Signals & Systems

Path search algorithms for application in W-CDMA systems

M.Sc. Thesis

Marijn C. Damstra

University of Twente
Department of Electrical Engineering,
Mathematics & Computer Science (EEMCS)
Signals & Systems Group (SAS)
P.O. Box 217
7500 AE Enschede
The Netherlands

Report Number:	SAS020-04
Report Date:	August 16, 2004
Period of Work:	01/10/2003 – 16/08/2004
Thesis Committee:	Prof. Dr. ir. C.H. Slump
	ir. F.W. Hoeksema
	ir. J. Potman

Abstract

The CADTES and SAS groups of the EEMCS faculty are working on the Adaptive Wireless Networking (AWGN) project. As part of this project adaptive algorithms are developed for digital signal processing in W-CDMA systems. One of these algorithms is the path search algorithm that estimates the delays of the paths between a transmitter and a receiver that are caused by reflections. Several options exist for implementing the path search algorithm. One of the questions posed within the AWGN project is: to what extent will it be useful for the path search function to switch between different algorithms, as the conditions between transmitter and receiver change?

First, this document presents an overview of path search algorithms from literature. About twenty papers are discussed, the algorithms they describe have been compared with each other on sixteen points. Based on the similarities that are discovered, the algorithms are classified in three classes: algorithms using a Power Delay Profile (PDP), algorithms based on a Maximum Likelihood Estimation (MLE) method and subspace-based algorithms.

Next, an algorithm is selected from each class. Both the Power Delay Profile and the Maximum Likelihood algorithms are implemented, the subspace algorithm is analyzed in theory only. In order to set up meaningful simulations channel models and simulation scenarios are investigated. The available simulator is discussed, as well as the modifications that were made.

The algorithms' performance is determined by simulation, results show that the MLE algorithm outperforms the PDP algorithm in most situations. The MLE algorithm however requires more computations under all circumstances. In view of this trade-off between performance and number of computations the MLE algorithm should be used in case of closely spaced paths, if time-variant path delays need to be estimated and if strong Doppler effects occur. BER simulations will have to be carried out to quantify the benefits of selecting the MLE algorithm in these cases.

Acknowledgments

For my M.Sc. thesis I was looking for an assignment in the area of algorithm development for wireless digital communications systems. Having discussed the options with professor Slump, I decided to visit ir. Potman to hear more about his Ph.D. dissertation as part of the Adaptive Wireless Network project. Inspired by his enthusiasm I opted for studying *Path search algorithms for application in W-CDMA systems*.

During the first part of my M.Sc. thesis I have spent quite some time rereading a number of books from previous courses on wireless communications, CDMA systems, digital signal processing, channel models, radio propagation etcetera. Having finished the literature study, I started working on the implementation of the path search algorithms, followed by the simulations. I enjoyed working on this part and I appreciate the inspiring discussions and enthusiasm of both ir. Potman and ir. Hoeksema during our fruitful bi-weekly meetings.

At the end of my thesis' work I would very much like to thank professor Slump, ir. Potman and ir. Hoeksema for their support.

Enschede, August 16^{th} 2004
Marijn Damstra

Contents

1

Introduction

Currently the CADTES group and the SAS group of the EEMCS faculty are working on the Adaptive Wireless Networking (AWGN) project that is sponsored by Freeband. The goal of the AWGN project is the development of adaptive algorithms for digital signal processing and the mapping of these algorithms on a flexible hardware architecture. Within the AWGN project mostly adaptive algorithms are investigated that can be applied in CDMA systems, such as UMTS.

One of these algorithms is the so-called path search algorithm. It is used to estimate the delays of the different paths between a transmitter and a receiver that are caused by reflections. Several options are available for implementing the path-search algorithm. One of the questions posed within the AWGN project is: to what extent will it be useful for a certain function to switch between different algorithms, as the conditions between transmitter and receiver change?

As part of the AWGN project, the goal of this M.Sc. thesis is to answer this question for the path search algorithm. In order to provide an answer, existing path search algorithms have been researched in literature. The possible implementations of the path search algorithm have been investigated and are presented in this document (Chapter 2). If possible the circumstances under which these algorithms show the best performance are noted, as well as a measure for their performance.

Before implementing the algorithms and setting up simulations first a number of channel models are analyzed in Chapter 3. Also scenarios have been formulated in Chapter 3 that describe the conditions in the channel between the transmitting base station and the receiving mobile terminal.

Next the algorithms that are selected in the literature study have been implemented. The implementation of these path searchers is discussed in detail in Chapters 4 and 5. The theory behind the subspace-based algorithm is discussed in Appendix B. The performance of the algorithms needs to be investigated in more detail, for this a number of simulations are carried out (Chapter 6). Finally the conclusions of this M.Sc. thesis are presented in Chapter 7.

2

Literature study

2.1 Overview of the literature study

To get a clearer picture of the different techniques for implementing the path search algorithm papers have been searched in literature about this. In Section 2.2 it is briefly discussed which sources have been used and according to which method the papers have been searched. This resulted in about twenty papers. In Section 2.3 a model of the received signal is presented, for a better understanding of the papers. The algorithms have been compared with each other on sixteen points; these points are addressed in Section 2.4.

The discussion of the algorithms that were found follows in Section 2.5. During the literature study it turned out that there are several resemblances between the algorithms that were found. Based on these resemblances it was possible to classify the algorithms in three classes.

The first class that was identified consists of algorithms that use a Power Delay Profile. This class of algorithms is described in Subsection 2.5.1, including an overview of the papers. A second group of algorithms is based on a Maximum Likelihood method, the papers are discussed in Subsection 2.5.2 in detail. Subspace-based algorithms form the last group (Subsection 2.5.3).

One of the questions of the AWGN project (Chapter 1) is: to what extent is it useful to switch between several types of algorithms that can carry out the path search function, as conditions change between transmitter and receiver? To be able to answer this question, the algorithms will be implemented during the next phase of this M.Sc. thesis. One algorithm from each of the classes is selected, as will be discussed in Section 2.6.

2.2 Selection of papers for the literature study

In order to find papers discussing path search algorithms a table of keywords was drafted. This table contains a variety of combinations of one or more words from the following list:

path	multipath	channel	algorithm
search∗	delay	estimat∗	cdma

∗ is a wildcard, "estimat∗" can be "estimator" or "estimation" and so on, "search∗" can be "searcher" or "searching". Search engines provided by a number of organizations were used to find papers discussing techniques for path searching, using the generated table of keywords. They include:

database:	**provided by:**	**available at:**
IEEExplore	IEEE	http://ieeexplore.ieee.org
Universiteits Bibliotheek	University of Twente	http://opc4.civ.utwente.nl
Picarta	OCLC PICA	http://picarta.pica.nl/
Web of Science	ISI	http://isi4.isiknowledge.com/portal.cgi/wos
Current contents	ISI	University of Twente library
Compendex	Elsevier Engineering Information	http://www.engineeringvillage2.org/
Inspec	Elsevier Engineering Information	http://www.engineeringvillage2.org/
Google	Google	http://www.google.com/
Altavista	Altavista	http://www.altavista.com/

This resulted in a large number of papers, which were processed in two phases. During the first phase the most relevant papers were selected based on their abstracts. These papers were then fully processed and the most relevant ones have been selected for the literature study. They will be discussed in detail in Section 2.5.

2.3 Received signal model

The papers present algorithms that process the signal that is received from a multipath channel. The signal that is transmitted by user k over the multipath channel can be described as $x_k(t)$:

$$x_k(t) = \sqrt{\frac{\varepsilon_{b,k}}{T_b}} \sum_{i=0}^{P_0-1} {b_k}^{(i)} s_k(t - iT_b) \tag{2.1}$$

In (2.1) ${b_k}^{(i)} \in \{-1, 1\}$ is the i^{th} symbol, transmitted with energy $\varepsilon_{b,k}$, T_b is the symbol duration, P_0 denotes the packet length and $s_k(t)$ is the spreading waveform:

$$s_k(t) = \sum_{n=0}^{N_c-1} s_k^{(n)} \psi(t - nT_c) \tag{2.2}$$

In (2.2) $s_k^{(n)} \in \{-1, 1\}$, T_c is the chip duration and $\psi(t)$ is the pulse waveform with duration T_c. The signal $x_k(t)$ is deformed by the channel.

The channel impulse response $h_k(t)$ can be described as a sum of L multipaths that result in a sum of superimposed versions of the transmitted signal $x_k(t)$.

$$h_k(t) = \sum_{l=1}^{L} \alpha_{k,l} \delta(t - \tau_{k,l}) e^{j\phi_{k,l}} \tag{2.3}$$

In (2.3) L indicates the number of paths in the channel and $\alpha_{k,l}$ and $\tau_{k,l}$ indicate respectively their complex attenuation and delay. The received signal $r(t)$ can be described as follows:

$$r(t) = \sum_{k=1}^{K} h_k x_k(t - \tau_{k,l}) + w(t) \tag{2.4}$$

In (2.4) K is the number of users sending over the channel and $w(t)$ white Gaussian noise.

2.4 Characteristics of path search algorithms

Of each paper it will be discussed in what kind of system the path search algorithm is applied and under what kind of circumstances. For this the channel model that is used will be discussed, as well as the scenario. The scenario describes the circumstances under which the algorithm is applied, for example whether the algorithm is used in a mobile terminal or in a base station, the number of multipaths that are present and whether there are other mobile terminals sending over the channel. If there are other users sending over the channel the received signal will contain signal components from these users as well, due to the non perfect orthogonality of their codes. This is called *Multiple Access Interference* (MAI) [34]. In the discussion of the channel model it will be indicated whether it includes Doppler effects and what type of fading (see [34]) occurs (fast/slow and flat/frequency selective fading). If possible the *delay spread* is noted, that indicates the maximum path delay.

For each paper the set-up of the path search algorithm will then be discussed. It will be indicated whether the algorithm is based on a correlator [27], a matched filter [27] or on another method. Algorithms can exploit pilot tones (pilot aided estimation) or not (blind estimation). Most of the algorithms consist of three parts. The first part estimates the parameters of the paths in the channel (delay and attenuation), the second part estimates the power of the paths that are found and the third part selects a number of these paths for demodulation by e.g. a Rake receiver [34].

Finally, the robustness and performance of each algorithm is investigated. The robustness of the algorithms is assessed by the extent to which the algorithm is sensitive to the near-far effect [34] and to Multiple Access Interference. The performance of the algorithms is verified by computer simulations in the majority of the papers.

To be able to compare the performance of the algorithms with each other, it will be discussed what aspect of the performance has been verified in the papers. Various performance measures are used, such as Bit Error Rate as function of Signal to Noise Ratio (BER-SNR) [34] [27], Receiver Operating Characteristics (ROC) curves [35] [30], Mean Square Error (MSE) of estimated delay and attenuation, performance with respect to Cramér-Rao Lower Bounds (CRLB) [10] [2].

2.5 Discussion of the papers

2.5.1 Power delay profile-based path searchers

A large number of papers analyzed so-called *Power Delay Profile* (PDP)-based path searchers. The PDP-based path searcher is depicted in Figure 2.1:

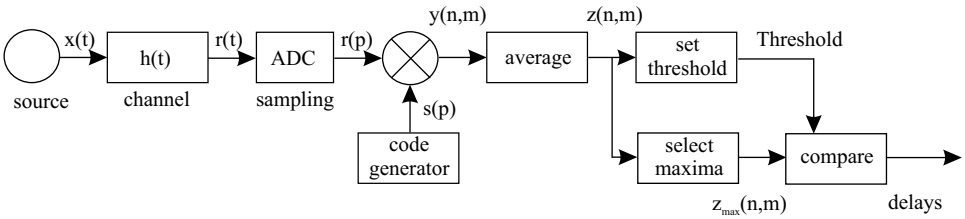

Figure 2.1: Power Delay Profile-based path searcher

Algorithms of this type search paths in three steps. During the first step the complex received baseband signal $r(p)$ is correlated with a replica of the user signature $s(p)$. Generally the received signal consists of a series of pilot symbols, which means that this signal contains the user signature. In this way a correlated signal $y(n,m)$ is obtained for the n^{th} frame, that contains a number of peaks for the different delays $\tau_{i,l}$. These peaks correspond to the paths in the channel between transmitter and receiver.

$$y(n,m) = \frac{1}{N}\sum_{p=0}^{N-1} h_i(p)\delta(m - \frac{\tau_{i,l}}{T_c}) + \tilde{w}(n,m) \tag{2.5}$$

N is the correlation window size, p the discrete time index and m the correlation index. $\delta(m - \frac{\tau_{i,l}}{T_c})$ results from the autocorrelation of the user signature $s_i(p)$ of the user i and $\tilde{w}(n,m)$ is the correlated noise.

The next step is to find the most significant paths. For this a Power Delay Profile $z(n,m)$ is determined based on the signal $y(n,m)$. This PDP is a measure for the received power in each of the paths that is found (M indicates the length of the summation).

$$z(n,m) = \frac{1}{M}\sum_{k=0}^{M-1} |y(n-k,m)|^2 \tag{2.6}$$

Finally a limited number of paths from the PDP is selected. In order to do this a threshold value is set for the power of the paths. There are several methods for calculating this threshold value, for example by means of an estimate of the power of the noise. Generally speaking, η is set as:

$$\eta = K{\sigma_n}^2 \tag{2.7}$$

Estimating σ_n^2 is crucial for setting threshold η. Most papers base their noise variance estimate on the PDP. As it contains peaks for the detected paths, the PDP needs to be modified in order to determine the noise variance estimate. Some papers suggest to subtract the strongest paths from the PDP [13] [28] or subtract a WMSA filtered signal from the PDP [12]. In [25] the threshold is simply set to

$$\eta = \frac{1}{P} \sum_{m=0}^{P-1} z(n,m)(a + bM^c) \tag{2.8}$$

M is the averaging length and P the correlation length.

In the next subsections the found power delay profile-based path searchers will be discussed. Because of the width of the main lobe of the autocorrelation most of these algorithms have a resolution of one chip period or more.

2.5.1.1. Path searcher for a WCDMA Rake receiver

In [25] an algorithm is described for finding paths that is suitable for W-CDMA systems. The channel is modelled as a filter in discrete time, no information concerning the type of fading is given. Paths are characterized in the model by their delay and attenuation and the channel has a certain delay spread (20 μs). Furthermore, no Doppler effects occur. The described algorithm is used in the mobile terminal (downlink), the number of users in the cell is not mentioned and their interference is considered as white noise.

The algorithm is based on a correlator that makes use of a pilot tone in the received signal from the Primary Common Pilot Channel (P-CPICH). By calculating the correlation over the length of a frame, a Power Delay Profile is found. This PDP is then averaged over a number of previous frames, to suppress fast fluctuations as a result of interference (non-coherent averaging). Based on this average the PDP threshold is calculated. Parallel to this maxima are searched in the average PDP based on a three points method. The height of the found maxima (attenuation) is then compared with the threshold. If it exceeds the threshold then the associated delay is assigned to a finger of a Rake receiver.

Only one simulation is presented in the paper, in which a PDP is shown. There is no data available concerning the performance of the algorithm under varying circumstances. The algorithm does not suppress the MAI in the received signal; this can only be compensated for by adapting the length of the non-coherent averaging. The paper elaborates on the implementation of the algorithm in C, the length of the code is 1688 bytes and it requires 32.8 MIPS.

2.5.1.2. Path-search algorithm introducing path-management tables for a DS-CDMA mobile terminal

In [13] a path search algorithm is presented for wideband direct sequence (DS) CDMA systems, that has been extended with three path management tables. The channel model shows Rayleigh fading (2 paths), and Doppler effects occur ($f_d = 5.6$ Hz). The described algorithm is used in the mobile terminal (downlink), the number of users in the cell is not mentioned and their interference is not modelled separately.

The path searcher consists of three parts: a spreading code generator, a correlator and a power measurement block. It uses the common pilot symbols in the received signal. The correlator performs the correlation of the received signal with the generated spreading code and the power measurement block then calculates a PDP. The paper discusses the selection of the paths in detail and proposes three management tables to aid this selection. The first table contains the data of the path searcher (ten strongest paths from PDP). The table is initialized on the basis of the momentaneous PDP and is adapted by using a first order forgetting filter.

The second table resides in the Rake combiner. This table contains the data of the paths which have been assigned to the fingers of the Rake receiver. For this the data from the channel estimator in the Rake receiver is used. This table is also updated using a filter with a forgetting factor. The third table is updated by multiplying the signal strength of each path in the second table by a weighing factor and adding it to the strength of the corresponding path in the first table. Finally the strength of the paths from the third table is compared to a threshold.

The performance of the presented algorithm has been examined by simulating the Block Error Rate (BLER) as a function of the SNR. For comparison the performance of a conventional path search algorithm (without management tables) and the performance of the ideal case (in which the two paths are selected) are used.

2.5.1.3. Performance analysis of multipath searcher in WCDMA system

In [28] a path search algorithm is presented that is based on the PDP as well. The channel shows slow frequency selective Rayleigh fading and Doppler effects occur. The described algorithm is used in the base station (uplink), the channel model takes only one user into account.

The algorithm is based on a correlator which uses the pilot tone in the received signal from the Dedicated Physical Control Channel (DPCCH). The correlated signal is then accumulated (coherent or non-coherent) over a number of slots. Finally a Power Delay Profile is calculated by squaring this signal. Based on this PDP the paths will be selected using a certain threshold value. Expressions are inferred for the chance of detection (P_d), miss (P_m) and false alarm (P_f).

The performance of the algorithm is examined by simulating P_d, P_m and P_f as a function of the threshold value. For this simulations have been carried out for both coherent and non-coherent accumulation, both at two different speeds of the mobile terminal. It is proven that coherent accumulation works the best at low speeds of the mobile terminal, whereas non-coherent accumulation performs the best at high speeds.

2.5.1.4. Path search performance and its parameter optimization of pilot symbol-assisted coherent Rake receiver for W-CDMA mobile radio

Another algorithm for W-CDMA systems that also uses coherent accumulation was found in [18]. The algorithm is used in the base station (uplink). The channel is characterized by fast frequency selective Rayleigh fading and Doppler effects occur (ITU-R Vehicular B model). The MAI induced by other users is not suppressed.

The algorithm is based on a correlator for pilot symbol aided path searching. The correlated signal is coherently averaged to obtain a momentaneous estimate of the channel. These estimates are averaged coherently over a number of slots R to generate a PDP. Finally, this PDP is averaged over a number of frames (during T_{avg}). Also the power of the interference and the noise is determined by subtracting the four strongest paths from the PDP and averaging the rest of the paths. This average is multiplied by a factor M to calculate a threshold value.

The performance of the algorithm is analyzed by determining the required E_b/N_0 of the receiver for a BER $= 10^{-3}$ as a function of the threshold value M, the averaging time T_{avg} and the number of slots R. This has been done both by simulation and by lab experiments. The antenna diversity was varied (with or without), as well as the number of users (one or two), the Doppler frequency (f_d = 80 or 320 Hz) and the number of paths (one up to four). On the basis of these simulations and the lab experiments optimal values for the parameters M, T_{avg} and R were found.

Finally experiments have been carried out on two locations, while the mobile terminal was moving. Here too the required E_b/N_0 of the transmitter for a BER = 10^{-3} was examined as a function of the threshold value M, averaging time T_{avg} and number of slots R. These experiments show that the path search algorithm is insensitive to fluctuations in the PDP such as those occurring in practice.

2.5.1.5. A novel channel estimation algorithm applied to UTRA-FDD

In order to be able to set a threshold value for the selection of paths in the PDP, it is important to establish a measure for the background noise and the interference. In [18] this measure is determined by subtracting the strongest paths from the average PDP. This method is sensitive to fluctuations in the background noise. In order to reduce this sensitivity an algorithm is presented in [12] that can calculate a better estimate of the background noise. The algorithm is used in the base station (uplink). The channel is modelled as a filter, it is assumed that slow frequency selective Rayleigh fading occurs without Doppler effects (ITU-R Vehicular B model). The MAI induced by other users is considered as noise.

The algorithm is based on a matched filter that is matched to the pilot sequence. The output of the matched filter is then filtered by a Weighted Multi-Slot Average (WMSA) filter. By subtracting the output of this filter from the momentaneous PDP, the unbiased variance of the PDP can be calculated. This can then be used to determine a threshold value for selecting the paths.

The performance of this algorithm has been assessed by calculating the MSE of the noise variance estimate. For comparison the MSE has been also determined for two algorithms that remove the strongest paths from the PDP in order to estimate the variance [18]. The results show that the new algorithm is significantly more precise.

Also the algorithms have been simulated in combination with a Rake receiver. In this case the required E_b/N_0 of the transmitter for a BER = 10^{-3} as a function of the threshold value was investigated. From this it can also be concluded that the presented algorithm performs better. Finally the complexity of the algorithms has been compared by calculating the number of MIPS as a function of the number of fingers of the Rake receiver. It turns out that in the case that three fingers or more are used the presented algorithm will require less MIPS. (approximately 6 MIPS).

2.5.1.6. Multipath-decorrelating receiver using adaptive path selection for synchronous CDMA frequency-selective fading channels

In [4] a path search algorithm is presented for CDMA systems, with frequency selective fading channels. The channel model shows Rayleigh fading (six paths), and no Doppler effects occur. The described algorithm is used in the base station (uplink), the number of users in the cell is ten and their interference is suppressed partially.

The path searcher is based on a MF that is matched to the user signature. This way a delay profile is determined, from which the strongest paths are selected using a threshold. The signals in the paths that are found are then multiplied by a weighing factor and are forwarded to a maximal ratio combiner [34].

