cdot \frac{dP_{\gamma_b}(\gamma_b)}{d\gamma_b} \tag{6.23}$$

The derivative of P_{γ_b} is just a non-central chi-square pdf with 2 degrees of freedom, so:

$$p_{\gamma_b|\beta_{max}}(\gamma_b) = M \left[\int_0^{\gamma_b} \frac{1}{2\sigma^2 E_b/N_o} \exp\left(- \frac{s^2 + \frac{z}{E_b/N_o}}{2\sigma^2} \right) \cdot I_0\left(\sqrt{\frac{z}{E_b/N_0}} \frac{s}{\sigma^2} \right) dz \right]^{M-1} \tag{6.24}$$

$$\cdot \frac{1}{2\sigma^2 E_b/N_o} \exp\left(- \frac{s^2 + \frac{\gamma_b}{E_b/N_o}}{2\sigma^2} \right) \cdot I_0\left(\sqrt{\frac{\gamma_b}{E_b/N_o}} \frac{s}{\sigma^2} \right)$$

We have plotted $p(\gamma_b)$ for several values of diversity in figure 6.3, where it should be noted that the pdf for larger signal-to-noise ratios is spread considerably.

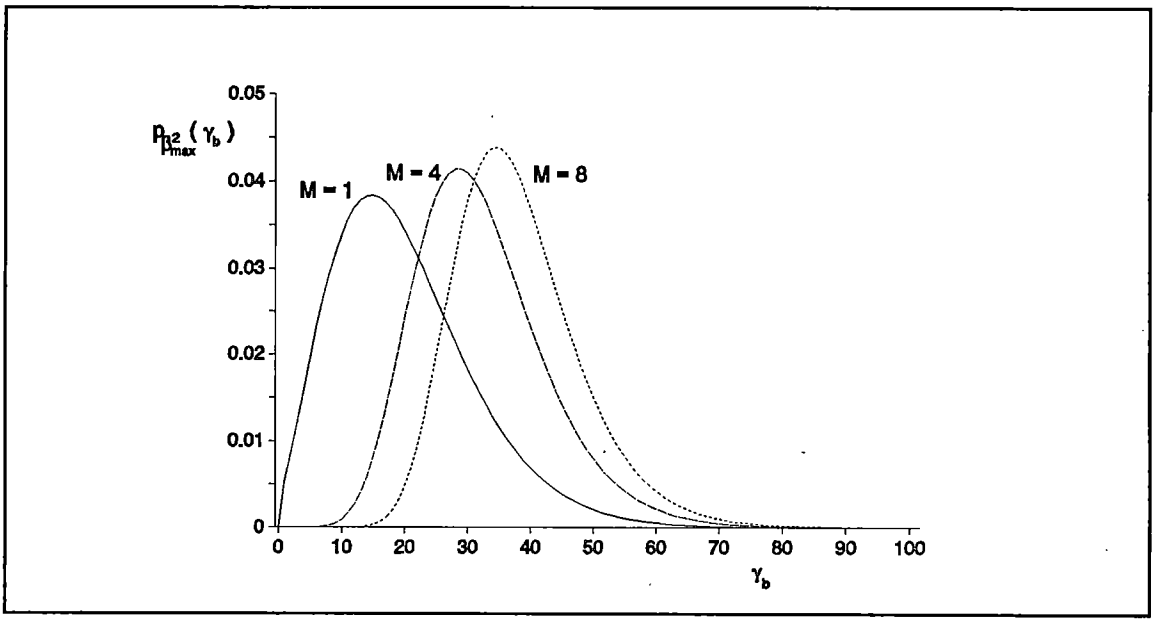

Figure 6.3 The pdf of the highest path gain for several values of the order of diversity M, with R = 6.8 dB and SNR = 10 dB

6.2.2 Expression for the Bit Error Probability

Now we know the conditional bit error probability and the probability density function of the highest path gain, we can write for the bit error probability using random hopping patterns:

$$P_e = \int_0^\infty P_e(\gamma_b) \cdot p_{\gamma_b}(\gamma_b)\, d\gamma_b \cdot \left(1 - \frac{1}{M}\left(1 + \frac{1}{N_b}\right)\right)^{K-1} + 1 - \left(1 - \frac{1}{M}\left(1 + \frac{1}{N_b}\right)\right)^{K-1} \tag{6.25}$$

In the case of deterministic hopping patterns, the probability of a hit is zero, as discussed in section 6.1. The average bit error probability can then be written as:

$$P_e = \int_0^\infty P_e(\gamma_b) \cdot p_{\gamma_b \mid \beta_{max}}(\gamma_b)\, d\gamma_b \tag{6.26}$$

6.3 Computational Results

Before we can calculate the average bit error probability, we have to write σ and s in terms of the signal-to-noise ratio and the Rician parameter R. The received signal power is written as:

$$P = \tfrac{1}{2}s^2 + \sigma^2 \tag{6.27}$$

By making use of the Rician parameter

$$R = \frac{s^2}{2\sigma^2} \tag{6.28}$$

we can write $\tfrac{1}{2}s^2$ and σ^2 as:

$$\begin{aligned} \tfrac{1}{2}s^2 &= \frac{PR}{R+1} \\ \sigma^2 &= \frac{P}{R+1} \end{aligned} \tag{6.29}$$

We can now write the statistical moments of equation 6.7 in terms of the bit signal-to-noise ratio and the channel signal-to-noise ratio:

$$a = \left| \sqrt{\frac{\gamma_b}{4\lambda + 2}} - \sqrt{\frac{\gamma_b}{2\lambda + 2}} \right| \qquad b = \left[\sqrt{\frac{\gamma_b}{4\lambda + 2}} + \sqrt{\frac{\gamma_b}{2\lambda + 2}} \right]$$

$$\frac{\mu}{\sqrt{\mu_{-1}\mu_0}} = \sqrt{\frac{\lambda^2}{2\lambda^2 + 3\lambda + 1}} \tag{6.30}$$

$$\text{with} \qquad \lambda = \frac{E_b}{N_o} \cdot \left((L-1) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + (L-1)\cdot(L-2) \cdot \tfrac{1}{2}s^2 \right)$$

Two important conclusions can be drawn from these equations. The first is that the bit error probability is independent of the channel bit rate; the variable T_b has dropped out of the expressions for the statistical moments and the variables a and b. The product PT_b is also known as the average energy per bit E_b, which is determined by the value of SNR and consequently also the product PT_b is fixed. Thus the bit error probability is independent of R_c.

The second conclusion that can be drawn is that $P_e(\gamma_b)$ for L=1 closely approximates the bit error probability function of DPSK transmission in an AWGN channel with selection diversity.

This can be explained by again looking at the statistical moments μ_0, μ_{-1} and μ. When L=1 the expressions simplify into

$$\begin{aligned} \mu_0 &= \mu_{-1} = \tfrac{1}{8} N_o T_b \\ \mu &= 0 \\ a &= 0, \qquad b = \frac{2m}{\sqrt{2\mu_0}} \end{aligned} \tag{6.31}$$

The expression for the conditional bit error probability can now be written as:

$$\begin{aligned} P_{e|\beta_{max}(L=1)} &= \tfrac{1}{2} \exp\left(- \frac{m^2}{\mu_0} \right) = \tfrac{1}{2} \exp\left(- \frac{PT\,\beta_{max}^2}{N_o} \right) \\ &= \tfrac{1}{2} \exp\left(- \frac{E_b}{N_o} \beta_{max}^2 \right) \end{aligned} \tag{6.32}$$

When we compare equation 6.32 with equation A.1, it is clear that the expressions for the bit error probability are equal.

Because the received signal power is a parameter in our expression for the average bit error probability, we normalise it with respect to the received scattered power, in order to obtain more general results. So:

$$\begin{aligned} P &= \sigma^2 + \tfrac{1}{2} s^2 = \sigma^2 \cdot (1 + R) \\ \sigma^2 &= 1 \;\Rightarrow\; P_N = 1 + R \end{aligned} \tag{6.33}$$

The integrals in equations 6.25 and 6.26 are calculated using the Romberg procedure from the Turbo Pascal Mathematical Methods Toolbox.
In the following sections we shall investigate the bit error probability for the two hopping techniques: deterministic hopping and random hopping.

6.3.1 Deterministic Hopping Patterns

In this section we calculate the average bit error probability as a function of the signal-to-noise ratio E_b/N_o for deterministic hopping patterns. The hopping patterns are chosen such to prevent the possibility that two users occupy the same frequency.

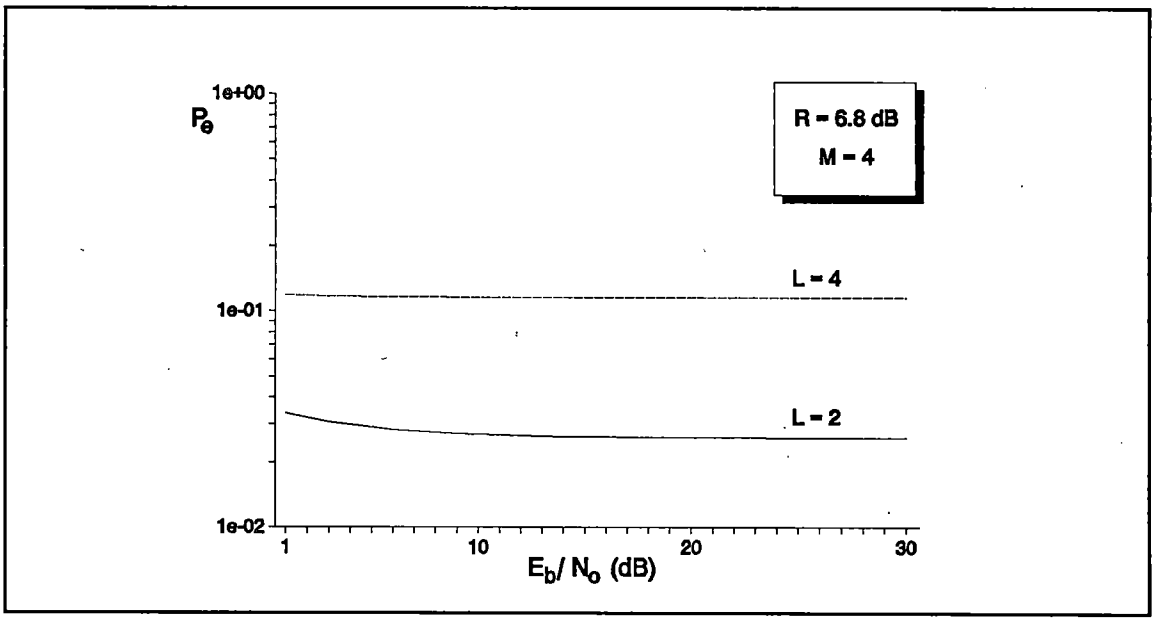

Figure 6.4 Average Bit Error Probability as a function of the signal-to-noise ratio for several values of the number of resolvable paths with R = 6.8 dB

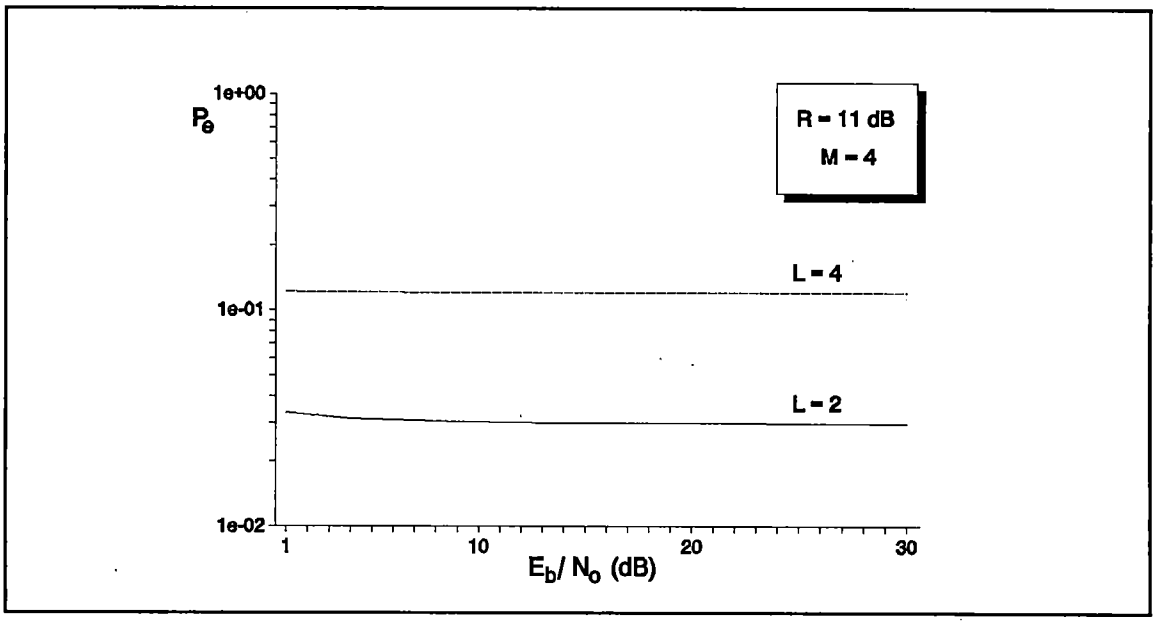

Figure 6.5 Average Bit Error Probability as a function of the signal-to-noise ratio for several values of the number of resolvable paths with R = 11 dB

The average bit error probability is plotted in figures 6.4 and 6.5 as a function of the signal-to-noise ratio. The received signal power P is normalised, but for higher values of P the average bit error probability will become better. We can see that the average bit error probability increases with increasing number of resolvable paths. When we compare the results of figure 6.4 with 6.5, it is clear that an increase of the Rice parameter enhances the performance of the system.

The influence of the order of diversity on the performance is investigated in figure 6.6. The results shown are for the orders of diversity M = 2, 4 and 8. As expected, we can see in figure 6.6 that the bit error probability decreases with increasing order of diversity.

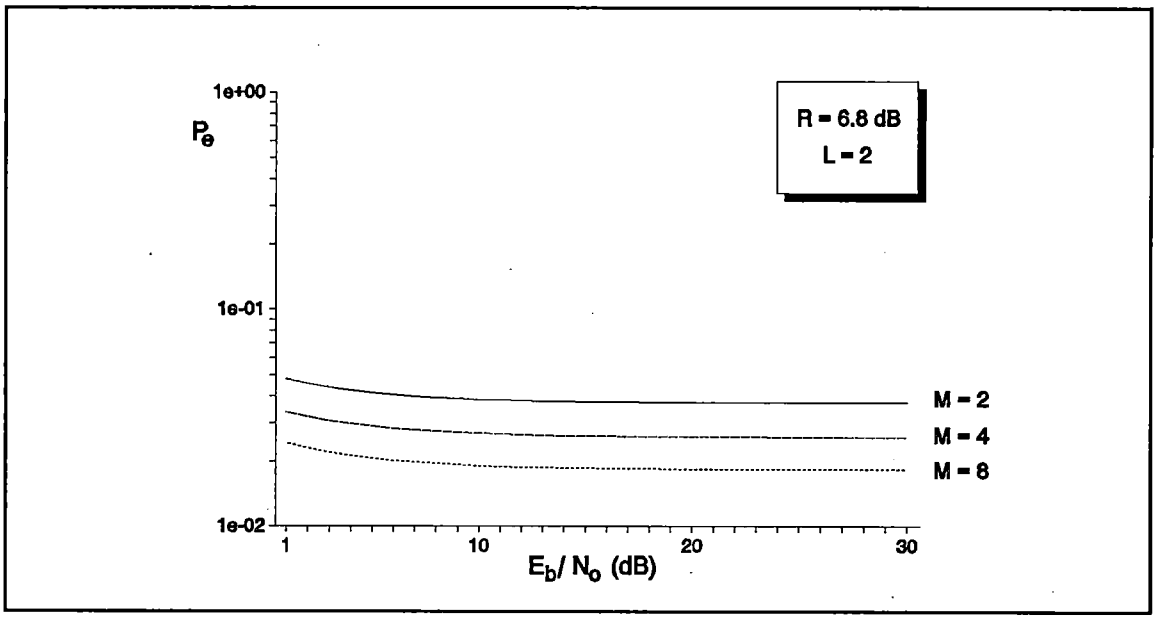

Figure 6.6 Average Bit Error Probability as a function of the signal-to-noise ratio for several values of the order of diversity M

6.3.2 Random Hopping Patterns

In this section we will investigate the influence of random hopping patterns on the bit error probability. As shown in equation 6.25, the probability of a hit forms a lower bound of the bit error probability that can be reached, so the number of simultaneously transmitting users becomes an important design variable. For the performance measures in this section we will assume that 12% of the system capacity is used at any time t. In a system of 128 frequency

slots this means that there are 15 users transmitting/receiving at any time. The probability of a hit is then given by equation 6.5 (for a system with 64 bits per hop):

$$\Pr\{\text{ hit }\} = 1 - \left(1 - \frac{1}{128}\left(1 + \frac{1}{64}\right)\right)^{14} = 0{,}106 \qquad (6.34)$$

The bit error probability has been plotted in figure 6.7 as a function of the signal-to-noise ratio for several values of the number of resolvable paths L. There we can see that for low values of L the bit error probability becomes equal to the probability of a hit.

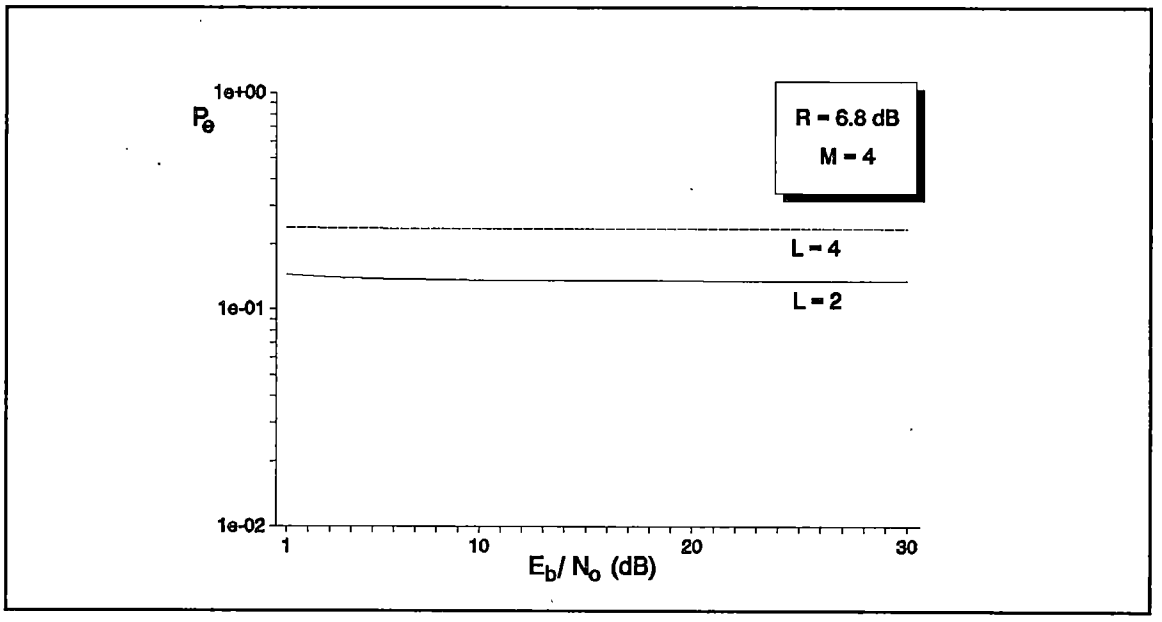

Figure 6.7 Average Bit Error Probability as a function of the signal-to-noise ratio for several values of the number of resolvable paths

6.4 Discussion of the Results

From the performance analysis performed in this chapter the following conclusions can be drawn:

- The average bit error probability decreases with increasing signal-to-noise ratios to an asymptote caused by the interference from the multiple resolvable paths.
- The performance increases with increasing order of diversity.
- For random hopping, the performance decreases with increasing number of simultaneously transmitting users.

The results obtained in this chapter show that the performance of a slow frequency hopping system in a discrete multipath channel is not very good. This can be explained by stating that the resolvable paths, other than the selected path, contain useful information that is marked as interference with this type of diversity. Better performance results can be expected when the transmitted signal power would not be equally distributed over the resolvable paths. So selection diversity would not be a preferred diversity technique if it would be possible to create a system with multiple resolvable paths for slow frequency hopping.

When we compare the results of the performance analysis of a slow frequency hopping system using DPSK modulation and selection diversity with the results of the performance analysis of a direct sequence system using DPSK modulation and selection diversity by Howard Sewberath Misser [28], it becomes clear that direct sequence has a far better performance than slow frequency hopping in a system where multiple users occupy the same frequencies. This can be explained because DS uses a form of data redundancy (a bit is transmitted in 127 or 255 coded parts). A better performance can be reached when fast frequency hopping is used, because another kind of data redundancy technique is, but unfortunately DPSK modulation can only be implemented in FFH with great difficulty.
Also the diversity technique of Selection Diversity is not suited for slow frequency hopping. A better technique will be discussed in the following chapter, where better performance results are expected.

7. SLOW FREQUENCY HOPPING WITH MAXIMAL RATIO COMBINING

Another diversity technique that can be used to overcome the problem to receive a radio signal in a fading environment and to increase the performance of the system is Maximal Ratio Combining (MRC). MRC is based on summing the demodulation results of a group of signals carrying the same information, and using the result as the decision variable.
This form of diversity is very attractive for spread spectrum with DPSK modulation, because it is very easily implemented and gives a significant improvement in performance.

We define the average bit error probability for slow frequency hopping with maximal ratio combining as:

$$\begin{aligned} P_e &\triangleq \Pr\{\text{bit error} \mid \text{no hit}\} \cdot \Pr\{\text{no hit}\} + \Pr\{\text{bit error} \mid \text{hit}\} \cdot \Pr\{\text{hit}\} \\ &= \Pr\left\{\begin{matrix}\text{bit error due to interference} \\ \text{from multiple resolvable paths}\end{matrix} \,\middle|\, \text{no hit}\right\} \cdot \Pr\{\text{no hit}\} + \Pr\{\text{hit}\} \end{aligned} \tag{7.1}$$

where we have assumed that all bits are destroyed when a hit occurs.
The probability of a hit has been discussed in chapter 6 and in this chapter we shall again look at the cases of deterministic hopping patterns and random hopping patterns.

7.1 Bit Error Probability due Interference from Multiple Resolvable Paths

For binary DPSK the combiner circuit will give as an output the decision variable, as in equation 5.19-5.21:

$$\xi = \mathrm{Re}\left[\sum_{l=1}^{M} V_{0l} V_{-1l}^{*}\right] = \sum_{l=1}^{M} \mathrm{Re}\left[V_{0l} V_{-1l}^{*}\right] = \sum_{l=1}^{M} \xi_l \tag{7.2}$$

Equation 7.2 is thus the sum of M decision variables. In literature this diversity technique is also called predetection combining [5.6] or equal gain combining [5.8] and can easily be implemented with a simple integrating circuit.

The bit error probability we investigate is the probability that ξ is less than zero and we again assume that the channel parameters remain constant over two successive signalling intervals.
This error probability is derived by using the error probability given in [5.7 eq. 4.4.13], which applies to binary DPSK modulated signals, transmitted over L, statistically independent, time-invariant channels and average it over the Rician fading channel statistics.

The error probability as a function of the signal-to-noise ratio per bit is given by:

$$P_e(\gamma_b) = \frac{1}{2^{2M-1}} \cdot \exp(-\gamma_b) \cdot \sum_{k=1}^{M-1} b_k \cdot \gamma_b^k \tag{7.3}$$

where γ_b is the signal-to-noise ratio per bit:

$$\gamma_b = \frac{E_b}{N_o} \cdot \sum_{k=1}^{M} \beta_k^2 \tag{7.4}$$

where β_k is the path gain of the k*th* combined path and

$$b_k = \frac{1}{k!} \cdot \sum_{n=0}^{M-1-k} \binom{2M-1}{n} \tag{7.5}$$

Since β is Rician distributed, β^2 has a non-central chi-square distribution with 2M degrees of freedom [5.7 eq. 1.1.115]. Consequently, the distribution for γ_b is given by the non-central chi-square pdf with 2M degrees of freedom and by making use of equation B.15 we can write the pdf of γ_b as:

$$p(\gamma_b) = \frac{1}{2\sigma^2 E_b/N_o} \cdot \left(\frac{\gamma_b}{E_b/N_0 \, S_M^2} \right)^{\frac{M-1}{2}} \cdot \exp\left(- \frac{S_M^2 + \gamma_b N_o/E_b}{2\sigma^2} \right) \cdot I_{M-1}\left(\sqrt{\frac{\gamma_b}{E_b/N_o}} \cdot \frac{S_M}{\sigma^2} \right) \tag{7.6}$$

where I_{M-1} is the (M-1)*th*-order modified Bessel function of the first kind and S_M is the non-centrality parameter, which is defined as:

$$S_M^2 = \sum_{i=1}^{M} m_i^2 \tag{7.7}$$

where m_i is the mean of the Rician distributed variables, so the value of S_M is given by

$$S_M = M \cdot (2\sigma^2 + s^2) = 2 \cdot M \cdot P \tag{7.8}$$

The expression for the bit error probability for maximal ratio combining is now obtained by averaging equation 7.3 over equation 7.6

$$P_e = \int_0^\infty P_e(\gamma_b) \cdot p(\gamma_b)\, d\gamma_b \tag{7.9}$$

7.2 Computational Results

In this section, we investigate the performance for the two types of hopping techniques, discussed in section 6.1, deterministic hopping patterns and random hopping patterns.
As in the previous chapter, we normalise the signal power in order to obtain more general results.

7.2.1 Deterministic Hopping Patterns

The influence of the order of diversity on the performance is investigated in figures 7.1 and 7.2. These figures show that maximal ratio combining improves the performance significantly when compared to the non-diversity case (M = 1). When we compare these two figures, it becomes clear that an increase of the Rician parameter also improves the performance.

In figure 7.3 the average bit error probability is plotted for the Rician channel with R = 6.8 dB and 11 dB. Also the results for a Rayleigh fading channel are shown. The Rician parameter is the ratio of the peak power and the power received over specular paths. When R = 0, there is no constant path and the Rician distribution becomes the Rayleigh distribution.
The probability density function $p(\gamma_b)$ is for a Rayleigh fading channel is given by the chi-square distribution with 2M degrees of freedom:

$$p(\gamma_b) = \frac{1}{(2\sigma^2)^M\,(M-1)!\,E_b/N_o} \cdot \left(\frac{\gamma_b}{E_b/N_o} \right)^{M-1} \cdot \exp\left(- \frac{\gamma_b}{2\sigma^2\,E_b/N_o} \right) \tag{7.10}$$

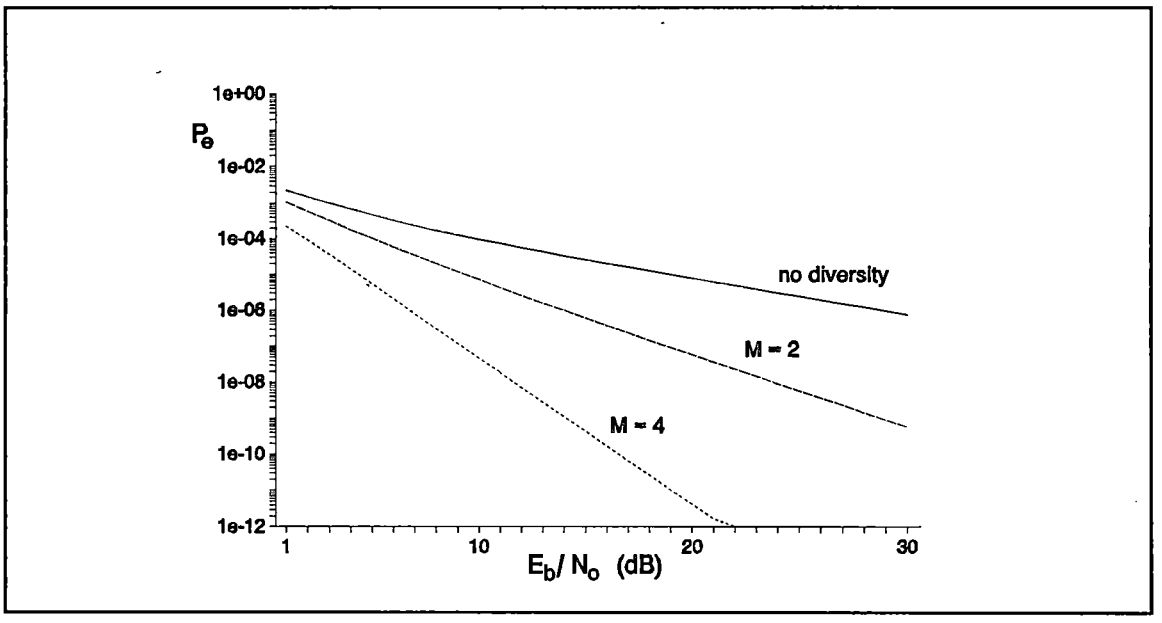

Figure 7.1 Average bit error probability as a function of the signal-to-noise ratio for several values of the order of diversity M with R = 6.8 dB

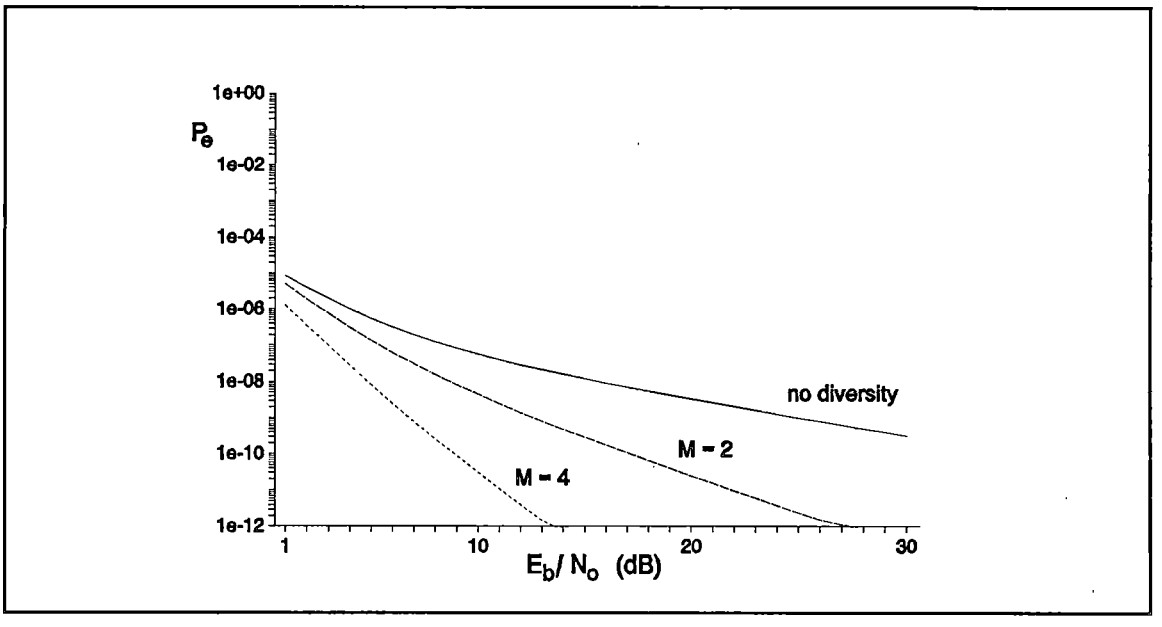

Figure 7.2 Average bit error probability as a function of the signal-to-noise ratio for several values of the order of diversity M with R = 11 dB

From [6] we know that R = 6.8 dB typically corresponds to a 30-year-old brick building with reinforced concrete and plaster, as well as some ceramic block interior partitions. R = 11 dB corresponds to a building that has the same construction, but has an open-office interior floor plan, and non-metallic ceiling tiles.

When we compare the Rician and the Rayleigh fading channels, it is clear that the Rician channel offers a better performance than the Rayleigh fading channel.

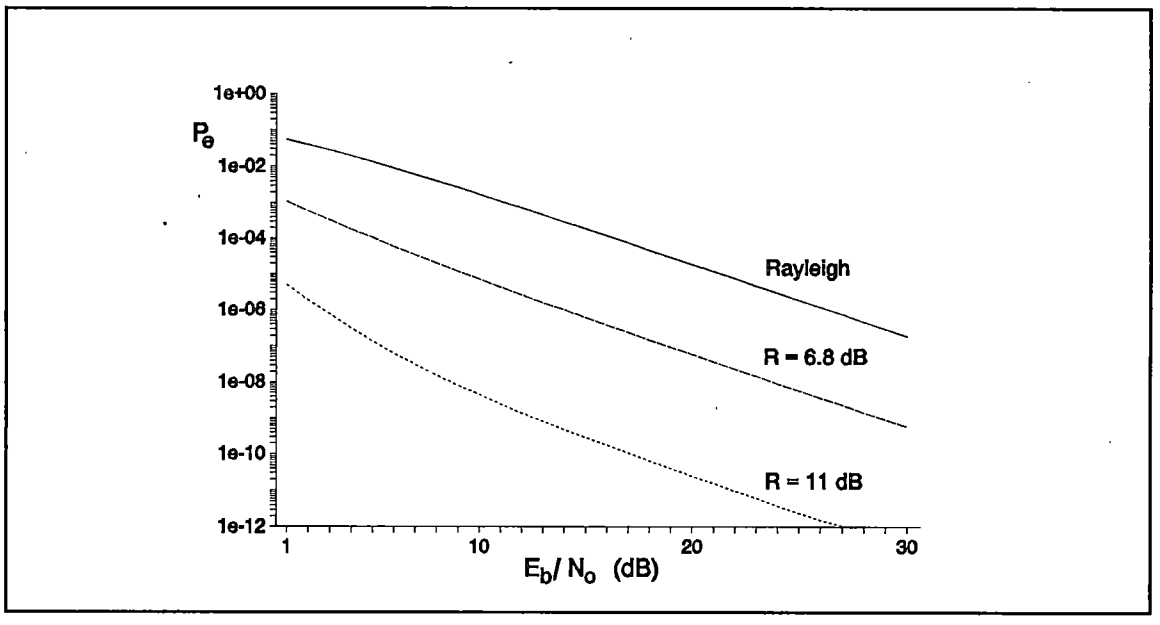

Figure 7.3 Average bit error probability as a function of the signal-to-noise ratio for a Rician channel with R = 6.8 dB and R = 11 dB, and for a Rayleigh channel, M = 2

7.2.2 Random Hopping Patterns

In figure 7.4 the influence of the order of diversity is investigated for a system that uses random hopping patterns. It can be seen that only for very small signal-to-noise ratios a higher order of diversity gives a marginal better performance. For higher signal-to-noise ratios, the curves approach the lower boundary caused by the probability of a hit.

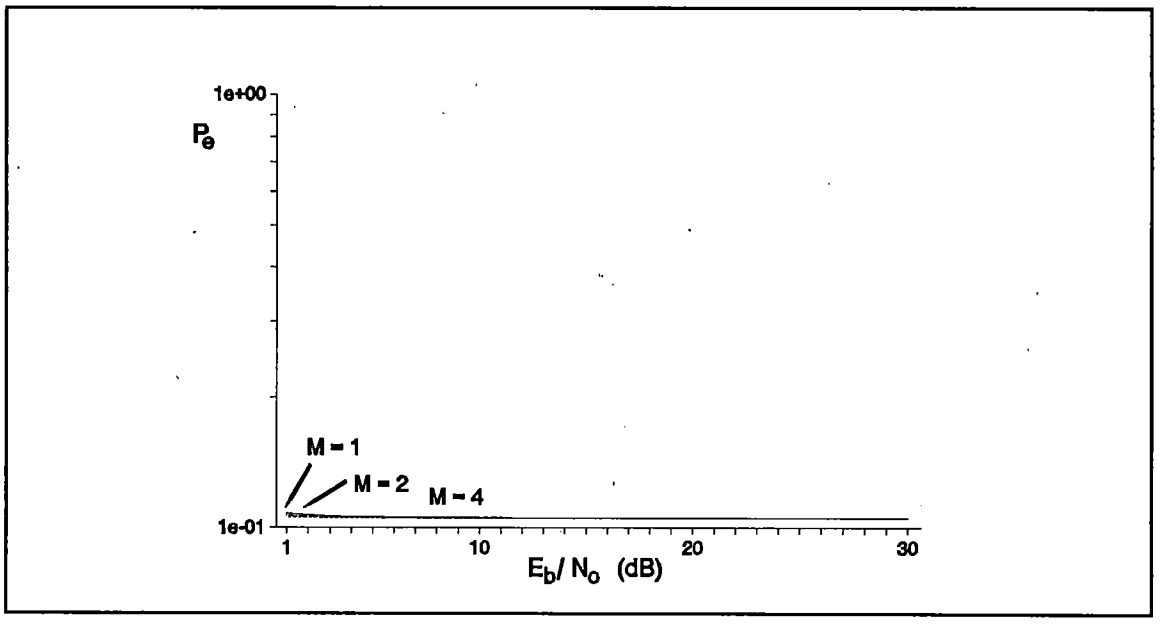

Figure 7.4 Average bit error probability as a function of the signal-to-noise ratio for several values of the order of diversity M with R = 6.8 dB

7.3 Discussion of the Results

From the performance analysis performed in this chapter the following conclusions can be drawn:

- The average bit error probability decreases with increasing signal-to-noise ratios to an asymptote caused by the interference from the multiple resolvable paths.
- The performance increases with increasing order of diversity.
- A larger Rician parameter gives an improved performance and the Rician channel is superior to the Rayleigh channel.
- The performance decreases dramatically when users are allowed to occupy the same frequency slot at the same time.

In this chapter we have shown that maximal ratio combining improves the performance when compared with no diversity, and that it gives a far better performance than selection diversity. Also it is shown that for random hopping patterns the probability of a hit dominates the average bit error probability.

8. HYBRID SFH/DS WITH DPSK MODULATION

Hybrid systems are attractive because they can combine the advantages of both DS and FH systems while avoiding some of their disadvantages. A hybrid system can combine the antimultipath effectiveness of DS systems with the good antipartial-band-jamming and the good anti-near/far problem features of FH systems. Hybrid systems may also use shorter signature sequences and hopping patterns, thus reducing the overall acquisition time. A disadvantage of hybrid systems is the increased complexity of their transmitters and receivers.

There a two ways of looking at the hybrid SFH/DS system:

- From the viewpoint of slow frequency hopping, the hybrid system is a manner to create a form of coded data redundancy in order to gain a better performance when confronted with multi-user interference.
- From the viewpoint of direct sequence, the hybrid system is a manner to diminish the "near-far" problem and to increase the system capacity in terms of maximum number of users.

In this chapter we define the transmitter, channel and receiver models respectively.

8.1 Transmitter Model

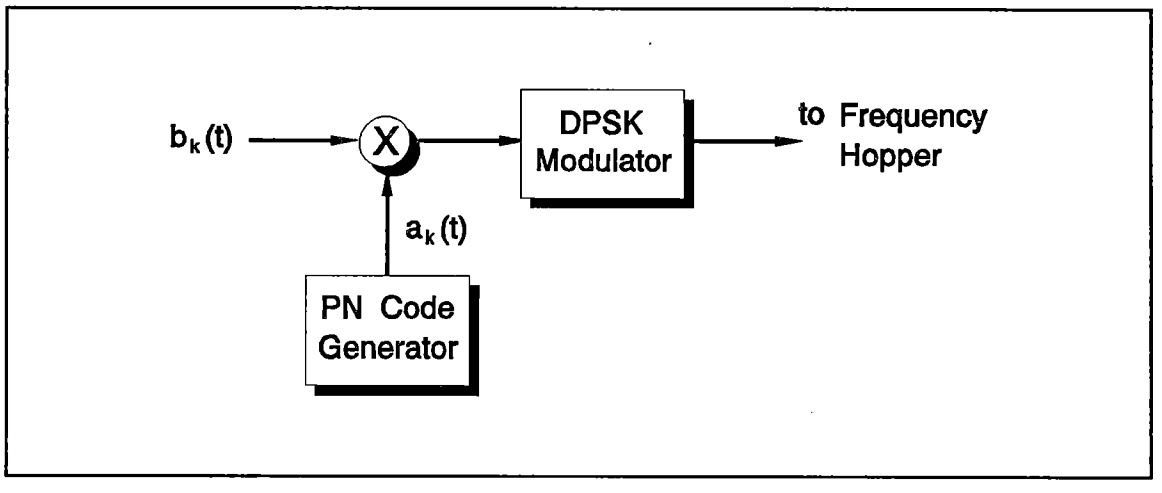

Figure 8.1 Transmitter Model

The transmitter for the slow frequency hopped/direct sequence spread spectrum signal with DPSK modulation, as shown in figure 8.1, is modelled as in [22]. There are K such transmitters in the spread spectrum multiple access system. The data waveform is written as:

$$b_k(t) = \sum_j b_k^j P_{T_b}(t - jT_b) \qquad b_k^j \in \{0,1\} \tag{8.1}$$

where b_k^j is the *jth* data bit of user k and P_{T_b} is a rectangular pulse of unit height and duration T_b. The signal bandwidth is first spread as shown in figure 8.2; every user has a unique spread spectrum code, in our case, a Gold code, with fixed length:

$$a_k(t) = \sum_i a_k^i P_{T_c}(t - iT_c) \qquad a_k^i \in \{-1,1\} \tag{8.2}$$

where P_{T_c} is a rectangular pulse of unit height and duration T_c. We assume that each user-code has length N and that $T_b/T_c = N$, so that one code sequence fits in one data bit interval. The Gold code has the nice feature of a low cross-correlation and a high auto-correlation peak (for more information refer to [4] and [28, appendix D])

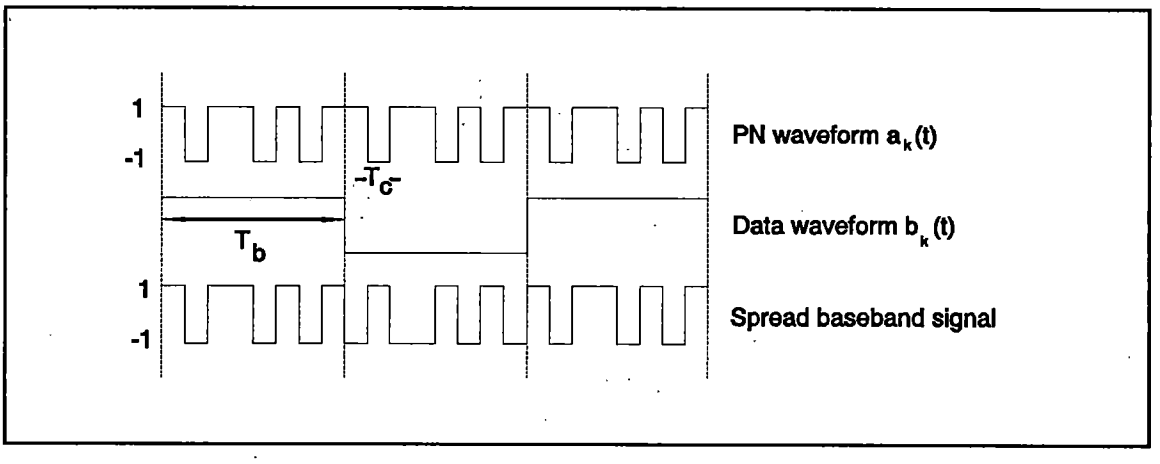

Figure 8.2 Direct Sequence Band Spreading

DPSK modulation of the resulting waveform gives

$$c_k(t) = \sum_j \cos(2\pi f_c t + \theta_k + \phi_k^j)\, P_{T_c}(t - jT_c) \qquad \phi \in \{0, \pi\} \tag{8.3}$$

where f_c is the carrier frequency, which is the same for all the users and $\theta_k(t)$ is the phase angle introduced by the *kth* modulator. Because the phase shift of π just reverses the sign of the cosine, equation 8.3 can be written as:

$$c_k(t) = a_k(t)\, b_k(t) \cos(2\pi f_c t + \theta_k) \tag{8.4}$$

This expression is a collection of user code sequences with a sign determined by the sign of $b_k(t)$. The DPSK signal is then frequency hopped according to the *kth* hopping pattern $f_k(t)$:

$$c_k^*(t) = a_k(t)\, b_k(t) \cos(2\pi f_c t + \theta_k) \cos(2\pi f_k(t) t + \alpha_k(t)) \tag{8.5}$$

The bandpass filter in the frequency hopper, shown in figure 5.2, removes unwanted frequency components present at the output of the multiplier. The signal at the output of the filter is

$$s_k(t) = \sqrt{2P}\, a_k(t)\, b_k(t) \cos\{2\pi[f_c + f_k(t)]t + \theta_k + \alpha_k(t)\} \tag{8.6}$$

where P is the transmitted power of the *kth* signal. The signal $\alpha_k(t)$ represents the phase shift introduced by the frequency hopper as it switches from one frequency to another. We assume that $\alpha_k(t)$ is constant during the time intervals that $f_k(t)$ is constant.

8.2 Channel Model

We model the channel like in [23] and [24] as a discrete multipath slow fading channel, as shown in figure 8.3. This means that there are L independently fading multiple signal paths links between each user and the receiver antenna at the base station.

The criterion for the existence of multiple resolvable paths, as derived in chapter 3, is that the signal bandwidth W, is much larger than the coherence bandwidth of the channel ($W \gg (\Delta f)_c$) and that two signals must be separated by one chip time, T_c, in order to resolved.

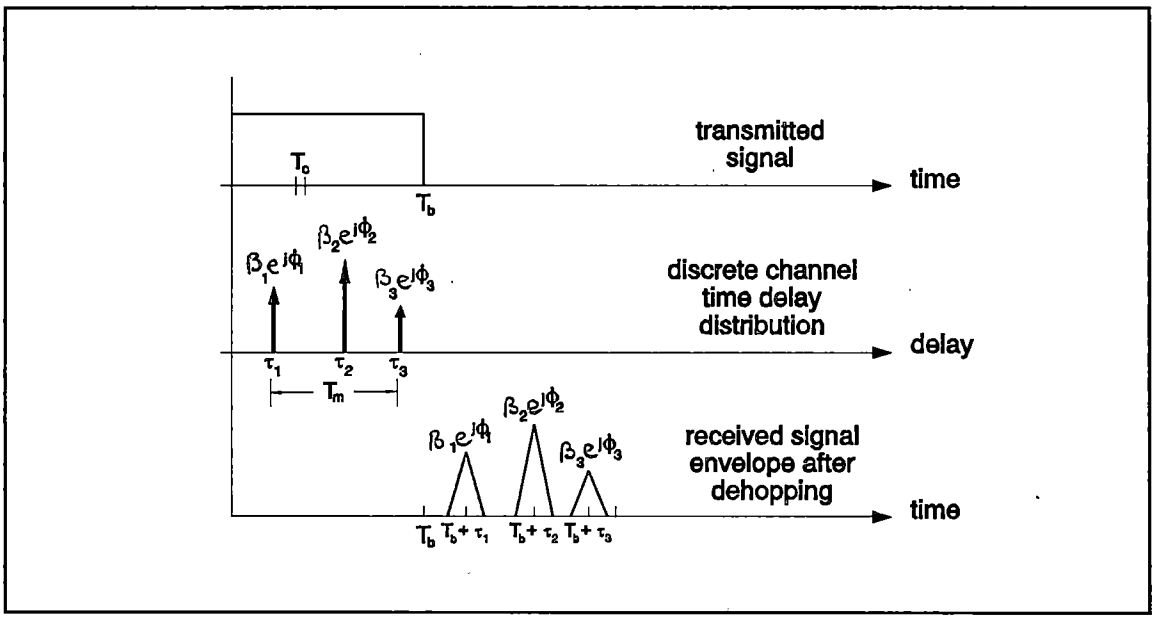

Figure 8.3 The Discrete Multipath Channel Model

The number of resolvable paths is given by [25, p.720] as:

$$L = T_m W \tag{8.7}$$

where T_m is the multipath delay spread. The hybrid SFH/DS signal bandwidth is defined as:

$$W = \frac{1}{T_c} \tag{8.8}$$

Now we can write equation 8.8 in terms of chip duration and multipath delay spread

$$L = \left\lfloor \frac{T_m}{T_c} \right\rfloor + 1 \tag{8.9}$$

We again model the multipath fading by means of the path gain β, as discussed in section 5.2, and we shall investigate the influence of the Rayleigh and Rician channel statistics on the performance of our system.

8.3 Receiver Model

The hybrid SFH/DS receiver model we describe in this section consists of two parts: the frequency dehopper and the DPSK demodulator. We again choose user 1 as the reference user and assume that the receiver is capable of acquiring the frequency hopping pattern, the direct sequence code, and time synchronisation with the signal of user 1. The first part of the receiver, the frequency dehopper, has already been shown in figure 5.4.

The received signal at the antenna for a certain user is the time convolution of equation 8.6 and 5.6 of the transmitted signal and the channel impulse response:

$$r(t) = \sqrt{2P} \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl}\, a_k(t-\tau_{kl})\, b_k(t-\tau_{kl}) \cos\{2\pi[f_c+f_k(t-\tau_{kl})]t+\phi_{kl}\} + n(t) \qquad (8.10)$$

$$\text{with } \phi_{kl} = \theta_k + \alpha(t-\tau_{kl}) - 2\pi[f_c+f_k(t-\tau_{kl})]\tau_{kl} - \gamma_{kl}$$

where n(t) is the white Gaussian noise process with two-sided power spectral density $\frac{1}{2}N_0$ W/Hz.

The received signal is the input to a bandpass filter. This filter is followed by the dehopper which introduces a phase waveform $\delta_1(t)$ analogous to that introduced by the frequency hopper and is assumed constant during the hop interval. The dehopper is followed by a bandpass filter which removes the high frequency components. The output of the filter is given by

$$r_d(t) = \sqrt{\tfrac{1}{2}P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot \delta[f_k(t-\tau_{kl}), f_1(t-\tau_{kl})] \cdot a_k(t-\tau_{kl}) \cdot b_k(t-\tau_{kl}) \cdot \qquad (8.11)$$

$$\cos\{2\pi f_c t + \Phi_{kl}(t)\} + \hat{n}(t) \qquad \text{with } \Phi_{kl}(t) = \phi_{kl}(t) + \delta_1(t)$$

where $\hat{n}(t)$ is the white Gaussian noise process with two-sided power spectral density $\frac{1}{8}N_0$ W/Hz and we consider $\delta(u,v)=0$ for $u \neq v$ and $\delta(u,v)=1$ for $u=v$ (both u and v are real).
For simplicity reasons we assume that Φ_{kl} is an independent random variable uniformly distributed in $[0,2\pi]$.

Equation 8.11 can be written in the lowpass equivalent form:

$$r_d(t) = x(t)\cdot\cos(2\pi f_c t) + y(t)\cdot\sin(2\pi f_c t) + \hat{n}(t) \qquad (8.12)$$

with

$$\hat{n}(t) = n_c(t)\cdot\cos(2\pi f_c t) + n_s(t)\cdot\sin(2\pi f_c t) \tag{8.13}$$

$$x(t) = \sqrt{½P}\cdot\sum_{k=1}^{K}\sum_{l=1}^{L}\beta_{kl}\cdot a_k(t-\tau_{kl})\cdot b_k(t-\tau_{kl})\cdot\cos(\Phi_{kl}) + n_c(t) \tag{8.14}$$

$$y(t) = \sqrt{½P}\cdot\sum_{k=1}^{K}\sum_{l=1}^{L}\beta_{kl}\cdot a_k(t-\tau_{kl})\cdot b_k(t-\tau_{kl})\cdot\sin(\Phi_{kl}) + n_s(t) \tag{8.15}$$

where x(t) and $n_c(t)$ are the in-phase components and y(t) and $n_s(t)$ the quadrature components.

As in chapter 5 we can express the decision variable ξ as:

$$\xi = X_{V_0} X_{V_{-1}} + Y_{V_0} Y_{V_{-1}} \tag{8.16}$$

The de-hopped output $r_d(t)$ is multiplied by either $a_k(t) \cdot \cos(2\pi f_c t)$ or $a_k(t) \cdot \sin(2\pi f_c t)$ and the product is integrated and sampled periodically at the end of each T-second signalling interval. The output of the integrator is reset to zero after sampling.
The integrator acts as a lowpass filter, thus rejecting the double frequency components resulting from the multiplication (The timing signal for sampling the output of the integrator is derived from the received signal).

At time t we can write for the outputs of the integrate-and-dump filters at the top half of figure 8.4:

$$X_{V_0} = \sqrt{⅛P}\cdot\sum_{k=1}^{K}\sum_{l=1}^{L}\beta_{kl}\cdot\cos(\Phi_{kl})\cdot\int_0^t a_1(s-t)\cdot a_k(t-\tau_{kl})\cdot b_k(s-\tau_{kl})\,ds + \int_0^t a_1(s-t)\cdot n_c(s)\,ds \tag{8.17}$$

and

$$Y_{V_0} = \sqrt{⅛P}\cdot\sum_{k=1}^{K}\sum_{l=1}^{L}\beta_{kl}\cdot\sin(\Phi_{kl})\cdot\int_0^t a_1(s-t)\cdot a_k(s-\tau_{kl})\cdot b_k(s-\tau_{kl})\,ds + \int_0^t a_1(s-t)\cdot n_s(s)\,ds \tag{8.18}$$

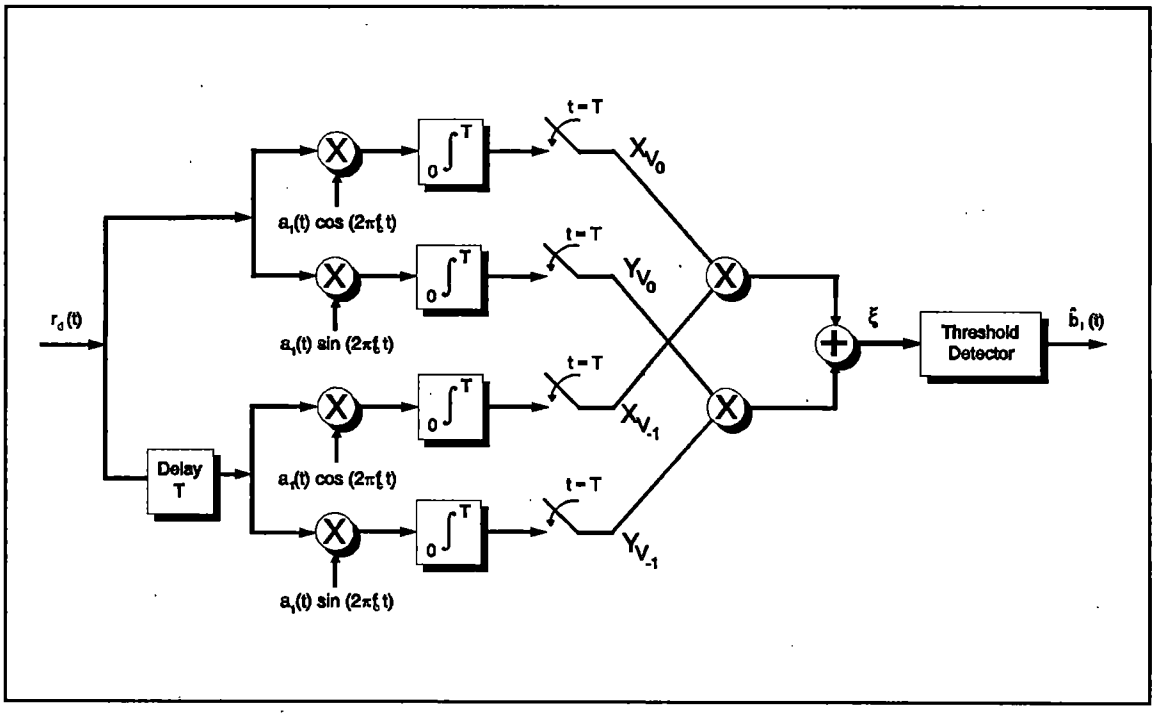

Figure 8.4 DPSK Demodulator for SFH/DS

8.3.1 Interference due to Multiple Resolvable Paths

We assume that the receiver can synchronise to an arbitrary path at t=0, then at $t=T=T_b$ the filter output is sampled.

The *jth*, the first and the last path signals of the reference user are shown in figure 5.6 during a hopping interval. The L-1 other path signals are at the same frequency as the *jth* path, so each bit of the *jth* path signal is fully hit L-1 times.

The contribution from the *ith* path signal, assuming that $\tau_{1i} > \tau_{1j}$, is incorporated into the expression for the signals at the sampling points $T=T_b$:

$$X_{V_0} = \sqrt{\tfrac{1}{8}P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot \cos(\Phi_{kl}) \cdot \left[b_k^{-1} R_{k1}(\tau_{kl}) + b_k^0 \hat{R}_{k1}(\tau_{kl}) \right] + \eta \qquad (8.19)$$

and

$$Y_{V_0} = \sqrt{\tfrac{1}{8}P} \cdot \sum_{k=1}^{K} \sum_{l=1}^{L} \beta_{kl} \cdot \sin(\Phi_{kl}) \cdot \left[b_k^{-1} R_{k1}(\tau_{kl}) + b_k^0 \hat{R}_{k1}(\tau_{kl}) \right] + \nu \qquad (8.20)$$

where b_k^{-1} and b_k^0 are the previous and current data bit respectively, and

$$R_{k1}(\tau) = \int_0^{\tau} a_k(t-\tau)\cdot a_1(t)\,dt \tag{8.21}$$

$$\hat{R}_{k1}(\tau) = \int_{\tau}^{T} a_k(t-\tau)\cdot a_1(t)\,dt \tag{8.22}$$

$$\eta = \int_0^T a_1(s)\cdot n_c(s)\,ds \quad , \quad \nu = \int_0^T a_1(s)\cdot n_s(s)\,ds \tag{8.23}$$

We now assume that the *jth* path between the transmitter of user 1 and the corresponding receiver is the reference path and all other paths constitute interference, so that without loss of generality, $\tau_{1j}=0$ and $\Phi_{1j}=0$.