The performance of the algorithm has been analyzed by simulations, the BER has been examined as a function of the SNR. Simulations have been carried out for several threshold values. For comparison the performance of a conventional path searcher from literature has been incorporated in the simulations as well. Also the performance of an algorithm has been used that only selects a fixed number of paths, regardless of the strength of these paths. From these simulations it becomes clear that the presented algorithm performs considerably better. Also it is stated that the computational cost of the algorithm is not higher than that of the other two algorithms. Finally the near-far resistance has been examined. This has decreased, but this deterioration can be limited by a proper choice of the threshold.

2.5.1.7. Performance analysis of multi-path searcher for mobile station in W-CDMA system employing transmit diversity

The properties of a *double-dwell serial search* (DDSS) path searcher for W-CDMA systems are investigated in [36]. The DDSS scheme detects a received signal in two phases, rather than one. A state diagram of DDSS is provided by [37]. The algorithm is applied in the mobile terminal (downlink). It uses Jakes' channel model [29] with two Rayleigh fading paths. Doppler effects occur, as the mobile terminal is moving. Interference by other users is not taken into consideration.

The path searcher uses the pilot symbols from the CPICH. The diagram of the path searcher exists of an in-phase part (I) and a quadrature part (Q), that are exactly equal to each other. First, the descrambled signal is correlated with the pilot symbols. The result is coherently summed (over four symbols) and then squared, to obtain a measure for the power in the

paths. Finally the results of the coherent summation from the I and Q parts are added and summed for post-detection integration. The system applies *Space-Time Transmit Diversity* (STTD), which means that an antenna array is used at the transmitter (in the current case two antennae are used).

To assess the performance of the algorithm, expressions are inferred for the chance of detection (P_d) and of false alarm (P_f). Then the ROC curve is determined by simulation for several lengths of the post-detection integration and for several speeds of the mobile terminal. By increasing the length of the post-detection integration, the ROC behavior is improved. This induces a larger sensitivity for Doppler effects however, which causes a trade-off.

2.5.1.8. Subchip multipath delay estimation for downlink WCDMA system based on Teager-Kaiser operator

Most path search algorithms have a resolution of one or several chips. In [17] an algorithm is presented for W-CDMA systems that can also distinguish between multipath components within a chip period. The algorithm has been intended for the mobile terminal (downlink). It is assumed that the channel has three paths, and exhibits Rayleigh fading without Doppler effects. The delay spread is smaller than the chip time. Two to thirty users are present in the cell.

The algorithm is based on the crosscorrelation of the received signal with a replica of the desired user signature. Then the result is manipulated with the *Teager-Kaiser* (TK) operator, which is defined in discrete time as follows:

$$\Psi_D[x(n)] = x(n-1)x^*(n-1) - \frac{1}{2}(x(n-2)x^*(n) + x(n)x^*(n-2)) \quad (2.9)$$

This produces a signal with a number of peaks that correspond to the different paths. Then square envelope detection is applied to this signal, after which noncoherent averaging is performed over a number of symbols. The first maxima in this signal correspond to the paths that are searched.

The performance of the algorithm has been simulated, for this the chance that the algorithm correctly detects all three paths in the channel has been investigated as a function of the near-far ratio. These simulations show the robustness of the algorithm against MAI and near-far effects. For comparison curves of three other algorithms from literature have also been determined. These are Multiple Signal Classification (MUSIC), Multipath Estimating Delayed-Locked loops (MEDLL) and Pulse Subtraction (PS).

It can be concluded from these simulations that the presented algorithm performs similar to MUSIC and performs better than MEDLL and PS. As the number of users increases in the cell (from two to thirty), the performance deteriorates. Also the pulse shape that is used influences the performance. The complexity of the algorithm is much lower than that of MUSIC.

2.5.1.9. Performance analysis of multi-path searcher for UE in W-CDMA systems

In [37] a path searcher is described that uses double-dwell serial search as well, as in [36]. The algorithm has been intended for usage in the mobile terminal (downlink) of a W-CDMA system. Slow frequency selective Rayleigh fading and Doppler effects occur while MAI is not taken into account.

The path searcher uses the CPICH to find the paths. The path searcher consists of an I and a Q branch. The received signal (both the I and Q component) is correlated, then coherently summed over a number of chips and squared. After this the I and Q components are added to each other. Finally post-detection integration is carried out. Expressions are inferred for the chance of false alarm P_f and the chance of detection P_d. P_d is maximized using the Neyman-Pearson criterion. The performance of the algorithm has been examined by simulations, this resulted in ROC curves at several values of the SNR. Finally a STTD-DDSS approach is presented to improve the performance of the algorithm.

2.5.2 Maximum Likelihood-based path searchers

A second class of path search algorithms is based on a *Maximum Likelihood* (ML) approach. In this set-up the received signal $r(t)$ is first filtered by a matched filter and then sampled. Based on the matched filter output signal y, a Maximum Likelihood function can be formulated. A parameter matrix is defined first as $\boldsymbol{\theta}$ with $\boldsymbol{\tau} = [\tau_{11}, \ldots, \tau_{kl}]^T$ and $\boldsymbol{\alpha} = [\alpha_{11}, \ldots, \alpha_{kl}]^T$:

$$\boldsymbol{\theta} = [\boldsymbol{\tau} \quad \boldsymbol{\alpha}] \tag{2.10}$$

with $\theta_{kl} = [\tau_{kl} \quad \alpha_{kl}]$. For a compact notation the shifted versions of the spreading waveforms $s_{kl}(t)$ (see equation 2.2) are defined as:

$$s(t;\boldsymbol{\theta}) = \sum_{k=1}^{K} \sum_{l=1}^{L} (s_{kl}(t)) \tag{2.11}$$

The likelihood function can now be defined as follows:

$$\Lambda(\boldsymbol{\theta}; y) = \frac{1}{N_0} \left[2 \int_{D_o} \Re\Big(s^H(t;\boldsymbol{\theta}) y(t) \Big) dt - \int_{D_o} ||s(t;\boldsymbol{\theta})||^2 dt \right] \tag{2.12}$$

In (2.12) $[\cdot]^H$ denotes the Hermitian operator. By maximizing the ML function (or a proportional cost function) over the time interval with length D_o, the estimates of the path delays are found:

$$\hat{\boldsymbol{\theta}}_{ML}(y) \in arg \max_{\boldsymbol{\theta}} \{\Lambda(\boldsymbol{\theta}; y)\} \tag{2.13}$$

This maximization can be carried out by several iterative algorithms, such as *Expectation Maximization* (EM) [8] [9] [10] [31] [5] and *Alternating Projections* (AP) [9]. Following the EM approach, the path parameters θ are determined by maximizing (2.13). The advantage of ML-based algorithms is that they have a higher resolution than the power delay profile-based algorithms.

2.5.2.1. Channel acquisition for wideband CDMA signals

A method for selecting paths in a channel that is suitable for Wideband Code Division Multiple Access systems is described in [21]. The channel between the base station and the mobile terminal is modelled as a tapped delay line filter with a certain delay spread. Furthermore no Doppler effects occur and the fading type is slow frequency selective Rayleigh fading.

The case is considered in which a base station has contact with ten mobile terminals in its cell (uplink). The parameters of the paths of these mobile terminals (attenuation, delay) are available in the base station. Under these circumstances it is investigated what happens if there is a new

mobile terminal coming into the cell. The question is how can the parameters of the paths of the new user be estimated. Using knowledge of the structure of the MAI it is possible to reduce its impact.

The path search algorithm is based on a chip-matched filter that makes use of a pilot tone in the received signal. As the channel is estimated, the channel model is approached by a finite tapped delay line filter with all taps on equal distance of each other. This means that only the delay of the first tap (initial delay) needs to be estimated, as well as the gains of the taps. Using a Minimum Mean Square Error (MMSE) estimate for the sent symbols of the interfering mobile terminals, a Maximum Likelihood estimate of the parameters of the new user's paths can be made. All taps are estimated at the same time (joint acquisition).

The second step of the algorithm is selecting the paths that contain the most energy. In order to do this a channel model is calculated with a limited number of taps that shows the highest correlation with the estimate of the channel that was found earlier. Besides this correlation-based method another method for tap-reduction is presented. This method simply chooses the most significant taps from the estimated channel model. This method can be extended by also looking at the taps that lie in the direct vicinity of the most significant taps.

The performance of the algorithm is examined by a number of simulations. The performance measure that is used is the correlation between the reduced channel model (step 2 of the algorithm) and the estimated channel model (step 1 of the algorithm). This correlation is determined as a function of the Signal to Noise Ratio. It appears that the method that selects the most significant taps has the highest correlation for different values of SNR. The algorithm can suppress a limited number of interfering users, using the MMSE approach. Because of this the algorithm is MAI resistant. By reducing the number of users to be suppressed, the algorithm can be made less complex. This will deteriorate the performance of the algorithm, which leads to a tradeoff.

2.5.2.2. A sequential algorithm for joint parameter estimation and multiuser detection in DS/CDMA systems with multipath propagation

In [8] an algorithm is presented that uses joint parameter estimation and detection (JED) in a W-CDMA system with several users. The algorithm will be used in the base station (uplink). It is assumed that there are six users in the cell and that the channel has three paths within a certain delay spread. The fading type is Rayleigh fading with a slow, frequency selective

character and no Doppler effects occur. The channel model exists of an impulse response that has been measured on five places in an urban area.

The algorithm has been based on a correlator and uses a preamble for initialization. A log-likelihood function is determined for valuing both the symbols that are sent and the parameters of the different paths. Given the complexity of this function, it is proposed to estimate the sent symbols for well-known parameters of the paths and to estimate the parameters of the paths as known symbols are sent. The path parameter estimation will be carried out by the Expectation Maximization algorithm (see also [31] for the JED-EM the method).

For this the conditional expectation of the log-likelihood function is derived. Next, this function can be maximized, starting from an initial value for the parameters of the paths. The maximization of the function is subdivided into two iterative algorithms, one for valuing the delays and one for valuing the attenuation of the paths. Initialization takes place by using a preamble (pilot symbols).

The performance of the algorithm is analyzed by calculating the Root Mean Squared Error (RMSE) of the attenuation estimate, for the case of only one user and one path of which the delay is known. Also an expression is found for the bit error probability, for the case of strong MAI induced by the presence of other users. Using simulations, the influence of the number of the iterations used by the algorithm on the RMSE of the attenuation estimate is determined. It appears that the algorithm can also give a good estimate of the attenuation under the influence of strong MAI.

The bit error probability of the algorithm has also been simulated for strong MAI. In order to compare the results the bit error probability of a Rake receiver with perfect estimates of the delays and attenuation is used, as well as the bit error probability of a BPSK system with only AWGN. The simulations show that the algorithm is much less sensitive to ISI and MAI than the Rake receiver and it closely approaches the performance of the BPSK/AWGN receiver.

2.5.2.3. A new time-delay estimation in multipath

In [16] the problem of path searching is discussed in a broader sense, it not only concerns W-CDMA systems, but also appears in sonar, radar, seismic research etceteras. A variety of techniques exist for estimating the delays of multipaths, e.g. Maximum Likelihood Estimation-based methods (MLE), Least Squares (LS), Expectation Maximization (EM) and Autocorrelation Estimation (AE).

In this paper MLE and AE are compared with each other. A channel is assumed that can be modelled by a tapped delay line, there are two multipaths and the noise is AWGN. An expression for the MLE is inferred, based on the Power Spectral Density (PSD) of the signal. Also an expression is presented for the sample autocorrelation function for EM. Both methods provide an estimate of the path delays.

Next the AE method is generalized to a General Autocorrelation Estimator (GAE). This method is no longer dependent on the PSD of the sent symbols, which means that a pilot tone is no longer necessary. A cost function is derived, that is to be maximized in order to value the delays. After estimation of the autocorrelation function the local maxima of this estimate can be determined. Using the cost function the time delays of the paths are then found.

The performance of the GAE algorithm is measured by calculating the MSE of the delay estimate. Simulations are carried out with the MSE as a function of the SNR, as well as simulations with the MSE as a function of the delay. For comparison the MLE and AE algorithms from the first part of the paper are used, as well as the CRLB. For low SNR values MLE seems to perform just as badly as AE. For higher SNR levels MLE performs considerably better than AE, but it requires more calculations than AE. Simulations show that GAE approaches the CRLB for both white and colored noise.

2.5.2.4. Iterative techniques for DS/CDMA multipath channel estimation

In [9] an algorithm is presented that uses a Maximum Likelihood approach in a W-CDMA system with several users (see also [10]). The algorithm will be used in the base station (uplink). It is assumed that there are ten users in the cell and that the channel has three paths within a certain delay spread. The tenth user has just arrived in the cell and the parameters of its paths are now to be valued. Rayleigh fading occurs with a slow, frequency selective character without Doppler effects. The MAI is modelled as colored Gaussian noise and an attempt is made to suppress it. The channel is modelled as a discrete tapped-delay line filter.

As the tenth user comes into the cell, it transmits a known training sequence. The received signal is first processed by a filter that is matched to the chip waveform. Then the output of the matched filter is sampled at the chip rate. A Maximum Likelihood function is derived for estimating the delay and attenuation of each of the new user's paths. This function also contains the covariance matrix of the training sequence, this matrix is estimated as well.

In the paper two methods are presented for maximizing the ML-function. These methods are based on Expectation Maximization (EM) and Alternating projection (AP). Both methods have been formulated as an iterative algorithm and produce an estimate of the delay and attenuation of the paths of the new user.

With the presented algorithms several simulations have been carried out. The deviation (RMSE) between the estimates and the real values of the attenuation and delays of the paths has been examined. Also the chance has been defined that all parameters of the new user's three paths are correctly valued. In the simulations the performances of the EM and AP algorithms are compared to those of a simple correlator from literature and the CRLB. Also the performances of the EM and AP algorithms are determined for the case they use the exact covariance matrix, instead of an estimated version.

From the simulations it becomes clear that the EM and the AP algorithms have a similar performance and approach the CRLB. The complexity of the calculations of the EM algorithm is considerably lower than those of the AP algorithm. On the other hand the AP algorithm requires less iterations in order to converge. Therefore a proposal is made to use the AP algorithm for initializing the EM algorithm.

2.5.2.5. Weighted RLS channel estimators for DS/CDMA signals in multipath

In [3] an algorithm is described that can be used for estimating multipath parameters in a W-CDMA system. No specific choice is made for using the algorithm in the base station or the mobile terminal. It is assumed that there are eight users in the cell. Rayleigh fading occurs with a slow, frequency selective character without Doppler effects. The channel is considered as a Finite Impulse Response (FIR) filter, that is constant over one symbol time.

The received signal is filtered and sampled. Then a Maximum Likelihood cost function is defined, that has to be minimized in order to calculate an estimate for the parameters of the channel. For this minimization a Recursive Weighted Least Squares (RWLS) algorithm is described. To make sure that the complexity of the calculations remains limited, a bank of parallel RWLS estimators is used.

Finally the sensitivity of the presented algorithm has been examined to synchronization mismatches using simulations. Its performance has been compared to that of a Single User Matched Filter (SUMF) receiver and a MMSE receiver.

2.5.2.6. Multipath time-delay detection and estimation

In [24] an algorithm is presented for estimating the delay and the attenuation of paths in a multipath system in a general sense. The presented algorithm is especially suitable for a channel in which the path delays are closely spaced. It is assumed that the channel imposes AWGN, no MAI or Doppler effects are mentioned.

The first step of the algorithm consists of matched filtering of the received signal. According to the reconstruction theorem the samples at the output of the matched filter can be expressed as a matrix multiplied by a weighing vector plus a noise vector. The indices of the components of the weighing vector that are nonzero, correspond to the delays of the paths. The required weighing factors are samples of an interpolation function, such as the sine cardinal function.

In case of oversampling with for example a factor two it is necessary to choose a reconstruction criterion for selecting the correct reconstruction function. Since the minimum l_2 norm always leads to the same sine cardinal function, it is proposed to choose the minimum l_1 norm for this. The minimization of this norm is formulated next and this turns out to be the so-called deconvolution criterion. By minimizing this expression, the path delays can be found.

A number of simulations have been carried out with the presented algorithm. The deviation between the estimates and the real values for the attenuations and delays of the paths has been examined. For comparison the performances of the Complex to Real Least Squares (CRALS) and Projection Onto Convex Sets (POCS) algorithms are used. It appears that the presented algorithm performs better than CRALS and POCS. The algorithm also functions at low SNR, up to 20 dB lower than CRALS and POCS. The complexity of the algorithm's calculations is described as reasonable. An additional advantage of the algorithm is that no initialization is required.

2.5.3 Subspace-based path searchers

A third group of path search algorithms is based on the covariance matrix of the received signal. First the baseband signal is correlated with the user signature and then sampled. The next step is to calculate an estimate of the covariance matrix of this signal.

By eigendecomposition of the covariance matrix, the eigenvectors of the signal subspace and the noise subspace can be found. In principle the signal subspace and the noise subspace are orthogonal and can therefore be separated from each other. In the papers that were found this orthogonality is exploited by respectively determining a Super Delay Profile and the use of a Toeplitz displacement.

This way the channel can be identified and the path delays can be determined. The algorithms have the advantage that they require much less knowledge of the user signatures of all the users in a cell (unlike the ML methods) and they can achieve a higher path delay resolution.

2.5.3.1. Multipath delay estimation using a superresolution PN-correlation method

A number of algorithms exist for multipath delay estimation with high resolution. One of these algorithms is the superresolution Pseudo-Noise sequence correlation method (SPM), based on MUSIC. In [11] a number of improvements for the SPM algorithm are proposed and the performance is examined. The paper does not address techniques specific for W-CDMA systems, but discusses the path searching problem in a broader context. Instead of analyzing the influence of MAI, the effects of a narrowband interferer are studied.

The SPM algorithm is based on a correlator that calculates the correlation of the received signal with the chipping sequence. For the delay profile vector that is determined this way, an expression is inferred:

$$\underline{y} = \sum_{l=1}^{L} h_l \underline{r}(\tau_l) + \underline{\tilde{w}} \tag{2.14}$$

In (2.14) $\underline{r}(\tau)$ denotes the steering vector containing the autocorrelation of the user signature. In short:

$$\underline{y} = \Gamma \underline{g} + \underline{\tilde{w}} \tag{2.15}$$

With $\Gamma = [\underline{r}(\tau_1) \dots \underline{r}(\tau_L)]$ and $\underline{g} = [h_1 \dots h_L]^T$.

Also an expression is found for the covariance matrix R_y of vector $\underline{y}$:

$$R_y = \Gamma G \Gamma^H + R_n \tag{2.16}$$

In (2.16) $G = E\{\underline{g}\underline{g}^H\}$ and $R_n = E\{\underline{\tilde{w}}\underline{\tilde{w}}^H\} = \sigma^2 R_0$. By estimating R_y and determining the eigenvectors $\underline{v}$ of this matrix a super resolution delay profile (SDP) can be determined as:

$$SDP(\tau) = \frac{\underline{r}^H(\tau) {R_0}^{-1} \underline{r}(\tau)}{\sum_{m=M+1}^{L} |\underline{r}^H(\tau)\underline{v}_m|^2} \tag{2.17}$$

With M as the number of samples at the output of the correlator. An estimate of the covariance matrix can be made for time-variant channels as well (slow frequency selective fading) to find the paths.

In many cases the covariance matrix can not be estimated precisely enough. This can be solved by exploiting the relation between the used carrier frequency and this matrix. By switching the carrier frequency while the PN sequence is sent (both in the transmitter and receiver), several estimates of the matrix can be made. Finally a frequency smoothed (FS) estimate of the covariance matrix is made by averaging these estimates.

If the jumps in the carrier frequency are not chosen carefully, it is possible that FS hampers the performance of SPM. In order to prevent this, a proposal is presented to choose the jumps in the carrier frequency at random: Random Frequency Smoothing (RFS). This technique is more able to find the paths than FS, even for a small number of jumps in the carrier frequency. The performance of the SPM/RFS algorithm has been investigated extensively in an artificial pond with a transducer and a hydrophone.

2.5.3.2. A Toeplitz displacement method for blind multipath estimation for long code DS/CDMA signals

An algorithm capable of making blind estimates of multipath parameters in a W-CDMA system is presented in [6]. No specific choice is made for use of the algorithm in the base station or the mobile terminal. Eight up to twelve users are assumed to be in the cell. No Doppler effects occur, while the type of fading is Rayleigh fading with a slow, frequency selective character. The PN code is long (randomized spreading) and the delay spread is much smaller than the symbol time. The MAI is modelled in detail.