Since the fading is slow, we may also assume that the delayed vector V_{-1} differs from V_0 only in the data bits involved, and in the additive Gaussian noise samples.

$$\begin{aligned} V_0 &= X_{V_0} + jY_{V_0} \\ &= \sqrt{\tfrac{1}{8}P}\cdot\beta_{1j}\cdot T\cdot b_1^0 + \sqrt{\tfrac{1}{8}P}\cdot\sum_{k=1}^{K}\left(b_k^{-1}X_k + b_k^0\hat{X}_k\right) \\ &\quad + j\sqrt{\tfrac{1}{8}P}\cdot\sum_{k=1}^{K}\left(b_k^{-1}Y_k + b_k^0\hat{Y}_k\right) + (\eta_1 + j\nu_1) \end{aligned} \tag{8.24}$$

where

$$\begin{aligned} X_1 &= \sum_{\substack{l=1\\ l\neq j}}^{L} R_{11}(\tau_{1l})\,\beta_{1l}\cos(\Phi_{1l}) \quad , Y_1 = \sum_{\substack{l=1\\ l\neq j}}^{L} R_{11}(\tau_{1l})\,\beta_{1l}\sin(\Phi_{1l}) \\ \hat{X}_1 &= \sum_{\substack{l=1\\ l\neq j}}^{L} \hat{R}_{11}(\tau_{1l})\,\beta_{1l}\cos(\Phi_{1l}) \quad , \hat{Y}_1 = \sum_{\substack{l=1\\ l\neq j}}^{L} \hat{R}_{11}(\tau_{1l})\,\beta_{1l}\sin(\Phi_{1l}) \end{aligned} \tag{8.25}$$

and for $k \geq 2$

$$\begin{aligned} X_k &= \sum_{l=1}^{L} R_{k1}(\tau_{kl})\,\beta_{kl}\cos(\Phi_{kl}) \quad , Y_k = \sum_{l=1}^{L} R_{k1}(\tau_{kl})\,\beta_{kl}\sin(\Phi_{kl}) \\ \hat{X}_k &= \sum_{l=1}^{L} \hat{R}_{k1}(\tau_{kl})\,\beta_{kl}\cos(\Phi_{kl}) \quad , \hat{Y}_k = \sum_{l=1}^{L} \hat{R}_{k1}(\tau_{kl})\,\beta_{kl}\sin(\Phi_{kl}) \end{aligned} \tag{8.26}$$

9. HYBRID SFH/DS WITH SELECTION DIVERSITY

As discussed in chapter 5, selection diversity is based on selecting the largest of a group of signals carrying the same information. The multiple resolvable paths can be used for selection diversity by selecting the path with the largest output of the matched filter. With selection diversity of order M, the decision variable is

$$\xi_{max}^{M} = \max_{l=1,...,M} (\xi_l) \tag{9.1}$$

where $\{\xi_i, i=1,...,M\}$ are identically distributed independent stochastic variables.

The order of diversity that can be achieved with selection diversity is defined as:

$$M = L \times \text{Number of Antennas} \tag{9.2}$$

We define the (average) bit error probability for the hybrid SFH/DS system with selection diversity as the error probability due to multi-user interference and interference from the multiple resolvable paths of the reference user, other than the selected path.
We assume that the hopping pattern of the frequency hopper is deterministic and a constant number of simultaneously transmitting users occupy the same frequency slot.

9.1 Bit Error Probability due to Multi-User Interference and Interference from Multiple Resolvable Paths

Because the data bits are equiprobable, we can write the bit error probability as:

$$\begin{aligned} P_e &= \tfrac{1}{2} \Pr\{ \xi_{max}^{M} < 0 \mid b_1^0 b_1^{-1} = 1 \} + \tfrac{1}{2} \Pr\{ \xi_{max}^{M} > 0 \mid b_1^0 b_1^{-1} = -1 \} \\ &= \Pr\{ \xi_{max}^{M} < 0 \mid b_1^0 b_1^{-1} = 1 \} = \Pr\{ \xi_{max}^{M} > 0 \mid b_1^0 b_1^{-1} = -1 \} \end{aligned} \tag{9.3}$$

We derive this probability using the formula for the bit error probability for binary DPSK modulated signals, with multichannel reception in a time-invariant Rician fading channel, given in appendix 7A.4 and derived in appendix 4B of [25]. Because with selection diversity we only use the strongest path and all other signals are seen as interference, we may use this equation in the case of L=1, i.e., for a single path:

$$P_{e|\beta_{max},\{\tau_{kl}\},L} = Q(a,b) - \tfrac{1}{2}\left[1 + \frac{\mu}{\sqrt{\mu_0 \mu_{-1}}}\right] \cdot \exp\left(-\frac{a^2+b^2}{2}\right) \cdot I_0(ab) \qquad (9.4)$$

where $I_0(ab)$ is the modified Bessel function of the first kind and zero order which is defined as:

$$I_0(ab) = \frac{1}{2\pi}\int_0^{2\pi} \exp(ab\cos\theta)\, d\theta \qquad (9.5)$$

and Q(a,b) is the Marcum Q-function which is defined as:

$$Q(a,b) = \int_b^{\infty} x \cdot \exp\left(-\frac{a^2+x^2}{2}\right) \cdot I_0(ax)\, dx \qquad (9.6)$$

where a and b are:

$$a = \frac{m}{\sqrt{2}}\left|\frac{1}{\sqrt{\mu_0}} - \frac{1}{\sqrt{\mu_{-1}}}\right| \quad , \quad b = \frac{m}{\sqrt{2}}\left[\frac{1}{\sqrt{\mu_0}} + \frac{1}{\sqrt{\mu_{-1}}}\right] \qquad (9.7)$$

and

$$m = E\left[V_0 \mid \beta_{max}, b_1^0\right] = E\left[V_{-1} \mid \beta_{max}, b_1^{-1}\right] \qquad (9.8)$$

$$\begin{aligned} \mu_0 &= \mathrm{var}\left(V_0 \mid \{\tau_{kl}\}, L\right) \\ \mu_{-1} &= \mathrm{var}\left(V_{-1} \mid \{\tau_{kl}\}, L\right) \\ \mu &= E\left[(V_0 - m)(V_{-1} - m)^* \mid \{\tau_{kl}\}, L\right] \end{aligned} \qquad (9.9)$$

9.1.1 Derivation of the statistical moments

m was defined in equation 9.8 as:

$$m = E\left[V_0 \mid \beta_{max}, b_1^0 \right] = E\left[V_{-1} \mid \beta_{max}, b_1^{-1} \right] \tag{9.10}$$

After inserting equation 8.22 we can rewrite this equation as:

$$m = \sqrt{\tfrac{1}{8}P}\,\beta_{max} T_b b_1^0 + \sqrt{\tfrac{1}{8}P}\cdot E\left[\sum_{k=1}^{K} \left(b_k^{-1} X_k + b_k^0 \hat{X}_k \right) + j\left(b_k^{-1} Y_k + b_k^0 \hat{Y}_k \right) \right] \tag{9.11}$$

We may interchange the expectation and the summation because the multiple resolvable paths are statistically independent and also because [27, p.107]:

$$z = x + jy \quad \Rightarrow \quad E[z] = E[x] + jE[y] \tag{9.12}$$

Equation 9.11 can be rewritten as:

$$\begin{aligned} &E\left[\sum_{k=1}^{K} \left(b_k^{-1} X_k + b_k^0 \hat{X}_k \right) + j\left(b_k^{-1} Y_k + b_k^0 \hat{Y}_k \right) \right] \\ &= \sum_{k=1}^{K} E\left[b_k^{-1} X_k + b_k^0 \hat{X}_k \right] + j\, E\left[b_k^{-1} Y_k + b_k^0 \hat{Y}_k \right] \end{aligned} \tag{9.13}$$

When we take a closer look at equation 9.13, we can see that

$$\begin{aligned} E\left[b_k^{-1} X_k + b_k^0 \hat{X}_k \right] &= E\left[\sum_{l=1}^{L} \left(b_k^{-1}\cdot R_{kl}(\tau_{kl}) + b_k^0 \cdot \hat{R}_{kl}(\tau_{kl}) \right) \cdot \beta_{kl} \cos(\Phi_{kl}) \right] \\ E\left[b_k^{-1} Y_k + b_k^0 \hat{Y}_k \right] &= E\left[\sum_{l=1}^{L} \left(b_k^{-1}\cdot R_{kl}(\tau_{kl}) + b_k^0 \cdot \hat{R}_{kl}(\tau_{kl}) \right) \cdot \beta_{kl} \sin(\Phi_{kl}) \right] \end{aligned} \tag{9.14}$$

In equation 9.14, the path delays $\{\tau_{kl}\}$ are given, so the partial correlation functions are constants. The Rician distributed path gains are multiplied by the cosine or sine of a uniformly distributed phase. As noted in chapter 5, the Rician pdf describes the envelope of the sum of two orthogonal non zero-mean Gaussian random variables with identical variance σ^2, so this multiplication extracts one of the non zero-mean Gaussian variables.

When we assume that the Gaussian variables which constitute the Rician variable have equal

means, the mean of the pdf is derived from equation 5.9 as $m_1 = m_2 = s/\sqrt{2}$. The pdf of this product is thus a non zero-mean Gaussian distribution, with mean $s/\sqrt{2}$ and variance σ^2:

$$p(y) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left[- \frac{\left(y - \frac{s}{\sqrt{2}} \right)^2}{2\sigma^2} \right] \tag{9.15}$$

This product is multiplied by the bit estimate $\hat{b}$ that has a value of 1 or -1 with probability ½. If we denote the product of the path gain and the cosine or sine of a uniformly distributed phase as x and the bit estimate as $\hat{b}$, then the pdf for $y = \hat{b}x$ can be written as [27, p.96]:

$$p_y(y) = \frac{1}{|\hat{b}|} \cdot p_x\left(\frac{y}{\hat{b}} \right) \tag{9.16}$$

Because

$$p(\hat{b}) = ½\,, \qquad \hat{b} \in \{ -1, 1 \} \tag{9.17}$$

we can write $p_y(y)$ as:

$$p_y(y) = \frac{1}{2\sqrt{2\pi}\,\sigma} \exp\left[- \frac{\left(y - \frac{s}{\sqrt{2}} \right)^2}{2\sigma^2} \right] + \frac{1}{2\sqrt{2\pi}\,\sigma} \exp\left[- \frac{\left(y + \frac{s}{\sqrt{2}} \right)^2}{2\sigma^2} \right] \tag{9.18}$$

where s is the constant component of the Rician fading channel.
It is clear from equation 9.18 that $p_y(y)$ is the sum of two Gaussian pdf's with opposite mean values. The mean value of y, E[y], is then 0.

Now we know that $E[X_{1l}] = E[Y_{1l}] = 0$, we conclude that

$$\begin{aligned} m &= E[\, V_0 \mid \beta_{max},\, b_1^0 \,] = E[\, V_{-1} \mid \beta_{max},\, b_1^{-1} \,] \\ &= \sqrt{⅛P} \cdot \beta_{max} \cdot T_b \cdot b_1^0 = \sqrt{⅛P} \cdot \beta_{max} \cdot T_b \cdot b_1^{-1} \end{aligned} \tag{9.19}$$

The statistical moments μ_0, μ_{-1} and μ, defined above have been calculated in [28, appendix B] and the results are given below.

$$\mu_0 = \tfrac{1}{8}P \cdot \sum_{k=1}^{K} E\left[X_k^2 + \hat{X}_k^2 + Y_k^2 + \hat{Y}_k^2 \mid \{\tau_{k1}\}, L \right] + \tfrac{1}{4}P \cdot E\left[X_1\hat{X}_1 + Y_1\hat{Y}_1 \mid \{\tau_{k1}\}, L \right] + 2\sigma_n^2 \tag{9.20}$$

$$\mu_{-1} = \tfrac{1}{8}P \cdot \sum_{k=1}^{K} E\left[X_k^2 + \hat{X}_k^2 + Y_k^2 + \hat{Y}_k^2 \mid \{\tau_{k1}\}, L \right] + 2\sigma_n^2 \tag{9.21}$$

$$\mu = \tfrac{1}{8}P \cdot \sum_{k=1}^{K} E\left[X_k\hat{X}_k + Y_k\hat{Y}_k \mid \{\tau_{k1}\}, L \right] + \tfrac{1}{8}P \cdot E\left[\hat{X}_1^2 + \hat{Y}_1^2 \mid \{\tau_{k1}\}, L \right] \tag{9.22}$$

The conditional expectations in the above equations have the following properties:

$$E\left[X_1^2 \mid \{\tau_{k1}\}, L \right] + E\left[Y_1^2 \mid \{\tau_{k1}\}, L \right] = \sum_{\substack{l=1 \\ l\neq j}}^{L} R_{11}^2(\tau_{1l}) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + \sum_{\substack{l=1 \\ l\neq j}}^{L} R_{11}(\tau_{1l}) \sum_{\substack{p=1 \\ p\neq l\neq j}}^{L} R_{11}(\tau_{1p}) \cdot \tfrac{1}{2}s^2 \tag{9.23}$$

$$E\left[\hat{X}_1^2 \mid \{\tau_{k1}\}, L \right] + E\left[\hat{Y}_1^2 \mid \{\tau_{k1}\}, L \right] = \sum_{\substack{l=1 \\ l\neq j}}^{L} \hat{R}_{11}^2(\tau_{1l}) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + \sum_{\substack{l=1 \\ l\neq j}}^{L} \hat{R}_{11}(\tau_{1l}) \sum_{\substack{p=1 \\ p\neq l\neq j}}^{L} \hat{R}_{11}(\tau_{1p}) \cdot \tfrac{1}{2}s^2 \tag{9.24}$$

$$E\left[X_1\hat{X}_1 \mid \{\tau_{k1}\}, L \right] + E\left[Y_1\hat{Y}_1 \mid \{\tau_{k1}\}, L \right] = \sum_{\substack{l=1 \\ l\neq j}}^{L} R_{11}(\tau_{1l})\,\hat{R}_{11}(\tau_{1l}) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + \sum_{\substack{l=1 \\ l\neq j}}^{L} R_{11}(\tau_{1l}) \sum_{\substack{p=1 \\ p\neq l\neq j}}^{L} \hat{R}_{11}(\tau_{1p}) \cdot \tfrac{1}{2}s^2 \tag{9.25}$$

and for $K \geq 2$:

$$E\left[X_k^2 \mid \{\tau_{k1}\}, L \right] + E\left[Y_k^2 \mid \{\tau_{k1}\}, L \right] = \sum_{l=1}^{L} R_{k1}^2(\tau_{kl}) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + \sum_{l=1}^{L} R_{k1}(\tau_{kl}) \sum_{\substack{p=1 \\ p\neq l}}^{L} R_{k1}(\tau_{kp}) \cdot \tfrac{1}{2}s^2 \tag{9.26}$$

$$E\left[\hat{X}_k^2 \mid \{\tau_{k1}\}, L\right] + E\left[\hat{Y}_k^2 \mid \{\tau_{k1}\}, L\right] = \sum_{l=1}^{L} \hat{R}_{k1}^2(\tau_{kl}) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + \sum_{l=1}^{L} \hat{R}_{k1}(\tau_{kl}) \sum_{\substack{p=1 \\ p \neq l}}^{L} \hat{R}_{k1}(\tau_{kp}) \cdot \tfrac{1}{2}s^2 \tag{9.27}$$

$$E\left[X_k \hat{X}_k \mid \{\tau_{k1}\}, L\right] + E\left[Y_k \hat{Y}_k \mid \{\tau_{k1}\}, L\right] = \sum_{l=1}^{L} R_{k1}(\tau_{kl})\, \hat{R}_{k1}(\tau_{kl}) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + \sum_{l=1}^{L} R_{k1}(\tau_{kl}) \sum_{\substack{p=1 \\ p \neq l}}^{L} \hat{R}_{k1}(\tau_{kp}) \cdot \tfrac{1}{2}s^2 \tag{9.28}$$

Selection diversity can be accomplished by letting the reference path β_{1j} be the strongest path, i.e., $\beta_{1j} = \beta_{max}$, and since the fading is slow this path will also be the strongest path for the next bit, so we have maximised the decision variable in equation 9.1.

In order to remove the conditioning on β_{max}, we must average the conditional error probability given in equation 9.4 over the Rician fading channel statistics. The pdf of the maximum path gain, will be derived in the following section.

9.1.2 The Probability Density Function of the Path Gain

Selection diversity is based on selecting the strongest signal from several statistically independent signals carrying the same data. In our case these signals are usually the multiple resolvable paths due the inherent diversity of spread spectrum, but these signals can also be obtained from different antennas, in order to increase the order of diversity.

Suppose we have M identically distributed random variables $\{ \beta_1, ..., \beta_M \}$. For the largest random variable, β_{max}, the following inequality holds:

$$\beta_{max} > \beta_l \qquad l \in \{ 1,2,...,M \} \tag{9.29}$$

The probability that $\beta_{max} < y$, i.e., $P_{\beta_{max}}(y)$ is now given by:

$$\begin{aligned} P_{\beta_{max}}(y) &= Pr\{ \beta_1 < y \} \cdot Pr\{ \beta_2 < y \} \cdot .... \cdot Pr\{ \beta_M < y \} \\ &= P_{\beta_1}(y) \cdot P_{\beta_2}(y) \cdot .... \cdot P_{\beta_M}(y) \end{aligned} \tag{9.30}$$

This means that the cdf of β_{max} is the product of the cdf's of the M path gains.
Since β is Rician distributed, the cdf of β is defined as:

$$P_\beta(\beta) = \int_0^\beta \frac{z}{\sigma^2} \cdot \exp\left(- \frac{s^2 + z^2}{2\sigma^2} \right) \cdot I_0\left(\frac{sz}{\sigma^2} \right) dz \tag{9.31}$$

The cdf of β_{max} is now given by:

$$P_{\beta_{max}}(\beta) = [\, P_\beta(\beta) \,]^M \tag{9.32}$$

The pdf of β_{max} can be obtained by calculating the derivative with respect to β:

$$p_{\beta_{max}}(\beta) = M [\, P_\beta(\beta) \,]^{M-1} \cdot \frac{dP_\beta(\beta)}{d\beta} \tag{9.33}$$

The derivative of P_β is just a Rician distribution, so:

$$\begin{aligned} p_{\beta_{max}}(\beta) = M \left[\int_0^\beta \frac{z}{\sigma^2} \cdot \exp\left(- \frac{s^2 + z^2}{2\sigma^2} \right) \cdot I_0\left(\frac{zs}{\sigma^2} \right) dz \right]^{M-1} \\ \cdot \frac{\beta}{\sigma^2} \cdot \exp\left(- \frac{s^2 + \beta^2}{2\sigma^2} \right) \cdot I_0\left(\frac{\beta s}{\sigma^2} \right) \end{aligned} \tag{9.34}$$

9.2 Expression for the Bit Error Probability

Now that we have removed the conditioning on β_{max}, we have to remove the conditioning on $\{\tau_{kl}\}$. This involves solving complicated integrals, but we can simplify our calculations by neglecting the term $\frac{1}{4}P \cdot E [\, X_1\hat{X}_1 + Y_1\hat{Y}_1 \mid \{\tau_{kl}\}, L \,]$ in equation 9.20. The contribution of this term to μ_0 is relatively small when K is large [24].
This simplification yields $\mu_0 = \mu_{-1}$, which results in a = 0 in equation 9.4. The average bit error probability then becomes:

$$\begin{aligned} P_{e|\beta_{max},\{\tau_{kl}\},L} &= Q(0,b) - \tfrac{1}{2}\left[1 + \frac{\mu}{\mu_0} \right] \cdot \exp\left(- \frac{b^2}{2} \right) \cdot I_0(0) \\ &= \tfrac{1}{2}\left[1 - \frac{\mu}{\mu_0} \right] \cdot \exp\left(- \frac{m^2}{\mu_0} \right) \end{aligned} \tag{9.35}$$

because $I_0(0) = 1$ and $Q(0,b) = \exp(-\frac{1}{2}b^2)$.

The conditioning on $\{\tau_{kl}\}$ is now removed by integrating over all (K×L)-1 pdf's of the path delays. τ_{1l} is excluded because it is known due to the selection of the strongest path.
The integration over all path delays is very time consuming. Instead we can utilise the central limit theorem, and assume that μ_0 and μ are Gaussian distributed random variables if K×L is sufficiently large.

The following expression for the average bit error probability is then found:

$$P_e = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{0}^{\infty} \frac{1}{2}\left[1 - \frac{\mu}{\mu_0}\right]\cdot\exp\left(-\frac{\frac{1}{8}P\beta_{max}^2 T_b^2}{\mu_0}\right)\cdot p_{\beta_{max}}(\beta)\cdot p_{\mu}(\mu)\cdot p_{\mu_0}(\mu_0)\, d\beta\, d\mu\, d\mu_0 \quad (9.36)$$

where p_μ and $p_{\mu o}$ are Gaussian pdf's.

We can solve the integral over μ analytically because

$$\int_{-\infty}^{\infty}\left[1 - \frac{\mu}{\mu_0}\right]\cdot p_{\mu}(\mu)\, d\mu = 1 - \frac{E_\tau[\mu]}{\mu_0} \quad (9.37)$$

So in order to determine the pdfs of μ_0 and μ, we have to calculate the mean and the variance of μ_0 and the mean of μ with respect to the path delays.

9.3 Gaussian Approximation

In this section we shall derive the means and variation needed for the gaussian approximation of the multi-user interference. For convenience we drop the condition of $l \neq j$ in equations 9.23-9.25. This means that user one has L instead of L-1 self-interference paths. The effect of this approximation is negligible if K×L is large.

First we write the partial correlation functions $R_{k1}(\tau)$ and $\hat{R}_{k1}(\tau)$ as:

$$\begin{aligned} R_{k1}(\tau) &= C_{k1}(n-N)\cdot T_c + \left(C_{k1}(n+1-N) - C_{k1}(n-N) \right)\cdot(\tau - nT_c) \\ &= A_{k1n}\cdot T_c + B_{k1n}\cdot(\tau - nT_c) \\ \hat{R}_{k1}(\tau) &= C_{k1}(n)\cdot T_c + \left(C_{k1}(n+1) - C_{k1}(n) \right)\cdot(\tau - nT_c) \\ &= \hat{A}_{k1n}\cdot T_c + \hat{B}_{k1n}\cdot(\tau - nT_c) \end{aligned} \tag{9.38}$$

where $0 \le nT_c \le (n+1)T_c$ and C_{k1} is the discrete aperiodic correlation which is related to the chip sequences $\{a_k^j\}$ and $\{a_1^j\}$ by:

$$C_{k1}(n) = \begin{cases} \sum_{j=0}^{N-1-n} a_k^j \cdot a_1^{j+n} & 0 \le n \le N-1 \\ \sum_{j=0}^{N-1+n} a_k^{j-n} \cdot a_1^j & 1-N \le n \le 0 \end{cases} \tag{9.39}$$

and where

$$\begin{aligned} A_{k1n} &= C_{k1}(n-N) \\ B_{k1n} &= C_{k1}n+1-N) - C_{k1}(n-N) \\ \hat{A}_{k1n} &= C_{k1}(n) \\ \hat{B}_{k1n} &= C_{k1}(n+1) - C_{k1}(n) \end{aligned} \tag{9.40}$$

A general expression for the expectation of the product of the partial correlation functions is given by [24] as:

$$\begin{aligned} I_k^{p,q} &= E_\tau \left[R_{k1}^{2q}(\tau_{k1}) \cdot \hat{R}_{k1}^{2(p-q)}(\tau_{k1}) \right] \\ &= \frac{T_c^{2p+1}}{T_b} \cdot \sum_{n=0}^{N-1} \sum_{l=0}^{2q} \frac{(-1)^l}{i+1} \cdot \frac{\binom{2q}{l}}{\binom{2(p-q)+i+1}{i+1}} \cdot \frac{B_{k1n}^l}{\hat{B}_{k1n}^{l+1}} \\ &\quad \cdot \left((A_{k1n} + B_{k1n})^{2q-l} \cdot (\hat{A}_{k1n} + \hat{B}_{k1n})^{2(p-q)+i+1} - A_{k1n}^{2q-l}\, \hat{A}_{k1n}^{2(p-q)+i+1} \right) \end{aligned} \tag{9.41}$$

where E_τ is the expectation with respect to the path delay(s).

Expectations not covered by this expression are [28, appendix C]:

$$\chi_k = E_\tau\left[R_{k1}(\tau_{k1}) \cdot \hat{R}_{k1}(\tau_{k1}) \right] = \frac{T_c^3}{T_b} \cdot \sum_{n=0}^{N-1} \left[A_{k1n} \cdot \hat{A}_{k1n} + \tfrac{1}{3} B_{k1n} \cdot \hat{B}_{k1n} + \tfrac{1}{2} A_{k1n} \cdot \hat{B}_{k1n} + \tfrac{1}{2} \hat{A}_{k1n} \cdot B_{k1n} \right] \tag{9.42}$$

$$\gamma_k = E_\tau\left[R_{k1}^2(\tau_{k1}) \cdot \hat{R}_{k1}(\tau_{k1}) \right] = \frac{T_c^4}{T_b} \cdot \sum_{n=0}^{N-1} \left[A_{k1n}^2 \cdot \hat{A}_{k1n} + A_{k1n} \cdot \hat{A}_{k1n} \cdot B_{k1n} + \tfrac{1}{3} B_{k1n}^2 \cdot \hat{A}_{k1n} + \tfrac{1}{2} A_{k1n}^2 \cdot \hat{B}_{k1n} + \tfrac{2}{3} A_{k1n} \cdot B_{k1n} \cdot \hat{B}_{k1n} + \tfrac{1}{4} B_{k1n}^2 \cdot \hat{B}_{k1n} \right] \tag{9.43}$$

$$\hat{\gamma}_k = E_\tau\left[\hat{R}_{k1}^2(\tau_{k1}) \cdot R_{k1}(\tau_{k1}) \right] = \frac{T_c^4}{T_b} \cdot \sum_{n=0}^{N-1} \left[\hat{A}_{k1n}^2 \cdot A_{k1n} + \hat{A}_{k1n} \cdot A_{k1n} \cdot \hat{B}_{k1n} + \tfrac{1}{3} \hat{B}_{k1n}^2 \cdot A_{k1n} + \tfrac{1}{2} \hat{A}_{k1n}^2 \cdot B_{k1n} + \tfrac{2}{3} \hat{A}_{k1n} \cdot \hat{B}_{k1n} \cdot B_{k1n} + \tfrac{1}{4} \hat{B}_{k1n}^2 \cdot B_{k1n} \right] \tag{9.44}$$

$$\zeta_k = E_\tau\left[R_{k1}^3(\tau_{k1}) \right] = \frac{T_c^4}{T_b} \cdot \sum_{n=0}^{N-1} \left[A_{k1n}^3 + \frac{3}{2} A_{k1n}^2 \cdot B_{k1n} + A_{k1n} \cdot B_{k1n}^2 + \tfrac{1}{4} B_{k1n}^3 \right] \tag{9.45}$$

$$\hat{\zeta}_k = E_\tau\left[\hat{R}_{k1}^3(\tau_{k1}) \right] = \frac{T_c^4}{T_b} \cdot \sum_{n=0}^{N-1} \left[\hat{A}_{k1n}^3 + \frac{3}{2} \hat{A}_{k1n}^2 \cdot \hat{B}_{k1n} + \hat{A}_{k1n} \cdot \hat{B}_{k1n}^2 + \tfrac{1}{4} \hat{B}_{k1n}^3 \right] \tag{9.46}$$

$$\upsilon_k = E_\tau\left[R_{k1}(\tau_{k1}) \right] = \frac{T_c^2}{T_b} \cdot \sum_{n=0}^{N-1} \left[A_{k1n} + \tfrac{1}{2} B_{k1n} \right] \tag{9.47}$$

$$\hat{\upsilon}_k = E_\tau\left[\hat{R}_{k1}(\tau_{k1}) \right] = \frac{T_c^2}{T_b} \cdot \sum_{n=0}^{N-1} \left[\hat{A}_{k1n} + \tfrac{1}{2} \hat{B}_{k1n} \right] \tag{9.48}$$

With these expressions we are able to express the means of μ_0 and μ, and the variance of μ_0 in terms of the discrete aperiodic correlation C_{kl}. Since the variance of the noise in μ_0 appears

as a constant when the expectation with respect to path delays is calculated, we remove it for the time being and add it to the mean afterwards. The mean values of μ_0 and μ are calculated by making use of equations 9.26-9.28.

$$\begin{aligned} E_\tau [\, \mu_0 \,] &= \tfrac{1}{4}P \cdot \sum_{k=1}^{K} E_\tau \left[E \left[X_k^2 \right] \right] + E_\tau \left[E \left[\hat{X}_k^2 \right] \right] + 2\sigma_n^2 \\ &= \tfrac{1}{4}P \cdot \left[\sum_{k=1}^{K} L \cdot \left(\Gamma_k^{1,1} + \Gamma_k^{1,0} \right) \cdot (\sigma^2 + \tfrac{1}{2}s^2) \right. \\ &\quad \left. + L\,(L-1) \left(v_k^2 + \hat{v}_k^2 \right) \cdot \tfrac{1}{2}s^2 \right] + 2\sigma_n^2 \end{aligned} \tag{9.49}$$

$$\begin{aligned} E_\tau [\, \mu \,] &= \tfrac{1}{4}P \cdot \sum_{k=1}^{K} E_\tau \left[E \left[X_k \hat{X}_k \right] \right] + \tfrac{1}{4}P \cdot E_\tau \left[E \left[\hat{X}_1^2 \right] \right] \\ &= \tfrac{1}{4}P \cdot \sum_{k=1}^{K} \left[L \cdot \chi_k \cdot (\sigma^2 + \tfrac{1}{2}s^2) + L\,(L-1) \cdot v_k \cdot \hat{v}_k \cdot \tfrac{1}{2}s^2 \right] \\ &\quad + L \cdot \Gamma_1^{1,0} \cdot (\sigma^2 + \tfrac{1}{2}s^2) + L\,(L-1) \cdot \hat{v}_1^2 \cdot \tfrac{1}{2}s^2 \end{aligned} \tag{9.50}$$

For calculating the variance we make another simplification: we drop the term: $\hat{X}_1^2 + \hat{Y}_1^2$ in equation 9.22. It is shown in [24] that μ is very small (nearly zero) and it is even ignored there. This implies that the assumption mentioned above is a reasonable one.

$$\mathrm{var}(\mu_0) = E_\tau \left[\mu_0^2 \right] - \left(E_\tau [\mu_0] \right)^2 \tag{9.51}$$

where

$$\begin{aligned} E_\tau \left[\mu_0^2 \right] &= E_\tau \left[\left(\tfrac{1}{8}P \cdot \sum_{k=1}^{K} E \left[2X_k^2 \right] + E \left[2\hat{X}_k^2 \right] \right)^2 \right. \\ &\quad \left. + \left(\tfrac{1}{8}P \cdot \sum_{k=1}^{K} E \left[2X_k^2 \right] + E \left[2\hat{X}_k^2 \right] \right) \cdot 2\sigma_n^2 + \left(2\sigma_n^2 \right)^2 \right] \\ &= (\tfrac{1}{8}P)^2 \cdot E_\tau \left[\sum_{k=1}^{K} \left(E \left[2X_k^2 \right] + E \left[2\hat{X}_k^2 \right] \right)^2 \right. \\ &\quad \left. + \left(\sum_{k=1}^{K} E \left[2X_k^2 \right] + E \left[2\hat{X}_k^2 \right] \right) \cdot \left(\sum_{\substack{l=1 \\ l \neq k}}^{K} E \left[2X_l^2 \right] + E \left[2\hat{X}_l^2 \right] \right) \right] \\ &\quad + E_\tau \left[\tfrac{1}{2}P \cdot \sigma_n^2 \cdot \sum_{k=1}^{K} E \left[2X_k^2 \right] + E \left[2\hat{X}_k^2 \right] \right] + E_\tau \left[\left(2\sigma^2 \right)^2 \right] \end{aligned} \tag{9.52}$$

and

$$
\begin{aligned}
\left(E_\tau [\mu_0] \right)^2 &= \left(\tfrac{1}{8}P \cdot \sum_{k=1}^{K} E_\tau \left[E \left[2X_k^2 \right] \right] + E_\tau \left[E \left[2\hat{X}_k^2 \right] \right] + 2\sigma_n^2 \right)^2 \\
&= (\tfrac{1}{8}P)^2 \cdot \left(\sum_{k=1}^{K} \left(E_\tau \left[E \left[2X_k^2 \right] \right] + E_\tau \left[E \left[2\hat{X}_k^2 \right] \right] \right)^2 \right. \\
&\quad + \left(\sum_{k=1}^{K} E_\tau \left[E \left[2X_k^2 \right] \right] + E_\tau \left[E \left[2\hat{X}_k^2 \right] \right] \right) \\
&\quad \left. \cdot \left(\sum_{\substack{i=1 \\ i \neq k}}^{K} E_\tau \left[E \left[2X_i^2 \right] \right] + E_\tau \left[E \left[2\hat{X}_i^2 \right] \right] \right) \right) \\
&\quad + \tfrac{1}{2}P \cdot \sigma_n^2 \cdot \sum_{k=1}^{K} E_\tau \left[E \left[2X_k^2 \right] \right] + E_\tau \left[E \left[2\hat{X}_k^2 \right] \right] + E_\tau \left[\left(2\sigma_n^2 \right)^2 \right]
\end{aligned}
\tag{9.53}
$$

So we can write for the variance of μ_0:

$$
\begin{aligned}
\mathrm{var}(\mu_0) &= E_\tau \left[\mu_0^2 \right] - \left(E_\tau [\mu_0] \right)^2 \\
&= (\tfrac{1}{8}P)^2 \cdot \left(\sum_{k=1}^{K} E_\tau \left[\left(E \left[2X_k^2 \right] \right)^2 + 2 E \left[2X_k^2 \right] E \left[2\hat{X}_k^2 \right] + \left(E \left[2\hat{X}_k^2 \right] \right)^2 \right] \right) \\
&\quad - (\tfrac{1}{8}P)^2 \cdot \left(\sum_{k=1}^{K} \left(E_\tau \left[E \left[2X_k^2 \right] + E \left[2\hat{X}_k^2 \right] \right] \right)^2 \right)
\end{aligned}
\tag{9.54}
$$

The expectations needed to calculate this variance are derived in appendix C. The results are given below

$$
\begin{aligned}
& E_\tau \left[\left(E \left[2X_k^2 \right] \right)^2 \right] = 4 \cdot E_\tau \left[\left(E \left[X_k^2 \right] \right)^2 \right] = \\
& 4 \cdot \left[(\sigma^2 + \tfrac{1}{2}s^2)^2 \cdot \left(L \cdot \Gamma_k^{2,2} + L(L-1) \cdot \left(\Gamma_k^{1,1} \right)^2 \right) \right. \\
& \quad + (\sigma^2 + \tfrac{1}{2}s^2) \cdot \tfrac{1}{2}s^2 \cdot \left(4 \cdot L(L-1) \cdot \zeta_k \cdot \upsilon_k + 2 \cdot L(L-1)(L-2) \cdot \Gamma_k^{1,1} \cdot \upsilon_k^2 \right) \\
& \quad \left. + (\tfrac{1}{2}s^2)^2 \cdot \left(2 \cdot L(L-1) \left(\Gamma_k^{1,1} \right)^2 + 4 \cdot L(L-1)(L-2) \cdot \Gamma_k^{1,1} \cdot \upsilon_k^2 \right. \right. \\
& \qquad \left. \left. + L(L-1)(L-2)(L-3) \cdot \upsilon_k^4 \right) \right]
\end{aligned}
\tag{9.55}
$$

$$
\begin{aligned}
&E_\tau\left[\left(E\left[2\hat{X}_k^2\right]\right)^2\right] = 4\cdot E_\tau\left[\left(E\left[\hat{X}_k^2\right]\right)^2\right] = \\
&4\cdot\Big[(\sigma^2+\tfrac{1}{2}s^2)^2\cdot\left(L\cdot\Gamma_k^{2,0}+L(L-1)\cdot\left(\Gamma_k^{1,0}\right)^2\right) \\
&\quad+(\sigma^2+\tfrac{1}{2}s^2)\cdot\tfrac{1}{2}s^2\cdot\left(4\cdot L(L-1)\cdot\hat{\zeta}_k\cdot\hat{\upsilon}_k+2\cdot L(L-1)(L-2)\cdot\Gamma_k^{1,0}\cdot\hat{\upsilon}_k^2\right) \\
&\quad+(\tfrac{1}{2}s^2)^2\cdot\Big(2\cdot L(L-1)\left(\Gamma_k^{1,0}\right)^2+4\cdot L(L-1)(L-2)\cdot\Gamma_k^{1,0}\cdot\hat{\upsilon}_k^2 \\
&\qquad\qquad+L(L-1)(L-2)(L-3)\cdot\hat{\upsilon}_k^4\Big)\Big]
\end{aligned}
\tag{9.56}
$$

$$
\begin{aligned}
&E_\tau\left[2E\left[2X_k^2\right]E\left[2\hat{X}_k^2\right]\right] = 8\cdot E_\tau\left[E\left[X_k^2\right]E\left[\hat{X}_k^2\right]\right] = \\
&8\cdot\Big[(\sigma^2+\tfrac{1}{2}s^2)^2\cdot\left(L^2\cdot\Gamma_k^{1,1}\cdot\Gamma_k^{1,0}\right) \\
&\quad+(\sigma^2+\tfrac{1}{2}s^2)\cdot\tfrac{1}{2}s^2\cdot\Big(2\cdot L(L-1)\cdot\gamma_k\cdot\hat{\upsilon}_k+L(L-1)(L-2)\cdot\Gamma_k^{1,1}\cdot\hat{\upsilon}_k^2 \\
&\qquad+2\cdot L(L-1)\cdot\hat{\gamma}_k\cdot\upsilon_K+L(L-1)(L-2)\cdot\Gamma_k^{1,0}\cdot\upsilon_k^2\Big) \\
&\quad+(\tfrac{1}{2}s^2)^2\cdot\left(L^2\cdot(L-1)^2\cdot\upsilon_k^2\cdot\hat{\upsilon}_k^2\right)\Big]
\end{aligned}
\tag{9.57}
$$

$$
\begin{aligned}
&\left(E_\tau\left[E\left[2X_k^2\right]+E\left[2\hat{X}_k^2\right]\right]\right)^2 = 4\cdot\left(E_\tau\left[E\left[X_k^2\right]+E\left[\hat{X}_k^2\right]\right]\right)^2 = \\
&4\cdot\left((\sigma^2+\tfrac{1}{2}s^2)\cdot L\cdot\left(\Gamma_k^{1,1}+\Gamma_k^{1,0}\right)+\tfrac{1}{2}s^2\cdot L(L-1)\cdot\left(\upsilon_k^2+\hat{\upsilon}_k^2\right)\right)^2
\end{aligned}
\tag{9.58}
$$

9.4 Computational Results

As in chapter 6, we write σ and s in terms of the signal-to-noise ratio and the Rician parameter R. The received signal power we use in our calculations is again normalised with respect to the received specular power.

The direct sequence codes used in the performance analysis performed in this section are Gold codes and the integrals in equations 9.36 are calculated using the Romberg procedure from the Turbo Pascal Mathematical Methods Toolbox 4.0.

The average bit error probability for several values of the number of resolvable paths is plotted in figure 9.1 and 9.2 as a function of the signal-to-noise ratio for bitrates of 32 kbit/s and 64 kbit/s.

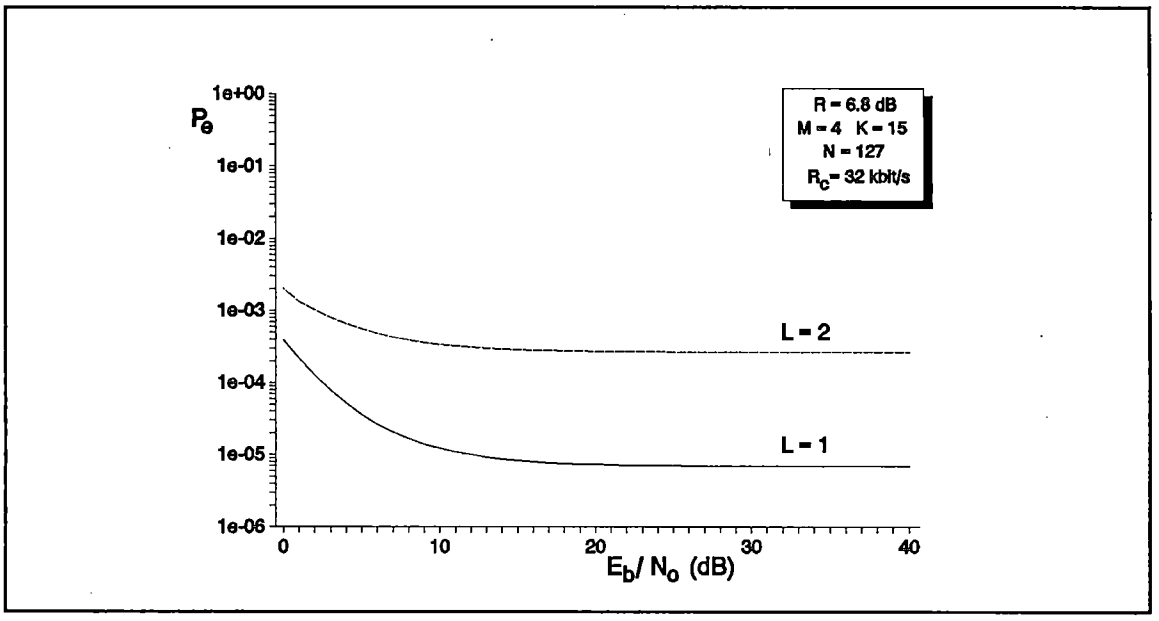

Figure 9.1 Average Bit Error Probability as a function of the signal-to-noise ratio for several values of resolvable paths with R_c = 32 kbit/s

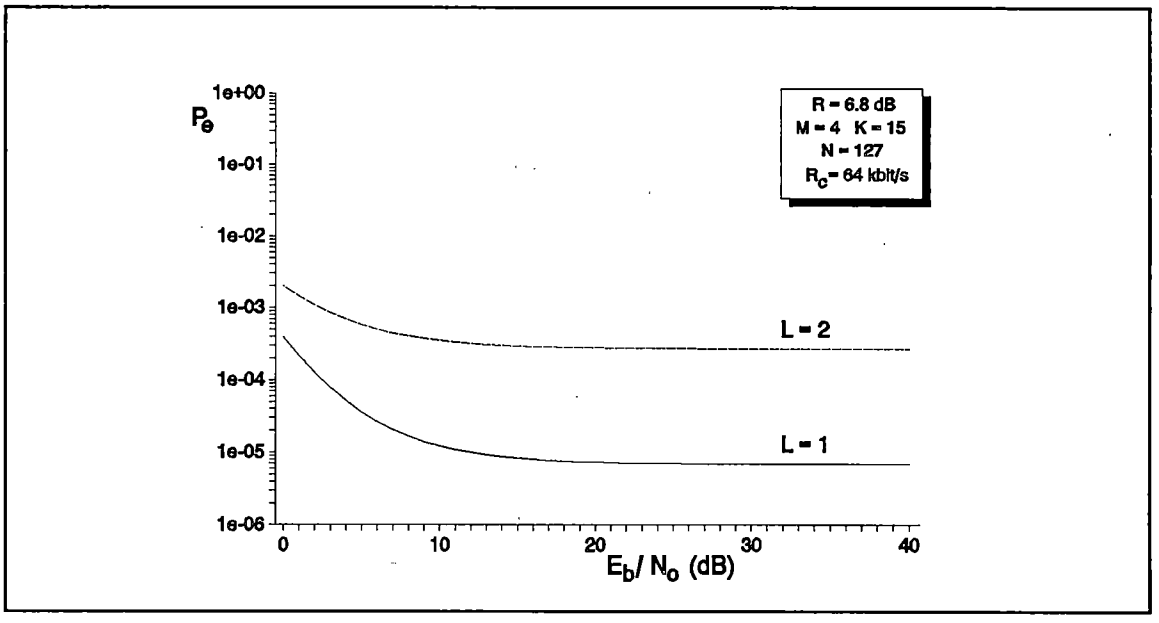

Figure 9.2 Average Bit Error Probability as a function of the signal-to-noise ratio for several values of resolvable paths with R_c = 64 kbit/s

In figures 9.1 and 9.2 we can see that P_e is approximately equal for an equal number of resolvable paths. This might be a surprising result, but one should take notice of the fact that with L = 1 and R_c = 32 kbit/s, the delay spread T_m of the channel lies in the range $0 < T_m < 246$ ns. The delay spread for R_c = 64 kbit/s with L = 1 lies in the range $0 < T_m < 123$ ns and there are two resolvable paths when $123 < T_m < 246$ ns. So these results apply to different channel characteristics.

The delay spreads that correspond to specific values of L for a specific channel bit rate, are given in table 9.1.

L	R_b=32 kbit/s		R_b=64 kbit/s		R_b=144 kbit/s	
	N=127	N=255	N=127	N=255	N=127	N=255
1	$T_m<246$	$T_m<123$	$T_m<123$	$T_m<62$	$T_m<55$	$T_m<27$
2	$246<T_m<492$	$123<T_m<246$	$123<T_m<246$	$62<T_m<124$	$55<T_m<110$	$27<T_m<54$
3		$246<T_m<370$	$246<T_m<370$	$124<T_m<186$	$110<T_m<165$	$54<T_m<81$
4				$186<T_m<248$	$165<T_m<220$	$81<T_m<108$
5				$248<T_m<310$	$220<T_m<275$	$108<T_m<135$
6						$135<T_m<162$
7						$162<T_m<189$
8						$189<T_m<216$

Table 9.1 Relation between the delay spread and the number of resolvable paths, for different data rates and code sequence lengths (T_m in ns)

In figure 9.3, P_e is plotted for 255 chips per bit. When we compare the plots with the plots for 127 chips per bit in figure 9.2, it is clear that for an equal number of resolvable paths, a higher number of chips per bit enhances the performance of the system.

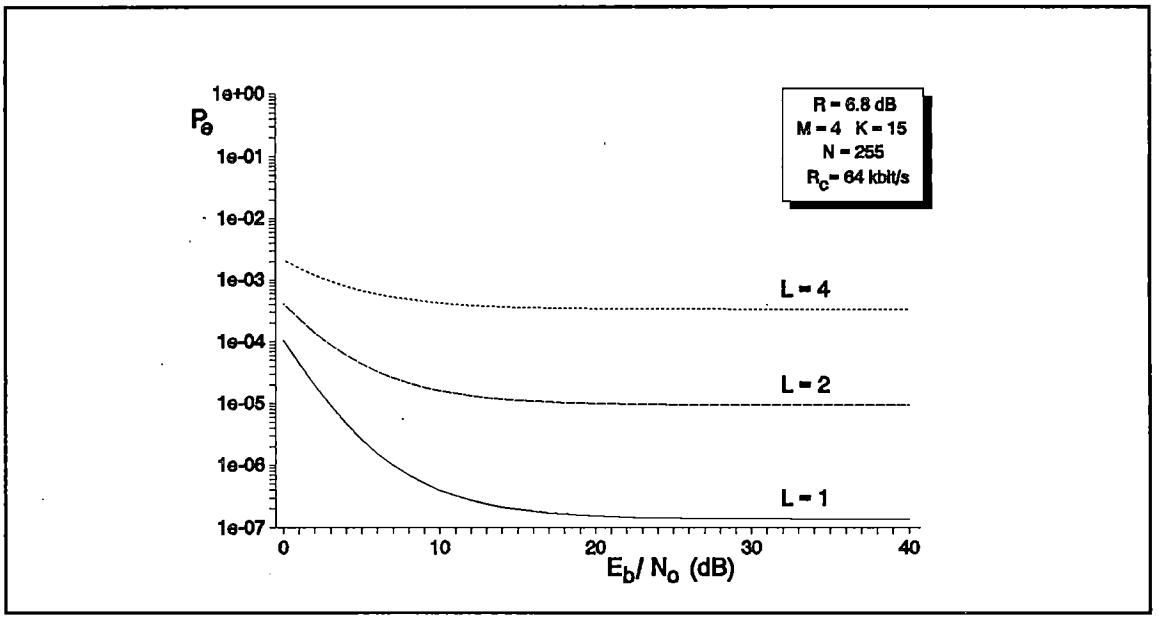

Figure 9.3 Average Bit Error Probability as a function of the signal-to-noise ratio for several values of resolvable paths with R_c = 64 kbit/s

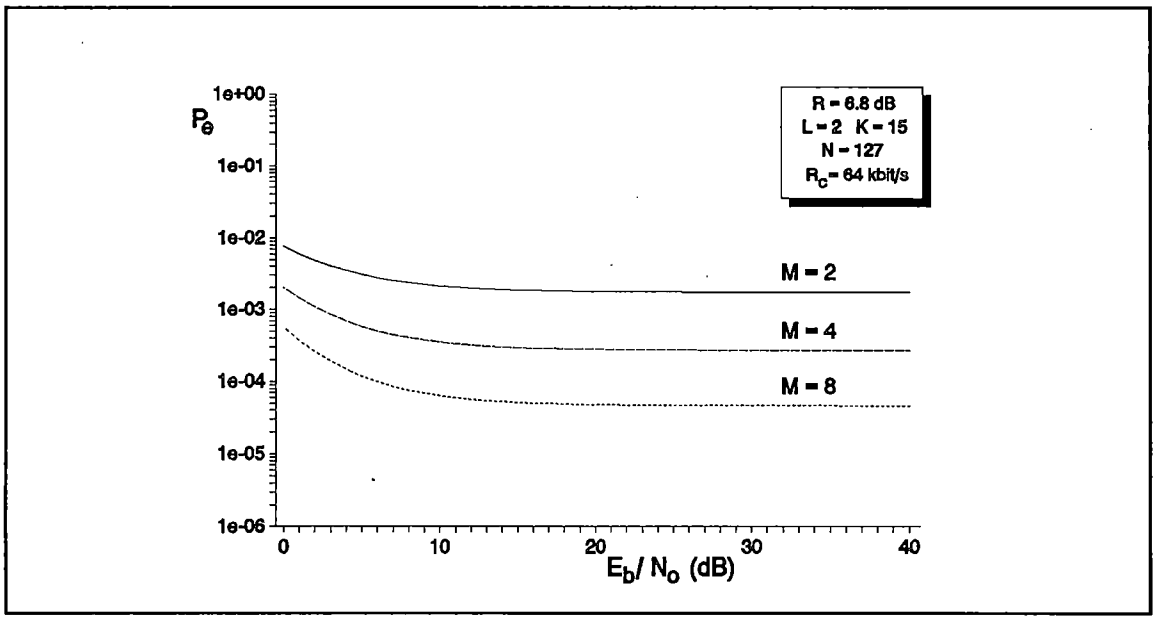

Figure 9.4 Average Bit Error Probability as a function of the signal-to-noise ratio for several values of diversity

The influence of the order of diversity is investigated in figure 9.4, where we have plotted P_e for several values of the order of diversity with the number of resolvable paths fixed at 2. We can clearly see that a higher order of diversity (more antenna's) enhances the performance.

In figure 9.5 the influence of the value of the Rician parameter is investigated. The Rician parameter is the ratio of the peak power and the power received over specular paths. When R = 0 , there is no dominant path and the Rician distribution becomes the Rayleigh distribution. From [6] we know that R = 6.8 dB corresponds to a 30-year-old brick building with reinforced concrete and plaster, as well as some ceramic block interior partitions. R = 11 dB corresponds to a building that has the same construction, but has an open-office interior floor plan, and non-metallic ceiling tiles.

It can be seen in figure 9.5, that the Rician channel gives a far better performance than the Rayleigh channel and we may say that the Rayleigh fading channel model gives a pessimistic prediction of the average bit error probability, especially for low signal-to-noise ratios.

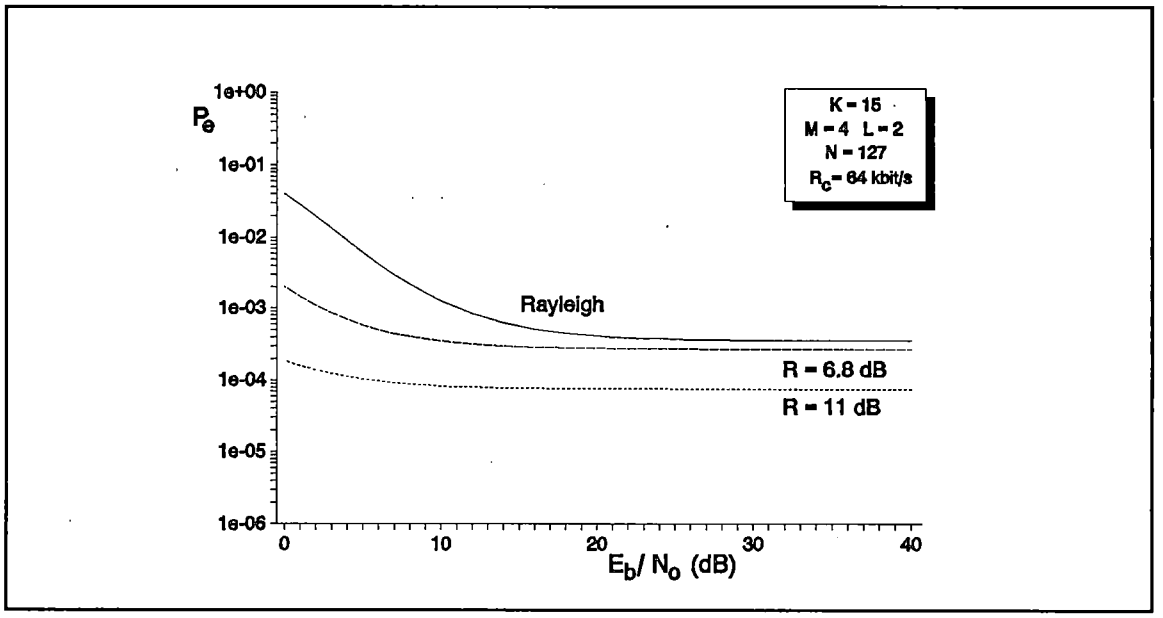

Figure 9.5 Average Bit Error Probability as a function of the signal-to-noise ratio for a Rician channel with R = 6.8 dB and R = 11 dB, and for a Rayleigh channel

The influence of the number of chips per bit is investigated in figure 9.6. In order to make a fair comparison between N = 127 and N = 255, we need to have an identical path delay for both plots. The consequence of this is, that for N = 127 we have 2 resolvable paths and for N 255 we have 4 resolvable paths.

From the plots we can see that P_e is approximately equal for both cases. The problem that arises now is why would we use 255 chips per bit when 127 chips per bit gives the same performance ? An answer could be that for 255 chips the system can accommodate twice the amount of users.

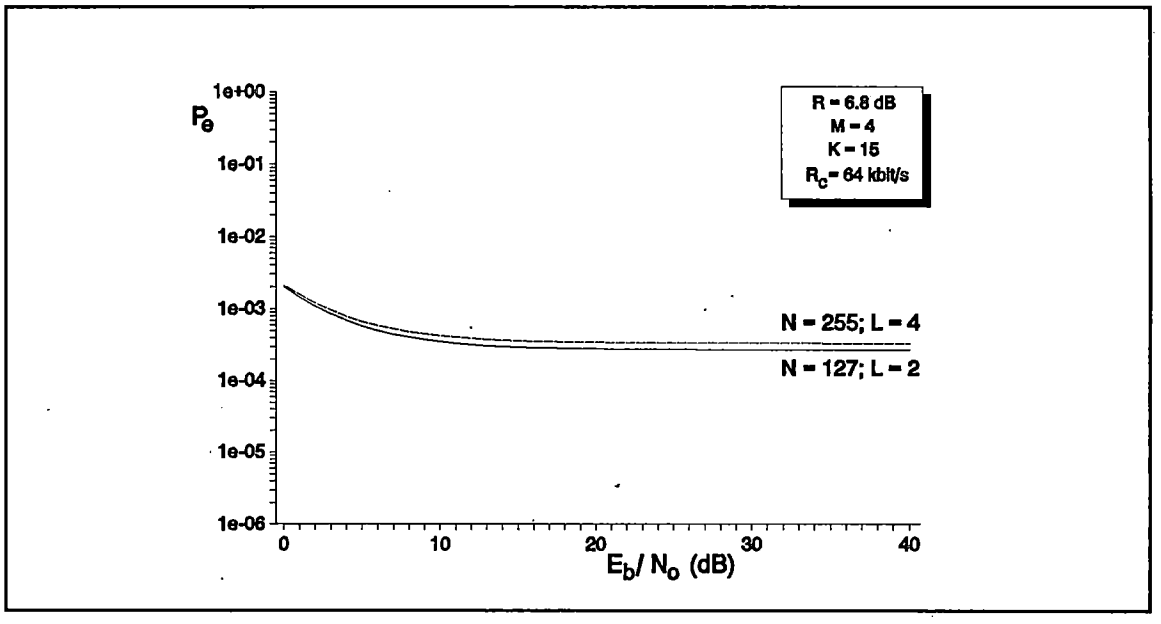

Figure 9.6 Average Bit Error Probability as a function of the signal-to-noise ratio for N = 127 and N = 255, T_m = 185 ns

The influence of the number of simultaneously transmitting users K is investigated in figure 9.7. In this figure we can see that the performance decreases with increasing K, as expected.

In figure 9.8 the average bit error probability is plotted for several values of the bit rate. We again have assumed a fixed delay spread of 200 ns, in order to make a fair comparison as in figure 9.6. We can see that the performance decreases with increasing channel bit rate, because the number of resolvable paths increases proportionally, which has a negative effect on the average bit error probability.