The received signal is chip-matched filtered and then sampled at the chip rate, the matched filter output $\underline{y}(n)$ is:

$$\underline{y}(n) = \sum_{k=1}^{K} \mathbf{C}_k(n)\mathbf{H}_k\mathbf{b}_k(n) + \mathbf{w}(n) \tag{2.18}$$

In (2.18) $\mathbf{C}_k(n)$ denotes a spreading code matrix, containing all spreading sequences. The channel is modelled as $\mathbf{H}_k$ and the sent data as $\mathbf{b}_k(n)$. Next an expression is inferred for the covariance matrix $\mathbf{R}_y$ of the received signal from user k.

$$\mathbf{R}_y = \sigma_k^2\mathbf{S}_k(n)\mathbf{C}_k(n)\mathbf{H}_k{\mathbf{H}_k}^H{\mathbf{C}_k}^H(n){\mathbf{S}_k}^H(n) + \mathbf{R}_I(n) + \mathbf{R}_w(n) \tag{2.19}$$

With $\mathbf{S}_k$ as a matched filtering matrix. By performing a Toeplitz displacement of $\mathbf{R}_y$ the noise $\mathbf{R}_w(n)$ and interference $\mathbf{R}_I(n)$ can be removed:

$$\mathbf{R}_h = \overline{\mathbf{SC}_k}^{+}\mathbf{H}_k{\mathbf{H}_k}^H{\overline{\mathbf{SC}_k}^{+}}^H - \overline{\mathbf{SC}_k}^{-}\mathbf{H}_k{\mathbf{H}_k}^H{\overline{\mathbf{SC}_k}^{-}}^H \tag{2.20}$$

The paper argues that the eigenvectors of this covariance matrix $\mathbf{R}_h$ contain the estimate of the channel parameters. Eigendecomposition of $\mathbf{R}_h$ leads to a matrix $\mathbf{V}$ containing the eigenvectors. The channel estimate is determined by:

$$\hat{\mathbf{h}}_k = \arg \min_{||\mathbf{h}||=1} \text{Trace}\{H^H\mathbf{P}H\} \tag{2.21}$$

with $H = [(\overline{\mathbf{SC}_k}\mathbf{H}_k)^+, (\overline{\mathbf{SC}_k}\mathbf{H}_k)^-]$ and $\mathbf{P} = \mathbf{I} - \mathbf{V}\mathbf{V}^H$

The performance of the presented algorithm is assessed by determining the MSE of the channel estimate using simulations. For comparison the MSE of an algorithm from literature is used. It appears that the performance of the presented algorithm is similar to that of the algorithm from literature.

ref.	system type	channel model	fading type	delay spread	Doppler effects	scenario	MAI modelling and suppression	channel estimation	evaluation of paths	path selection
[21]	w-cdma	tapped delay line	Rayleigh fading slow + freq. sel.	not specified	none	uplink 10 users L paths	MAI suppresion with MMSE technique	MMSE-ML technique	determining power in paths	select maxima with Max-L_r technique
[25]	w-cdma	filter in discrete time	not specified	20 μs	none	downlink K users L paths	MAI is modelled as AWGN	correlation of P-CPICH with replica	calculate PDP (non-coher. averaging)	setting threshold for max. in PDP
[13]	w-cdma	not specified	Rayleigh fading	not specified	f_d = 5.6 Hz	downlink 1 user 2 paths	none	correlation of common pilot symbols	PDP based, 3 man. tables for path data	select ten strong. paths from PDP
[28]	w-cdma	L-path model	Rayleigh fading slow + freq. sel.	not specified	v = 30 km/h 120 km/h	uplink 1 user L paths	none	correlation of DPCCH	accumulation (coherent + non-coh.) sq. env. detect.	thresholding
[18]	w-cdma	ITU-R Veh. B L-path model	Rayleigh fading slow + freq. sel.	not specified	f_d = 80 Hz 320 Hz	uplink 1-2 users 1-4 paths	none (fast TPC)	correlation of pilot symbols with replica	calculate PDP (coherent averaging)	thresholding: PDP - 4 strongest paths
[12]	utra-fdd	ITU-R Veh. B 3GPP specs	Rayleigh fading slow + freq. sel.	not specified	none	uplink K users L paths	none	correlation of pilot symbols with replica	calculate PDP	thresholding PDP - WMSA filtered signal
[4]	w-cdma	not specified	Rayleigh fading freq. sel.	not specified	none	uplink 10 users 6 paths	partial MAI suppression	correlation with spreading sequence	calculate PDP select subset of paths by power	thresholding with respect to strongest path
[36]	w-cdma	Jakes' model	Rayleigh fading slow + freq. sel.	not specified	v = 3 km/h 30 km/h 160 km/h	downlink K users 2 paths	none	correlation of CPICH coh. summ.	square envelope detect. + post detect. integr.	thresholding?
[8]	w-cdma	meas. impulse response at 5 locations	Rayleigh fading slow + freq. sel.	$< T_b$	none	uplink 6 users 3 paths	MAI suppression with ML-JED technique	correlation of preamble	ML-JED	Expectation-Maximization

ref.	**verification**	**signal source**	**performance measure**	**compared with**	**cost**
[21]	simulations	BPSK, SF = 31, sync (9 chips)	correlation of estimated channel and reduced channel	opt. correl., conventional algorithm	
[25]	simulations	not specified	a single PDP	no comparison	1689 byte 32.8 MIPS
[13]	simulations	QPSK, specified in great detail	BLER as function of SNR	conventional receiver + perfect rcv.	
[28]	simulations	$SF_c = 256$ $SF_d = 64$ $N_p = 9$	P_d, P_m and P_f, mean acquisition time	no comparison	
[18]	simulations field experiments	QPSK specified in great detail	required E_b/N_0 of the transmitter for a BER = 10^{-3}	no comparison	
[12]	simulations	12.2 kbps RRC pulse	MSE of noise var. req. E_b/N_0 transm. for a BER = 10^{-3}	conv. recv. thr = PDP - 4 thr = PDP - 6	6 MIPS
[4]	simulations	not specified	BER as function of SNR	conv. rcv. + rcv. selecting fixed paths	
[36]	simulations	not specified	ROC	no comparison	
[8]	simulations	not specified (BPSK?)	RMSE of amplitude estim. BER	perfect Rake BPSK/ AWGN	

ref.	system type	channel model	fading type	delay spread	Doppler effects	scenario	MAI modelling and suppression	channel estimation	evaluation of paths	path selection
[16]	general case	tapped delay line	not specified	not specified	none	both links K users 2 paths	none	GAE	determine local maxima of estimated autocorrelation	find path delays with cost function
[17]	w-cdma	sub-chip: overlap. paths	Rayleigh fading	$< T_c$	none	downlink 2-30 users 3 paths	MAI modelled as AWGN	correlation and Teager Kaiser operation	squared envelope det. + non-coh. averag.	maximum selection
[11]	general case	path delay variation	slow + freq. sel.	$< 0.2T_c$	none	high path resolution 3-4 paths	no MAI, only narrowband interference	estimation of covariance matrix	calculate super-resolution delay profile	
[24]	general case	not specified close paths	not specified	not specified	none	link not specif. K users 3 paths	none		deconvol. method with with min. L_1-norm	minimize using the min. L_1-norm
[37]	w-cdma	not specified	Rayleigh fading slow + freq. sel.	not specified	v = 160 km/h	downlink K users L paths	none	correlation of CPICH coh. summ.	sq. envelope detect. + post detection integration	thresholding?
[9]	w-cdma	discrete tapped delay line	Rayleigh fading slow + freq. sel.	$< 0.5T_s$	none	uplink 10 users 3 paths	MAI is modelled as colored Gauss. noise	matched filtering of training sequence	ML criterion	maximize using EM and AP
[6]	w-cdma	not specified	Rayleigh fading slow + freq. sel.	$<< T_s$	none	both links 8-12 users L paths	MAI is described in channel model	estimation of covariance matrix	Toeplitz Displacement and eigen decomposition	
[3]	w-cdma	FIR filter	Rayleigh fading slow + freq. sel.	$< 16T_c$	none	both links 8 users L paths	none		ML cost function	minimize using RWLS technique

ref.	**verification**	**signal source**	**performance measure**	**compared with**	**cost**
[16]	simulations	not specified	MSE of delay	MLE algorithm + CRLB	
[17]	simulations	QPSK, rect RRC pulses Walsh SF = 64	chance of detecting all paths	MUSIC + MEDLL + PS	
[11]	measurem.	transducer and hydrophone	several PDP's	SPM + SPM-FS	
[24]	simulations	linear FM	error in delay and amplitude estimates	CRALS + POCS	
[37]	simulations	not specified	ROC	no comparison	
[9]	simulations	Gold seq. (31 bits), SNR = MAI = 10 dB	RMSE of amp. and delay, chance of detecting all paths	simple correlator + CRLB	
[6]	simulations	not specified	MSE of channel estimate	algorithm from literature	
[3]	simulations	Walsh-Hadamard 4PSK	?	SUMF + MMSE receiver	

2.6 Selection of three algorithms

One of the questions within the AWGN project (Chapter 1) is: to what extent will it be useful to switch between different types of algorithms for the path search function? In order to get a clear picture of the different techniques for the implementation of the path search algorithm a number of papers has been discussed in the literature study. The algorithms that were found have been compared with each other (Section 2.4) and have been discussed in detail in Section 2.5. The algorithms have been classified in three classes, based on the similarities discovered among the various algorithms. Three algorithms are now selected for further research. Two algorithms will be implemented: the Power Delay Profile-based path searcher (Chapter 4) and the Maximum Likelihood-based path searcher (Chapter 5). The subspace-based algorithm will be discussed in theory only, see Appendix B.

The first class consists of the Power Delay Profile-based algorithms (Subsection 2.5.1). These algorithms correlate the received signal with a replica of the user signature. Next a Power Delay Profile (PDP) is calculated based on this correlated signal, from which the strongest paths are selected using a threshold value. In [18] extensive simulations and measurements were carried out with this method. The implementation of the PDP-based path searcher will be performed using [25], as it describes its principle in great detail (see Chapter 4). Improvement of this algorithm will then be possible by calculating the threshold value according to [12].

The second group of algorithms is based on a Maximum Likelihood approach (Subsection 2.5.2). The received signal is filtered first with a matched filter and then sampled. Based on this signal a Maximum Likelihood Estimate of the path delays is formulated. By maximizing the MLE function the delays can be determined. This maximization is carried out in [8], [16], [10] and [9] using Expectation Maximization, as well as in [31] and [5]. The MLE-EM method provides good performances for a low computational cost and is therefore selected for optimizing the Maximum Likelihood function. The implementation of the MLE-EM path searcher will be based on [8], as it describes its principle in great detail (see Chapter 5).

Subspace-based algorithms (Subsection 2.5.3) constitute the final group. These algorithms correlate the received signal with the user signature, whereupon the covariance matrix of this correlated signal is estimated. Using eigendecomposition of the covariance matrix the eigenvectors of the signal and noise subspaces are found. The orthogonality of these subspaces [6] [11] is exploited to estimate the path delays. As [6] describes an algorithm specific for CDMA systems it provides a basis for the third path search algorithm. The theory of the algorithm will be discussed in Appendix B.

3

Simulation environment for path search algorithms

3.1 Multipath channel models

Now that the various options for the implementation of the path search algorithm have been investigated in the literature study, the selected algorithms will be implemented. In order to set up meaningful simulations with these algorithms first a number of channel models are analyzed in this section. Also scenarios will be formulated in the next section that describe the conditions in the channel between the transmitting base station and the receiving mobile terminal.

The main features of the channel model become clear from formula (2.3), which is repeated here for convenience:

$$h_k(t) = \sum_{l=1}^{L} \alpha_{k,l}(t)\delta(t - \tau_{k,l}(t))e^{j\phi_{k,l}} \tag{3.1}$$

In equation (3.1) L indicates the number of paths for user k in the channel and $\alpha_{k,l}(t)$ and $\tau_{k,l}(t)$ indicate respectively the path attenuation and delay. The formula describes that the channel can be viewed as a sum of L multipaths, each having a specific delay $\tau_{k,l}(t)$ and attenuation $\alpha_{k,l}(t)$.

During the literature study various channel models were encountered that try to capture the behavior of this Rayleigh fading channel between the base station and the mobile terminal. In the next subsections the models will be discussed that are used in the papers from the literature study. They include Jakes' channel model (Subsection 3.1.1), the L-path channel model (Subsection 3.1.2) and models from ETSI (Subsection 3.1.3).

3.1.1 Jakes' channel model

A classic model for Rayleigh fading that is widely used (e.g. in [36] and [19]) can be found in [29]. The field is modelled as:

$$E(t) = Re[T(t)e^{j\omega_c t}] \tag{3.2}$$

with

$$T(t) = \frac{E_0}{\sqrt{N}} \sum_{n=1}^{N} e^{j(\omega_m t \cos \alpha_n + \phi_n)} \tag{3.3}$$

It is assumed that the angles of arrival are uniformly distributed, $\alpha_n = 2\pi n/N$ with n in $[1, N]$. The Doppler frequency is denoted as ω_m. A problem of this model is that it is not Wide Sense Stationary. This issue is analyzed in [32] and an improvement of the Jakes' channel model is proposed. Also a new block scheme for the channel simulator is presented.

Unfortunately, part of the second order statistics of the improved Jakes' channel model is not modelled correctly. In [7] the autocorrelation and cross-correlation of the signal quadrature component and the autocorrelation of the squared envelope are analyzed. In [20] the correlations of the Jakes' channel model are investigated as well. Especially the distribution of the scatterers in the channel are investigated.

Another drawback of the Jakes' channel model is that it does not produce uncorrelated paths under all circumstances. To solve this problem, improvements are proposed in [15].

3.1.2 L-path channel model

In most of the papers the L-path channel model was used: [28], [18], [8] and [11]. This model is also referred to as the tapped delay line channel model in [21], [25], [16], [9] and [3]. It is described in [34] as:

$$h_k(t) = \sum_{l=1}^{L_k} h_{k,l}(t)\delta(\tau - \tau_{k,l}(t))e^{j\phi_{k,l}} \tag{3.4}$$

Both the attenuation and delay can change over time. This means that a path can change in length (and thus delay), it can disappear ($h_{k,l}(t) = 0$ for certain values of t) or a new path can appear ($h_{k,l}(t) > 0$ for certain values of t).

In [22] $h_{k,l}(t)$ is referred to as a Wide Sense Stationary narrow-band complex Gaussian process with a Jakes' power spectrum. Also [14] denotes the character of the stochastic process as Wide Sense Stationary Uncorrelated

Scattering, with $h_{k,l}(t)$ as an independent Rayleigh distributed random variable. The Rayleigh distribution is defined in [34] and [27] with the following probability density function:

$$p(r) = \begin{cases} \frac{r}{\sigma^2} e^{\frac{-r^2}{2\sigma^2}} & 0 \leq r \leq \infty \\ 0 & r < 0 \end{cases} \tag{3.5}$$

3.1.3 3GPP and ITU-R Vehicular B channel models

In [12] one of the channel models that is used is the model specified by 3GPP (see ETSI document [1]). Of this model Case 2 and Case 3 are used for simulations. In [1] the multipaths are defined in terms of their relative delay and attenuation, the Case 2 channel model has three paths and the Case 3 channel model has four paths. These models also include Doppler effects (for respectively $v = 3$ km/h and $v = 120$ km/h), but in [12] these models were made stationary.

The other channel model that is used in [12] (and in [18] as well) is the ITU-R Vehicular B model (see ETSI document [23]). It models a rural or suburban outdoor macro cell environment at a maximum velocity of $v = 500$ km/h. Again, the model is made stationary in [12]. It consists of a tapped delay line and has six paths, of which the delay and attenuation are defined. Also traffic models are established for the services that are carried by the links in the cell (e.g. speech and web browsing).

The propagation effects are divided in three groups. The first group is formed by the effects that cause the mean path loss. The path loss is modelled as:

$$L = 40(1 - 4 \cdot 10^{-3} \Delta h_b) \log_{10}(R) - 18 \log_{10}(\Delta h_b) + 21 \log_{10}(f) + 80dB \tag{3.6}$$

In equation (3.6) R denotes the base station - mobile terminal separation in km, f is the carrier frequency and Δh_b is the base station antenna height. The second group causes slow signal variation, by shadowing and scattering. The slow signal variation is log-normally distributed. The last group causes rapid signal variation, due to multipath effects. This is described by the channel impulse response model, of which the six path delays and attenuations are defined.

3.1.4 Measurements of channel impulse responses

An alternative for modelling the channel's impulse response is the use of actual measurements of its impulse response. In [8] 500 different channel impulse responses have been obtained in urban areas. For each simulation a channel impulse is selected randomly from these measurements.

In [18] simulations are accompanied by laboratory experiments. Both are used to optimize the path searcher's parameters. Finally field experiments are carried out to investigate the performance of the optimized path searcher. For this a measurement vehicle has been driven along two different courses (of length 0.6 and 1.1 km) at a number of speeds (20 to 40 km/h). The first course is located in a factory area and the second course near a residential area. The corresponding power delay profile measurements are shown as a function of time in [18] and clearly display the different characteristics of the two radio environments.

In the simulation environment that is available the L-path model is used. The fading type can be set to either Rayleigh or AWGN. Also two vectors of length L can be used to set the channel parameters for user k in terms of the path delays $\tau_{k,l}$ and path gains $\alpha_{k,l}$. The values that will be used for these channel parameters will be discussed in the next section.

3.2 Scenarios for simulations

As previously noted, it is necessary to analyze the channel model first in order to set up meaningful simulations with the path search algorithms from the literature study. Also scenarios need to be formulated that describe the conditions in the channel between the transmitting base station and the receiving mobile terminal. This section gives an overview of likely scenarios, in Subsection 3.2.1 the characteristics of an outdoor rural area are discussed, the outdoor urban area is analyzed in Subsection 3.2.2 and Subsection 3.2.3 describes an indoor office environment. Finally two effects are commented upon in Subsection 3.2.4. These are variable path delays over time and the so-called birth-death sequence of paths. Table 3.1 gives an overview of these scenarios . More information on scenarios can be found in [1] and [23].

3.2.1 Outdoor rural area

The outdoor rural area is divided into large cells (macro cell) that are covered by a small number of base stations with high transmit powers. Signals travel over relatively long distances which causes a root mean squared delay spread of about 370 to 4000 ns. The outdoor rural area is characterized by a low path loss, following a R^{-2} rule. This is due to the low number of scatterers

Scenario	**cell size**	**transmit power**	**path loss**	**r.m.s. delay spread**	**mobile speed**
Indoor office area	pico / micro	low	variable	35 - 100 ns	walking speed
Outdoor urban area	micro / macro	low	R^{-6} R^{-4}	45 - 750 ns	vehicle speed
Outdoor rural area	macro	high	R^{-2}	370 - 4000 ns	vehicle speed

Scenario	**Doppler effects**	**fading effects**	**shadowing effects**
Indoor office area	v = 3 km/h	Ricean, Rayleigh	log-normal 12 dB
Outdoor urban area	v = 50 km/h	Ricean, Rayleigh	log-normal 10 dB
Outdoor rural area	v = 120 km/h	Rayleigh	log-normal <10 dB

Table 3.1: Scenarios for mobile terminal environment

and obstructions in the paths of the channel, as the terrain is mostly flat. The fading type is mainly Rayleigh and shadowing effects are modelled by a log-normal distribution with a standard deviation below 10 dB. Doppler effects occur as the mobile terminal can have speeds up to 120 km/h. This scenario corresponds to the Vehicular B channel parameters in Table 3.2.

3.2.2 Outdoor urban area

The outdoor rural area is divided into smaller cells (micro to macro cells) that are covered by base stations with low transmit powers. The root mean squared delay spread lies between 45 and 750 ns. The outdoor urban area is characterized by a range of path losses (R^{-6} to R^{-4}), depending on the number of high buildings, base station antenna height, obstructions by trees etcetera. The number of scatterers and obstructions in the paths of the channel can vary. The fading type can be both Ricean and Rayleigh, shadowing effects (see [34]) are modelled by a log-normal distribution with a standard deviation of 10 dB. Doppler effects occur as the mobile terminal can have speeds up to 50 km/h. This scenario corresponds to the Case 3 and Pedestrian B channel parameters in Table 3.2.

3.2.3 Indoor office area

The indoor office area is divided into small cells (pico to micro cells) that are covered by many base stations with low transmit powers. The root mean squared delay spread lies between 35 and 100 ns, as the path lengths tend to be short. The indoor office area is characterized by a large range of path losses, as the number of scatterers and obstructions in the paths of the channel can vary a lot. Walls, floors and surfaces of e.g. filing cabinets cause both attenuation and scattering. The fading type can therefore be both Ricean and Rayleigh, shadowing effects are modelled by a log-normal distribution with a standard deviation of 12 dB. Doppler effects occur due to walking speeds of the mobile terminal (3 km/h). This scenario corresponds to the Indoor Office B channel parameters in Table 3.2.

Vehicular B	0	300	8900	12900	17100	20000	ns
	-2.5	0.0	-12.8	-10.0	-25.2	-16.0	dB
Case 3	0	260	521	781	-	-	ns
	0.0	-3.0	-6.0	-9.0	-	-	dB
Pedestrian B	0	200	800	1200	2300	3700	ns
	0.0	-0.9	-4.9	-8.0	-7.8	-23.9	dB
Indoor office B	0	100	200	300	500	700	ns
	0.0	-3.6	-7.2	-10.8	-18.0	-25.2	dB

Table 3.2: Four path delay and power vectors

3.2.4 Propagation conditions

In most of the papers that were found during the literature study it was assumed that the channel is stationary, or at least stationary over one symbol period. This is not necessarily true, as both $h_{k,l}(t)$ and $\tau_{k,l}(t)$ can be functions of time. As the mobile moves from one location the next within a cell, or if an object passes the mobile terminal nearby, it is possible that changes occur in the path parameters or even in the number of paths.

The path searcher needs to be sensitive to changes in path delays after it has converged to an estimated path delay for the first time. If for example, the delay of the l^{th} path $\tau_{k,l}(t)$ changes, it needs to be estimated again. An example of a changing path delay is provided in [11].