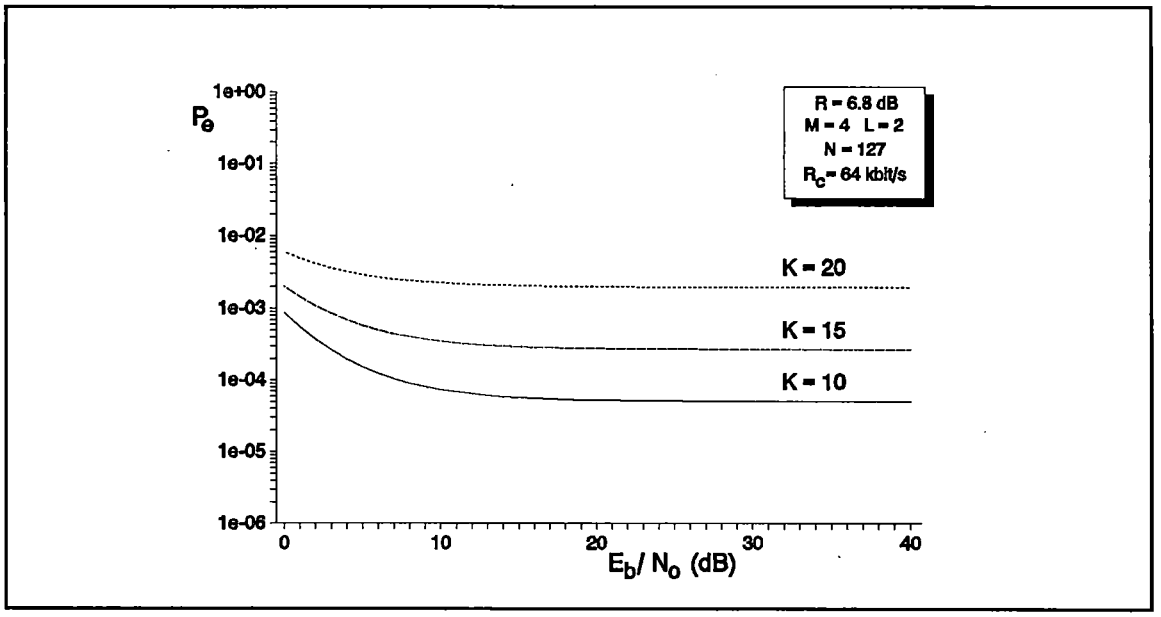

Figure 9.7 Average Bit Error Probability as a function of the signal-to-noise ratio for a different number of simultaneously transmitting users

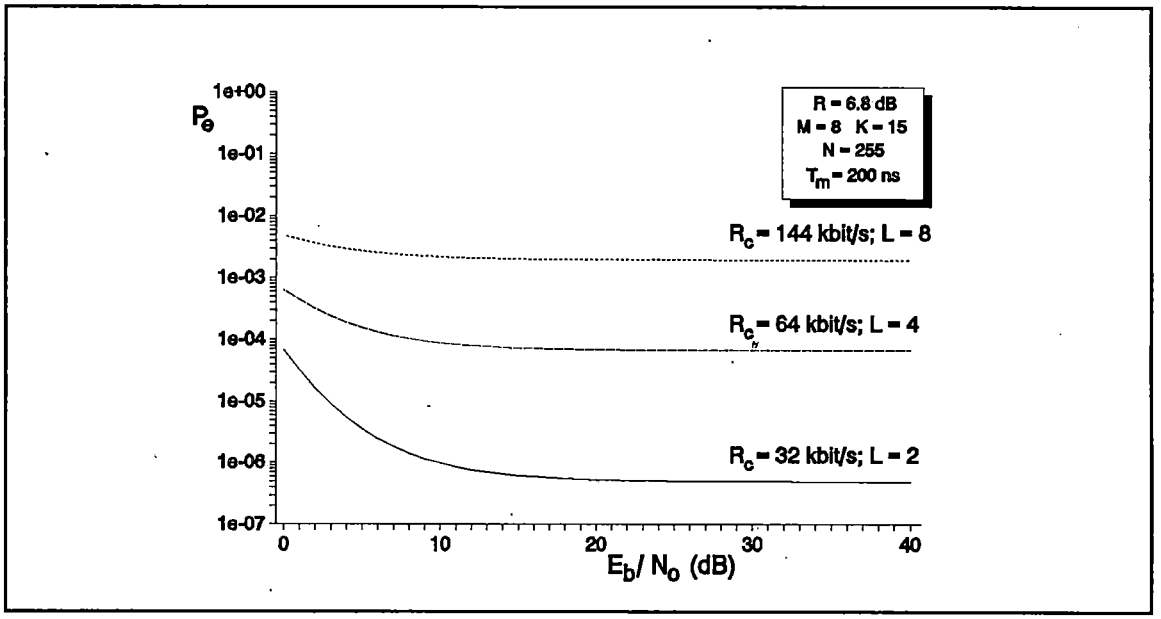

Figure 9.8 Average Bit Error Probability as a function of the signal-to-noise ratio for T_m = 200 ns

9.5 Discussion of the Results

From the performance analysis performed in this chapter the following conclusions can be drawn:

- The average bit error probability decreases with increasing signal-to-noise ratios to an asymptote caused by the multi-user interference and interference from the multiple resolvable paths.
- For equal number of resolvable paths, the performance increases with increasing number of chips per bit.
- The performance increases with increasing order of diversity.
- The Rician channel model enhances performance when compared with the Rayleigh channel model.
- The performance decreases with increasing number of simultaneously transmitting users.
- The performance decreases with increasing bit rate.

When we compare the performance of a hybrid SFH/DS system that has an average number of simultaneously transmitting users of 15 and N = 127, R_c = 64 kbit/s with the performance of a slow frequency hopping system, as discussed in chapter 6 and 7, that has Q = 128, K = 15, R_c = 64 kbit/s and uses random hopping patterns, it is obvious that the hybrid system has a far better performance than the SFH systems for equal bandwidth utilisation and number of users. This can be explained by the correlation step in the receiver for the hybrid system, because of which we can allow multiple users occupy the same frequencies at the same time.

10. CONCLUSIONS AND RECOMMENDATIONS

The performance of frequency hopping systems with DPSK modulation and diversity in the indoor Rician fading channel were investigated in this report.

First we looked at a fast frequency hopping system with DPSK modulation, but we found that it was unable to comply to our system demands, with respect to multiple access and mainly because of the lack of flexibility in this system.

Next we investigated the performance of a slow frequency hopping system with DPSK modulation for two kinds of diversity, namely selection diversity and maximal ratio combining. From this performance analysis the following conclusions can be drawn:

- When there are multiple resolvable paths, the average bit error probability was found to decrease with increasing signal-to-noise ratios to an asymptote caused by the interference from the multiple resolvable paths.
- The performance was found to decrease dramatically when users were allowed to occupy the same frequency slot at the same time.
- Maximal ratio combining yields a far better performance than selection diversity.

The obtained results showed that for deterministic hopping patterns, the performance of a slow frequency hopping system with selection diversity in a discrete multipath channel is not very good. So selection diversity would not be a good method to use for such a system. Instead we would use maximal ratio combining as the diversity technique.

We also investigated the performance of a hybrid slow frequency hopping/direct sequence system with DPSK modulation in an indoor Rician fading channel with selection diversity. From this performance analysis the following conclusions can be drawn:

- When there are simultaneously transmitting users transmitting on the same frequency, the average bit error probability was found to decrease with increasing signal-to-noise ratios to an asymptote caused by the multi-user interference and interference from the multiple resolvable paths.
- For an equal number of resolvable paths, the performance increases with increasing number of chips per bit.
- The performance was found to decrease with increasing bit rate.

There are also conclusion which are valid for both systems:

- The performance was found to increase with increasing order of diversity.
- A larger Rician parameter gives an improved performance.
- It was found that a Rician fading channel model enhances the performance when compared with a Rayleigh fading channel model.

When we compare the systems we have investigated in this report with the system requirements set for a wireless CDMA system for the indoor environment, we can conclude that the hybrid system is the best suited for this task. Mainly because of its ability to allow multiple users to share the same frequencies, without a dramatic loss in performance.

Recommendations for further research:

Subjects for further research in the field of indoor wireless communications, especially for the proposed hybrid SFH/DS system, that may give interesting results, are:

- Other performance measures could be used to validate the hybrid SFH/DS system, such as packet throughput and delay. Packet arrival models as proposed in [33] and models for the distribution of terminals over the cell would then have to be incorporated into the mathematical models proposed in this report. The distribution of terminals is especially interesting, because the received signal from a terminal would depend on the distance of that terminal to the receiver and it would be possible to use signal capture in our receiver model.

- When the (packet) throughput of the hybrid SFH/DS system is investigated, also the use of forward error correcting coding and its influence on the performance can be investigated.

- Also the performance of hybrid SFH/DS with maximal ratio combining can be investigated. But such a system would be very hard to build in reality, because of the fact that the resolvable paths arrive at the receiving antenna with a minimal time difference of a chip time, which makes tracking of the signal very difficult.

- We can investigate how the transmitted signal power is distributed over the multiple resolvable paths and whether it is possible to control this. When the distribution of signal power can be controlled, a diversity technique such as selection diversity will give an enhanced performance approaching that of maximal ratio combining.

REFERENCES

[1] **R.L. Pickholz, D.L. Schilling and L.B. Millstein,**
"Theory of Spread-Spectrum Communications - A Tutorial"
IEEE Transactions on Communications,
vol. COM-30, pp. 855-884, May 1982, 56 refs.

[2] **D.L Schilling, R.L. Pickholz and L.B. Millstein,**
"Spread spectrum goes commercial"
IEEE Spectrum,
pp. 40-45, August 1990.

[3] **K. Pahlavan,**
"Wireless Communications for Office Information Networks"
IEEE Communications Magazine,
vol. 23, no. 6, June 1985, pp. 19-27, 31 refs.

[4] **R.C. Dixon,**
"Spread Spectrum Systems - Second Edition"
John Wiley & Sons, New York, 1984.

[5] **G.R. Cooper and C.D. McGillem,**
"Modern Communications and Spread Spectrum"
McGraw-Hill Book Company, New York, 1986.

[6] **R.J.C. Bultitude,**
"Measurement, Characterization and Modeling of Indoor 800/900 MHz Radio Channels for Digital Communications"
IEEE Communications Magazine,
vol. 25, no. 6, June 1987, pp. 5-12, 8 refs.

[7] **T.S. Rappaport and C.D. McGillem,**
"UHF Fading in Factories"
IEEE Journal on Selected Areas in Communications,
vol. 7, no. 1, January 1989, pp. 40-48, 32 refs.

[8] **A.A.M. Saleh and R.A. Valenzuela**
"A statistical model for indoor multipath propagation"
IEEE Journal on Selected Areas on Communications,
vol. SAC-5, no. 2, February 1987, pp. 128-137, 20 refs.

[9] **D.M.J. Devasirvatham,**
"Time Delay Spread and Signal Level Measurements of 850 MHz Radio Waves in Building Environments"
IEEE Transactions on Antennas and Propagation,
vol. AP-34, no. 11, November 1986, pp. 1300-1305.

[10] **S. Stein,**
"Fading Channel Issues in System Engineering"
IEEE Journal on Selected Areas in Communications,
vol. SAC-5, no. 2, February 1987, pp. 68-89, 19 refs.

[11] **S.J. Howard and K. Pahlavan,**
"Doppler Spread Measurements of Indoor Radio Channel"
Electronics Letters,
vol. 26, no. 2, 18th January 1990, pp. 107-109, 6 refs.

[12] **G.R. Cooper and R.W. Nettleton,**
"A Spread-Spectrum Technique for High-Capacity Mobile Communications"
IEEE Transactions on Vehicular Technology,
vol. VT-27, pp. 264-275, November 1978, 22 refs.

[13] **P.S. Henry,**
"Spectrum Efficiency of a Frequency-Hopped-DPSK Spread-Spectrum Mobile Radio System"
IEEE Transactions on Vehicular Technology,
vol. VT-28, pp. 327-332, November 1979, 14 refs.

[14] **O. Yue,**
"Frequency-Hopping, Multiple-Access, Phase-Shift-Keying System Performance in a Rayleigh Fading Environment"
Bell System Technical Journal,
vol. 59, no. 6, pp. 861-879, July-August 1980, 14 refs.

[15] **O. Yue,**
"Hard-Limited Versus Linear Combining for Frequency-Hopping Multiple-Access Systems in a Rayleigh Fading Environment"
IEEE Transactions on Vehicular Technology,
vol. VT-30, pp. 10-14, February 1981, 4 refs.

[16] **R.W. Nettleton and G.R. Cooper,**
"Performance of a Frequency-Hopped Differentially Modulated Spread-Spectrum Receiver in a Rayleigh Fading Channel"
IEEE Transactions on Vehicular Technology,
vol. VT-30, pp. 14-29, February 1981, 25 refs.

[17] **M. Matsumoto and G.R. Cooper,**
"Multiple Narrow-Band Interferers in an FH-DPSK Spread-Spectrum Communication System"
IEEE Transactions on Vehicular Technology,
vol. VT-30, pp. 37-42, February 1981, 6 refs.

[18] **M. Matsumoto and G.R. Cooper,**
"Performance of a Nonlinear FH-DPSK Spread-Spectrum Receiver with Multiple Narrow-Band Interfering Signals"
IEEE Transactions on Communications,
vol. COM-30, pp. 937-942, May 1982, 8 refs.

[19] **S.V. Pizzi, P.J. Leary, J. deLellis,**
"A frequency-hopped Ungerboeck-encoded 4-ary DPSK modem"
MILCOM 88: 21st Century Military Communications - What's Possible ?
IEEE Military Communications Conference (Cat. No. 88CH2537-9), San Diego, CA, USA, 23-26 Oct. 1988, p. 133-141, vol. 1, 12 refs.

[20] **W.C. Lindsey, S.H. An, R.M. Vacek,**
"Performance of M-ary FH-DPSK systems in the presence of jamming"
MILCOM 88: 21st Century Military Communications - What's Possible ?
IEEE Military Communications Conference (Cat. No. 88CH2537-9), San Diego, CA, USA, 23-26 Oct. 1988, p. 605-609, vol. 2, 5 refs.

[21] **K.S. Gong,**
"Performance analysis of FH/DPSK in additive white Gaussian noise (AWGN) and multitone jamming"
MILCOM 88: 21st Century Military Communications - What's Possible ?
IEEE Military Communications Conference (Cat. No. 88CH2537-9), San Diego, CA, USA, 23-26 Oct. 1988, p. 947-953, vol. 3, 5 refs.

[22] **E.A. Geraniotis and M.B. Pursley,**
"Error Probabilities for Slow-Frequency-Hopped Spread-Spectrum Multiple-Access Communications Over Fading Channels"
IEEE Transactions on Communications,
vol. COM-30, May 1982, pp. 996-1009.

[23] **M. Kavehrad and P.J. McLane,**
"Performance of Low-Complexity Channel Coding and Diversity for Spread Spectrum in Indoor, Wireless Communication"
AT&T Technical Journal,
vol. 64, no. 6, October 1985, pp. 1927-1965, 28 refs.

[24] **M. Kavehrad and B. Ramamurthi,**
"Direct-Sequence Spread Spectrum with DPSK Modulation and Diversity for Indoor Wireless Communications"
IEEE Transactions on Communications,
vol. COM-35, February 1987, pp. 224-236, 11 refs.

[25] **J.G. Proakis,**
"Digital Communications - 2nd Edition"
McGraw-Hill, New York, 1983

[26] **J. Wang and M. Moeneclaey,**
"Performance of fast-frequency-hopping spread-spectrum multiple-access for indoor wireless communications"
IEEE International Conference on Communications ICC'90, Communications - Foundation for the 21st Century, April 16-19, vol. 4, p. 1358-1362, 8 refs.

[27] **A. Papoulis,**
"Probability, Random Variables and Stochastic Processes - 2nd Edition"
McGraw-Hill, Singapore, 1984

[28] **H. Sewberath Misser,**
"Performance Analysis of a Direct-Sequence Spread-Spectrum Multiple Access Communication System in an Indoor Rician Fading Radio Channel with Differential Phase Shift Keying"
Delft University of Technology, Telecommunications and Traffic-Control Systems Group, Graduation Thesis, 26th June 1990.

[29] **P.Z. Peebles Jr.,**
"Probability, Random Variables, and Random Signal Principles"
McGraw-Hill, New York, 1981

[30] **A. Kajihara and M. Nakagawa,**
"A New PLL Synthesizer for Fast FH Spread Spectrum Applications"
IEEE Conference ?, 1989, pp. 1602-1606, 6 refs.

[31] **E.A. Geraniotis,**
"Noncoherent Hybrid DS-SFH Spread-Spectrum Multiple-Access Communications"
IEEE Transactions on Communications,
vol. COM-34, pp. 862-872, September 1986.

[32] **M. Moeneclaey and J. Wang,**
"Performance of Hybrid DS/SFH Spread-Spectrum Multiple-Access with Predetection Diversity for Indoor Radio"
Archiv für Elektronik und Übertragungs Technik,
vol. 45, no. 1, pp. 11-17, January 1991, 5 refs.

[33] **C.A.F.J. Wijffels,**
"Throughput, delay and stability analysis of a slotted code division multiple access system for indoor wireless communication"
Delft University of Technology, Telecommunications and Traffic-Control Systems Group, Graduation Thesis, 5th March 1991.

[34] **A.B. Carlson,**
"Communication Systems - 3rd Edition"
McGraw-Hill, New York, 1986

APPENDIX A: DPSK Modulation

It is possible to generate a type of coherent detection processor without using a coherent local oscillator. This is done by the use of differential phase shift keying in which consecutive bits are given the same phase for a one and opposite phase for a zero. The local oscillator is replaced by the signal delayed an amount equal to the bit spacing. This is illustrated in figure A.1.

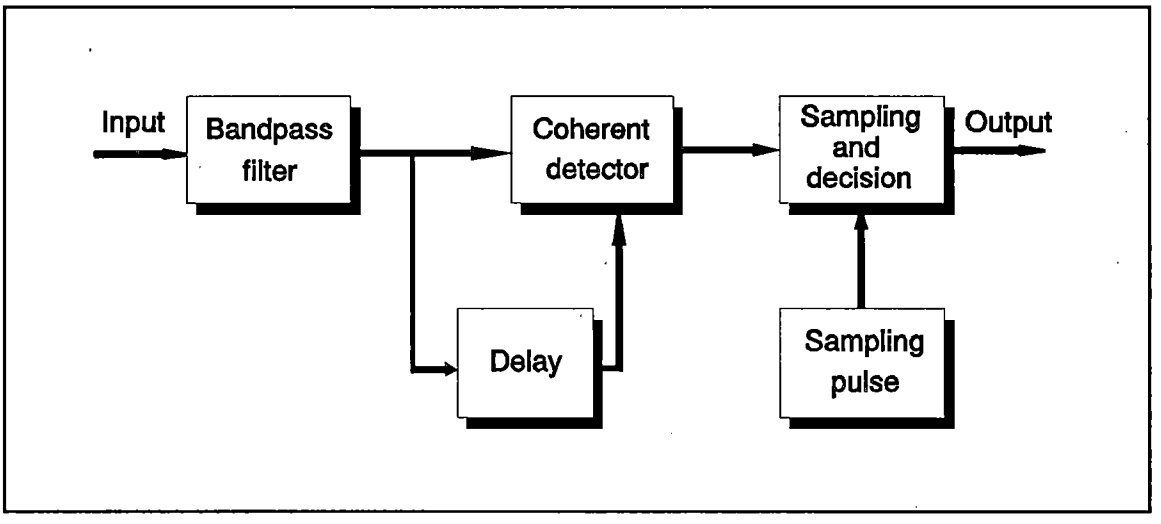

Figure A.1 DPSK Decoder

Differential encoding starts arbitrarily with the first bit and then indicates message bits by phase transition or no phase transition. As an example, consider the following bit sequence and encoded message.

Message		1	1	0	1	0	1	1	0	0	1
Encoded Message	1	1	1	0	0	1	1	1	0	1	1
Phase	0	0	0	π	π	0	0	0	π	0	0

The performance of DPSK is somewhat poorer than PSK because, although the detection is coherent, it is carried out with a noisy reference (the previous bit consists of signal plus noise).

The bit error probability in an AWGN channel is defined as:

$$P_e = \tfrac{1}{2} \cdot \exp(-\gamma_b) = \tfrac{1}{2} \cdot \exp\left(-\frac{E_b}{N_o}\beta^2\right) \tag{A.1}$$

The equivalent low-pass spectrum of binary DPSK is defined as [34]:

$$G_{lp}(f) = \frac{1}{r} \cdot \mathrm{sinc}^2\left(\frac{f}{r}\right) = T_b \cdot \mathrm{sinc}^2(fT_b) \tag{A.2}$$

The power spectrum of DPSK with carrier frequency f_c is plotted in figure A.2 and the transmission bandwidth is determined as $1/T_b$.

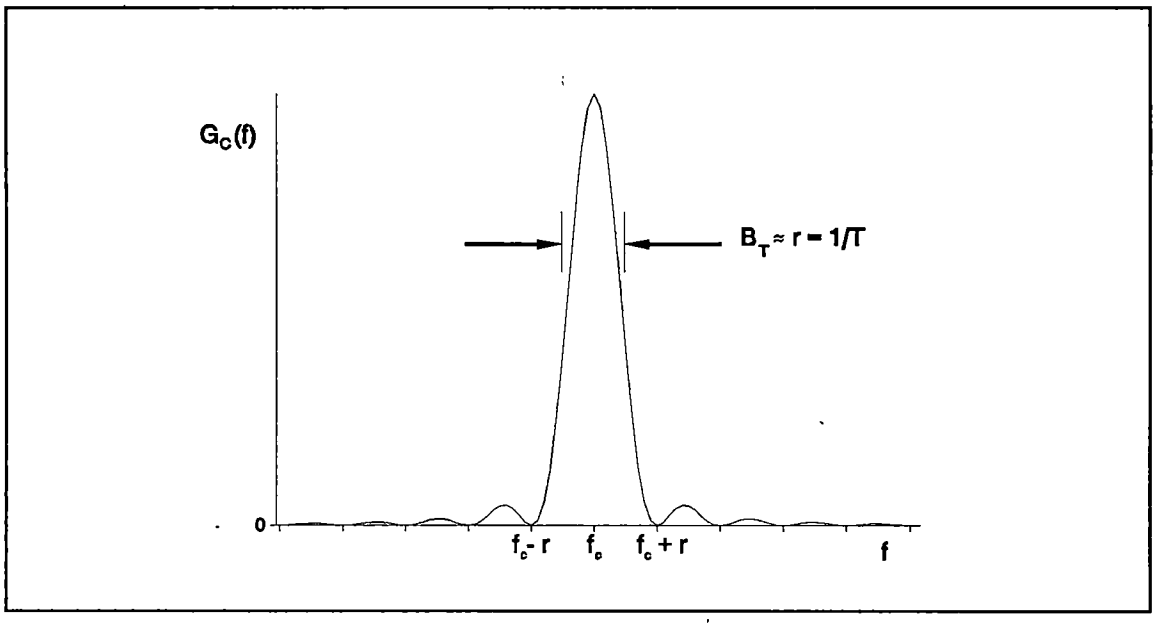

Figure A.2 DPSK Power Spectrum

APPENDIX B: Derivation of the Statistical Moments Used in the Bit Error Probability for Slow Frequency Hopping

The statistical moments needed in the mathematical model of the bit error probability in equation 6.7 will be derived in this appendix.

Because of the way equation 7A.26 in [25] is derived it is clear that we look at the case:

$$P_e = Pr\{ \xi_{max} < 0 \mid b_1^0 b_1^{-1} = 1 \} \tag{B.1}$$

so $b_1^{-1} = b_1^0 = 1$ or $b_1^{-1} = b_1^0 = -1$.

In the event when there is no hit, we can ignore the multi-user interference components in equation 5.24 - 5.25. We can now write for the outputs of the matched filters at the sampling points, for reference user 1:

$$X_{V_0} = \sum_{l=1}^{L} \sqrt{\tfrac{1}{8}P} \cdot \beta_{1l} \cos(\Phi_{1l}) \cdot \left[\int_0^{\tau_{1l}} b_1^{-1}(t-\tau_{1l})\,dt + \int_{\tau_{1l}}^{T_b} b_1^0(t-\tau_{1l})\,dt \right] + \eta \tag{B.2}$$

and

$$Y_{V_0} = \sum_{l=1}^{L} \sqrt{\tfrac{1}{8}P} \cdot \beta_{1l} \sin(\Phi_{1l}) \cdot \left[\int_0^{\tau_{1l}} b_1^{-1}(t-\tau_{1l})\,dt + \int_{\tau_{1l}}^{T_b} b_1^0(t-\tau_{1l})\,dt \right] + \nu \tag{B.3}$$

where b_1^{-1} and b_1^0 are the previous and current data bit respectively, and

$$\eta = \int_0^{T_b} n_c(s)ds \quad , \qquad \nu = \int_0^{T_b} n_s(s)ds \tag{B.4}$$

The noise samples η and ν are independent zero-mean Gaussian random variables with identical variance $\sigma_n^2 = \frac{1}{16}N_0T_b$. We assumed that the *jth* path between the transmitter of user 1 and the corresponding receiver is the reference path and all other paths constitute interference, so that without loss of generality, we can assume that $\tau_{1j}=0$ and $\Phi_{1j}=0$.

$$V_0 = \sqrt{\tfrac{1}{8}P}\cdot\beta_{max}\cdot T_b\cdot b_1^0 + \sqrt{\tfrac{1}{8}P}\cdot\left[\sum_{\substack{l=1\\ l\neq j}}^{L} X_{1l} + j\,Y_{1l}\right] + (\eta_1 + j\nu_1) \tag{B.5}$$

where

$$\begin{aligned} X_{1l} &= \left[\, b_1^{-1}\,\tau_{1l} + b_1^0\,(T_b - \tau_{1l})\,\right]\cdot\beta_{1l}\cos(\Phi_{1l}) \\ Y_{1l} &= \left[\, b_1^{-1}\,\tau_{1l} + b_1^0\,(T_b - \tau_{1l})\,\right]\cdot\beta_{1l}\sin(\Phi_{1l}) \end{aligned} \tag{B.6}$$

B.1 Derivation of m

m was defined in equation 6.13 as:

$$m = E\,[\,V_0 \mid \beta_{max},\, b_1^0\,] = E\,[\,V_{-1} \mid \beta_{max},\, b_1^{-1}\,] \tag{B.7}$$

After inserting equation B.5 we can rewrite this equation as:

$$m = \sqrt{\tfrac{1}{8}P}\cdot\beta_{max}\cdot T_b\cdot b_1^0 + \sqrt{\tfrac{1}{8}P}\cdot E\left[\sum_{\substack{l=1\\ l\neq j}}^{L} X_{1l} + jY_{1l}\right] \tag{B.8}$$

We may interchange the expectation and the summation because the multiple resolvable paths are statistically independent and also because [27, p.107]:

$$z = x + jy \;\Rightarrow\; E\,[z] = E\,[x] + jE\,[y] \tag{B.9}$$

Equation B.8 can be rewritten as:

$$E\left[\sum_{\substack{l=1\\ l\neq j}}^{L} X_{1l} + jY_{1l}\right] = \sum_{\substack{l=1\\ l\neq j}}^{L}\left[\,E\,[\,X_{1l}\,] + j\,E\,[\,Y_{1l}\,]\,\right] \tag{B.10}$$

When we take a closer look at equation B.10, we can see that

$$E[X_{1l}] = E\left[\left[b_1^{-1}\tau_{1l} + b_1^0(T_b-\tau_{1l})\right]\cdot\beta_{1l}\cos(\Phi_{1l})\right]$$
$$E[Y_{1l}] = E\left[\left[b_1^{-1}\tau_{1l} + b_1^0(T_b-\tau_{1l})\right]\cdot\beta_{1l}\sin(\Phi_{1l})\right] \quad (B.11)$$

When the number of resolvable paths is large we can write

$$E[X_{1l}] = E[\hat{b}\,\beta_{1l}\cos(\Phi_{1l})]$$
$$E[Y_{1l}] = E[\hat{b}\,\beta_{1l}\sin(\Phi_{1l})] \quad (B.12)$$

because

$$E[\tau_{1l}] = \tfrac{1}{2}T_b$$
$$E[b_1^{-1}\tau_{1l} + b_1^0(T_b-\tau_{1l})] = \tfrac{1}{2}T_b\left[b_1^{-1} + b_1^0\right] = \hat{b} \quad (B.13)$$

In equation B.12, the Rician distributed path gains are multiplied by the cosine or sine of a uniformly distributed phase. As noted in chapter 5, the Rician pdf describes the envelope of the sum of two orthogonal non zero-mean Gaussian random variables with identical variance σ^2, so this multiplication extracts one of the non zero-mean Gaussian variables.

When we assume that the Gaussian variables which constitute the Rician variable have equal means, the mean of the pdf is derived from equation 5.9 as $m_1 = m_2 = s/\sqrt{2}$. The pdf of this product is thus a non zero-mean Gaussian distribution, with mean $s/\sqrt{2}$ and variance σ^2:

$$p(y) = \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left[-\frac{\left(y-\frac{s}{\sqrt{2}}\right)^2}{2\sigma^2}\right] \quad (B.14)$$

This product is multiplied by the bit estimate $\hat{b}$ which value is T_b or $-T_b$ with probability ½. If we denote the product of the path gain and the cosine or sine of a uniformly distributed phase as x and the bit estimate as $\hat{b}$, then the pdf for $y = \hat{b}x$ can be written as [27, p. 96]:

$$p_y(y) = \frac{1}{|\hat{b}|}\cdot p_x\left(\frac{y}{\hat{b}}\right) \quad (B.15)$$

Because

$$p(\hat{b}) = \tfrac{1}{2}, \qquad \hat{b} \in \{ -T_b, T_b \}$$

we can write $p_y(y)$ as:

$$p_y(y) = \frac{1}{2T_b\sqrt{2\pi}\,\sigma} \exp\left[- \frac{\left(\frac{y}{T_b} - \frac{s}{\sqrt{2}}\right)^2}{2\sigma^2} \right] + \frac{1}{2T_b\sqrt{2\pi}\,\sigma} \exp\left[- \frac{\left(\frac{y}{T_b} + \frac{s}{\sqrt{2}}\right)^2}{2\sigma^2} \right] \quad \text{(B.17)}$$

where s is the constant component of the Rician fading channel.
It is clear from equation B.17 that $p_y(y)$ is the sum of two Gaussian pdf's with opposite mean values. The mean value of y, E[y], is then 0.

Now we know that $E[X_{1l}] = E[Y_{1l}] = 0$, we conclude that

$$\begin{aligned} m &= E[\, V_0 \mid \beta_{max}, b_1^0 \,] = E[\, V_{-1} \mid \beta_{max}, b_1^{-1} \,] \\ &= \sqrt{\tfrac{1}{8}P} \cdot \beta_{max} \cdot T_b \cdot b_1^0 = \sqrt{\tfrac{1}{8}P} \cdot \beta_{max} \cdot T_b \cdot b_1^{-1} \end{aligned} \quad \text{(B.18)}$$

B.2 Derivation of μ_0

μ_0 was defined in equation 6.14 as:

$$\mu_0 = \text{var}\,(\, V_0 \mid L \,) = E\left[\, (V_0 - m)\,(V_0 - m)^* \mid L \,\right] \quad \text{(B.19)}$$

where the superscripted asterisk * represents the complex conjugate.
After inserting equations B.5 and B.18 we can write equation B.19 as:

$$\mu_0 = E\left[\left(\sum_{\substack{l=1 \\ l \neq j}}^{L} \sqrt{\tfrac{1}{8}P}\, X_{1l} \right)^2 + \left(\sum_{\substack{l=1 \\ l \neq j}}^{L} \sqrt{\tfrac{1}{8}P}\, Y_{1l} \right)^2 \right] + 2\sigma_n^2 \quad \text{(B.20)}$$

Because

$$E\,[\, b_1^l\, b_1^l \,] = 1 \quad \text{(B.21)}$$

and

$$E[\, b_1^{-1}\, b_1^0 \,] = 1\,, \qquad E[\, \tau_{11} \,] = \tfrac{1}{2}T_b \tag{B.22}$$

the expectation and summation can be interchanged and equation B.20 can be written as:

$$\mu_0 = \tfrac{1}{8}P \cdot T_b^{\,2} \cdot \sum_{\substack{p=1\\p\neq j}}^{L} \sum_{\substack{q=1\\q\neq j}}^{L} E[\, \beta_{1p}\cos\Phi_{1p}\ \beta_{1q}\cos\Phi_{1q} + \beta_{1p}\sin\Phi_{1p}\ \beta_{1q}\sin\Phi_{1q} \,] + 2\sigma_n^2 \tag{B.23}$$

As proven in the preceding section, the products $\beta\cos\Phi$ and $\beta\sin\Phi$ are non zero-mean Gaussian random variables. The pdf's of the squares of these products are non-central chi-square distributions with 1 degree of freedom and are given by [27]:

$$p_y(y) = \frac{1}{2\sqrt{y}}\left[\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left[-\frac{\left(\sqrt{y}-\frac{s}{\sqrt{2}}\right)^2}{2\sigma^2}\right] + \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left[-\frac{\left(\sqrt{y}+\frac{s}{\sqrt{2}}\right)^2}{2\sigma^2}\right]\right] = \frac{1}{\sqrt{2\pi y}\,\sigma}\exp\left[-\frac{y+\tfrac{1}{2}s^2}{2\sigma^2}\right]\cdot\cosh\left(\frac{\sqrt{y}\,s}{\sqrt{2}\,\sigma^2}\right) \tag{B.24}$$

The expectation of this distribution is given by [25, p. 29] as:

$$E[\, y \,] = \sigma^2 + \tfrac{1}{2}s^2 \tag{B.25}$$

β_{1p} and β_{1q} are equally distributed Rician random variables, and Φ_{1p} and Φ_{1q} are equally distributed uniform random variables, so we can write equation B.23 as:

$$\mu_0 = \tfrac{1}{8}P \cdot T_b^{\,2} \cdot \left[\left(\sum_{\substack{p=1\\p\neq j}}^{L} E[\,(\beta_{1p}\cos\Phi_{1p})^2\,] + E[\,(\beta_{1p}\sin\Phi_{1p})^2\,]\right) + \left(\sum_{\substack{p=1\\p\neq j}}^{L}\sum_{\substack{q=1\\q\neq p\neq j}}^{L} E[\,\beta_{1p}\cos\Phi_{1p}\ \beta_{1p}\cos\Phi_{1q}\,] + E[\,\beta_{1p}\sin\Phi_{1p}\ \beta_{1q}\sin\Phi_{1q}\,]\right)\right] + 2\sigma_n^2 \tag{B.26}$$

We conclude that the expression for μ_0 is:

$$\mu_0 = \tfrac{1}{4}P \cdot T_b^2 \cdot \left((L-1) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + (L-1)(L-2) \cdot \tfrac{1}{2}s^2 \right) + \tfrac{1}{8}N_0 T_b \qquad (B.27)$$

B.3 Derivation of μ_{-1}

μ_{-1} was defined in equation 6.15 as:

$$\mu_{-1} = \mathrm{var}\left(V_{-1} \mid L \right) = E\left[(V_{-1} - m)\,(V_{-1} - m)^* \mid L \right] \qquad (B.28)$$

After inserting equations B.5 and B.18 we can write equation B.28 as:

$$\mu_{-1} = E\left[\left(\sum_{\substack{l=1 \\ l \neq j}}^{L} \sqrt{\tfrac{1}{8}P}\, X_{1l} \right)^2 + \left(\sum_{\substack{l=1 \\ l \neq j}}^{L} \sqrt{\tfrac{1}{8}P}\, Y_{1l} \right)^2 \right] + 2\sigma_n^2 \qquad (B.29)$$

Because

$$\begin{aligned} E\left[b_1^l\, b_1^j \right] &= 0\,, \quad i \neq j \\ E\left[b_1^l\, b_1^l \right] &= 1 \end{aligned} \qquad (B.30)$$

and

$$E\left[b_1^{-1}\, b_1^0 \right] = 1\,, \qquad E\left[\tau_{1l} \right] = \tfrac{1}{2}T_b \qquad (B.31)$$

the expectation and summation can be interchanged by writing equation B.29 as:

$$\mu_{-1} = \tfrac{1}{8}P \cdot \tfrac{1}{2}T_b^2 \cdot \sum_{\substack{p=1 \\ p \neq j}}^{L} \sum_{\substack{q=1 \\ q \neq j}}^{L} E\left[\beta_{1p}\cos\Phi_{1p}\ \beta_{1q}\cos\Phi_{1q} + \beta_{1p}\sin\Phi_{1p}\ \beta_{1q}\sin\Phi_{1q} \right] + 2\sigma_n^2 \qquad (B.32)$$

Following the derivation of the expectation in the preceding section, β_{1p} and β_{1q} are equally distributed Rician random variables, and Φ_{1p} and Φ_{1q} are equally distributed uniform random variables, so we can write equation B.32 as:

$$\mu_{-1} = \frac{1}{16}P \cdot T_b^2 \cdot \left[\left(\sum_{\substack{p=1 \\ p \neq j}}^{L} E\left[\left(\beta_{1p}\cos\Phi_{1p} \right)^2 \right] + E\left[\left(\beta_{1p}\sin\Phi_{1p} \right)^2 \right] \right) + \left(\sum_{\substack{p=1 \\ p \neq j}}^{L} \sum_{\substack{q=1 \\ q \neq p \neq j}}^{L} E\left[\beta_{1p}\cos\Phi_{1p}\ \beta_{1p}\cos\Phi_{1q} \right] + E\left[\beta_{1p}\sin\Phi_{1p}\ \beta_{1q}\sin\Phi_{1q} \right] \right) \right] + 2\sigma_n^2 \tag{B.33}$$

The expression for μ_{-1} is obtained as:

$$\mu_{-1} = \tfrac{1}{8}P \cdot T_b^2 \cdot \left((L-1) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + (L-1)(L-2) \cdot \tfrac{1}{2}s^2 \right) + \tfrac{1}{8}N_0 T_b \tag{B.34}$$

B.4 Derivation of μ

μ was defined in equation 6.16 as:

$$\mu = E\left[(V_0 - m)\,(V_{-1} - m)^* \mid L \right] \tag{B.35}$$

where

$$\begin{aligned} V_0 - m &= \sqrt{\tfrac{1}{8}P} \cdot \left[\sum_{\substack{l=1 \\ l \neq j}}^{L} X_{1l}^0 + jY_{1l}^0 \right] + (\eta_1 + j\nu_1) \\ V_{-1} - m &= \sqrt{\tfrac{1}{8}P} \cdot \left[\sum_{\substack{l=1 \\ l \neq j}}^{L} X_{1l}^{-1} + jY_{1l}^{-1} \right] + (\eta_1 + j\nu_1) \end{aligned} \tag{B.36}$$

Inserting equation B.36, we can write equation B.35 as:

$$\begin{aligned} \mu = {} & \tfrac{1}{8}P \cdot E\left[\sum_{\substack{l=1 \\ l \neq j}}^{L} X_{1l}^0 \sum_{\substack{l=1 \\ l \neq j}}^{L} X_{1l}^{-1} + \sum_{\substack{l=1 \\ l \neq j}}^{L} Y_{1l}^0 \sum_{\substack{l=1 \\ l \neq j}}^{L} Y_{1l}^{-1} \right] \\ & + j\,\tfrac{1}{8}P \cdot E\left[\sum_{\substack{l=1 \\ l \neq j}}^{L} Y_{1l}^0 \sum_{\substack{l=1 \\ l \neq j}}^{L} X_{1l}^{-1} + \sum_{\substack{l=1 \\ l \neq j}}^{L} X_{1l}^0 \sum_{\substack{l=1 \\ l \neq j}}^{L} Y_{1l}^{-1} \right] \end{aligned} \tag{B.37}$$

The noise terms are dropped here because these are four statistically independent zero-mean Gaussian random variables. The expectation and summation can be interchanged by writing

equation B.37 as:

$$\mu = \tfrac{1}{8}P \cdot \left[\sum_{\substack{p=1 \\ p \neq j}}^{L} \sum_{\substack{q=1 \\ q \neq j}}^{L} E\left[X_{1p}^{0} X_{1q}^{-1} + Y_{1p}^{0} Y_{1q}^{-1} \right] + j\, E\left[Y_{1p}^{0} X_{1q}^{-1} + X_{1p}^{0} Y_{1q}^{-1} \right] \right] \tag{B.38}$$

where

$$\begin{aligned} E\left[X_{1p}^{0} X_{1q}^{-1} \right] &= \tfrac{1}{2}T_b^2\, E\left[\beta_{1p}\cos\Phi_{1p}\ \beta_{1q}\cos\Phi_{1q} \right] \\ E\left[Y_{1p}^{0} Y_{1q}^{-1} \right] &= \tfrac{1}{2}T_b^2\, E\left[\beta_{1p}\sin\Phi_{1p}\ \beta_{1q}\sin\Phi_{1q} \right] \\ E\left[Y_{1p}^{0} X_{1q}^{-1} \right] &= \tfrac{1}{2}T_b^2\, E\left[\beta_{1p}\sin\Phi_{1p}\ \beta_{1q}\cos\Phi_{1q} \right] \\ E\left[X_{1p}^{0} Y_{1q}^{-1} \right] &= \tfrac{1}{2}T_b^2\, E\left[\beta_{1p}\cos\Phi_{1p}\ \beta_{1q}\sin\Phi_{1q} \right] \end{aligned} \tag{B.39}$$

because

$$\begin{aligned} E\left[b_1^{l}\, b_1^{j} \right] &= 0, \quad i \neq j \\ E\left[b_1^{l}\, b_1^{l} \right] &= 1 \end{aligned} \tag{B.40}$$

and

$$E\left[b_1^{-1}\, b_1^{0} \right] = 1, \qquad E\left[\tau_{11} \right] = \tfrac{1}{2}T_b \tag{B.41}$$

Using the trigonometric identities and the fact that sin(-x) = -sin(x), the imaginary part of equation B.38 becomes zero and we can rewrite equation B.38

$$\begin{aligned} \mu = \tfrac{1}{16}P \cdot T_b^2 \cdot \Bigg[& \sum_{\substack{p=1 \\ p \neq j}}^{L} \sum_{\substack{q=1 \\ q \neq j}}^{L} E\left[\beta_{1p}\cos\Phi_{1p} \cdot \beta_{1q}\cos\Phi_{1q} \right] \\ & + E\left[\beta_{1p}\sin\Phi_{1p} \cdot \beta_{1q}\sin\Phi_{1q} \right] \Bigg] \end{aligned} \tag{B.42}$$

Following the derivation of the expectation in section B.2, β_{1p} and β_{1q} are equally distributed Rician random variables, and Φ_{1p} and Φ_{1q} are equally distributed uniform random variables, so we can write equation B.42 as:

$$\mu = \frac{1}{16}P \cdot T_b^2 \cdot \left[\left(\sum_{\substack{p=1 \\ p \neq j}}^{L} E\left[\left(\beta_{1p}\cos\Phi_{1p} \right)^2 \right] + E\left[\left(\beta_{1p}\sin\Phi_{1p} \right)^2 \right] \right) + \left(\sum_{\substack{p=1 \\ p \neq j}}^{L} \sum_{\substack{q=1 \\ q \neq p \neq j}}^{L} E\left[\beta_{1p}\cos\Phi_{1p}\ \beta_{1p}\cos\Phi_{1q} \right] + E\left[\beta_{1p}\sin\Phi_{1p}\ \beta_{1q}\sin\Phi_{1q} \right] \right) \right] \quad \text{(B.43)}$$

The expression for μ is obtained as:

$$\mu = \tfrac{1}{8}P \cdot T_b^2 \cdot \left((L-1) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + (L-1)(L-2) \cdot \tfrac{1}{2}s^2 \right) \quad \text{(B.44)}$$

APPENDIX C: Derivation of the Statistical Moments for the Gaussian Approximation of the Multi-User Interference

The statistical moments needed in the Gaussian approximation of the multi-user interference in equations 9.49-9.54 will be derived in this appendix.

First we will derive the expectations of X_k^2, $\hat{X}_k^2$, Y_k^2 and $\hat{Y}_k^2$ with respect to the path delay. These were defined in chapter 8 as:

$$X_k = \sum_{l=1}^{L} R_{kl}(\tau_{kl})\ \beta_{kl} \cos(\Phi_{kl}) \quad , Y_k = \sum_{l=1}^{L} R_{kl}(\tau_{kl})\ \beta_{kl} \sin(\Phi_{kl})$$

$$\hat{X}_k = \sum_{l=1}^{L} \hat{R}_{kl}(\tau_{kl})\ \beta_{kl} \cos(\Phi_{kl}) \quad , \hat{Y}_k = \sum_{l=1}^{L} \hat{R}_{kl}(\tau_{kl})\ \beta_{kl} \sin(\Phi_{kl}) \tag{C.1}$$

The path gains and path phases are statistically independent, so we can write:

$$E_\tau\left[E\left[X_k^2\right]\right] = E_\tau\left[E\left[Y_k^2\right]\right]$$

$$= E_\tau\left[\sum_{l=1}^{L} R_{kl}^2(\tau_{kl}) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + \sum_{l=1}^{L} R_{kl}(\tau_{kl}) \sum_{\substack{p=1 \\ p\neq l}}^{L} R_{kl}(\tau_{kp}) \cdot \tfrac{1}{2}s^2\right] \tag{C.2}$$

$$= L \cdot \Gamma_k^{1,1} \cdot (\sigma^2 + \tfrac{1}{2}s^2) + L(L-1) \cdot \upsilon_k^2 \cdot \tfrac{1}{2}s^2$$

$$E_\tau\left[E\left[\hat{X}_k^2\right]\right] = E_\tau\left[E\left[\hat{Y}_k^2\right]\right]$$

$$= E_\tau\left[\sum_{l=1}^{L} \hat{R}_{kl}^2(\tau_{kl}) \cdot (\sigma^2 + \tfrac{1}{2}s^2) + \sum_{l=1}^{L} \hat{R}_{kl}(\tau_{kl}) \sum_{\substack{p=1 \\ p\neq l}}^{L} \hat{R}_{kl}(\tau_{kp}) \cdot \tfrac{1}{2}s^2\right] \tag{C.3}$$

$$= L \cdot \Gamma_k^{1,0} \cdot (\sigma^2 + \tfrac{1}{2}s^2) + L(L-1) \cdot \hat{\upsilon}_k^2 \cdot \tfrac{1}{2}s^2$$

In the following pages we will derive the expectations needed in the calculation of var(μ_0).

$$
\begin{aligned}
&E_\tau\left[\left(E\left[2X_k^2\right]\right)^2\right] = 4\cdot E_\tau\left[\left(E\left[X_k^2\right]\right)^2\right] \\
&= 4\cdot E_\tau\Bigg[(\sigma^2+\tfrac{1}{2}s^2)^2\cdot\left(\sum_{l=1}^{L}R_{k1}^4(\tau_{kl})+\sum_{l=1}^{L}R_{k1}^2(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}^2(\tau_{kp})\right) \\
&\quad+2\cdot(\sigma^2+\tfrac{1}{2}s^2)\cdot\tfrac{1}{2}s^2\cdot\Bigg(2\cdot\sum_{l=1}^{L}R_{k1}^3(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}(\tau_{kp})+ \\
&\qquad+\sum_{l=1}^{L}R_{k1}^2(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}(\tau_{kp})\sum_{\substack{q=1\\q\neq p\neq l}}^{L}R_{k1}(\tau_{kq})\Bigg) \\
&\quad+(\tfrac{1}{2}s^2)^2\cdot\Bigg(2\cdot\sum_{l=1}^{L}R_{k1}^2(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}^2(\tau_{kp}) \\
&\qquad+4\cdot\sum_{l=1}^{L}R_{k1}^2(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}(\tau_{kp})\sum_{\substack{q=1\\q\neq p\neq l}}^{L}R_{k1}(\tau_{kq}) \\
&\qquad+\sum_{l=1}^{L}R_{k1}(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}(\tau_{kp})\sum_{\substack{q=1\\q\neq p\neq l}}^{L}R_{k1}(\tau_{kq})\sum_{\substack{r=1\\r\neq q\neq p\neq l}}^{L}R_{k1}(\tau_{kr})\Bigg)\Bigg]
\end{aligned}
\tag{C.4}
$$

By making use of equations 9.42 - 9.48 we can write this expression as:

$$
\begin{aligned}
&E_\tau\left[\left(E\left[2X_k^2\right]\right)^2\right] = 4\cdot E_\tau\left[\left(E\left[X_k^2\right]\right)^2\right] \\
&= 4\cdot\Big[(\sigma^2+\tfrac{1}{2}s^2)^2\cdot\left(L\cdot\Gamma_k^{2,2}+L(L-1)\cdot\left(\Gamma_k^{1,1}\right)^2\right) \\
&\quad+(\sigma^2+\tfrac{1}{2}s^2)\cdot\tfrac{1}{2}s^2\cdot\left(4\cdot L(L-1)\cdot\zeta_k\cdot\upsilon_k\right. \\
&\qquad\left.+2\cdot L(L-1)(L-2)\cdot\Gamma_k^{1,1}\cdot\upsilon_k^2\right) \\
&\quad+(\tfrac{1}{2}s^2)^2\cdot\left(2\cdot L(L-1)\left(\Gamma_k^{1,1}\right)^2+4\cdot L(L-1)(L-2)\cdot\Gamma_k^{1,1}\cdot\upsilon_k^2\right. \\
&\qquad\left.+L(L-1)(L-2)(L-3)\cdot\upsilon_k^4\right)\Big]
\end{aligned}
\tag{C.5}
$$

The expression for $E_\tau\left[\left(E\left[2\hat{X}_k^2\right]\right)^2\right]$ can be determined by replacing $\Gamma_k^{2,2}$ by $\Gamma_k^{2,0}$, $\Gamma_k^{1,1}$ by $\Gamma_k^{1,0}$, υ_k by $\hat{\upsilon}_k$ and ζ_k by $\hat{\zeta}_k$.

$$
\begin{aligned}
&E_\tau\left[\,2E\left[\,2X_k^2\,\right]E\left[\,2\hat{X}_k^2\,\right]\right] = 8\cdot E_\tau\left[\,E\left[\,X_k^2\,\right]E\left[\,\hat{X}_k^2\,\right]\right] \\
&= 8\cdot E_\tau\Bigg[\,(\sigma^2+\tfrac{1}{2}s^2)^2\cdot\Bigg(\sum_{l=1}^{L}R_{k1}^2(\tau_{kl})\sum_{p=1}^{L}\hat{R}_{k1}^2(\tau_{kp})\Bigg) \\
&\quad+(\sigma^2+\tfrac{1}{2}s^2)\cdot\tfrac{1}{2}s^2\cdot\Bigg(2\cdot\sum_{l=1}^{L}R_{k1}^2(\tau_{kl})\hat{R}_{k1}(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}\hat{R}_{k1}(\tau_{kp}) + \\
&\qquad+\sum_{l=1}^{L}R_{k1}^2(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}\hat{R}_{k1}(\tau_{kp})\sum_{\substack{q=1\\q\neq p\neq l}}^{L}\hat{R}_{k1}(\tau_{kq}) \\
&\qquad+2\cdot\sum_{l=1}^{L}\hat{R}_{k1}^2(\tau_{kl})R_{k1}(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}(\tau_{kp}) \\
&\qquad+\sum_{l=1}^{L}\hat{R}_{k1}^2(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}(\tau_{kp})\sum_{\substack{q=1\\q\neq p\neq l}}^{L}R_{k1}(\tau_{kq})\Bigg) \\
&\quad+(\tfrac{1}{2}s^2)^2\cdot\Bigg(\sum_{l=1}^{L}R_{k1}(\tau_{kl})\sum_{\substack{p=1\\p\neq l}}^{L}R_{k1}(\tau_{kp})\sum_{q=1}^{L}\hat{R}_{k1}(\tau_{kq})\sum_{\substack{r=1\\r\neq q}}^{L}\hat{R}_{k1}(\tau_{kr})\Bigg)\Bigg]
\end{aligned}
\tag{C.6}
$$

By making use of equations 9.42 - 9.48 we can write this expression as:

$$
\begin{aligned}
&E_\tau\left[\,2E\left[\,2X_k^2\,\right]E\left[\,2\hat{X}_k^2\,\right]\right] = 8\cdot E_\tau\left[\,E\left[\,X_k^2\,\right]E\left[\,\hat{X}_k^2\,\right]\right] \\
&= 8\cdot\Big[\,(\sigma^2+\tfrac{1}{2}s^2)^2\cdot\left(L^2\cdot\Gamma_k^{1,1}\cdot\Gamma_k^{1,0}\right) \\
&\quad+(\sigma^2+\tfrac{1}{2}s^2)\cdot\tfrac{1}{2}s^2\cdot\left(2\cdot L(L-1)\cdot\gamma_k\cdot\hat{\upsilon}_k+L(L-1)(L-2)\cdot\Gamma_k^{1,1}\cdot\hat{\upsilon}_k^2\right. \\
&\qquad\left.+2\cdot L(L-1)\cdot\hat{\gamma}_k\cdot\upsilon_K+L(L-1)(L-2)\cdot\Gamma_k^{1,0}\cdot\upsilon_k^2\right) \\
&\quad+(\tfrac{1}{2}s^2)^2\cdot\left(L^2\cdot(L-1)^2\cdot\upsilon_k^2\cdot\hat{\upsilon}_k^2\right)\Big]
\end{aligned}
\tag{C.7}
$$

APPENDIX D: Computer Programmes Used to Derive the Computational Results for the Bit Error Probability

D.1 Slow Frequency Hopping with Selection Diversity

```
{$N+}      { Compile for Mathematical Co-Processor }

program BitErrorProbability_SFH_SD(input,output);

uses  Dos, Crt;

const
  TNArraySize = 50;                     { Size of the vectors }

type
  TNvector = array[0..TNArraySize] of extended;

var
  Rice                       : extended;      { Rician Parameter }
  M                          : integer;       { Order of Diversity }
  L                          : integer;       { Number of Resolvable Paths }
  P                          : extended;      { Received Signal Power }
  s, s_square                : extended;      { LOS Power }
  sigma_square               : extended;      { Multipath Power }
  snr_db                     : integer;       { Signal to Noise Ratio in dB }
  snr                        : extended;      { Signal to Noise Ratio }
  Pe                         : extended;      { Average Bit Error Probability }
  a, b                       : extended;      { Variables for Pe }
  PrIt                       : integer;       { Flags if something went wrong }
  PrEr                       : byte;          { Tolerance variable }
  PrValue                    : extended;      { Outcome of the integral }
  counter                    : integer;       { Number of the subintegral }
  point1, point2             : extended;      { Start and End point of the subintegral }
  ValuePoint1, ValuePoint2 : extended;        { Values of Betapdf of these points }
  done                       : boolean;       { Check if calculations are finished }
  uitvoer                    : text;          { Output file }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

{$I Romberg.pas}

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

function power(a,b:extended):extended;

{--------------------------------------------------------}
{-                  Calculates a^b                      -}
{--------------------------------------------------------}

var
  hlp : extended;

begin
  if a=0 then power:=0 else
  begin
    if a>0 then
    begin
      power:=exp(b*ln(a));
    end
    else
    begin
```

```
      hlp:=exp(b*ln(abs(a)));
      if b/2 = trunc(b/2) then power:=hlp else power:=-hlp;
    end;
  end;
end; { function power }

{--------------------------------------------------------------}

function bessel(y : extended) : extended;

{------------------------------------------------}
{-  modified bessel function of first kind and zero  -}
{-     order. Formulas from Abramowitz pp. 378       -}
{-              formulas 9.8.1 & 9.8.2               -}
{------------------------------------------------}

var
  t,b   :extended;

begin
  t:=y/3.75;
  if y<=3.75 then
  begin
    bessel:= 1+3.5156229*sqr(t)+3.0899424*power(t,4)
             +1.2067492*power(t,6)+0.2659732*power(t,8)
             +0.0360768*power(t,10)+0.0045813*power(t,12);
  end
  else
  begin
    b:= 0.39894228+0.01328592/t+0.00225319/sqr(t)
        -0.00157565/(power(t,3))+0.00916281/(power(t,4))
        -0.02057706/(power(t,5))+0.02635537/(power(t,6))
        -0.01647633/(power(t,7))+0.00392377/(power(t,8));
    bessel:=b/(sqrt(y)*exp(-y));
  end;
end; { function bessel }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

{$F+}
function Gamma_Cdf(z:extended) : extended;

{------------------------------------------------}
{-       calculates the cdf of the non-central      -}
{-              chi-square distribution             -}
{------------------------------------------------}

begin
  Gamma_Cdf:=(1/(2*snr*sigma_square))*exp(-(s_square+(z/snr))/(2*sigma_square))*
           bessel((s*sqrt(z/snr))/sigma_square);
end; { function Gamma_Cdf }
{$F-}

function Gamma_Pdf(y:extended) : extended;

{------------------------------------------------}
{-       calculates the pdf of the bit snr          -}
{------------------------------------------------}

var
   IntValue, hlp : extended;
   it            : integer;
   er            : byte;

begin
  Romberg(0,y,1e-7,1000,IntValue,it,er,@Gamma_Cdf);
  Gamma_Pdf:=M*power(IntValue,M-1)*(1/(2*snr*sigma_square))
             *exp(-(s_square+(y/snr))/(2*sigma_square))
             *bessel((s*sqrt(y/snr))/sigma_square);
end; { function Gamma_Pdf }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}
```

```
{$F+}
function Qab_func(x:extended) : extended;

{-------------------------------------------------}
{-       calculates the marcum Q-function        -}
{-------------------------------------------------}

begin
  Qab_func:=x*exp(-(sqr(a)+sqr(x))/2)*bessel(a*x);
end; { function Qab_func }
{$F-}

function Qab(Qb:extended) : extended;

var
  Qvalue : extended;
  it     : integer;
  er     : byte;

begin
  Romberg(Qb,30,1e-6,1000,Qvalue,it,er,@Qab_func);
  Qab:=Qvalue;
end; { function Qab }

{-------------------------------------------------------------}

function P2(g:extended) : extended;

{-------------------------------------------------}
{-       calculates the Bit Error Probability    -}
{-               as a function of g              -}
{-------------------------------------------------}

var
  hlp, hlp1, hlp2 : extended;
  factor : extended;

begin
  hlp := snr*(((L-1)*P)+((L-1)*(L-2)*s_square/2));
  hlp1 := sqrt(g/(4*hlp+2));
  hlp2 := sqrt(g/(2*hlp+2));
  a := abs(hlp1 - hlp2);
  b := hlp1 + hlp2;
  factor := (1+sqrt(sqr(hlp)/(2*sqr(hlp)+3*hlp+1)))/2;
  P2:=(Qab(b)-factor*exp(-(sqr(a)+sqr(b))/2)*bessel(a*b));
end; { function P2 }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

{$F+}
function PrError(gamma:extended) : extended;

{-------------------------------------------------}
{-       calculates the Bit Error Probability    -}
{-               as a function of SNR            -}
{-------------------------------------------------}

begin
  PrError:=P2(gamma)*Gamma_Pdf(gamma);
end; { function PrError }
{$F-}

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

procedure GetData(var GetRice        : extended;
                  var GetM           : integer;
                  var GetL           : integer;
                  var GetUitvoer     : text);

{-------------------------------------------------}
{-  This procedure assigns values to the above   -}
{-       variables from keyboard input           -}
{-------------------------------------------------}
```

```
var
  PlaceX, PlaceY: integer;

procedure GetRicianParameter (var GetRice : extended);

var
  confirm : char;

{-------------------------------------------------------}
{-   This procedure reads in the Rician Parameter     -}
{-                from the keyboard                    -}
{-------------------------------------------------------}

begin
  Writeln;
  GetRice := 6.8;
  Write('Rician Parameter : ' : 24);
  TextColor(white);
  Write(GetRice:2:1,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetRice:=GetRice else Readln(GetRice);
  GotoXY(PlaceX-8,PlaceY);
  if (GetRice=0) then Writeln(0,'        ')
  else Writeln(GetRice:2:1,' dB   ');
end; { procedure GetRicianParameter }

{-------------------------------------------------------------------}

procedure GetOrderOfDiversity (var GetM : integer);

var
  confirm : char;

{-------------------------------------------------------}
{-   This procedure reads in the Order of Diversity   -}
{-                 from the keyboard                   -}
{-------------------------------------------------------}

begin
  Writeln;
  GetM := 4;
  Write('Order of Diversity : ' : 24);
  TextColor(white);
  Write(GetM,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetM:=GetM else Readln(GetM);
  GotoXY(PlaceX-6,PlaceY);
  Writeln(GetM,'      ');
end; { procedure GetOrderOfDiversity }

{-------------------------------------------------------------------}

procedure GetResolvablePaths (var GetL: integer);

var
  confirm : char;

{-------------------------------------------------------}
{-      This procedure reads in the number of        -}
{-        resolvable paths from the keyboard         -}
{-------------------------------------------------------}

begin
  Writeln;
  GetL := 2;
  Write('Resolvable Paths : ' : 24);
  TextColor(white);
  Write(GetL,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
```

```
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetL:=GetL else Readln(GetL);
  GotoXY(PlaceX-6,PlaceY);
  Writeln(GetL,'      ');
end; { procedure GetResolvablePaths }

{-------------------------------------------------------------------}

procedure GetOutputFile (var GetUitvoer : text);

var
  test, confirm : string;
  answer        : char;

{-----------------------------------------------------}
{-      This procedure reads in the outputfile       -}
{-          for the data from the keyboard           -}
{-----------------------------------------------------}

begin
  repeat
    Writeln;
    Write('   Output File : ');
    test:='';
    Readln(confirm);
    test:=fsearch(confirm,'c:');
    if test <> '' then
    begin
      Textcolor(White);
      writeln('   This file already exists !');
      writeln('   Overwrite this file ?  Y/N ');
      answer:=ReadKey;
      clrscr; writeln;
      Textcolor(Yellow);
      if (answer='y') or (answer='Y') then test:='';
    end;
  until test = '';
  clrscr;
  Writeln; Writeln;
  Write('   Output File : ',confirm);
  assign(GetUitvoer,confirm);
end; { procedure GetOutputFile }

{-------------------------------------------------------------------}

begin { procedure GetData }
  TextBackground(Black);
  TextColor(Cyan);
  ClrScr;
  writeln;
  Writeln('SLOW FREQUENCY HOPPING' : 32 );
  TextColor(Red);
  Write(' SELECTION DIVERSITY ' : 31 );

  Window(4,5,39,17);
  TextBackground(Blue);
  Textcolor(Yellow);
  ClrScr;
  Writeln; Writeln; Writeln;
  GetRicianParameter(GetRice);
  GetOrderOfDiversity(GetM);
  GetResolvablePaths(GetL);

  Window(4,20,39,24);
  TextBackground(Red);
  Textcolor(Yellow);
  ClrScr;
  Writeln;
  GetOutputFile(GetUitvoer);
  rewrite(GetUitvoer);
  close(GetUitvoer);

  Window(44,1,76,4);
  TextBackground(5);
  Textcolor(White);
```

```
  ClrScr;
  Writeln;
  Write('  SNR  Avg Bit Error Probability ');

  Window(44,4,76,24);
  TextBackground(5);
  Textcolor(Yellow);
  ClrScr;
end; { Procedure GetData }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

begin { Main Program }
  ClrScr;
  GetData(Rice,M,L,uitvoer);
  if (Rice<>0) then Rice:=power(10,Rice/10);
  P:=Rice+1;
  s_square:=(2*Rice*P)/(Rice+1);
  s:=sqrt(s_square);
  sigma_square:=P/(Rice+1);

  For snr_db:=1 to 30 do
  begin
    snr:=power(10,snr_db/10);
    done:=false;
    Pe:=0;
    point1:=0;
    ValuePoint1:=0;
    for counter:=1 to 15 do
    begin
      case trunc(snr_db/5) of
        0 : point2 := 10*counter;        {1-4}
        1 : point2 := 50*counter;        {5-9}
        2 : point2 := 100*counter;       {10-14}
        3 : point2 := 250*counter;       {15-19}
        4 : point2 := 1000*counter;      {20-24}
      else  point2 := 2500*counter;      {25-30}
      end;
      ValuePoint2:=Gamma_Pdf(point2);
      if (ValuePoint1 > (1E-5/(power(10,trunc(snr_db/10)))))
         or (ValuePoint2 > (1E-5/(power(10,trunc(snr_db/10))))) then
      begin
        done:=true;
        Romberg(point1,point2,1e-6,1000,PrValue,PrIt,PrEr,@PrError);
        Pe:=Pe+PrValue;
        writeln('  #',counter:2,'  ',PrValue);
      end
      else if done=true then counter:=15;
      point1:=point2;
      ValuePoint1:=ValuePoint2;
    end;
    Writeln('       ---------------------- +');
    write('  ');
    Textbackground(red);
    Textcolor(white);
    write(snr_db:2,'   ',Pe);
    Textbackground(5);
    Textcolor(yellow);
    Writeln; Writeln;
    append(uitvoer);
    Writeln(uitvoer,Pe);
    close(uitvoer);
  end;
  Window(63,25,78,25);
  TextBackground(Black);
  Textcolor(White);
end.
```