Another possibility is that the number of paths changes over time. This can happen for example as the mobile terminal is moving from a rural area into an urban area. The number of scatterers will increase, causing a larger number of paths. This can be described as a change of the l^{th} path attenuation $h_{k,l}(t)$ from zero to a certain value. This is denoted as the 'birth' of a path. The opposite, $h_{k,l}(t)$ goes to zero, is called the 'death' of a path. These effects are described in [1].

3.2.5 Modifications to the simulator

A model of the downlink in a UMTS system is available in C++. As the algorithms will operate in the mobile terminal they need to communicate with the downlink receiver (see Appendix C.2). The number of modifications to the downlink receiver (see Appendix C.3) can be kept to a minimum by creating a virtual base class for the path search algorithm (see Appendix C.1). A number of set() and get() functions handle the communication between the downlink receiver and the path searcher. All functions are pure virtual functions with an empty body, as they are members of the abstract path searcher class.

In the header file of the downlink receiver (see Appendix C.2) the path searchers' abstract base classes need to be included as well as the various versions of the path searcher (such as the path searcher classes in Appendixes C.4 and D.1). Using a pointer to the abstract base class, it is possible to switch between different implementations of the path searcher in the source file of the downlink receiver (see Appendix C.3).

In order to support the simulations discussed in Section 3.2.4 a second basestation is added to the simulator. This way the situation shortly before a handover can be simulated: the mobile terminal is at the boundary of the first basestation's cell and detects a second base station with a signal strength equal or greater than that of the first basestation.

The current simulator only supports channels of which the parameters are constant during the entire simulation run. By updating the channel parameters for every new frame that is processed, it is possible to investigate whether the algorithms can adapt to the changing channel parameters. Moving propagation conditions are simulated by using a channel with two paths at 0 ns of strength 0 dB. Each frame the second path is moved over 512 ns.

4

A Power Delay Profile-based path searcher

4.1 Introduction

As part of the literature study several papers have been discussed that analyze so-called *Power Delay Profile* (PDP)-based path searchers. One of these algorithms [25] has been selected for implementation. This chapter discusses the PDP-based path search algorithm and its implementation in detail. The algorithm consists of five functional blocks that will be discussed in the following paragraphs.

The algorithm correlates the complex received signal with the pilot tone from the Primary Common Pilot Channel (P-CPICH). By calculating the correlation over the length of a frame, a delay profile is found (Section 4.3.1). This delay profile is averaged over a number of previous frames (Section 4.3.2), to suppress fast fluctuations induced by interference. This results in a Power Delay Profile (PDP). The local maxima in the PDP are selected using a three-points method (Section 4.3.3). The height of these maxima (attenuation) is compared with a threshold (Section 4.3.4). If it exceeds the threshold a path is detected and the associated delay is forwarded to a Rake receiver (Section 4.3.5). The PDP path searcher is shown in Figure 4.1.

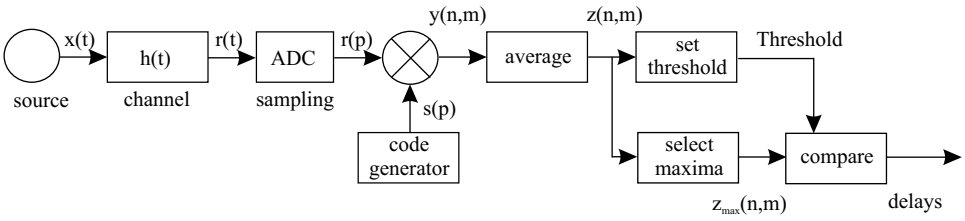

Figure 4.1: Power Delay Profile-based path searcher

4.2 Setting up interfaces and structuring the algorithm

A model of the downlink in a UMTS system is available in C++. As the algorithm will operate in the mobile terminal it needs to communicate with the downlink receiver. The downlink receiver provides the path searcher with both the conjugate of the user signature and the received signal. It also sets the spreading factor of the path searcher as it is initialized. The path searcher returns the number of paths and their delays to the downlink receiver. Based on the number of paths and the path delays the downlink receiver can set the Rake fingers accordingly.

The path searcher algorithm has been implemented in Matlab at first to gain more insight in its structure. As the model of the UMTS downlink is available in C++ the PDP path searcher needs to be modelled in C++ as well. The number of modifications to the downlink receiver can be kept to a minimum by creating a virtual base class for the path search algorithm (see Appendix C.1). A number of set() and get() functions handle the communication between the downlink receiver and the path searcher. All functions are pure virtual functions with an empty body, as they are members of the abstract path searcher class.

In the header file of the downlink receiver (see Appendix C.2) the path searcher's abstract base class needs to be included as well as the various versions of the path searcher (such as the PDP-based path searcher class, see Appendix C.4). Using a pointer to the abstract base class, it is possible to switch between different implementations of the path searcher in the source file of the downlink receiver (see Appendix C.3).

4.3 Discussion of the PDP algorithm in detail

4.3.1 Estimating the delay profile using correlation

The path searcher itself is described in the source file, see Appendix C.5. The algorithm requires initialization before the path search operation can be performed. As mentioned previously the algorithm consists of five functional blocks. The first block correlates the complex received signal $r(p)$ with a replica of the user signature $s(p)$ for the n^{th} frame (p is the discrete time index). This replica of the user signature is provided to the path searcher by the downlink receiver (it is obtained by the cell searcher). Generally the received signal consists of a series of pilot symbols, which means that this signal contains the user signature $s(p)$.

$$r(p) = \sum_{l=1}^{L} h_l(p)s(p - \frac{\tau_l}{T_c}) + w(p) \tag{4.1}$$

The received signal consists of the signal components of the transmitter travelling through L paths with complex channel coefficients $h_l(p)$. Both the Multiple Access Interference and the Additive White Gaussian Noise are contained by $w(p)$.

Next the correlated signal $y(n, m)$ is obtained, that contains a number of peaks for the different delays (n is the frame index and m is the correlation index). These peaks correspond to the paths in the channel between transmitter and receiver.

$$y(n, m) = \frac{1}{N} \sum_{p=0}^{N-1} r(p + m)s^*(p) \tag{4.2}$$

$$y(n, m) = \frac{1}{N} \sum_{p=0}^{N-1} \sum_{l=1}^{L} h_l(p)\delta(m - \frac{\tau_l}{T_c}) + \tilde{w}(n, m) \tag{4.3}$$

N is the correlation window size, $m \in [0, P - 1]$, P is the correlation length, $\delta(m - \frac{\tau_l}{T_c})$ results from the autocorrelation of the user signature $s(p)$ and $\tilde{w}(n, m)$ is the correlated noise. The correlated signal is stored in a matrix, each row corresponds to a frame with index n.

4.3.2 Calculating the Power Delay Profile

Next the correlated signal $y(n, m)$ is averaged with the correlated signal from the previous M frames, in order to find the most significant paths. This way a Power Delay Profile $z(n, m)$ is determined. The PDP is a measure for the received power in each of the paths that is found (M indicates the length of the summation).

$$z(n, m) = \frac{1}{M} \sum_{k=0}^{M-1} |y(n - k, m)|^2 \qquad m \in [0, P - 1] \tag{4.4}$$

4.3.3 Detection of local maxima in the PDP

Local maxima are detected in the PDP by checking the height of each sample in the PDP. If it exceeds the height of both the preceding and subsequent sample a local maximum is found. Its index and height are each stored in a vector for further processing. The number of maxima is counted as well.

4.3.4 Setting a threshold for the PDP

Finally a limited number of peaks from the PDP is selected. In order to do this a threshold value η is set for the power of the peaks. In [25] the threshold is set to:

$$\eta = \frac{1}{P} \sum_{m=0}^{P-1} z(n,m)(2 + \frac{4}{\sqrt{M}}) \tag{4.5}$$

M is the length of the averaging process and P the length of the correlation.

4.3.5 Selecting the paths from the PDP

Each peak from the PDP is compared to the threshold, if it exceeds the threshold a path is detected. The corresponding index and height are each stored in a vector. The number of paths is counted as well. Finally signal values are stored in a status file for visualization of the path searching process.

4.4 Simulation of the PDP path searcher

The correct functioning of the algorithm has been verified by simulation (see Figure 4.2). A path search was carried out for a Rayleigh fading channel with paths at 0 and 8 samples (0 ns and 2050 ns). Both paths had a gain of 0 dB, the SNR was 10 dB and the mobile speed 3 km/h. Both paths were found correctly by the algorithm. More detailed simulations will be carried out to analyze the performance differences of the Power Delay Profile-based algorithm compared to a Maximum Likelihood-based algorithm.

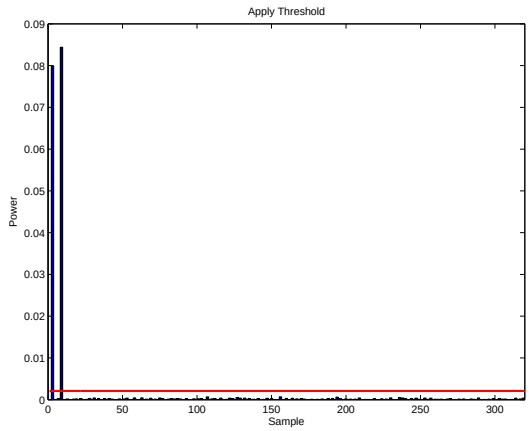

Figure 4.2: Simulation for a channel with two paths

5

A Maximum Likelihood Estimation algorithm

5.1 Introduction

As part of the literature study several papers have been discussed that analyze so-called *Maximum Likelihood Estimation* (MLE) path searchers using *Expectation Maximization* (EM). One of these algorithms [8] has been selected for implementation. This chapter presents the MLE-EM path search algorithm and its implementation in detail. The algorithm is implemented in C++ and uses the same interfaces to the downlink receiver as the Power Delay Profile algorithm.

For a better understanding of the algorithm a model for the received complex baseband signal will be discussed first (Section 5.2). Then the principles of both Maximum Likelihood Estimation (Section 5.3) and Expectation Maximization (Section 5.4) are stated. The MLE-EM algorithm consists of five functional blocks and will be discussed in detail in Section 5.5. The C++ code can be found in Appendices D.1 and D.2.

5.2 Model of the baseband received signal

The complex baseband signal $u(t)$ of the transmitter travels to the receiver through a channel via L paths [5][26]. Each path contributes to the complex baseband signal $y(t)$ in the receiver:

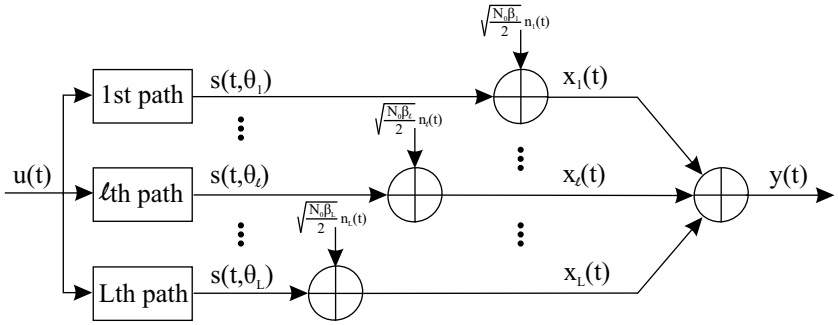

Figure 5.1: Multipaths

The contribution of the l^{th} path with complex gain α_l and delay τ_l is:

$$s(t, \theta_l) = \alpha_l u(t - \tau_l) \qquad \theta_l \triangleq [\alpha_l, \tau_l] \tag{5.1}$$

Each signal is corrupted by complex white gaussian noise $n_l(t)$, which leads to $x_l(t)$. The signal $x_l(t)$ is defined as:

$$x_l(t) = s(t, \theta_l) + \sqrt{\frac{N_0 \beta_l}{2}} n_l(t) \tag{5.2}$$

$$n(t) = \sum_{l=1}^{L} n_l(t) \qquad n_l(t) = \sqrt{\beta_l} n(t) \tag{5.3}$$

The noise $n(t)$ is divided over $x_l(t)$ by the factors β_l, with $\sum_{l=1}^{L} \beta_l = 1$. The received signal $y(t)$ is related to the signals $x_l(t)$ as follows:

$$y(t) = \sum_{l=1}^{L} x_l(t) \tag{5.4}$$

The complex baseband received signal $y(t)$ can therefore be described as the sum of the path signals and a complex white gaussian noise term $n(t)$:

$$y(t) = \sum_{l=1}^{L} s(t, \theta_l) + \sqrt{\frac{N_0}{2}} n(t) \tag{5.5}$$

5.3 Principle of Maximum Likelihood Estimation

A likelihood function can now be defined in terms of $y(t)$ and the path parameters $\boldsymbol{\theta} \triangleq [\theta_1, \ldots, \theta_L]^T$ as was done in [8], [5] and [26]:

$$\Lambda(\boldsymbol{\theta}; y) \triangleq \frac{1}{N_0}\left[2\int_{D_o} \Re\Big(s^*(t,\boldsymbol{\theta})y(t)\Big)dt - \int_{D_o} ||s(t,\boldsymbol{\theta})||^2 dt\right] \tag{5.6}$$

In (5.6) D_o denotes the time span over which paths are searched (the correlation window), $s(t,\boldsymbol{\theta}) = \sum_{l=1}^{L} s(t,\theta_l)$. By maximizing the likelihood function the estimates of the path parameters are found:

$$\hat{\boldsymbol{\theta}}_{ML}(y) \in arg\max_{\boldsymbol{\theta}}\{\Lambda(\boldsymbol{\theta}; y)\} \tag{5.7}$$

As the complexity of this calculation requires a vast amount of computational power, the likelihood function is split up into L parts (see (5.4)):

$$\Lambda(\theta_l; x_l) \triangleq \frac{1}{N_0\beta_l}\left[2\int_{D_o} \Re\Big(s^*(t,\theta_l)x_l(t)\Big)dt - \int_{D_o} ||s(t,\theta_l)||^2 dt\right] \tag{5.8}$$

$$\Big(\hat{\theta}_l\Big)_{ML}(x_l) \in arg\max_{\theta_l}\{\Lambda(\theta_l; x_l)\} \tag{5.9}$$

5.4 Principle of Expectation Maximization

The maximization (5.9) of the derived likelihood function (5.8) can be carried out by Expectation Maximization (EM). As the data $x_l(t)$ is not available, it is estimated in terms of the conditional expectation of $x_l(t)$ using $y(t)$ and a previous estimate of the channel parameters $\hat{\boldsymbol{\theta}}$.

Each iteration μ of the algorithm is divided into two steps. The first step provides an estimate $\hat{x_l}(t)$ of $x_l(t)$ and the second step calculates the value of θ_l that maximizes the likelihood function.

Expectation step:

$$\hat{x}_l(t, \hat{\boldsymbol{\theta}}(\mu)) \triangleq E_{\hat{\boldsymbol{\theta}}(\mu)}\Big[x_l(t)\Big|y\Big] \qquad (l = 1, \ldots, L) \tag{5.10}$$

Maximization step:

$$\hat{\theta}_l(\mu+1) = arg\max_{\theta_l}\left\{\Lambda\Big(\theta_l, \hat{x}_l\big(t, \hat{\boldsymbol{\theta}}(\mu)\big)\Big)\right\} \qquad (l = 1, \ldots, L) \tag{5.11}$$

A general diagram of the Expectation Maximization algorithm is shown in Figure 5.2 [5]. It can be implemented by first estimating the signals $\hat{x}_l(t)$ using the estimate of $\hat{\boldsymbol{\theta}}$ from the previous iteration. This decomposition of $y(t)$ follows from (5.4), $\hat{\boldsymbol{\theta}}(0)$ is the initial value.

The next step is to carry out L Maximum Likelihood Estimations to find all values for $\hat{\boldsymbol{\theta}}$. If the algorithm has converged sufficiently $\hat{\boldsymbol{\theta}}_{ML}$ is assigned its value.

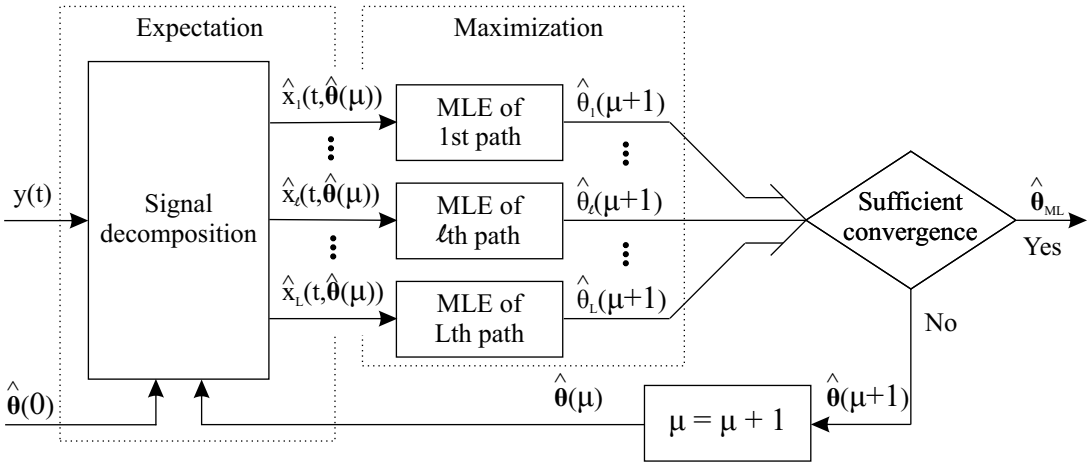

Figure 5.2: Expectation Maximization

5.5 Discussion of the MLE-EM algorithm in detail

The algorithm (see Figure 5.3) is an iterative algorithm that consists of five different operations. These operations will be discussed in the following subsections. The algorithm attempts to decompose the complex received signal $y(t)$ into L path signals $\hat{x}_l(t)$. This is referred to as the Expectation step. First the shifted scrambling codes are formed (Subsection 5.5.1) and then the path signals are estimated (Subsection 5.5.2).

The next step of the algorithm is the Maximization step. First, a maximum likelihood function is calculated for the estimated path signals (Subsection 5.5.3). This function is then maximized to determine an estimate of the path parameters (Subsection 5.5.4).

Before proceeding to the next iteration, the convergence rate of the algorithm is determined (Subsection 5.5.5). If the algorithm has converged sufficiently, it stops iterating.

The following diagram shows the MLE-EM algorithm in more detail [26]:

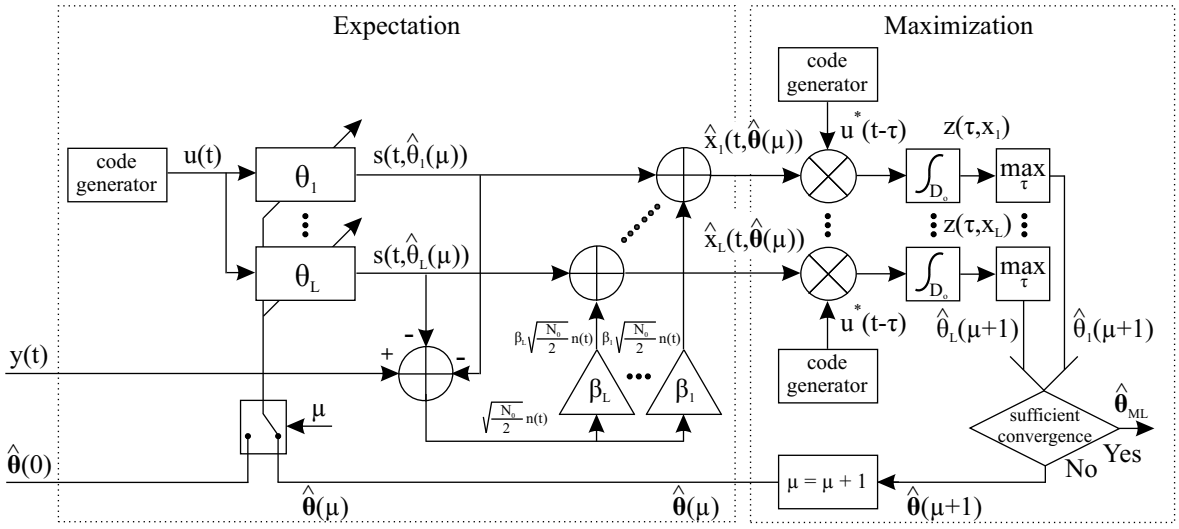

Figure 5.3: Expectation Maximization in detail

5.5.1 Expectation: Forming the shifted scrambling codes

As discussed in the previous sections (see Figure 5.1 and equation (5.5)), the received signal $y(t)$ can be described in terms of the shifted scrambling codes $s(t, \theta_l)$ and a noise term $n(t)$. In order to do this, the first step of the MLE-EM algorithm is to form the shifted scrambling codes. Each signal $s(t, \theta_l)$ can be formed by shifting the complex baseband signal $u(t)$ of the transmitter over τ_l and multiplying it with a complex gain α_l. Equation (5.1) is repeated for convenience here:

$$s(t, \theta_l) = \alpha_l u(t - \tau_l) \qquad \theta_l \triangleq [\alpha_l, \tau_l] \tag{5.12}$$

As a known pilot signal is transmitted, $u(t)$ is available in the receiver after a coarse timing synchronization has been achieved. During the first iteration of the algorithm, an initial value $\hat{\boldsymbol{\theta}}(0)$ will be used to form all $s(t, \theta_l)$. After the first iteration an estimate $\hat{\boldsymbol{\theta}}(1)$ of the path parameters is available, which will be used instead of $\hat{\boldsymbol{\theta}}(0)$. The initial value $\hat{\boldsymbol{\theta}}(0)$ is set with all values of τ_l and α_l to zero. As the first iteration will lead to a non-zero value for $\hat{\boldsymbol{\theta}}$ the algorithm will start to converge on the channel's paths, due to its monotonicity property (see [5]).