D.2 Slow Frequency Hopping with Maximal Ratio Combining

```
{$N+}      { Compile for Mathematical Co-Processor }

program BitErrorProbability_SFH_MRC(input,output);

uses  Dos, Crt;

const
  TNArraySize = 50;                  { Size of the vectors }

type
  TNvector = array[0..TNArraySize] of extended;

var
  Rice           : extended;         { Rician Parameter }
  M              : integer;          { Order of Diversity }
  P              : extended;         { Received Signal Power }
  snr_db         : integer;          { Signal to Noise Ratio in dB }
  snr            : extended;         { Signal to Noise Ratio }
  s, s_square    : extended;         { LOS Power }
  sigma_square   : extended;         { Reflected Power }
  Pe             : extended;         { Average Bit Error Probability }
  PrIt, BessIt   : integer;          { Flags if something went wrong }
  PrEr, BessEr   : byte;             { Tolerance variable }
  PrValue        : extended;         { Outcome of the integral }
  counter        : integer;          { Number of the subintegral }
  point1, point2 : extended;         { Start and End point of the subintegral }
  uitvoer        : text;             { Output file }
  z              : extended;
  v              : integer;

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

{$I Romberg.pas}

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

function power(a,b:extended):extended;

{---------------------------------------------------}
{-                 Calculates a^b                  -}
{---------------------------------------------------}

var
  hlp : extended;

begin
  if a=0 then power:=0 else
  begin
    if a>0 then
    begin
      power:=exp(b*ln(a));
    end
    else
    begin
      hlp:=exp(b*ln(abs(a)));
      if b/2 = trunc(b/2) then power:=hlp else power:=-hlp;
    end;
  end;
end; { function power }

{-----------------------------------------------------------------}

function fac(n:integer):extended;

{---------------------------------------------------}
{-                   calculates n !                -}
{---------------------------------------------------}

var
  som_fac : extended;
```

```
begin
  som_fac :=1.0;
  while n>1 do
  begin
    som_fac:=som_fac*(n);
    n:=n-1;
  end;
  fac:=som_fac;
end; { function fac }

{--------------------------------------------------------------}

function bin(a,b : integer) : extended;

{--------------------------------------------------}
{-                calculates a over b             -}
{--------------------------------------------------}

begin
  bin:=fac(a)/(fac(b)*fac(a-b));
end;

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

function Bessel_A(x: integer;y : extended) : extended;

{--------------------------------------------------}
{-    modified bessel function of first kind and  -}
{-    order x. Formulas from Abramowitz pp. 378   -}
{-              formulas 9.8.1 & 9.8.2            -}
{--------------------------------------------------}

var
  t,b   : extended;

begin
  if (x>1) then Bessel_A:=Bessel_A(x-2,y)-(2*(x-1)/y)*Bessel_A(x-1,y)
  else
  begin
    if x=0 then
    begin
      t:=y/3.75;
      if y<=3.75 then
      begin
        Bessel_A:= 1.0+3.5156229*sqr(t)+3.0899424*power(t,4)+
                   1.2067492*power(t,6)+0.2659732*power(t,8)+
                   0.0360768*power(t,10)+0.0045813*power(t,12);
      end
      else
      begin
        b:=0.39894228+0.01328592/t+0.00225319/sqr(t)-
           0.00157565/(power(t,3))+0.00916281/(power(t,4))-
           0.02057706/(power(t,5))+0.02635537/(power(t,6))-
           0.01647633/(power(t,7))+0.00392377/(power(t,8));
        Bessel_A:=b/(sqrt(y)*exp(-y));
      end;
    end
    else
    begin
      t:=y/3.75;
      if y<= 3.75 then
      begin
        b:=0.5+0.87890594*sqr(t)+0.51498869*power(t,4)+
           0.15084934*power(t,6)+0.02658733*power(t,8)+
           0.00301532*power(t,10)+0.00032411*power(t,12);
        Bessel_A:=b*y;
      end
      else
      begin
        b:=0.39894228-0.03988024/t-0.00362018/sqr(t)+
           0.00163801/(power(t,3))-0.01031555/(power(t,4))+
           0.02282967/(power(t,5))-0.02895312/(power(t,6))+
           0.01787654/(power(t,7))-0.00420059/(power(t,8));
        Bessel_A:=b/(sqrt(y)*exp(-y));
      end;
    end;
```

```
  end;
end; { function Bessel_A }

{--------------------------------------------------------------}

{$F+}
function BessFunc(in1:extended) : extended;

var
  bs1, bs2, bs3 : extended;

begin
  bs1:=1/Pi;
  bs2:=exp(z*cos(in1));
  bs3:=cos(v*in1);
  BessFunc:=bs1*bs2*bs3;
end;
{$F-}

function Bessel_B(vi:integer ; zi:extended) : extended;

{-----------------------------------------------}
{-   modified bessel function of first kind and   -}
{-   order x. Formula from Abramowitz pp. 376     -}
{-                formulas 9.6.19                 -}
{-----------------------------------------------}

var
  bess1   : extended;

begin
  v:=vi;
  z:=zi;
  Romberg(0,Pi,1e-6,100,bess1,BessIt,BessEr,@BessFunc);
  Bessel_B:=bess1;
end;

{--------------------------------------------------------------}

function Bessel(x:integer;y:extended): extended;

{-----------------------------------------------}
{-        When M=4 or higher, a stack overflow    -}
{-              occurs in the coprocessor         -}
{-        Solution: in Dos: set 87=N and          -}
{-                  in TP55: Emulation on (E+)    -}
{-----------------------------------------------}

begin
  If (M>3) then Bessel:=Bessel_B(x,y) else Bessel:=Bessel_A(x,y);
end;

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

function p1(k:integer) : extended;

{-----------------------------------------------}
{-              calculates the value of p1        -}
{-  M is global and equal to the order of diversity  -}
{-----------------------------------------------}

var
  n    : integer;
  hlp  : extended;

begin
  hlp:=0;
  for n:= 0 to (M-1-k) do
  begin
    hlp:=hlp+bin(2*M-1,n);
  end;
  p1:=hlp/fac(k);
end; { function p1 }

{--------------------------------------------------------------}
```

```
function P2(x:extended) : extended;

{--------------------------------------------------}
{-             calculates the value of P2         -}
{--------------------------------------------------}

var
  k    : integer;
  som  : extended;

begin
  som:=0;
  For k:=0 to (M-1) do
  begin
    som:=som+p1(k)*power(x,k);
  end;
  P2:=(1/power(2,2*M-1))*exp(-x)*som;
end; { function P2 }

{-----------------------------------------------------------------}

function Gamma_Pdf(x:extended) : extended;

{--------------------------------------------------}
{-             calculates the pdf of gamma        -}
{-      the signal to noise ratio SNR is global   -}
{--------------------------------------------------}

var
  Sm, Sm_square : extended;

begin
  Sm_square:=2*P;
  Sm:=sqrt(Sm_square);
  if Rice<>0 then
    Gamma_Pdf:=(1/(2*sigma_square*snr))*power(x/(snr*Sm_square),(M-1)/2)
               *exp(-(Sm_square+(x/snr))/(2*sigma_square))
               *bessel(M-1,sqrt(x/snr)*(Sm/sigma_square))
  else
    Gamma_Pdf:=1/(power(2*sigma_square,M)*fac(M-1)*snr)
               *power(x/snr,M-1)*exp(-x/(2*snr*sigma_square));
end; { function Gamma_Pdf }

{-----------------------------------------------------------------}

{$F+}
function PrError(gamma:extended) : extended;

{--------------------------------------------------}
{-       calculates the Bit Error Probability     -}
{-               as a function of SNR             -}
{--------------------------------------------------}

begin
  PrError:=P2(gamma)*Gamma_Pdf(gamma);
end; { function PrError }
{$F-}

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

procedure GetData(var GetR            : extended;
                  var GetM            : integer;
                  var GetUitvoer      : text);

{--------------------------------------------------}
{-  This procedure assigns values to the above    -}
{-        variables from keyboard input           -}
{--------------------------------------------------}

var
  PlaceX, PlaceY: integer;


procedure GetRicianParameter (var GetR : extended);

var
```

```
  confirm : char;

{-------------------------------------------------}
{-   This procedure reads in the Rician Parameter  -}
{-                from the keyboard                -}
{-------------------------------------------------}

begin
  Writeln;
  GetR := 6.8;
  Write('Rician Parameter : ' : 24);
  TextColor(white);
  Write(GetR:2:1,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetR:=GetR else Readln(GetR);
  GotoXY(PlaceX-8,PlaceY);
  if (GetR=0) then Writeln(GetR:2:1,'       ')
  else Writeln(GetR:2:1,' dB    ');
end; { procedure GetRicianParameter }

{-------------------------------------------------------------------}

procedure GetOrderOfDiversity (var GetM : integer);

var
  confirm : char;

{-------------------------------------------------}
{-   This procedure reads in the Order of Diversity  -}
{-                from the keyboard                -}
{-------------------------------------------------}

begin
  Writeln;
  GetM := 4;
  Write('Order of Diversity : ' : 24);
  TextColor(white);
  Write(GetM,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetM:=GetM else Readln(GetM);
  GotoXY(PlaceX-6,PlaceY);
  Writeln(GetM,'      ');
end; { procedure GetOrderOfDiversity }

{-------------------------------------------------------------------}

procedure GetOutputFile (var GetUitvoer : text);

var
  test, confirm : string;
  answer        : char;

{-------------------------------------------------}
{-      This procedure reads in the outputfile     -}
{-          for the data from the keyboard         -}
{-------------------------------------------------}

begin
  repeat
    Writeln;
    Write('   Output File : ');
    test:='';
    Readln(confirm);
    test:=fsearch(confirm,'c:');
    if test <> '' then
    begin
      Textcolor(White);
      writeln('   This file already exists !');
      writeln('   Overwrite this file ?  Y/N ');
      answer:=ReadKey;
```

```
      clrscr; writeln;
      Textcolor(Yellow);
      if (answer='y') or (answer='Y') then test:='';
    end;
  until test = '';
  clrscr;
  Writeln; Writeln;
  Write('   Output File : ',confirm);
  assign(GetUitvoer,confirm);
end; { procedure GetOutputFile }

{-----------------------------------------------------------------}

begin { procedure GetData }
  TextBackground(Black);
  TextColor(Cyan);
  ClrScr;
  Writeln;
  Writeln('SLOW FREQUENCY HOPPING' : 32 );
  TextColor(Red);
  Write(' MAXIMAL RATIO COMBINING' : 33 );

  Window(4,5,39,17);
  TextBackground(Blue);
  Textcolor(Yellow);
  ClrScr;
  Writeln; Writeln; Writeln;
  Writeln;
  GetRicianParameter(GetR);
  GetOrderOfDiversity(GetM);

  Window(4,20,39,24);
  TextBackground(Red);
  Textcolor(Yellow);
  ClrScr;
  Writeln;
  GetOutputFile(GetUitvoer);
  rewrite(GetUitvoer);
  close(GetUitvoer);

  Window(44,1,76,4);
  TextBackground(5);
  Textcolor(White);
  ClrScr;
  Writeln;
  Write('  SNR  Avg Bit Error Probability ');

  Window(44,4,76,24);
  TextBackground(5);
  Textcolor(Yellow);
  ClrScr;
end; { Procedure GetData }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

begin { Main Program }
  ClrScr;
  GetData(Rice,M,uitvoer);
  if (Rice<>0) then Rice := power(10,Rice/10);
  P := Rice+1;                    { normalised signal power }
  s_square := (2*Rice*P)/(Rice+1);
  s := sqrt(s_square);
  sigma_square := P/(Rice+1);
  For snr_db:=1 to 30 do
  begin
    snr:=power(10,snr_db/10);
    Pe:=0;
    point1:=0;
    for counter:=1 to 10 do
    begin
      point2:=counter*4;
      Romberg(point1,point2,1e-7,1000,PrValue,PrIt,PrEr,@PrError);
      Pe:=Pe+PrValue;
      writeln('  #',counter:2,'  ',PrValue);
      point1:=point2;
    end;
```

```
    Writeln('       ---------------------- +');
    write('  ');
    Textbackground(red);
    Textcolor(white);
    write(snr_db:2,'   ',Pe);
    Textbackground(5);
    Textcolor(yellow);
    Writeln; Writeln;
    append(uitvoer);
    Writeln(uitvoer,Pe);
    close(uitvoer);
  end;
  Window(63,25,78,25);
  TextBackground(Black);
  Textcolor(White);
end.
```