5.5.2 Expectation: Determining the path signal estimates

The next step will be to estimate the path signals $\hat{x_l}(t)$ using equation (5.2). This step is referred to as the Expectation step of the MLE-EM algorithm (5.10). Exploiting the knowledge of the composition of $y(t)$, the path signals can be calculated as following (see [8], [5] and [26]):

$$\hat{x}_l(t, \hat{\boldsymbol{\theta}}) = s(t, \hat{\theta}_l) + \beta_l \left[y(t) - \sum_{l'=1}^{L} s(t, \hat{\theta}_{l'}) \right] \tag{5.13}$$

β_l is a parameter of the algorithm that determines what amount of $n(t)$ is added to each signal $s(t, \theta_l)$, to form the path signal estimate $\hat{x}_l(t)$. The values of β_l need to be set and can be chosen freely, except for the constraint that $\sum_{l=1}^{L} \beta_l = 1$. The noise will be equally divided over the path signals by setting $\beta_l = \frac{1}{L} \forall l$, as was done in [8]. This concludes the Expectation step (5.10), in which decomposition of the received signal $y(t)$ into the signals $\hat{x}_l(t)$ is performed.

5.5.3 Maximization: Correlating the path signal estimates with the scrambling code

Having finished the Expectation step (5.10), the algorithm proceeds with the Maximization step (5.11). This requires calculating the likelihood functions $\Lambda\Big(\theta_l, \hat{x}_l\big(t, \hat{\boldsymbol{\theta}}\big)\Big) \quad l = 1, \ldots, L$. The likelihood function can be implemented by correlating each estimated path signal $\hat{x}_l(t)$ with the pilot signal u(t) (see [5] and [26]):

$$z(\tau, \hat{x}_l) \triangleq \int_{D_o} u^*(t - \tau) \hat{x}_l(t) dt \tag{5.14}$$

5.5.4 Maximization: Estimating the path parameters

Having obtained expressions for each path's likelihood function in the form of the MLEs $z(\tau, \hat{x}_l)$, the path parameters can now be estimated by maximizing these MLEs. Expressions for the values that maximize the likelihood function have been derived:

$$\Big(\hat{\tau_l}\Big)_{ML}(x_l) = \arg\max_{\tau} \{|z(\tau, x_l)|\} \tag{5.15}$$

$$\Big(\hat{\alpha_l}\Big)_{ML}(x_l) = \frac{1}{TP} z\left(\Big(\hat{\tau_l}\Big)_{ML}(x_l), x_l \right) \tag{5.16}$$

T denotes the length of the transmitted pilot signal $u(t)$ and P the transmitted power. As the MLE-EM algorithm is an iterative algorithm each iteration leads to an estimate $\hat{\boldsymbol{\theta}}$ of the path parameters.

5.5.5 Determining the convergence rate of the parameter estimates

Now that a new parameter estimate $\hat{\boldsymbol{\theta}}$ is available, the first iteration of the algorithm has finished. During the second iteration the scrambling codes $s(t, \theta_l)$ can be formed using the parameter estimate $\hat{\boldsymbol{\theta}}(1)$ that has just been obtained, replacing the previous value of $\hat{\boldsymbol{\theta}}(0)$.

If the algorithm has converged sufficiently, that is $d_\theta(\mu) < c$, with c a criterion for convergence, $\hat{\boldsymbol{\theta}}_{ML}$ is assigned its value and the algorithm stops iterating. The criterion d_θ is calculated by determining the sum of the difference between the current parameter estimate and the previous estimate of each path:

$$d_\theta(\mu) \triangleq \sum_{l=1}^{L} \frac{\left|\hat{\theta}_l(\mu+1) - \hat{\theta}_l(\mu)\right|}{\left|\hat{\theta}_l(\mu)\right|} < c \tag{5.17}$$

To make sure that the number of iterations that the algorithm can carry out is limited a parameter μ_{max} is defined. If the algorithm still has not converged sufficiently to meet the criterion c and is about to start a new iteration with μ equal to μ_{max}, it will stop iterating as well.

5.6 Simulation of the MLE-EM path searcher

The correct functioning of the algorithm has been verified by simulation (see Figure 5.4). A path search was carried out for a Rayleigh fading channel with two paths at 0 and 1 samples. The second path was 3 dB weaker than the first, the SNR was 3 dB and the mobile speed 1 km/h. Both paths were found correctly by the algorithm. More detailed simulations will be carried out to analyze the performance differences of the MLE-EM algorithm compared to the Power Delay Profile-based algorithm.

A number of remarks can be made with regard to the properties of the MLE-EM algorithm. First of all, it requires the number of paths L to be known before the path search operation starts. The value of L can be estimated using Akaike's and Rissanen's criteria [5]. If the incorrect number of paths is provided, e.g. three paths are searched while the channel has only two paths, the algorithm will not converge. The MLE-EM algorithm finds the channel's two paths and will hop between different delays for the third path. As the algorithm can not converge for the third path, it will use the maximum number of iterations, resulting in a slow behavior. If the correct number of paths is known, slow convergence can be dealt with by setting a less strict constraint c (5.17) and a lower value for μ_{max}.

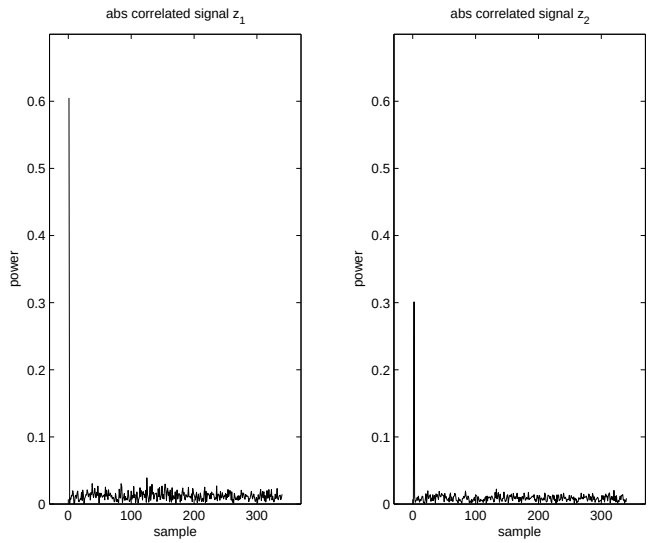

Figure 5.4: Simulation for a channel with two paths

Currently, the value of β_l is set to $\frac{1}{L}$. Another scheme is possible, in which $\beta_l = 1$. This is called the SAGE algorithm [5]. The SAGE algorithm differs from the MLE-EM algorithm in that it does not re-estimate the entire value of $\hat{\boldsymbol{\theta}}$ during each iteration, but only a subset of α_l and τ_l values. Therefore, the computational complexity of an entire iteration cycle of the SAGE algorithm is equal to that of a single MLE-EM iteration step [5].

Another issue is the algorithm's behavior during fast fades or birth and death of paths, causing the path delay profile to change within a frame. Currently, a low velocity of the mobile terminal is assumed resulting in negligible Doppler effects and slow fading. This is equivalent to a large coherence time T_{coh} of the channel: $T_{coh} \approx \frac{1}{B_d}$ with B_d the Doppler spread [33].

The current MLE-EM algorithm will search paths in the interval $\tau_l \in [0, D_o]$ in each frame. According to [1] the delay profile of a channel can change after 191 ms, which is rather slow compared to the frame time of 10 ms. Also the simulator used for testing the MLE-EM path search algorithm does not support path delay profiles that change over time. However, should the algorithm need to track fast changes of the delay profile, it is possible to split the received frames in blocks and perform the path search operation for each block [8].

6

Performance of the PDP and MLE algorithms

6.1 Goal of the simulations

The correctness of the functional behavior of both the PDP and MLE algorithm was verified by simulation as the algorithms were implemented. In order to investigate the performance of the two algorithms, additional simulations have been carried out. This chapter discusses the results of these simulations, the conclusions can be found in Section 7.2. The goal is to answer the following questions:

1. How do the algorithms perform at various levels of the SNR?
2. Which algorithm parameters influence the behavior the most?
3. Which algorithm can resolve closely spaced paths the best?
4. Do the algorithms have an acquiring and a tracking phase ?

In order to answer the first question (Section 6.2) simulations have been carried out with a Rayleigh fading channel as described in the UMTS standard ([1] and [23]), see Table 6.1. Doppler effects are investigated in Section 6.3, at first the mobile terminal velocity v is set to 3 km/h. The SNR is varied from -5 dB to 20 dB in 5 dB steps. The chip energy E_c is set to unit energy, the levels of both the DPCH and P-CPICH are set to -10 dB.

The second question, which parameters influence the algorithms' behavior the most, can be answered by varying one parameter at a time (Section 6.4). For the PDP algorithm the averaging length M and the threshold η can be varied. For the MLE algorithm the maximum number of iterations μ_{max} and the distribution of the noise β_l can be varied. Simulations are carried out at both low and high values of the SNR.

Vehicular B	0	300	8900	12900	17100	20000	ns
	-2.5	0.0	-12.8	-10.0	-25.2	-16.0	dB
Case 3	0	260	521	781	-	-	ns
	0.0	-3.0	-6.0	-9.0	-	-	dB
Pedestrian B	0	200	800	1200	2300	3700	ns
	0.0	-0.9	-4.9	-8.0	-7.8	-23.9	dB
Indoor office B	0	100	200	300	500	700	ns
	0.0	-3.6	-7.2	-10.8	-18.0	-25.2	dB

Table 6.1: Four path delay and power vectors

The resolution issue (Section 6.5) - which algorithm resolves closely spaced paths the best - is investigated by simulating a 2-path channel with the first delay equal to 0 ns. The second delay is varied by setting it to a small value and making it smaller in each simulation. The minimal distance between two paths can be one sample, by oversampling (OSF $>$ 1) it is possible to search paths within a chip. It is interesting to see whether such a high resolution estimate will be still possible if the SNR is lower.

Finally the results of the simulations are analyzed to see whether an acquiring and a tracking phase can be identified (Section 6.6). Possibly the channel estimate improves for every new frame that is processed, or the number of iterations required by the MLE algorithm goes down.

Two additional questions are of interest (Section 6.7):

1. How sensitive are both algorithms to interference?
2. Can the algorithms handle changes in the channel parameters?

In order to answer these questions, a second basestation was added to the simulator. This way the situation shortly before a handover can be simulated: the mobile terminal is at the edge of the first basestation's cell and detects a second basestation with a signal strength equal or greater than that of the first basestation. The question is: how is the path searcher's performance affected by the interfering signal of the second base station?

The current simulator only supports channels of which the parameters are constant during the entire simulation run. By modifying the simulator dynamic channel parameters can be supported. The parameters can then be varied from one frame to the next. This way it is possible to investigate whether the algorithms can adapt to the changing channel parameters. Two situations of interest are described in the UMTS standard [1]: moving propagation conditions (path delay variations in time) and birth-death propagation conditions (appearance of new paths etc.).

6.2 Performance of the algorithms in Rayleigh fading channels

6.2.1 Vehicular B channel model

A number of simulations has been carried out using the Vehicular B channel parameters (see Table 6.1) with the SNR varying over a range of -5 dB to 20 dB and an OSF equal to one. All data is averaged over twelve frames. Figure 6.1 shows the average number of correct path delay estimates. Especially the MLE algorithm benefits from the increasing SNR level, whereas the PDP algorithm performs the same. Also, the MLE algorithm finds on average one path more than the PDP algorithm.

Figure 6.2 shows the averaged power delay profiles (normalized w.r.t. the strongest path). All curves follow the same trend, the main difference is that the MLE algorithm does find the weak path at 17100 ns at high SNR. Figure 6.3 shows this as well: the MLE algorithm detects weak paths much better than the PDP algorithm. Also, the PDP algorithm cannot detect both paths at 0 ns and 300 ns at the same time (the sum of these detection percentages equals 100%), while the MLE algorithm detects both paths over 80% of the frames. The influence of the SNR on detection of weaker paths can be seen by taking a look at the path at 20,000 ns, the MLE algorithm detects this path 17% of the frames at -5 dB up to 75% of the frames at 20 dB.

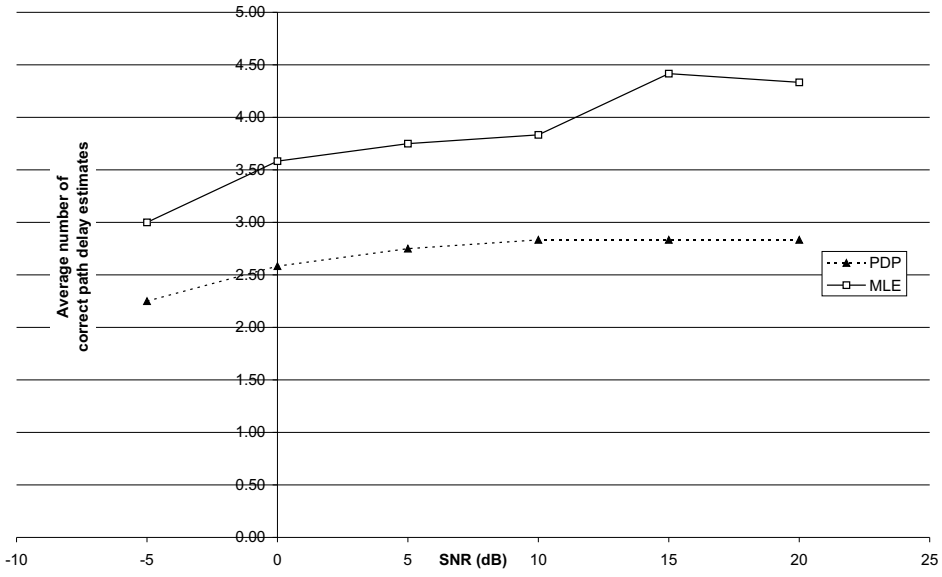

Figure 6.1: Vehicular B - Average number of correct path delay estimates

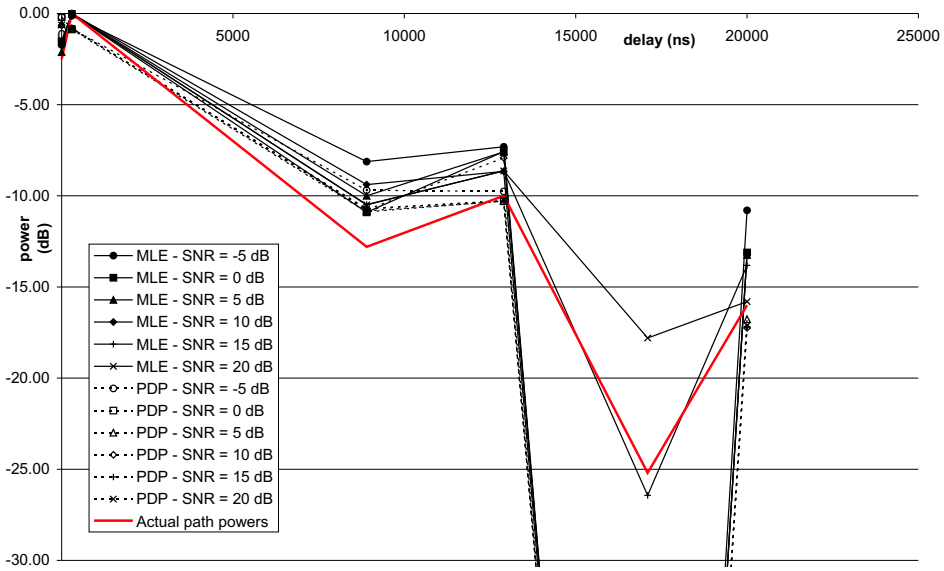

Figure 6.2: Vehicular B - Average power in paths with correct delay estimates

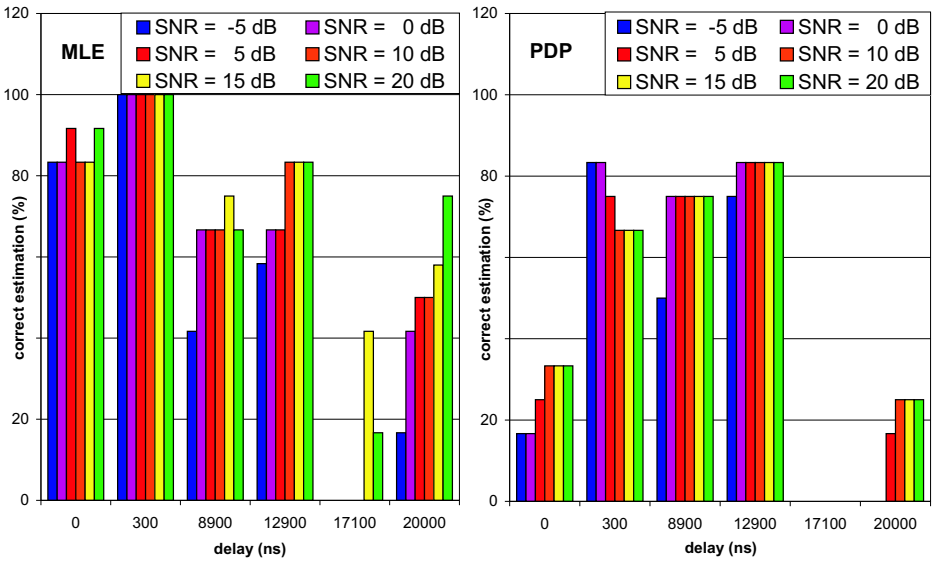

Figure 6.3: Vehicular B - Correct estimation of path delays (SNR = -5 to 20 dB)

6.2.2 Pedestrian B channel model

Using the Pedestrian B channel parameters the next series of simulations has been carried out, again with the SNR varying over a range of -5 dB to 20 dB. The OSF is set to four. All data is averaged over twelve frames. Figure 6.4 shows the average number of correct path delay estimates. Both algorithms benefit from the increasing SNR level. For low SNR levels the MLE algorithm finds on average one path more than the PDP algorithm.

Figure 6.5 shows the averaged power delay profiles (normalized w.r.t. the strongest path). It can be seen clearly that the MLE algorithm estimates the path power much better, especially at 0 ns, 200 ns and 800 ns. Again, weak paths are detected better by the MLE algorithm. It does find the weak path at 3700 ns for SNR levels of 5 dB and higher. Figure 6.6 shows this as well. Also, the PDP algorithm has trouble detecting the path at 1200 ns, while the MLE algorithm detects this path much better.

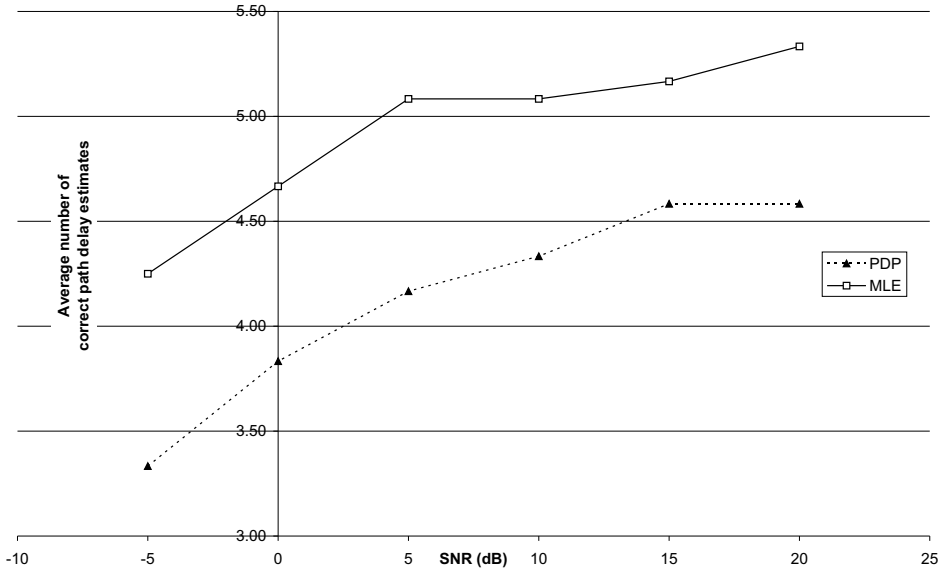

Figure 6.4: Pedestrian B - Average number of correct path delay estimates

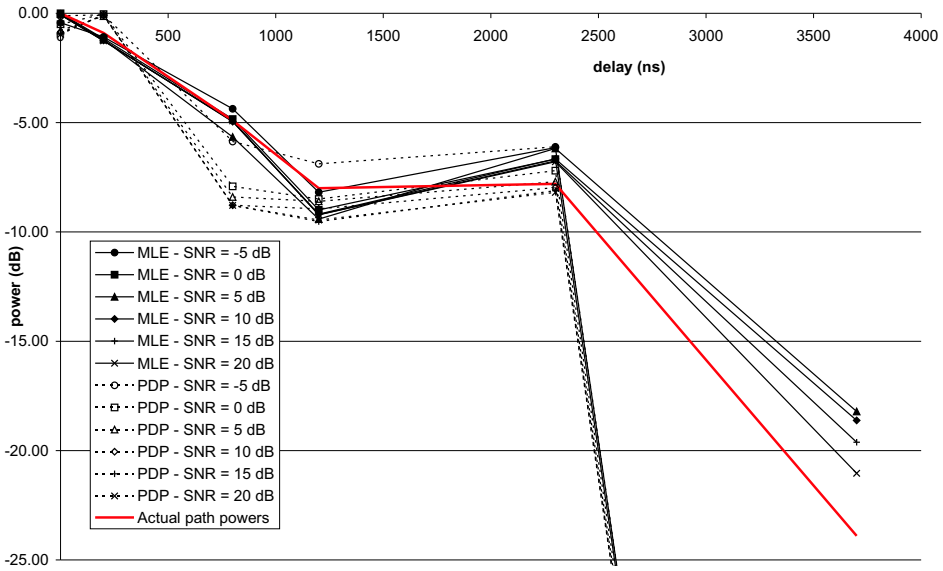

Figure 6.5: Pedestrian B - Average power in paths with correct delay estimates

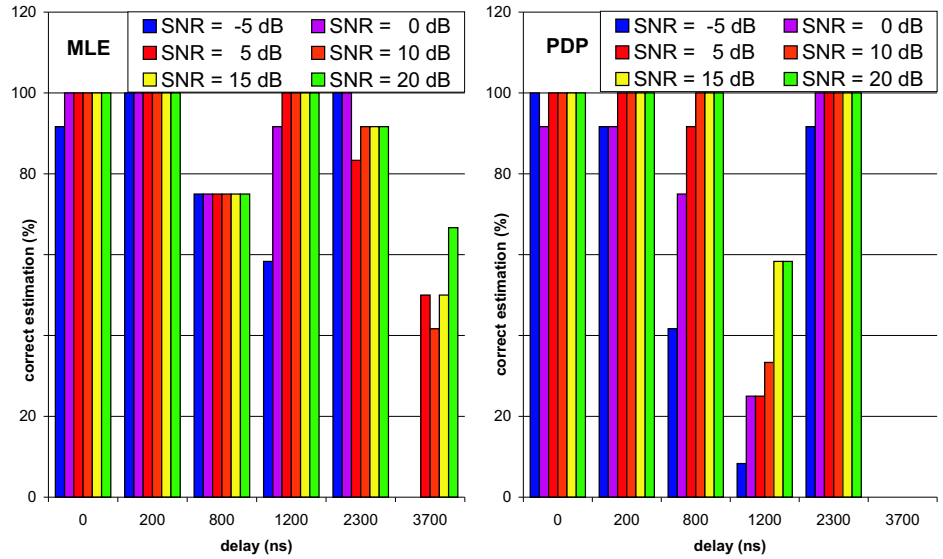

Figure 6.6: Pedestrian B - Correct estimation of path delays (SNR = -5 to 20 dB)

6.2.3 Case 3 channel model

A number of simulations has been carried out using the Case 3 channel parameters with the SNR varying over a range of -5 dB to 20 dB and an OSF equal to four. All data is averaged over twelve frames. Figure 6.7 shows the average number of correct path delay estimates. Both algorithms benefit from the increasing SNR level.

Figure 6.8 shows the averaged power delay profiles (normalized w.r.t. the strongest path). The PDP and MLE curves do not follow the same trend, the PDP algorithm estimates less power in the paths than the MLE algorithm. The MLE algorithm finds the weak path at 781 ns more often, as can be seen in Figure 6.9 as well.

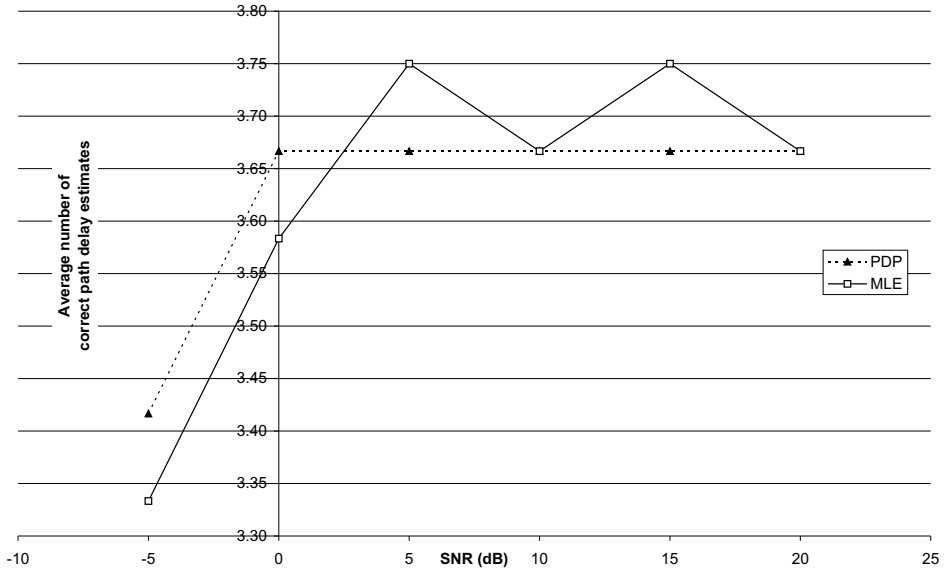

Figure 6.7: Case 3 - Average number of correct path delay estimates

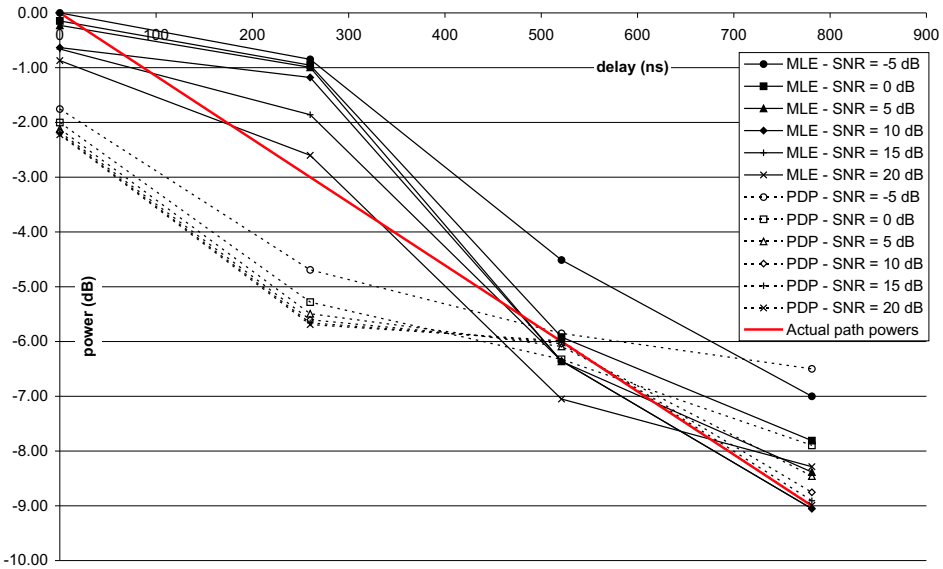

Figure 6.8: Case 3 - Average power in paths with correct delay estimates

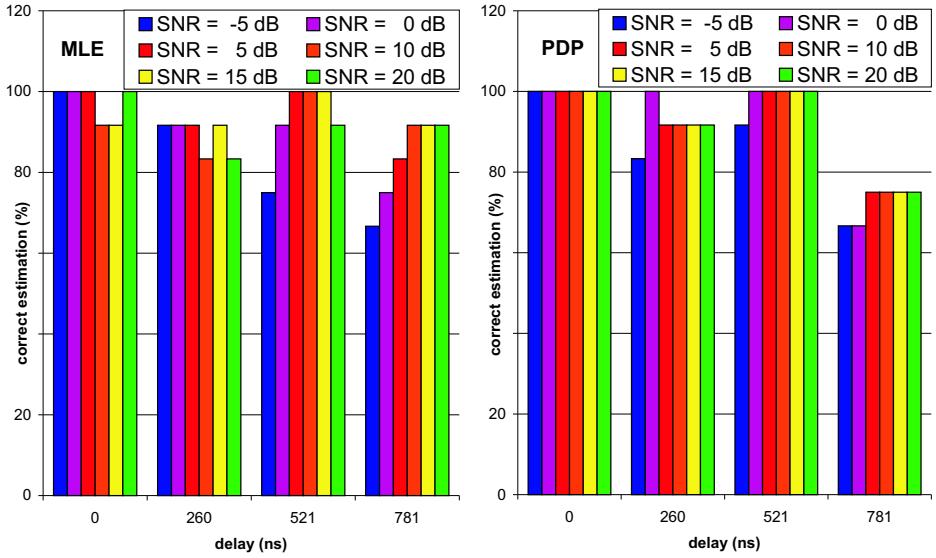

Figure 6.9: Case 3 - Correct estimation of path delays (SNR = -5 to 20 dB)

6.2.4 Office B channel model

Using the Office B channel parameters the next series of simulations has been carried out, again with the SNR varying over a range of -5 dB to 20 dB. The OSF is set to four. All data is averaged over twelve frames. Figure 6.10 shows the average number of correct path delay estimates. Only the MLE algorithm benefits from the increasing SNR level. On average the PDP algorithm finds about one path, whereas the MLE algorithm finds two to three paths depending on the SNR.

Figure 6.11 shows the averaged power delay profiles (normalized w.r.t. the strongest path). It can be seen clearly that paths can only be detected if they are separated in time by at least 100 ns. Both algorithms miss the paths at 100 ns and 300 ns. Again, weak paths are detected better by the MLE algorithm. It does find the weak paths at 500 ns and 700 ns. Figure 6.12 shows this as well. The Office B model clearly illustrates the resolution issues, which will be investigated in Section 6.5 in more detail.

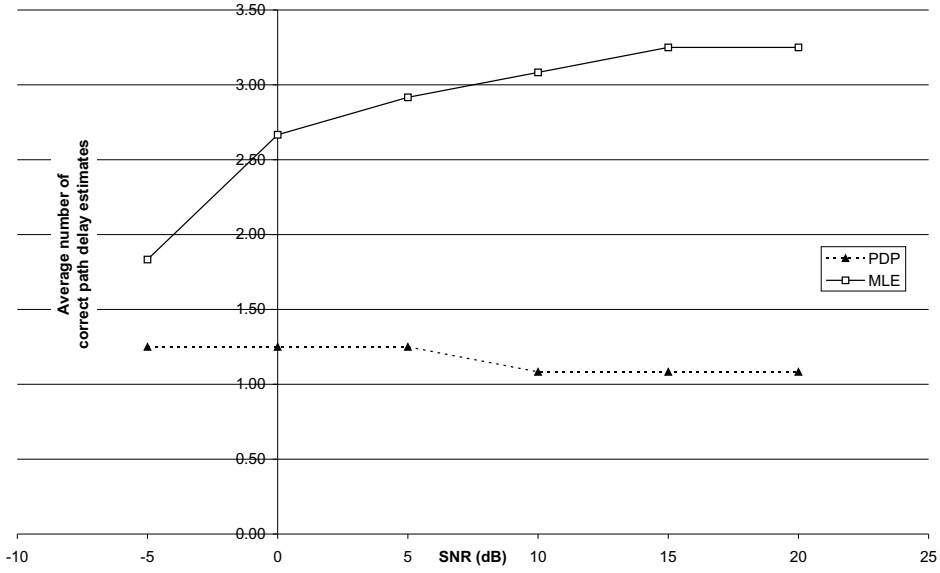

Figure 6.10: Office B - Average number of correct path delay estimates

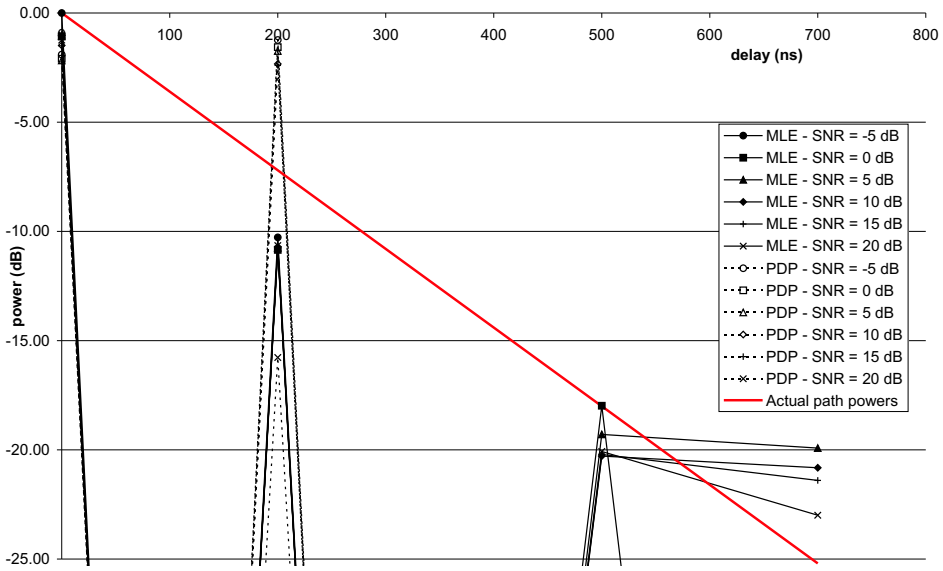

Figure 6.11: Office B - Average power in paths with correct delay estimates

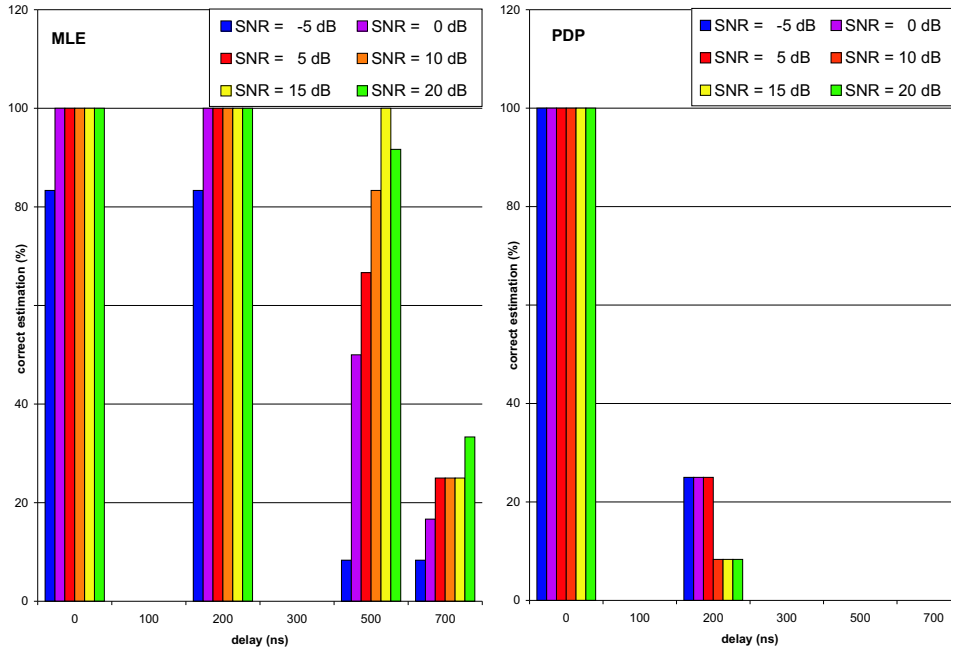

Figure 6.12: Office B - Correct estimation of path delays (SNR = -5 to 20 dB)

6.3 Doppler effects

The influence of Doppler effects on the performance of the algorithms has been investigated by carrying out a number of simulations using the Case 3 channel parameters. The SNR is set to both 0 dB and 10 dB and the OSF equals four. All data is averaged over twelve frames. Figure 6.13 shows the averaged power delay profiles (normalized w.r.t. the strongest path) and Figure 6.14 shows the path delay detection percentage. The PDP algorithm seems to be affected more by the Doppler effects than the MLE algorithm, that only shows a small degradation in path delay detection.

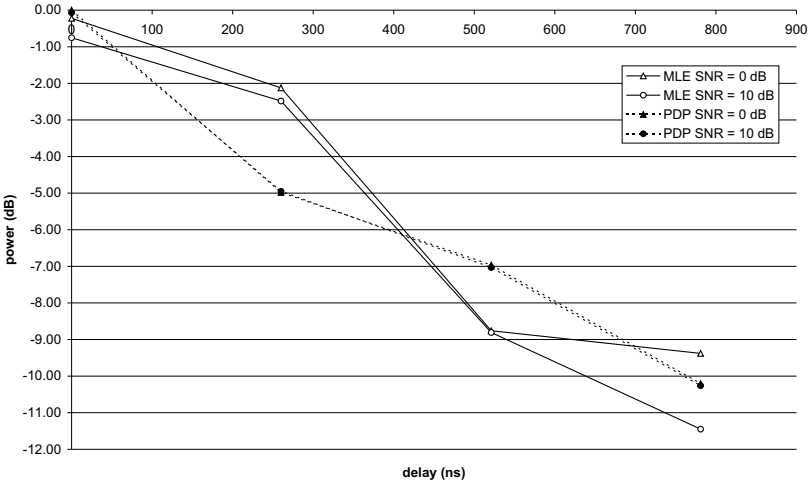

Figure 6.13: Doppler: Average power in paths with correct delay estimates

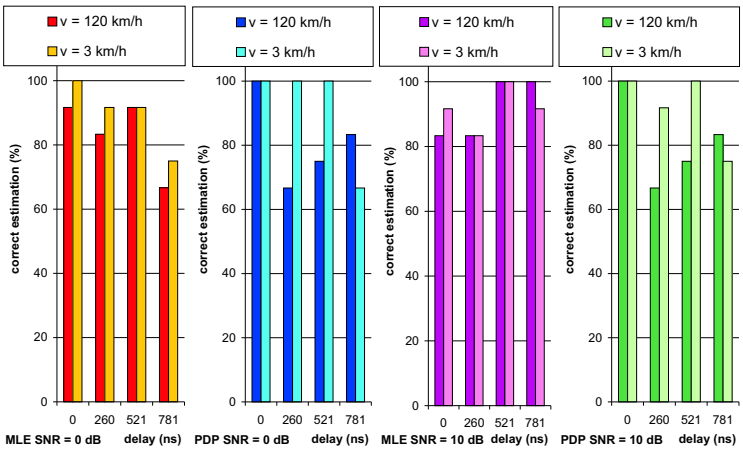

Figure 6.14: Doppler: Correct estimation of path delays

6.4 Influence of the algorithms' parameters on their performance

6.4.1 Parameters of the PDP algorithm

Averaging length M

The averaging process with length M causes the PDP algorithm to be less sensitive to fluctuations in the channel parameters. In case of changing channel parameters this becomes a disadvantage and M needs to be set to one (see Section 6.7). On the other hand, averaging is necessary to detect weak paths. To illustrate this, simulations have been carried out with a channel with paths at 700 ns (0 dB) and 2100 ns (-15 dB). The OSF is set to one, v to 3 km/h and the SNR to 0 dB. The results are presented in Table 6.2. The strong path at 700 ns is always detected, but the detection of the weak path at 2100 ns only improves if M increases.

M	paths found	detect path 1	detect path 2	avg. power path 1	avg. power path 2	average threshold
1	1.50	100%	50%	0.00 dB	-13.07 dB	-15.21 dB
2	1.67	100%	67%	-0.11 dB	-14.04 dB	-16.29 dB
3	1.83	100%	83%	-0.16 dB	-14.82 dB	-16.86 dB
4	2.00	100%	100%	-0.21 dB	-15.24 dB	-17.28 dB

Table 6.2: Influence of averaging length M on detection of weak paths

Also, as M increases, the threshold is set to a lower value. This is due to the way the threshold is calculated, using M.

Threshold η

The threshold η is calculated using the averaged power delay profile $z(m)$:

$$\eta = \frac{1}{P} \sum_{m=0}^{P-1} z(m)(a + bM^c) \tag{6.1}$$

See also Equations (2.8) and (4.5) from [25]. P is the correlation length, m the correlation index and M the averaging length. According to [25] $a = 2$, $b = 4$, $c = -0.5$ gives the best results. A number of simulations has been carried out using the Case 3 channel model with OSF $= 4$, $v = 3$ km/h and SNR $= 0$ dB. At first, a is varied from 4 down to 0, which makes no difference in the performance of the PDP algorithm. Setting $b = 2$ while $a = 0$ results in a threshold value that is too low, in this case up to four false paths are detected in addition to the desired paths. The algorithm does function correctly with these settings at higher SNR levels (10 dB).

In general it can be said that the peaks at the output of the correlator that correspond to actual paths are significantly higher than the peaks resulting from the noise. This allows for high threshold values. Other methods for calculating a threshold value based on $z(m)$ are proposed in [13] and [28] (subtracting the strongest paths from $z(m)$) and in [12] (subtracting a WMSA filtered signal from $z(m)$).

6.4.2 Parameters of the MLE algorithm

Number of paths to be estimated L

Simulations have been carried out using the Vehicular B channel (see Table 6.1) with SNR = 15 dB and OSF = 1. All data is averaged over twelve frames. Figure 6.15 shows the averaged power delay profiles (normalized w.r.t. the strongest path). It can be seen clearly that if not all six paths ($L < 6$) need to be found the weakest path at 17100 ns is omitted. Figure 6.16 shows this as well. In Table 6.3 the average number of correctly found paths and the average number of iterations is shown. Allowing the algorithm to search just the strongest four paths results in almost the same performance as in the case that five paths are searched. This way the number of iterations can be reduced by 30% causing only a slight drop in performance.