D.3 Hybrid SFH/DS with Selection Diversity

```
{$N+}      { Compile for Mathematical Co-Processor }

program BitErrorProbability_HYB_SD(input,output);

uses  Dos, Crt;

const
  TNArraySize = 50;                 { Size of the vectors }

type
  TNvector = array[0..TNArraySize] of extended;

var
  Bitrate                          : integer;         { Bitrate in Kbps }
  Rice                             : extended;        { Rician Parameter }
  M                                : integer;         { Order of Diversity }
  L                                : integer;         { Number of Resolvable Paths }
  K                                : integer;         { Number of simultaneously transmitting }
                                                      { users per frequency slot }
  Chips                            : integer;         { Number of chips per bit }
  Tb                               : extended;        { Bit duration }
  Tc                               : extended;        { Chip duration }
  No                               : extended;        { Gaussian Noise density...}
  P                                : extended;        { Received Signal Power }
  s, s_square                      : extended;        { LOS Power }
  sigma_square                     : extended;        { Multipath Power }
  snr_db                           : integer;         { Signal to Noise Ratio in dB }
  snr                              : extended;        { Signal to Noise Ratio }
  mu0                              : extended;
  sigma_mu0                        : extended;        { Standard deviation of mu0 }
  Mmu0, Mmu                        : extended;        { Mean values of mu and mu0 }
  Pe                               : extended;        { Average Bit Error Probability }
  PrIt                             : integer;         { Flags if something went wrong }
  PrEr                             : byte;            { Tolerance variable }
  PrValue                          : extended;        { Outcome of the integral }
  counter                          : integer;         { Number of the subintegral }
  point1, point2                   : extended;        { Start and End point of the subintegral }
  uitvoer                          : text;            { Output file }
  gtfile                           : text;            { Input/Output file }
  gtdata                           : string;
  K_str, Bitrate_str, Chips_str    : string;
  gold                             : array[1..30,0..254] of integer;  { Array of Gold Codes }
  GT10, GT11, GT20, GT22           : array[1..30] of extended;
  GGX, GGV, GGVCAP                 : array[1..30] of extended;
  GGY, GGYCAP, GGZ, GGZCAP         : array[1..30] of extended;

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

{$I Romberg.pas }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

function power(a,b:extended) : extended;

{----------------------------------------------------}
{-                  Calculates a^b                  -}
{----------------------------------------------------}

var
  hlp : extended;

begin
  if a=0 then power:=0 else
  begin
    if a>0 then
    begin
      power:=exp(b*ln(a));
    end
    else
    begin
      hlp:=exp(b*ln(abs(a)));
```

```
      if b/2 = trunc(b/2) then power:=hlp else power:=-hlp;
    end;
  end;
end; { function power }

{---------------------------------------------------------------}

function fac(n:integer) : extended;

{-------------------------------------------------}
{-                calculates n !                 -}
{-------------------------------------------------}

var
  som_fac : extended;

begin
  som_fac :=1.0;
  while n>1 do
  begin
    som_fac:=som_fac*(n);
    n:=n-1;
  end;
  fac:=som_fac;
end; { function fac }

{---------------------------------------------------------------}

function bin(a,b:integer) : extended;

{-------------------------------------------------}
{-              calculates a over b              -}
{-------------------------------------------------}

begin
  bin:=fac(a)/(fac(b)*fac(a-b));
end;

{---------------------------------------------------------------}

function bessel(y:extended) : extended;

{-------------------------------------------------}
{-  modified bessel function of first kind and zero  -}
{-     order. Formulas from Abramowitz pp. 378      -}
{-              formulas 9.8.1 & 9.8.2              -}
{-------------------------------------------------}

var
  t, b : extended;

begin
  t:=y/3.75;
  if y<=3.75 then
  begin
    bessel:= 1 + 3.5156229*sqr(t) + 3.0899424*power(t,4)
             + 1.2067492*power(t,6) + 0.2659732*power(t,8)
             + 0.0360768*power(t,10) + 0.0045813*power(t,12);
  end
  else
  begin
    b:= 0.39894228 + 0.01328592/t + 0.00225319/sqr(t)
        - 0.00157565/(power(t,3)) + 0.00916281/(power(t,4))
        - 0.02057706/(power(t,5)) + 0.02635537/(power(t,6))
        - 0.01647633/(power(t,7)) + 0.00392377/(power(t,8));
    bessel:=b/(sqrt(y)*exp(-y));
  end;
end; { function bessel }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

function C(k,y:integer) : integer;

var
  j, ch : integer;
```

```
begin
  ch:=0;
  if (y>=0) and (y<=Chips-1) then
  begin
    for j:=0 to (Chips-1-y) do ch:=ch+gold[k,j]*gold[1,j+y];
  end
  else
  begin
    if (y>=1-Chips) and (y<0) then
    begin
      for j:=0 to (Chips-1+y) do ch:=ch+gold[k,j-y]*gold[1,j];
    end;
  end;
  C:=ch;
end;  { function C }

{---------------------------------------------------------------}

function A(k,x:integer) : integer;

begin
  A:=C(k,x-Chips);
end;

{---------------------------------------------------------------}

function ACAP(k,x:integer) : integer;

begin
  ACAP:=C(k,x);
end;

{---------------------------------------------------------------}

function B(k,x:integer) : integer;

begin
  B:=C(k,x+1-Chips) - C(k,x-Chips);
end;

{---------------------------------------------------------------}

function BCAP(k,x:integer) : integer;

begin
  BCAP:=C(k,x+1) - C(k,x);
end;

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

{-----------------------------------------------------}
{-       The following functions calculate           -}
{-    the statistical moments for the gaussian       _}
{-       approximation of the interference           -}
{-----------------------------------------------------}

function GT(k,p,q:integer) : extended;

var
  h1, h2, h3 : extended;
  x, i       : integer;

begin
  h3:=0;
  for x:=0 to Chips-1 do
  begin
    for i:= 0 to 2*q do
    begin
      h2:=0;
      if i/2=trunc(i/2) then h1:=1 else h1:=-1;
      h2:=(h1/(i+1)) * (bin(2*q,i)/bin(2*(p-q)+i+1,i+1))
          * (power(B(k,x),i)/power(BCAP(k,x),i+1))
          * (power(A(k,x)+B(k,x),2*q-i)
             * power(ACAP(k,x)+BCAP(k,x),2*(p-q)+i+1)
             - (power(A(k,x),2*q-i) * power(ACAP(k,x),2*(p-q)+i+1)));
      h3:=h3+h2;
```

```
    end;
  end;
  h3:=h3*power(Tc,2*p+1)/Tb;
  GT:=h3;
end;

{-------------------------------------------------------------------}

function GX(k:integer) : extended;

var
  x   : integer;
  gxh : extended;

begin
  gxh:=0;
  for x:=0 to Chips-1 do
  begin
    gxh:=gxh + A(k,x)*ACAP(k,x) + (1/3)*B(k,x)*BCAP(k,x)
         + (1/2)*A(k,x)*BCAP(k,x) + (1/2)*ACAP(k,x)*B(k,x);
  end;
  GX:=gxh*power(Tc,3)/Tb;
end;

{-------------------------------------------------------------------}

function GY(k:integer) : extended;

var
  x   : integer;
  gyh : extended;

begin
  gyh:=0;
  for x:=0 to Chips-1 do
  begin
    gyh:= gyh + sqr(A(k,x))*ACAP(k,x) + A(k,x)*ACAP(k,x)*B(k,x)
          + (1/3)*sqr(B(k,x))*ACAP(k,x) + (1/2)*sqr(A(k,x))*BCAP(k,x)
          + (2/3)*A(k,x)*B(k,x)*BCAP(k,x) + (1/4)*sqr(B(k,x))*BCAP(k,x);
  end;
  GY:=gyh*power(Tc,4)/Tb;
end;

{-------------------------------------------------------------------}

function GYCAP(k:integer):extended;

var
  x   : integer;
  gyc : extended;

begin
  gyc:=0;
  for x:=0 to Chips-1 do
  begin
    gyc:= gyc + sqr(ACAP(k,x))*A(k,x) + ACAP(k,x)*A(k,x)*BCAP(k,x)
          + (1/3)*sqr(BCAP(k,x))*A(k,x) + (1/2)*sqr(ACAP(k,x))*B(k,x)
          + (2/3)*ACAP(k,x)*BCAP(k,x)*B(k,x) + (1/4)*sqr(BCAP(k,x))*B(k,x);
  end;
  GYCAP:=gyc*power(Tc,4)/Tb;
end;

{-------------------------------------------------------------------}

function GZ(k:integer) : extended;

var
  x   : integer;
  gzh : extended;

begin
  gzh:=0;
  for x:= 0 to Chips-1 do
  begin
    gzh:= gzh + power(A(k,x),3) + (3/2)*sqr(A(k,x))*B(k,x)
          + A(k,x)*sqr(B(k,x)) + (1/4)*power(B(k,x),3);
```

```
  end;
  GZ:=gzh*power(Tc,4)/Tb;
end;

{-----------------------------------------------------------------}

function GZCAP(k:integer) : extended;

var
  x   : integer;
  gzc : extended;

begin
  gzc:=0;
  for x:= 0 to Chips-1 do
  begin
    gzc:= gzc + power(ACAP(k,x),3) + (3/2)*sqr(ACAP(k,x))*BCAP(k,x)
          + ACAP(k,x)*sqr(BCAP(k,x)) + (1/4)*power(BCAP(k,x),3);
  end;
  GZCAP:=gzc*power(Tc,4)/Tb;
end;

{-----------------------------------------------------------------}

function GV(k:integer) : extended;

var
  x   : integer;
  gvh : extended;

begin
  gvh:=0;
  for x:= 0 to Chips-1 do
  begin
    gvh := gvh + A(k,x) + (1/2)*B(k,x);
  end;
  GV:=gvh*sqr(Tc)/Tb;
end;

{-----------------------------------------------------------------}

function GVCAP(k:integer) : extended;

var
  x   : integer;
  gvc : extended;

begin
  gvc:=0;
  for x:= 0 to Chips-1 do
  begin
    gvc := gvc + ACAP(k,x) + (1/2)*BCAP(k,x);
  end;
  GVCAP:=gvc*sqr(Tc)/Tb;
end;

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

function emu0(k:integer) : extended;

{---------------------------------------------------}
{-         calculates the expectation of mu0       -}
{---------------------------------------------------}

var
  kk    : integer;
  emu0h : extended;

begin
  emu0h:=0;
  for kk:= 1 to K do
  begin
    emu0h:=emu0h + L*(GT11[kk]+GT10[kk])*(sigma_square+s_square/2)
           + L*(L-1)*(sqr(GGV[kk])+sqr(GGVCAP[kk]))*(s_square/2);
  end;
  emu0:=2*(P/8)*emu0h + (No*Tb)/8;
```

```
end;

{---------------------------------------------------------------}

function emu(k:integer) : extended;

{-----------------------------------------------------}
{-        calculates the expectation of mu           -}
{-----------------------------------------------------}

var
  kk   : integer;
  emuh : extended;

begin
  emuh:=0;
  for kk:= 1 to K do
  begin
    emuh:=emuh + L*GGX[kk]*(sigma_square+s_square/2)
           + L*(L-1)*GGV[kk]*GGVCAP[kk]*(s_square/2)
           + L* GT10[1]*(sigma_square+s_square/2)
           + L*(L-1)*sqr(GGVCAP[1])*(s_square/2);
  end;
  emu:=2*(P/8)*emuh;
end;

{---------------------------------------------------------------}

function vmu0(k:integer) : extended;

{-----------------------------------------------------}
{-         calculates the variance of mu0            -}
{-----------------------------------------------------}

var
  kk                             : integer;
  L1, L2, L3                     : integer;
  vmuh1, vmuh2, vmuh3, vmuh4     : extended;
  s1, s2, s3, s4                 : extended;

begin
  s1:=s_square/2;
  s2:=sqr(s1);
  s3:=sigma_square+s_square/2;
  s4:=sqr(s3);
  L1:=L*(L-1);
  L2:=L*(L-1)*(L-2);
  L3:=L*(L-1)*(L-2)*(L-3);
  vmuh1:=0; vmuh2:=0; vmuh3:=0; vmuh4:=0;
  for kk:= 1 to K do
  begin
    vmuh1:=vmuh1 + s4*(L*GT22[kk] + L1*sqr(GT11[kk]))
           + s1*s3*(4*L1*GGZ[kk]*GGV[kk] + 2*L2*GT11[kk]*sqr(GGV[kk]))
           + s2*(2*L1*sqr(GT11[kk]) + 4*L2*GT11[kk]*sqr(GGV[kk])
                 + L3*power(GGV[kk],4));
    vmuh2:=vmuh2 + s4*(sqr(L)*GT11[kk]*GT10[kk])
           + s1*s3*(2*L1*GGY[kk]*GGVCAP[kk] + L2*GT11[kk]*sqr(GGVCAP[kk])
                    + 2*L1*GGYCAP[kk]*GGV[kk] + L2*GT10[kk]*sqr(GGV[kk]))
           + s2*(sqr(L)*sqr(L1)*sqr(GGV[kk])*sqr(GGVCAP[kk]));
    vmuh3:=vmuh3 + s4*(L*GT20[kk] + L1*sqr(GT10[kk]))
           + s1*s3*(4*L1*GGZCAP[kk]*GGVCAP[kk] + 2*L2*GT10[kk]*sqr(GGVCAP[kk]))
           + s2*(2*L1*sqr(GT10[kk]) + 4*L2*GT10[kk]*sqr(GGVCAP[kk])
                 + L3*sqr(sqr(GGVCAP[kk])));
    vmuh4:=vmuh4 + sqr(s3*L*(GT11[kk] + GT10[kk])
                       + s1*L1*(sqr(GGV[kk]) + sqr(GGVCAP[kk])));
  end;
  vmu0:=(4*vmuh1 + 8*vmuh2 + 4*vmuh3 - 4*vmuh4)*sqr(P/8);
end;

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

procedure initcodes;

{-----------------------------------------------------}
{-   This procedure reads the Gold Codes from disk   -}
{-----------------------------------------------------}
```

```
var
  counter1, counter2 : integer;
  invoer             : text;

begin
  assign(invoer,'gold'+Chips_str+'.dat');
  reset(invoer);
  for counter1:=1 to K do
  begin
    for counter2:=0 to Chips-1 do
    begin
      read(invoer,gold[counter1,counter2]);
    end;
  end;
  close(invoer);
end; { procedure initcodes }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

procedure gtinit;

{------------------------------------------------------}
{-    This procedure calculates the statistical       -}
{-          moments and writes them to disk           -}
{------------------------------------------------------}

var
  kk, x         : integer;
  test          : string[20];
  Xpos1, Xpos2  : integer;
  Ypos1, Ypos2  : integer;
  counter       : integer;

begin
  test:=fsearch(gtdata,'c:');
  if test='' then
  begin
    initcodes;
    rewrite(gtfile);
    Window(44,1,76,4);
    TextBackground(5);
    Textcolor(White);
    ClrScr;
    Writeln;
    Write('    Calculating Stat. Moments     ');
    Window(44,4,76,24);
    TextBackground(5);
    Textcolor(Yellow);
    ClrScr;
    writeln;writeln;
    writeln('      Calculating / Writing');
    write('     Stat. Moments for K = ');
    Xpos1:=WhereX;
    Ypos1:=WhereY;
    writeln;writeln;
    write('          Moment : ');
    Xpos2:=WhereX;
    Ypos2:=WhereY;
    write('   of 11');
    for kk:=1 to K do
    begin
      GotoXY(Xpos1,Ypos1);
      write(kk:2);
      counter:=0;
      GotoXY(Xpos2,Ypos2);  counter:=counter+1;          write(counter:2);
      GT22[kk]:=GT(kk,2,2); writeln(gtfile,gt22[kk]);
      GotoXY(Xpos2,Ypos2);  counter:=counter+1;          write(counter:2);
      GT11[kk]:=GT(kk,1,1); writeln(gtfile,gt11[kk]);
      GotoXY(Xpos2,Ypos2);  counter:=counter+1;          write(counter:2);
      GT10[kk]:=GT(kk,1,0); writeln(gtfile,gt10[kk]);
      GotoXY(Xpos2,Ypos2);  counter:=counter+1;          write(counter:2);
      GT20[kk]:=GT(kk,2,0); writeln(gtfile,gt20[kk]);
      GotoXY(Xpos2,Ypos2);  counter:=counter+1;          write(counter:2);
      GGX[kk]:=GX(kk);      writeln(gtfile,ggx[kk]);
      GotoXY(Xpos2,Ypos2);  counter:=counter+1;          write(counter:2);
      GGY[kk]:=GY(kk);      writeln(gtfile,ggy[kk]);
```

```
      GotoXY(Xpos2,Ypos2);   counter:=counter+1;         write(counter:2);
      GGYCAP[kk]:=GYCAP(kk); writeln(gtfile,ggycap[kk]);
      GotoXY(Xpos2,Ypos2);   counter:=counter+1;         write(counter:2);
      GGZ[kk]:=GZ(kk);       writeln(gtfile,ggz[kk]);
      GotoXY(Xpos2,Ypos2);   counter:=counter+1;         write(counter:2);
      GGZCAP[kk]:=GZCAP(kk); writeln(gtfile,ggzcap[kk]);
      GotoXY(Xpos2,Ypos2);   counter:=counter+1;         write(counter:2);
      GGV[kk]:=GV(kk);       writeln(gtfile,ggv[kk]);
      GotoXY(Xpos2,Ypos2);   counter:=counter+1;         write(counter:2);
      GGVCAP[kk]:=GVCAP(kk); writeln(gtfile,ggvcap[kk]);
    end;
    close(gtfile);
  end
  else
  begin
    reset(gtfile);
    for kk:=1 to K do
    begin
      read(gtfile,gt22[kk]);
      read(gtfile,gt11[kk]);
      read(gtfile,gt10[kk]);
      read(gtfile,gt20[kk]);
      read(gtfile,ggx[kk]);
      read(gtfile,ggy[kk]);
      read(gtfile,ggycap[kk]);
      read(gtfile,ggz[kk]);
      read(gtfile,ggzcap[kk]);
      read(gtfile,ggv[kk]);
      read(gtfile,ggvcap[kk]);
    end;
    close(gtfile);
  end;
end; { procedure gtinit }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

{$F+}
function Beta_Cdf(z:extended) : extended;

{---------------------------------------------------}
{-   calculates the cdf of the Rician distribution  -}
{---------------------------------------------------}

begin
  Beta_Cdf:=(z/sigma_square)*exp(-(sqr(z)+s_square)/(2*sigma_square))*
            bessel((s*z)/sigma_square);
end; { function Beta_Cdf }
{$F-}

{-------------------------------------------------------------------}

{$F+}
function Beta_Int(y:extended) : extended;

{---------------------------------------------------}
{-       calculates the first integrant with        -}
{-                 respect to beta                  -}
{---------------------------------------------------}

var
   IntValue : extended;
   it       : integer;
   er       : byte;
   hlp      : extended;

begin
  Romberg(0,y,1e-6,1000,IntValue,it,er,@Beta_Cdf);
  hlp:=((P*sqr(y)*sqr(Tb))/(8*mu0))
       + ((sqr(y)+s_square)/(2*sigma_square));
  Beta_Int:=(1/2) * (1-(Mmu/mu0))
            * M * power(IntValue,M-1) * (y/sigma_square)
            * exp(-hlp) * bessel((s*y)/sigma_square);
end; { function Beta_Int }
{$F-}
```

```
{$F+}
function Mu0_Int(x:extended) : extended;

{--------------------------------------------------}
{-      calculates the second integrant with      -}
{-                respect to mu0                  -}
{--------------------------------------------------}

var
   IntValue : extended;
   it       : integer;
   er       : byte;
   Mu0_Pdf  : extended;

begin
  mu0:=x;
  Romberg(1,5,1e-6,1000,IntValue,it,er,@Beta_Int);
  Mu0_Pdf:=(1/(sqrt(2*Pi)*sigma_mu0))*exp(-sqr(x-Mmu0)/(2*sqr(sigma_mu0)));
  Mu0_Int:=IntValue*Mu0_Pdf;
end; { function Mu0_Int }
{$F-}

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

procedure GetData(var GetRc          : integer;
                  var GetRice        : extended;
                  var GetM           : integer;
                  var GetL           : integer;
                  var GetK           : integer;
                  var GetChips       : integer;
                  var GetUitvoer     : text);

{--------------------------------------------------}
{-   This procedure assigns values to the above   -}
{-         variables from keyboard input          -}
{--------------------------------------------------}


var
  PlaceX, PlaceY: integer;


procedure GetBitRate (var GetRc : integer);

var
  confirm : char;

{--------------------------------------------------}
{-     This procedure reads in the Bit Rate       -}
{-              from the keyboard                 -}
{--------------------------------------------------}

begin
  Writeln;
  GetRc := 64;
  Write('Bit Rate : ' : 24);
  TextColor(white);
  Write(GetRc,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetRc:=GetRc else Readln(GetRc);
  GotoXY(PlaceX-7,PlaceY);
  Writeln(GetRc,' Kb/s   ');
end; { procedure GetBitRate }

{-----------------------------------------------------------------}

procedure GetRicianParameter (var GetRice : extended);

var
  confirm : char;
```

```
{-------------------------------------------------------}
{-   This procedure reads in the Rician Parameter     -}
{-                from the keyboard                    -}
{-------------------------------------------------------}

begin
  Writeln;
  GetRice := 6.8;
  Write('Rician Parameter : ' : 24);
  TextColor(white);
  Write(GetRice:2:1,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetRice:=GetRice else Readln(GetRice);
  GotoXY(PlaceX-8,PlaceY);
  if (GetRice=0) then Writeln(0,'        ')
  else Writeln(GetRice:2:1,' dB    ');
end; { procedure GetRicianParameter }

{-------------------------------------------------------------------}

procedure GetOrderOfDiversity (var GetM : integer);

var
  confirm : char;

{-------------------------------------------------------}
{-   This procedure reads in the Order of Diversity   -}
{-                 from the keyboard                   -}
{-------------------------------------------------------}

begin
  Writeln;
  GetM := 4;
  Write('Order of Diversity : ' : 24);
  TextColor(white);
  Write(GetM,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetM:=GetM else Readln(GetM);
  GotoXY(PlaceX-6,PlaceY);
  Writeln(GetM,'        ');
end; { procedure GetOrderOfDiversity }

{-------------------------------------------------------------------}

procedure GetResolvablePaths (var GetL: integer);

var
  confirm : char;

{-------------------------------------------------------}
{-      This procedure reads in the number of        -}
{-        resolvable paths from the keyboard         -}
{-------------------------------------------------------}

begin
  Writeln;
  GetL := 2;
  Write('Resolvable Paths : ' : 24);
  TextColor(white);
  Write(GetL,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetL:=GetL else Readln(GetL);
  GotoXY(PlaceX-6,PlaceY);
  Writeln(GetL,'        ');
end; { procedure GetResolvablePaths }
```

```
procedure GetNumberOfUsers (var GetK : integer);

var
  confirm : char;

{-------------------------------------------------}
{-      This procedure reads in the number of     -}
{-        simultaneously transmitting users       -}
{-                 from the keyboard              -}
{-------------------------------------------------}

begin
  Writeln;
  GetK := 15;
  Write('Number of Users : ' : 24);
  TextColor(white);
  Write(GetK,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetK:=GetK else Readln(GetK);
  GotoXY(PlaceX-7,PlaceY);
  Writeln(GetK,'        ');
end; { procedure GetNumberofUsers }

{-----------------------------------------------------------------}

procedure GetNumberOfChips (var GetChips : integer);

var
  confirm : char;

{-------------------------------------------------}
{-    This procedure reads in the number of Chips  -}
{-              per bit from the keyboard          -}
{-------------------------------------------------}

begin
  Writeln;
  GetChips := 127;
  Write('Number of Chips : ' : 24);
  TextColor(white);
  Write(GetChips,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetChips:=GetChips else Readln(GetChips);
  GotoXY(PlaceX-8,PlaceY);
  Write(GetChips,'        ');
end; { procedure GetNumberOfChips }

{-----------------------------------------------------------------}

procedure GetOutputFile (var GetUitvoer : text);

var
  test, confirm : string;
  answer        : char;

{-------------------------------------------------}
{-      This procedure reads in the outputfile     -}
{-          for the data from the keyboard         -}
{-------------------------------------------------}

begin
  repeat
    Writeln;
    Write('   Output File : ');
    test:='';
    Readln(confirm);
    test:=fsearch(confirm,'c:');
    if test <> '' then
    begin
      Textcolor(White);
```

```
      writeln('   This file already exists !');
      writeln('   Overwrite this file ?  Y/N ');
      answer:=ReadKey;
      clrscr; writeln;
      Textcolor(Yellow);
      if (answer='y') or (answer='Y') then test:='';
    end;
  until test = '';
  clrscr;
  Writeln; Writeln;
  Write('   Output File : ',confirm);
  assign(GetUitvoer,confirm);
end; { procedure GetOutputFile }

{---------------------------------------------------------------}

begin { procedure GetData }
  TextBackground(Black);
  TextColor(Cyan);
  ClrScr;
  writeln;
  Writeln('     HYBRID SFH/DS     ' : 32 );
  TextColor(Red);
  Write(' SELECTION DIVERSITY ' : 31 );

  Window(4,5,39,17);
  TextBackground(Blue);
  Textcolor(Yellow);
  ClrScr;
  GetBitRate(GetRc);
  GetRicianParameter(GetRice);
  GetOrderOfDiversity(GetM);
  GetResolvablePaths(GetL);
  GetNumberOfUsers(GetK);
  GetNumberOfChips(GetChips);

  Window(4,20,39,24);
  TextBackground(Red);
  Textcolor(Yellow);
  ClrScr;
  Writeln;
  GetOutputFile(GetUitvoer);
  rewrite(GetUitvoer);
  close(GetUitvoer);
end; { Procedure GetData }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

begin { Main Program }
  ClrScr;
  GetData(BitRate,Rice,M,L,K,Chips,uitvoer);
  if (Rice<>0) then Rice:=power(10,Rice/10);
  Tb:=(1/BitRate)/1000;
  Tc:=Tb/Chips;
  str(K,k_str);
  str(Bitrate,Bitrate_str);
  str(Chips,Chips_str);
  gtdata:='gt'+K_str+'_'+Bitrate_str+'.'+Chips_str+'';
  assign(gtfile,gtdata);
  gtinit;

  Window(44,1,76,4);
  TextBackground(5);
  Textcolor(White);
  ClrScr;
  Writeln;
  Write('  SNR  Avg Bit Error Probability ');
  Window(44,4,76,24);
  TextBackground(5);
  Textcolor(Yellow);
  ClrScr;

  P:=Rice+1;                        { normalised signal power   }
  s_square:=(2*Rice*P)/(Rice+1);
  s:=sqrt(s_square);
  sigma_square:=P/(Rice+1);
```

```
  Mmu:=emu(K);
  sigma_mu0:=sqrt(abs(vmu0(K)));

  for snr_db:=0 to 40 do
  begin
    snr:=power(10,snr_db/10);
    No:=(P*Tb)/snr;                    { snr = Eb/No ; Eb = P*Tb  }
    Mmu0:=emu0(K);                     { emu0 is a function of No }
    Pe:=0;
    point1:=Mmu0 - 2*sigma_mu0;
    for counter:=1 to 7 do
    begin
      if Chips=255 then
      begin
        point1:=Mmu0 - 1*sigma_mu0;
        counter:=2;
      end;
      point2:=point1 + sigma_mu0;
      Romberg(point1,point2,1e-6,1000,PrValue,PrIt,PrEr,@Mu0_Int);
      Pe:=Pe+PrValue;
      writeln('  #',counter:2,'  ',PrValue);
      point1:=point2;
    end;
    Writeln('       ---------------------- +');
    write('  ');
    Textbackground(red);
    Textcolor(white);
    write(snr_db:2,'   ',Pe);
    Textbackground(5);
    Textcolor(yellow);
    Writeln; Writeln;
    append(uitvoer);
    Writeln(uitvoer,Pe);
    close(uitvoer);
  end;
  Window(63,25,78,25);
  TextBackground(Black);
  Textcolor(White);
end.
```

D.4 Romberg Integration Routine

```
{$F+}
{$L Integrat.OBJ}
function UserFunction(X : extended; ProcAddr : Pointer) : extended; external;
{$F-}

{-------------------------------------------------------------------}

procedure Romberg(LowerLimit  : extended;
                  UpperLimit  : extended;
                  Tolerance   : extended;
                  MaxIter     : integer;
              var Integral    : extended;
              var Iter        : integer;
              var Error       : byte;
                  FuncPtr     : Pointer);

var
  Spacing : extended;                 { Spacing between points }
  NewEstimate,
  OldEstimate : TNvector;             { Iteration variables }
  TwoToTheIterMinus2 : integer;

{-------------------------------------------------------------------}

procedure TestAndInitialize(LowerLimit         : extended;
                            UpperLimit         : extended;
                            Tolerance          : extended;
                            MaxIter            : integer;
                        var Iter               : integer;
                        var Spacing            : extended;
                        var OldEstimate        : TNvector;
                        var TwoToTheIterMinus2 : integer;
                        var Error              : byte);

{-------------------------------------------------------------------}
{- Input: LowerLimit, UpperLimit, Tolerance, MaxIter              -}
{- Output: Iter, Spacing, OldEstimate, TwoToTheIterMinus2, Error  -}
{-                                                                -}
{- This procedure tests Tolerance and MaxIter for errors (they    -}
{- must be greater than zero) and initializes the above           -}
{- variables.                                                     -}
{-------------------------------------------------------------------}

begin
  Error := 0;
  if Tolerance <= 0 then
    Error := 1;
  if MaxIter <= 0 then
    Error := 2;
  if Error = 0 then
  begin
    Spacing := UpperLimit - LowerLimit;
    OldEstimate[1] := Spacing *
                      (UserFunction(LowerLimit, FuncPtr) +
                       UserFunction(UpperLimit, FuncPtr)) / 2;
    Iter := 1;
    TwoToTheIterMinus2 := 1;
  end;
end; { procedure TestAndInitialize }

{-------------------------------------------------------------------}

procedure Trapezoid(TwoToTheIterMinus2 : integer;
                    LowerLimit         : extended;
                    Spacing            : extended;
                    OldEstimate        : extended;
                var NewEstimate        : extended);
```

```
{-------------------------------------------------------}
{- Input: TwoToTheIterMinus2, LowerLimit, Spacing,      -}
{-        OldEstimate                                   -}
{- Output: NewEstimate                                  -}
{-                                                      -}
{- This procedure uses the trapezoid rule to            -}
{- improve the integral approximation (OldEstimate)     -}
{- on the interval [LowerLimit, LowerLimit + Spacing].  -}
{- The results are returned in the variable NewEstimate -}
{-------------------------------------------------------}

var
  Sum : extended;
  Dummy : integer;

begin
  Sum := 0;
  for Dummy := 1 to TwoToTheIterMinus2 do
    Sum := Sum + UserFunction(LowerLimit + (Dummy - 0.5) * Spacing, FuncPtr);
  NewEstimate := 0.5 * (OldEstimate + Spacing * Sum);
end; { procedure Trapezoid }

{-------------------------------------------------------------------}

procedure Extrapolate(Iter        : integer;
                      OldEstimate : TNvector;
                  var NewEstimate : TNvector);

{-------------------------------------------------------}
{- Input: Iter, OldEstimate                             -}
{- Output: NewEstimate                                  -}
{-                                                      -}
{- This procedure uses Richardson extrapolation         -}
{- to improve the current approximation to the integral -}
{- (OldEstimate).  The result is returned in the        -}
{- variable NewEstimate                                 -}
{-------------------------------------------------------}

var
  Extrap : integer;
  FourToTheExtrapMinus1 : extended;

begin
  FourToTheExtrapMinus1 := 1;
  for Extrap := 2 to Iter do
  begin
    FourToTheExtrapMinus1 := FourToTheExtrapMinus1 * 4;
    NewEstimate[Extrap] :=
                      (FourToTheExtrapMinus1 * NewEstimate[Extrap - 1] -
                      OldEstimate[Extrap - 1]) / (FourToTheExtrapMinus1 - 1);
  end;
end; { procedure Extrapolate }

{-------------------------------------------------------------------}

begin { procedure Romberg }
  TestAndInitialize(LowerLimit, UpperLimit, Tolerance, MaxIter, Iter,
                    Spacing, OldEstimate, TwoToTheIterMinus2, Error);
  if Error = 0 then
  begin
    repeat
      Iter := Succ(Iter);
      Trapezoid(TwoToTheIterMinus2, LowerLimit, Spacing, OldEstimate[1],
                NewEstimate[1]);
      TwoToTheIterMinus2 := TwoToTheIterMinus2 * 2;
      Extrapolate(Iter, OldEstimate, NewEstimate);
      Spacing := Spacing / 2;
      OldEstimate := NewEstimate;
    until { The fractional difference between iterations is within Tolerance }
          (ABS(NewEstimate[Iter - 1] - NewEstimate[Iter]) <=
           ABS(Tolerance * NewEstimate[Iter])) or (Iter >= MaxIter);
    if Iter >= MaxIter then
      Error := 3;
    Integral := NewEstimate[Iter];
  end;
end; { procedure Romberg }
```

D.5 Programme used to create the Gold Codes

```
program Create_Gold_Codes(input,output);

uses  Dos, Crt;

type
  ra = array[1..9] of byte;

var
  a,b               : ra;
  goldrijen         : array[0..50,1..255] of byte;
  uitvoer           : text;
  xpos,ypos         : integer;
  offset            : byte;
  nieuwe_goldkode   : byte;
  kar               : char;
  Register_Offset   : array[0..50] of integer;
  N                 : integer;              { Length of code sequence }
  K                 : integer;              { number of Gold codes }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

procedure Gold127(K : integer);

var
  counter, code, regset : integer;
  pnrija, pnrijb        : byte;

procedure register(var reginh: ra);

begin
  reginh[7]:=reginh[6] ; reginh[6]:=reginh[5];
  reginh[5]:=reginh[4] ; reginh[4]:=reginh[3];
  reginh[3]:=reginh[2] ; reginh[2]:=reginh[1];
  reginh[1]:=reginh[8] ;
end; { procedure register }

begin
  for code:=0 to K-1 do
  begin
    gotoxy(Xpos,Ypos);
    write(code : 2);
    offset:=Register_Offset[code];
    for regset:=1 to 7 do
    begin
      a[regset]:=1;
      b[regset]:=1;
    end;
    a[8]:=0;
    b[8]:=0;
    for counter:=1 to (offset mod 127) do
    begin
      b[8]:=b[7] xor b[6] xor b[3] xor b[1];
      register(b);
    end;
    for counter:=1 to 127 do
    begin
      a[8]:=a[7] xor a[6] xor a[4] xor a[2];
      b[8]:=b[7] xor b[6] xor b[3] xor b[1];
      pnrija:=a[7];
      pnrijb:=b[7];
      goldrijen[code,counter]:= pnrija xor pnrijb ;
      register(a);
      register(b);
    end;
  end;
end; { procedure Gold127 }
```

```
procedure Gold255(K:integer);

var
  counter, code, regset : integer;
  pnrija, pnrijb        : byte;

procedure register(var reginh: ra);

begin
  reginh[8]:=reginh[7]; reginh[7]:=reginh[6];
  reginh[6]:=reginh[5]; reginh[5]:=reginh[4];
  reginh[4]:=reginh[3]; reginh[3]:=reginh[2];
  reginh[2]:=reginh[1]; reginh[1]:=reginh[9];
end; { procedure register }

begin
  for code:=0 to K-1 do
  begin
    gotoxy(Xpos,Ypos);
    write(code : 2);
    offset:=Register_Offset[code];
    for regset:=1 to 8 do
    begin
      a[regset]:=1;
      b[regset]:=1;
    end;
    a[9]:=0;
    b[9]:=0;
    for counter:=1 to (offset mod 255) do
    begin
      b[9]:=b[8] xor b[7] xor b[6] xor b[5] xor b[2] xor b[1];
      register(b);
    end;
    for counter:=1 to 255 do
    begin
      a[9]:=a[8] xor a[6] xor a[4] xor a[3] xor a[2] xor a[1];
      b[9]:=b[8] xor b[7] xor b[6] xor b[5] xor b[2] xor b[1];
      pnrija:=a[8];
      pnrijb:=b[8];
      goldrijen[code,counter]:= pnrija xor pnrijb ;
      register(a);
      register(b);
    end;
  end;
end; { procedure Gold255 }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

procedure ReadOffsets(k : integer);

var
  ready   : boolean;
  counter : integer;

begin
  ready:=False;
  repeat
    ClrScr;
    writeln;
    Textcolor(White);
    writeln('     Give ',k,' offsets for');
    writeln('     the ',k,' Gold Codes:');
    writeln;
    Textcolor(Yellow);
    for counter:=0 to k-1 do
    begin
      write( '     Offset for Code ',counter :2,' : ');
      readln(Register_Offset[counter]);
    end;
    writeln;
    write('   All offsets correct ? (Y/N) ');
    GotoXY(29,12);
    kar:=readkey;
    if (kar='y') or (kar='Y') then ready:=true;
  until ready;
end;
```

```
procedure CodeToFile;

var
  counter1, counter2 : integer;

begin
  rewrite(uitvoer);
  for Counter1:=K-1 downto 0 do
  begin
    gotoxy(Xpos,Ypos);
    write(Counter1:2);
    for counter2:=1 to N do
    begin
        kar:=chr(goldrijen[counter1,counter2]);
        write(uitvoer,2*ord(kar)-1,' ');
        {ascii number of kar is written to file}
    end;
    writeln(uitvoer);
  end;
  close(uitvoer);
end;

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

procedure GetData(var GetChips       : integer;
                  var GetK           : integer;
                  var GetUitvoer     : text);

{-----------------------------------------------------}
{-   This procedure assigns values to the above      -}
{-         variables from keyboard input             -}
{-----------------------------------------------------}

var
  PlaceX, PlaceY: integer;


procedure GetNumberOfChips (var GetChips : integer);

var
  confirm : char;

{-----------------------------------------------------}
{-    This procedure reads in the number of Chips    -}
{-              per bit from the keyboard            -}
{-----------------------------------------------------}

begin
  Writeln;
  GetChips := 127;
  Writeln('Number of Chips   ' : 22);
  Write('(127 or 255) : ' : 22);
  TextColor(white);
  Write(GetChips,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetChips:=GetChips else Readln(GetChips);
  GotoXY(PlaceX-8,PlaceY);
  Write(GetChips,'         ');
end; { procedure GetNumberOfChips }

{------------------------------------------------------------------}

procedure GetNumberOfCodes (var GetK : integer);

var
  confirm : char;

{-----------------------------------------------------}
{-      This procedure reads in the number of        -}
{-          Gold Codes from the keyboard             -}
{-----------------------------------------------------}
```

```
begin
  Writeln; Writeln;
  GetK := 15;
  Writeln('Number of   ' : 22);
  Write('Gold Codes : ' : 22);
  TextColor(white);
  Write(GetK,' Y/N ');
  PlaceX:=WhereX;
  PlaceY:=WhereY;
  Textcolor(Yellow);
  confirm:= ReadKey;
  if confirm = 'y' then GetK:=GetK else Readln(GetK);
  GotoXY(PlaceX-7,PlaceY);
  Writeln(GetK,'       ');
end; { procedure GetNumberofUsers }

{-------------------------------------------------------------------}


procedure GetOutputFile (var GetUitvoer : text);

var
  test, confirm : string;
  answer        : char;

{-----------------------------------------------------}
{-      This procedure reads in the outputfile       -}
{-          for the data from the keyboard           -}
{-----------------------------------------------------}

begin
  repeat
    Writeln;
    Write('   Output File : ');
    test:='';
    Readln(confirm);
    test:=fsearch(confirm,'c:');
    if test <> '' then
    begin
      Textcolor(White);
      writeln('   This file already exists !');
      writeln('   Overwrite this file ?  Y/N ');
      answer:=ReadKey;
      clrscr; writeln;
      Textcolor(Yellow);
      if (answer='y') or (answer='Y') then test:='';
    end;
  until test = '';
  clrscr;
  Writeln; Writeln;
  Write('   Output File : ',confirm);
  assign(GetUitvoer,confirm);
end; { procedure GetOutputFile }

{-------------------------------------------------------------------}

begin { procedure GetData }
  TextBackground(Black);
  TextColor(Red);
  ClrScr;
  writeln;
  Writeln('      GOLD CODES      ' : 32 );
  Write(' FOR SPREAD SPECTRUM ' : 31 );

  Window(4,5,39,17);
  TextBackground(Blue);
  Textcolor(Yellow);
  ClrScr;
  Writeln; Writeln;
  GetNumberOfChips(GetChips);
  GetNumberOfCodes(GetK);

  Window(4,20,39,24);
  TextBackground(Red);
  Textcolor(Yellow);
  ClrScr;
```

```
  Writeln;
  GetOutputFile(GetUitvoer);

  Window(44,1,76,24);
  TextBackground(5);
  Textcolor(Yellow);
  ClrScr;
end; { Procedure GetData }

{-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-}

begin {Main Program}
  ClrScr;
  Getdata(N,K,uitvoer);
  ReadOffsets(K);
  ClrScr;
  writeln;writeln;
  write('   Generating Gold Codes : ');
  Xpos:=WhereX;
  Ypos:=WhereY;
  if N=127 then Gold127(K) else Gold255(K);
  writeln;writeln;
  write('   Writing Code : ');
  Xpos:=WhereX;
  Ypos:=WhereY;
  CodeToFile;
  Writeln;Writeln;Writeln;Writeln;
  TextColor(White);
  write('              Ready !    ');
  Readln;
end.
```