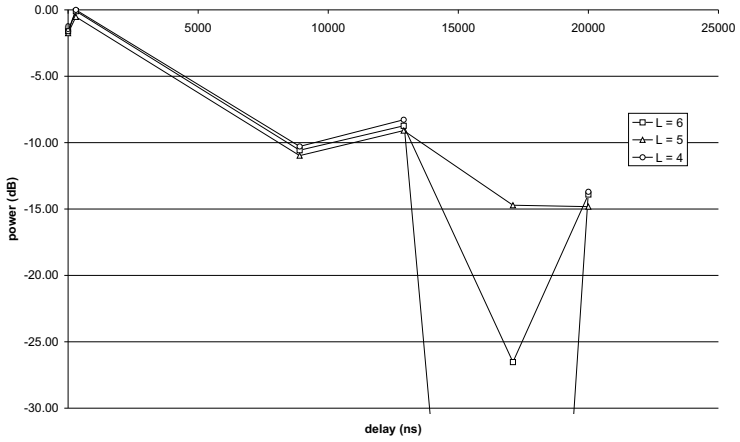

Figure 6.15: Average power in paths with correct delay estimates as function of L

Number of paths searched	Number of paths found	Number of iterations
6	4.42	47.4
5	4.17	40.3
4	3.83	28.3

Table 6.3: Effect of searching less paths than the existing number of paths

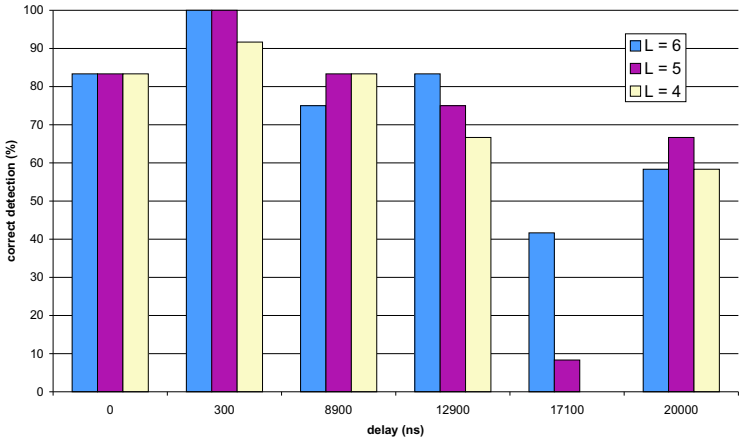

Figure 6.16: Correct estimation of path delays as function of L

Maximum number of iterations allowed μ_{max}

In order to study the effect of forcing the MLE algorithm to search paths using a fixed maximum number of iterations simulations have been carried out using the Vehicular B channel parameters. The SNR is set to 15 dB and the OSF equal to one. All data is averaged over twelve frames. Figure 6.17 shows the averaged power delay profiles (normalized w.r.t. the strongest path). For μ_{max} equal to ten the algorithm is not capable of finding the weakest path at 17100 ns. Also for twenty iterations the power delay profile differs from those for higher values of μ_{max}, but all paths are detected. This can be concluded from Figure 6.18 as well. Only in the case of ten iterations the weakest path is not found.

Distribution of noise over path signal estimates β_l

The factors β_l divide the estimate of the noise over the estimates of the path signals. Currently, the noise is spread evenly over all path signal estimates by setting $\beta_l = \frac{1}{L}$. Another option is to spread the noise estimate over the path signal estimates by strength: a larger share of the noise estimate is added to a stronger path signal estimate: $\beta_l = \frac{\alpha_l}{\sum_{l=1}^{L} \alpha_l}$. Using the Case 3 channel parameters simulations have been carried out to investigate the effects (OSF = 1, v = 3 km/h). Figure 6.19 shows that for high SNR the $\frac{\alpha_l}{\sum_{l=1}^{L} \alpha_l}$ scheme does not perform better than the $\frac{1}{L}$ scheme and for low SNR performs far worse. Also, the $\frac{1}{L}$ scheme requires less calculations.

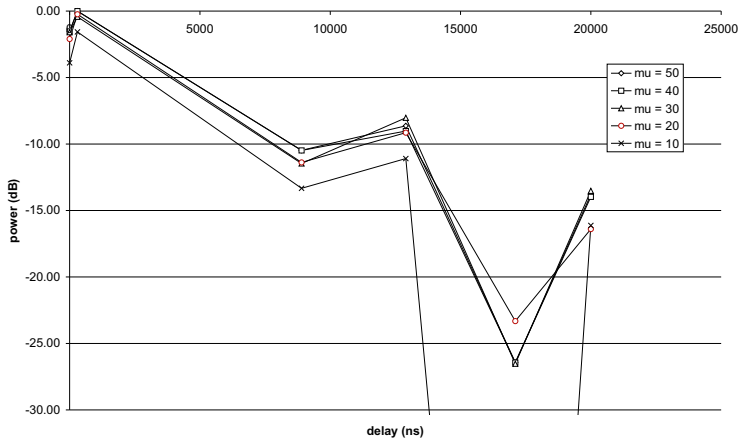

Figure 6.17: Average power in paths with correct delay estimates as function of μ

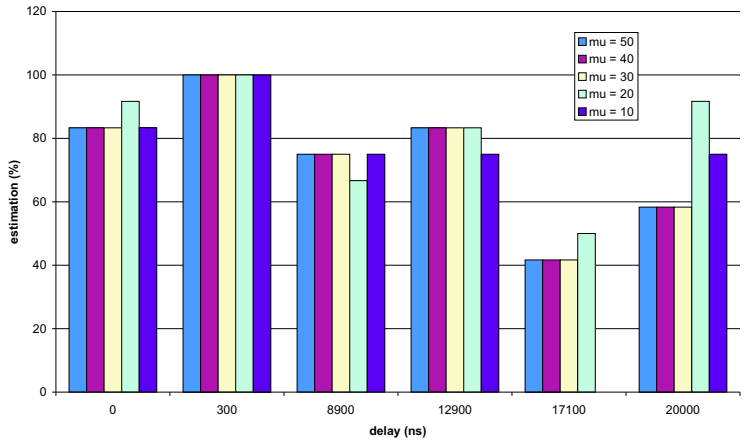

Figure 6.18: Correct estimation of path delays as function of μ

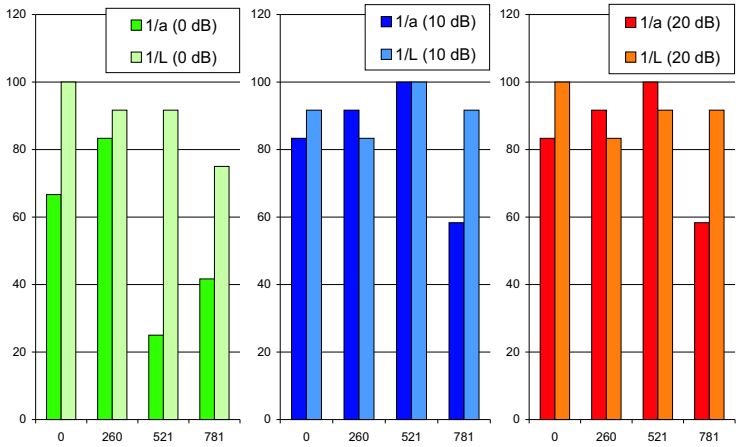

Figure 6.19: Correct estimation of path delays as function of β_l

6.5 Resolution of the algorithms: resolving closely spaced paths

In order to find out to what extent it is possible to separate paths from each other with the PDP and MLE algorithm a number of simulations has been carried out. A channel is used with paths at 0 ns and 391 ns, both with a gain of 0 dB. The SNR = 0 dB and v = 3 km/h. Next the path delay of the second path is decreased step by step. If the two paths can no longer be separated from each other the oversample factor is increased. The results are shown in Figure 6.20. It can be seen that if the two paths are spaced at a distance of 131 ns the MLE algorithm is still able to separately detect both paths using an OSF = 1. The PDP algorithm already requires an OSF = 3. Also for paths at a distance of 66 ns the MLE algorithm requires an OSF = 2, while the PDP algorithm requires OSF = 6.

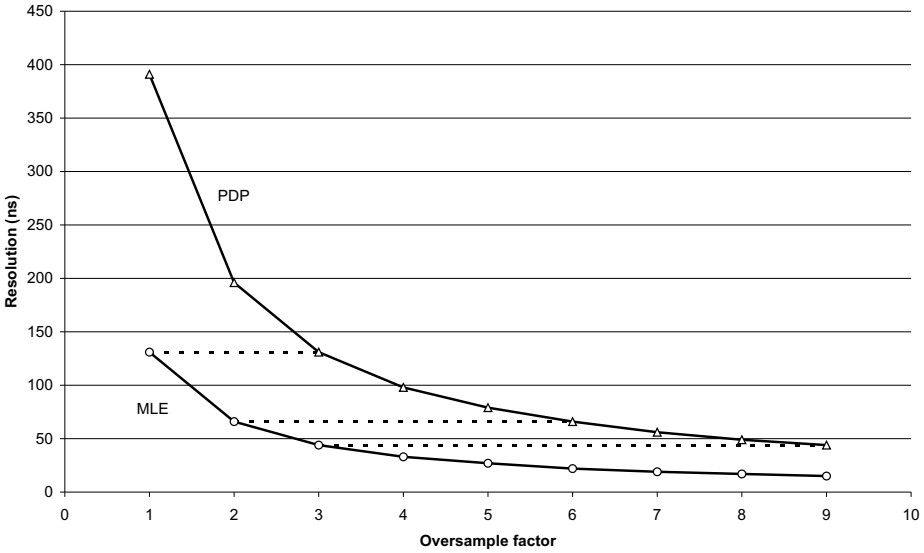

Figure 6.20: Resolution as function of the over sample factor

The performance of both algorithms for equal resolutions (i.e. 131 ns, 66 ns and 44 ns) is listed in Table 6.4. Due to the oversampling process less power is contained in the samples. Therefore the peaks at the correlator output of both algorithms will be lower, as can be concluded from the detected power in dB for the two paths. As the PDP algorithm requires an OSF of factor three higher than the MLE algorithm to achieve the same resolution, the path power diminishes rapidly. At 44 ns the first path is no longer detected and the second path only 8% of time. With an average detection of 0.08 paths the PDP algorithm no longer functions.

resolution	131 ns	131 ns	66 ns	66 ns	44 ns	44 ns
algorithm	MLE	PDP	MLE	PDP	MLE	PDP
OSF	1	3	2	6	3	9
paths found	1.83	1.58	1.42	1.08	1.25	0.08
iterations	13.1	-	22.3	-	25.3	-
det. path 1	83%	58%	50%	83%	58%	0%
pow. path 1	-0.34 dB	-10.13 dB	-8.06 dB	-11.62 dB	-7.29 dB	-
det. path 2	100%	100%	92%	25%	67%	8%
pow. path 2	0.00 dB	-9.07 dB	-3.71 dB	-15.76 dB	-6.56 dB	-20.72 dB

Table 6.4: Performance at higher over sample factors

6.6 Acquiring and tracking

Finally all simulations results are analyzed to see whether an acquiring and a tracking phase can be identified. Possibly the channel estimate improves for every new frame that is processed. Another question is: does the number of iterations required by the MLE algorithm go down as more frames have been processed? This way an acquiring phase can be defined, during which the algorithms search the channel's paths. Having found the paths, the algorithms need to hold on to these paths during the tracking phase. During the tracking phase the channel estimates should converge and less computations should be required than during the acquiring phase.

All simulation data of the PDP algorithm for the four channels (see Table 6.1) at $v = 3$ km/h and SNR = -5 dB to 20 dB is analyzed. The power of a single path (of strength 0 dB) is determined as a function of the frame number. It turns out that the path power is estimated as -5 dB in frame 1 and rises to 0 dB in frame 8 (Figure 6.21). This makes sense, as the averaging length $M = 8$. Most simulations show a 3 dB drop in frame 9. Additional simulations show that a different realization of the noise signal does not lead to this power drop. Also simulations over the double amount of frames show that the algorithm recovers after the drop. Simulations with longer averaging ($M = 12$) show a slower convergence of the power estimate.

The MLE algorithm does not show this build up in path power, also the number of iterations does not decrease over time. In fact, the algorithm requires 20 to 30 iterations to estimate all paths, only frame 9 results in 40 to 50 iterations. Additional simulations have been carried out with L paths of 0 dB at equal distance (300, 600, 900, 1200, 1500 and 1800 ns), SNR = 20 dB and $v = 3$ km/h. These simulations show that for for $L = 1$ to 6 the number of iterations does not go down in time. In fact, the number of iterations can be approximated as $\mu \approx 8.2L - 4.4$.

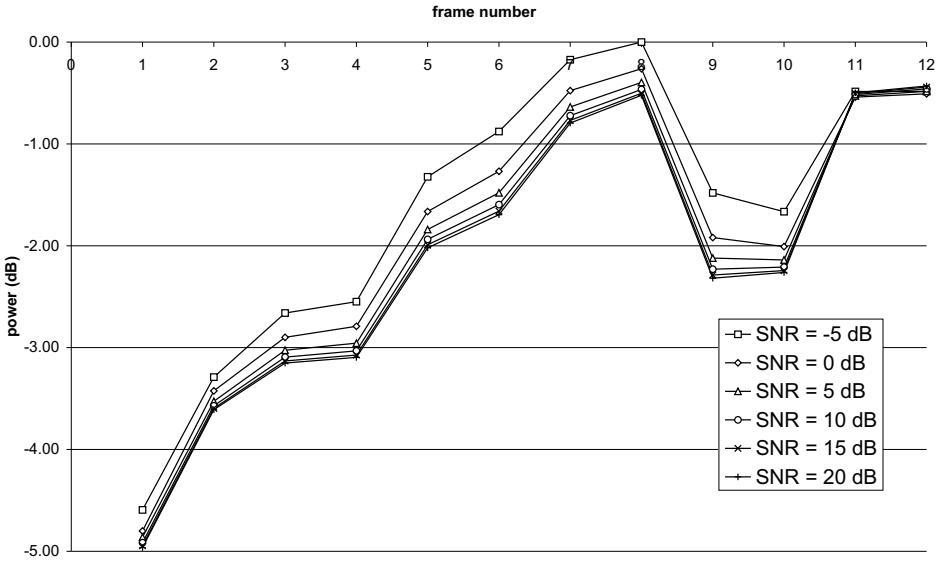

Figure 6.21: Case 3: Estimated power in path at 0 ns (0 dB) over 12 frames

6.7 Extension of the simulator

Now that the simulations with the current simulator have been concluded, it will be extended in order to answer two other questions: how sensitive are both algorithms to interference? Can the algorithms handle changes in the channel parameters?

6.7.1 Handover to a second basestation

First of all, a second basestation is added to the simulator. This way the situation shortly before a handover can be simulated: the mobile terminal is at the boundary of the first basestation's cell and detects a second base station with a signal strength equal or greater than that of the first basestation. The first basestation transmits trough a channel with paths at 521 ns (0 dB) and 1573 ns (-3 dB) and the interfering basestation's channel has paths at 0 ns (0 dB), 1060 ns (-3 dB) and 1573 ns (-3 dB).

algorithm	SIR	SNR	paths	path 1		path 2		iterations
MLE	0 dB	0 dB	1.83	92%	-5.24 dB	92%	-6.07 dB	12.2
PDP	0 dB	0 dB	1.92	92%	-3.68 dB	100%	-3.85 dB	-
MLE	-8 dB	0 dB	0.67	25%	0.00 dB	42%	-3.32 dB	12.3
PDP	-8 dB	0 dB	1.83	83%	-2.97 dB	100%	-2.59 dB	-

Table 6.5: Simulation conditions for interference

As can be seen from Table 6.5, the PDP algorithm outperforms the MLE algorithm in the presence of a second basestation (both at low and high signal to interference ratios). The algorithm detects more paths and estimates the delays correctly more often than the MLE algorithm.

6.7.2 Moving propagation conditions

The current simulator only supports channels of which the parameters are constant during the entire simulation run. By modifying the simulator dynamic channel parameters can be supported. The parameters can then be varied from one frame to the next. This way it is possible to investigate whether the algorithms can adapt to the changing channel parameters. Moving propagation conditions are simulated by using a channel with two paths at 0 ns of strength 0 dB. Each frame the second path is moved over 512 ns.

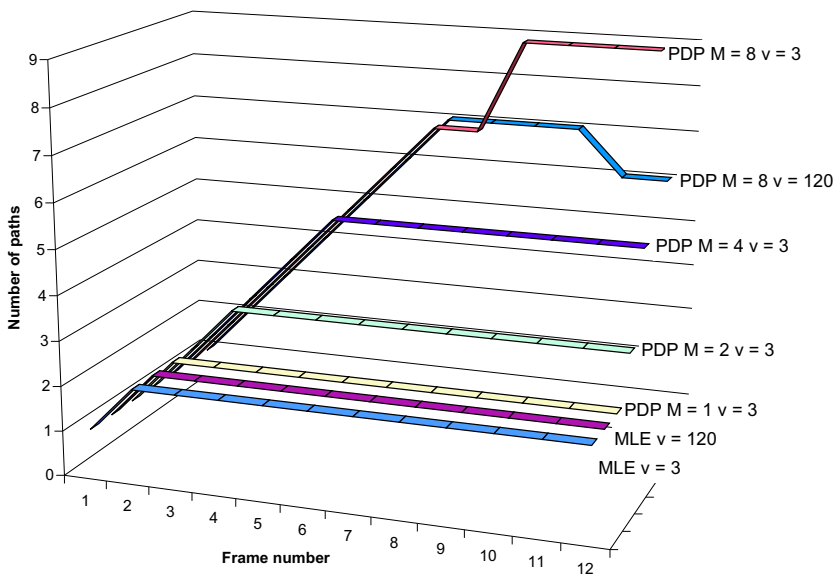

Figure 6.22: Moving propagation conditions

Figure 6.22 shows the effect of the averaging length M on the performance of the PDP algorithm. Only when no averaging occurs ($M = 1$) the PDP algorithm performs as well as the MLE algorithm. For $M = 2$ the PDP algorithm detects three paths, where only two paths exist. For larger values of M this behavior becomes even more problematic (five paths for $M = 4$ and up to nine paths are detected for $M = 8$). Hence, M should be kept as low as possible if channel parameters change rapidly. If M is too large, the number of paths that is found rises linearly with the frame number, up to frame $M + 1$.

6.7.3 Birth-death propagation conditions

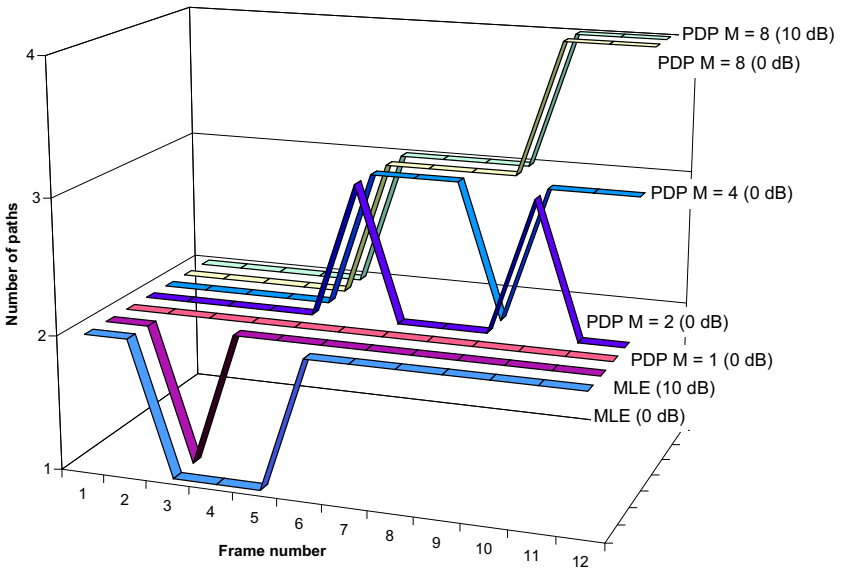

Figure 6.23: Birth-death propagation conditions

Another situation in which channel parameters change is in the case of birth-death propagation conditions. In this case paths suddenly appear or disappear. Simulations have been carried out with paths at 521 ns and 2050 ns (both 0 dB), the SNR is set to 10 dB. These channel parameters are used during the first four frames. Next, the path at 521 ns disappears and a new path appears at 1563 ns (0 dB). Again this situation holds for four frames, after which the path at 2050 ns disappears. At the same time a path appears at 1025 ns (0 dB).

Figure 6.23 shows that the MLE algorithm has difficulties finding the second path in the third frame (SNR = 0 and 10 dB) and in the fourth and fifth frame (SNR = 0 dB). The PDP algorithm on the other hand tends to find too many paths for higher values of M. For $M = 8$ (SNR is 0 dB and 10 dB) the algorithm finds a third path from the sixth frame on and a fourth path from the tenth frame on. A lower value of M results in the detection of less paths, for $M = 4$ a third path is found in frames 6, 7, 8, 10, 11 and 12. When M is set to two, a third path is found only in frames 6 and 10. This shows that the length M of the averaging process influences the detection of paths that have already disappeared. Without averaging ($M = 1$) the best results are obtained.

6.8 Computational power requirements

An important issue is the cost of the two algorithms in terms of computations. For this, the source code of both algorithms has been analyzed in order to determine the number of operations O they require for processing a single frame. Additions, subtractions, multiplications and divisions of variables are considered as operations. This has led to the following expressions:

$$O_{PDP} = OSF_{PDP}\,(4 + 6P + 2NP) \tag{6.2}$$

$$O_{MLE} = \mu OSF_{MLE}\,(N + 4L + PL + 4NL + 2NPL) \tag{6.3}$$

$$R = \frac{O_{MLE}}{O_{PDP}} = \mu \frac{OSF_{MLE}}{OSF_{PDP}} \frac{(N + 4L + PL + 4NL + 2NPL)}{(4 + 6P + 2NP)} \tag{6.4}$$

In these equations N and P are respectively the window size and length of the correlation, L is the number of searched paths and μ is the number of iterations required. From Figure 6.24 it can be concluded that the ratio R between the number of operations of the MLE and PDP algorithms increases linearly with the number of iterations μ. Figures 6.25 and 6.26 show that R is independent of both N and P and increases linearly with L:

$$R \approx \mu L \frac{OSF_{MLE}}{OSF_{PDP}} \qquad N, P \gg 1 \tag{6.5}$$

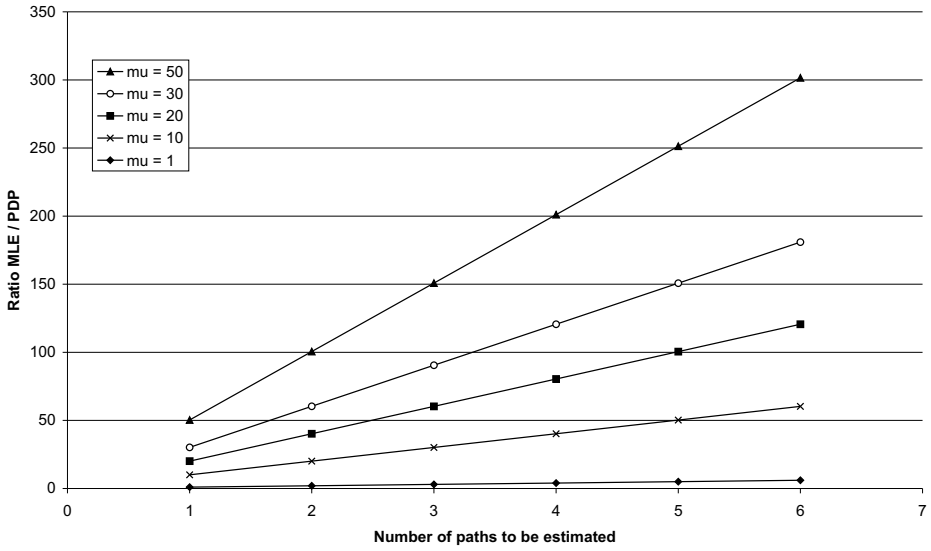

Figure 6.24: Ratio of no. of operations as function of μ ($N = 2560$, $P = 340$)

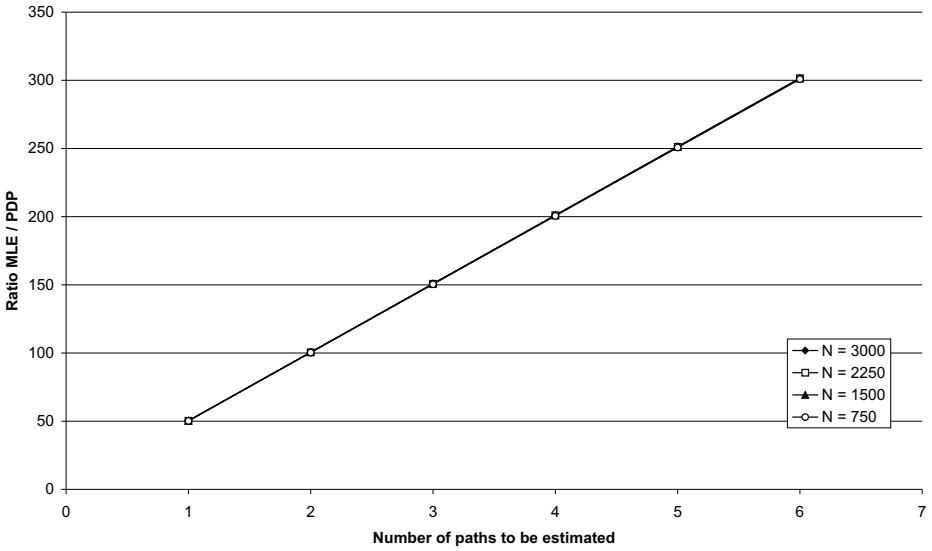

Figure 6.25: Ratio of no. of operations as function of N ($\mu = 50$, $P = 340$)

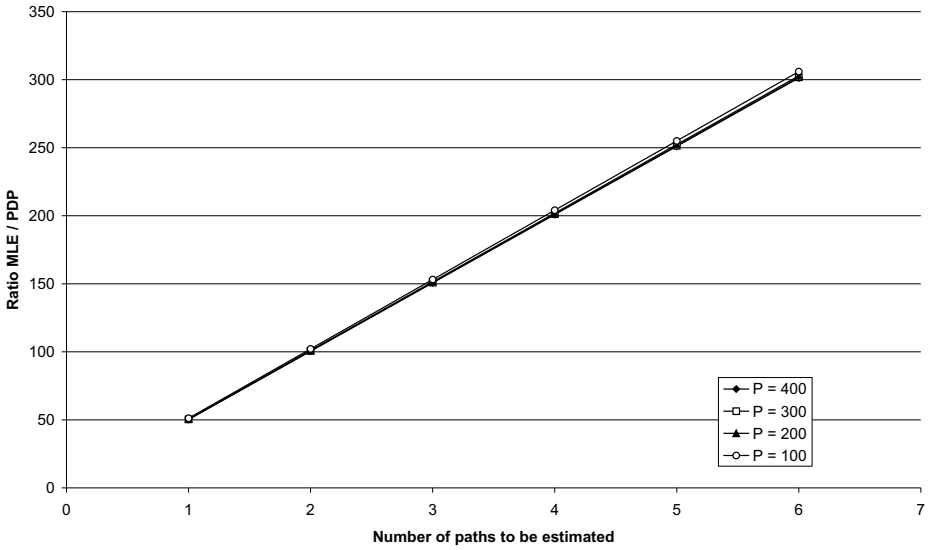

Figure 6.26: Ratio of no. of operations as function of P ($N = 2560$, $\mu = 50$)

7

Conclusions and future work

7.1 Summary of work

In this M.Sc. thesis it has been investigated to what extent it is useful for the path search function to switch between different algorithms, as conditions between transmitter and receiver change. First, a number of papers were discussed in a literature study to get a clear picture of different algorithms that were candidates for implementing the path search function (Chapter 2). These algorithms were compared with each other and have been classified in three classes, based on the similarities that were discovered.

Before implementing the algorithms and setting up simulations first a number of channel models were analyzed in Section 3.1. Also scenarios have been formulated in Section 3.2 that describe the conditions in the channel between the transmitting base station and the receiving mobile terminal. A number of modifications to the simulator have been discussed, as well as their implementation in C++ (Subsection 3.2.5).

Next the algorithms that were selected in the literature study have been implemented. The implementation of the Power Delay Profile (PDP)-based path searcher has been discussed in Chapter 4. In Chapter 5 the implementation of the Maximum Likelihood Estimation (MLE) path searcher has been presented. The theory behind the subspace-based algorithm has been discussed in Appendix B, but this algorithm has not been implemented.

The correctness of the functional behavior of both the PDP and MLE algorithm was verified by simulation as the algorithms were implemented. In order to investigate the performance of both algorithms in more detail, a number of simulations has been carried out (Chapter 6). The simulation results are discussed in Section 7.2, switching conditions are discussed in Section 7.3 and issues for further research are listed in Section 7.4.

7.2 Comparison of the PDP and MLE algorithms

The goal of the simulations of the PDP and MLE algorithms was to answer to following questions:

1. How do the algorithms perform at various levels of the SNR?
2. Which algorithm parameters influence the behavior the most?
3. Which algorithm can resolve closely spaced paths the best?
4. Do the algorithms have an acquiring and a tracking phase?
5. How sensitive are both algorithms to interference?
6. Can the algorithms handle changes in the channel parameters?

1. How do the algorithms perform at various levels of the SNR?

From the simulation results in Section 6.2 it can be concluded that the MLE algorithm detects one path more than the PDP algorithm for the Vehicular B and Pedestrian B channel models. In case of the Office B channel model the MLE algorithm even detects two paths more. This is due to the fact that that MLE algorithm can detect weak paths much better than the PDP algorithm, as can be seen clearly in Figures 6.3, 6.6, 6.9 and 6.12. Both algorithms benefit from high SNR values. The influence of Doppler effects is the largest on the PDP algorithm's performance (Section 6.3).

2. Which algorithm parameters influence the behavior the most?

From Section 6.4 it can be concluded that the PDP algorithm's performance strongly improves by increasing the averaging length. Weaker paths can then be detected better. The threshold value η has much less influence on the performance of the PDP algorithm.

The MLE algorithm is mainly sensitive to the number of paths L it has to detect and the maximum number of iterations μ_{max} it is allowed to use for this. Simulation results show that the number of iterations can be reduced by allowing the algorithm to skip searching the weakest path in the channel. In this case, the other paths can be found in only 10 iterations. In the case that all paths need to be detected it is useful to strongly decrease μ_{max}. This will lead to a minor loss in performance. Finally an alternative distribution of β_l was investigated which has not lead to an improvement of the algorithm.

3. Which algorithm can resolve closely spaced paths the best?

The ability of both algorithms to detect closely spaced paths has been discussed in Section 6.5. From the simulation results it can be concluded that the MLE algorithm can detect path delays that are a factor three smaller than the path delays that the PDP algorithm can detect for the same oversample factor. Also, the PDP algorithm can not detect weaker paths as the oversample factor is increased.

4. Do the algorithms have an acquiring and a tracking phase?

The behavior of both algorithms over time has been discussed in Section 6.6. The PDP algorithm shows an improvement of the path power estimates as more frames are processed. Depending on the averaging length M the convergence of the PDP algorithm is either fast or slow. The MLE algorithm on the other hand does not show this convergence in its power estimates. Also the number of iterations it requires does not become smaller as more frames are processed. Therefore it is not possible to identify an acquiring and tracking phase.

5. How sensitive are both algorithms to interference?

The influence of interference on the algorithms' performance has been investigated in Subsection 6.7.1. Simulations have been carried out with an interferer that transmits with a power equal to or larger than that of the desired base station, as is the case shortly before a handover. In both cases the PDP algorithm is less sensitive to the interference than the MLE algorithm.

6. Can the algorithms handle changes in the channel parameters?

In addition to the discussion in Section 6.6 on acquiring and tracking, simulations have been carried out for channels with path parameters that change from frame to frame. Changing path delays were investigated (Subsection 6.7.2), as well as the birth and death of paths (Subsection 6.7.3). In both cases the PDP's ability to track changing paths is hampered by the averaging length M. Without averaging the best results (close to those of the MLE algorithm) can be obtained.

Computational cost

In most cases the MLE algorithm outperforms the PDP algorithm. This is however at a certain cost, see Section 6.8. The MLE algorithm requires a factor $\mu L \frac{OSF_{MLE}}{OSF_{PDP}}$ more computations than the PDP algorithm. For equal oversample factors this means that if several paths need to be estimated, the MLE algorithm requires far more computations than the PDP algorithm.

By setting μ_{max} to a low value, the number of computations performed by the MLE algorithm can be kept relatively small. Also, when only a few paths need to be estimated, the influence of the factor L is relatively small. Under these circumstances the MLE algorithm requires the lowest amount of computations and will outperform the PDP algorithm, especially for channels with weak paths that need to be estimated. In the case of closely spaced paths the ratio $\frac{OSF_{MLE}}{OSF_{PDP}}$ can be as low as $\frac{1}{3}$, further reducing the number of computations for the MLE algorithm with respect to the PDP algorithm.

7.3 Switching conditions

From the discussion in Section 7.2 it becomes clear that the MLE algorithm outperforms the PDP algorithm in a number of situations. Also, the MLE algorithm will require more computations under all circumstances. In the case of a Vehicular B or Pedestrian B channel the PDP algorithm can not detect the weaker paths, while the MLE algorithm can. But the advantage of using the MLE algorithm is small, as these weak paths contain little energy and require a large number of iterations (Subsection 6.4). The PDP algorithm should therefore be selected.

In the case of strong paths that are closely spaced (e.g. Office B channel) the MLE algorithm is superior to the PDP algorithm. If only a few paths exist ($L = 2$) at 131 ns distance from each other, 13.1 iterations are required resulting in a ratio of operations $R = \mu L \frac{OSF_{MLE}}{OSF_{PDP}} = 8.73$. In this case the PDP algorithm detects almost 10 dB less power in the paths (Table 6.4) and the path detection percentages are lower. The improvement of the path estimation justifies the increase of the number of computations.

Also the ability of the MLE algorithm to estimate time-variant path delays is superior to that of the PDP algorithm (Section 6.7). Again, if only two or three strong paths need to be estimated, the MLE algorithm should be selected. In addition the influence of Doppler effects should be taken into account (Section 6.3). If strong Doppler effects occur, the MLE algorithm is superior and should therefore be selected.

Another set of circumstances that is worth further investigation is in the case of strong interference (Subsection 6.7.1). Sofar no explanation is available for the superior performance of the PDP algorithm at high SIR levels. Other issues qualified for further investigation are discussed in Section 7.4.

7.4 Recommendations

The trade-off between performance and number of computations should be investigated further for the cases described in Section 7.3. BER simulations are required to quantify the benefits of selecting the MLE algorithm instead of the PDP algorithm.

Some issues concerning the PDP algorithm are of interest for further research. First of all, the estimation of the noise variance σ_n^2 can be improved by using the method of [12]. This way a better calculation of the threshold is possible. Another option is to subtract the strongest paths from the power delay profile, before using it to set the threshold (as was proposed in [13] and [28]). Finally, the detection of closely spaced paths can be improved by performing the Teager-Kaiser operation after the correlation step in the PDP algorithm [17].

Some improvements to the MLE algorithm are worth further investigation as well. Should the MLE algorithm need to track fast changes of the delay profile, it is possible to split the received frames in blocks and perform the path search operation for each block [8].

Also another scheme is possible for β_l in which it is set to one. Currently, the value of β_l is set to $\frac{1}{L}$. The alternative scheme is called the SAGE algorithm [5]. The SAGE algorithm differs from the MLE-EM algorithm in that it does not re-estimate the entire value of $\hat{\boldsymbol{\theta}}$ during each iteration, but only a subset of α_l and τ_l values. Therefore, the computational complexity of an entire iteration cycle of the SAGE algorithm is equal to that of a single MLE-EM iteration step [5].

Finally, it would be useful to determine the convergence of the MLE algorithm per path. This way, path parameter estimates that have already converged in the first few iterations of the algorithm, do not need to be estimated again as the algorithm attempts to further improve the other parameter estimates. A reduction of the number of computations in subsequent iterations can be achieved this way.

A

List of acronyms

A - B

AE	Autocorrelation Estimation
AP	Alternating Projections
AWGN	Adaptive Wireless Network
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BLER	Block Error Rate
BPSK	Binary Phase Shift Keying

C

CADTES	Computer Architecture Design and Test for Embedded Systems
CCPCH	Common Control Physical Channel
CDMA	Code Division Multiple Access
CPICH	Common Pilot Channel
CRALS	Complex to Real Least Squares
CRLB	Cramér-Rao Lower Bound

D

DDSS	Double-Dwell Serial Search
DPCCH	Dedicated Physical Control Channel
DPDCH	Dedicated Physical Data Channel
DS-CDMA	Direct Sequence Code Division Multiple Access
DSSS	Direct Sequence Spread Spectrum

E - L

EEMCS	Electrical Engineering, Mathematics and Computer Science
EM	Expectation Maximization
FIR	Finite Impulse Response
FS	Frequency Smoothing
GAE	General Autocorrelation Estimation
ISI	Inter Symbol Interference
ITU-R	International Telecommunications Union - Radio communication
JED	Joint Estimation and Detection
LS	Least Squares

M

MAC	Medium Access Control
MAI	Multiple Access Interference
MEDLL	Multipath Estimating Delayed-Locked Loops
MF	Matched Filter
MIPS	Million Instructions Per Second
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimation
MMSE	Minimum Mean Squared Error
MSE	Mean Squared Error
MUSIC	Multiple Signal Classification

P

P-CCPCH	Primary Common Control Physical Channel
P-CPICH	Primary Common Pilot Channel
PDF	Probability Density Function
PDP	Power Delay Profile
PN	Pseudo Noise
POCS	Projection Onto Convex Sets
PS	Pulse Subtraction
PSD	Power Spectral Density

R

RFS	Random Frequency Smoothing
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristics
RWLS	Recursive Weighted Least Squares

S

SAS	Signals And Systems
SDP	Super Delay Profile
SIR	Signal to Interference Ratio
SNR	Signal to Noise Ratio
SPM	Super-resolution Pseudo-noise sequence correlation Method
STTD	Space-Time Transmit Diversity
SUMF	Single User Matched Filter

T - W

TDD	Time Division Duplex
TK	Teager-Kaiser
UMTS	Universal Mobile Telecommunications System
UTRA	UMTS Terrestrial Radio Access
W-CDMA	Wideband Code Division Multiple Access
WMSA	Weighted Multi-Slot Average
QPSK	Quadrature Phase Shift Keying

B

A Subspace-based path searcher

B.1 Introduction

As part of the literature study several papers have been discussed that analyze so-called *Subspace-based* path searchers. One of these algorithms [6] has been selected from the class of Subspace-based algorithms. This chapter presents the theory behind the selected Subspace-based path search algorithm.

For a better understanding of the algorithm a model for the received complex baseband signal will be discussed first (Section B.2). Then the principle of the Subspace algorithm is stated in Section B.3. In short, the baseband signal is correlated with the user signature and then an estimate of the covariance matrix of the correlated signal is calculated. By eigendecomposition of this covariance matrix, the eigenvectors of the signal subspace and the noise subspace can be found. In principle the signal subspace and the noise subspace are orthogonal and can therefore be separated from each other. This orthogonality is exploited by using a Toeplitz displacement.

In this way, the channel can be identified and the path delays can be determined. The algorithm has the advantage that it requires much less knowledge of the user signatures of all the users in a cell compared to the MLE method and it can achieve a higher path delay resolution [6].

B.2 Model of the baseband received signal

The received baseband signal $\mathbf{r}_k(n)$ for user k is of length $aN + M - 1$, with N the spreading gain, M the channel length and a the number of symbols. This signal can be modelled as:

$$\mathbf{r}_k(n) = \sum_{k=1}^{K} \alpha_k C_k H_k \mathbf{b}_k(n) + \mathbf{w}(n) \tag{B.1}$$

With $\mathbf{r}_k(n) = [r_k(n), \ldots, r_k(n + aN + M - 2)]^T$, α_k the complex amplitude and n the discrete time index. The additive, circularly symmetric Gaussian noise process with variance σ_w^2 is denoted as: $\mathbf{w}(n) = [w(n), \ldots, w(n + aN + M - 2)]^T$ and the $(a+2)\times 1$ -sized vector of data bits as: $\mathbf{b}(n) = \left[b\left\lfloor \frac{n}{N} \right\rfloor - 1, \ldots, b\left\lfloor \frac{n}{N} \right\rfloor + a\right]^T$. The channel matrix for user k is then:

$$H_k = \mathbf{h}_k \otimes I_{a+2} \tag{B.2}$$

In (B.2) the Kronecker operator is denoted as $\otimes$, $\mathbf{h}_k = [h_k(0), \ldots, h_k(M-1)]^T$ is the filtered and sampled complex channel impulse response and I_{a+2} is the $(a+2) \times (a+2)$ identity matrix.

Next, expressions for the spreading code matrix C_k in (B.1) are derived. First of all, a matrix of width M for vector $\mathbf{c}_k(n)$ is denoted as: $C(\mathbf{c}_k(n), M)$. This is the convolution matrix of the spreading code vector $\mathbf{c}_k(n) = [c_k(n), \ldots, c_k(n + N - 1)]^T$ of length N with another vector of length M. Therefor, $C(\mathbf{c}_k(n), M)$ is of size $(N + M - 1) \times M$.

$C^1_{k,M}(n)$ is then defined as the first N rows of $C(\mathbf{c}_k(n), M)$ and $C^2_{k,M}(n)$ as the last $M-1$ rows of $C(\mathbf{c}_k(n), M)$. This way, the spreading code matrix $C_k(n)$ for user k of size $(aN + M - 1) \times (a + 2)M$ can be denoted as:

$$C_k(n) = \begin{pmatrix} C^2_{k,M}(n-N) & C^1_{k,M}(n) & & & & \\ & C^2_{k,M}(n) & C^1_{k,M}(n+N) & & 0 & \\ & & C^2_{k,M}(n+N) & C^1_{k,M}(n+2N) & & \\ & & & \ddots & \ddots & \\ & 0 & & & C^2_{k,M}(n+aN) & \tilde{C}^1_{k,M}(n+(a+1)N) \end{pmatrix}$$

In the expression for $C_k(n)$ the matrix $\tilde{C}^1_{k,M}(n + (a + 1)N)$ consists of the first $M - 1$ rows of $C^1_{k,M}(n + (a + 1)N)$.

Instead of processing the received signal itself, it is filtered by a set of M matched filters per symbol, which leads to the observation vector $\mathbf{y}(n)$ of length aM:

$$\mathbf{y}(n) = S_1(n)\mathbf{r}(n) = S_1(n)\left(\sum_{k=1}^{K} \alpha_k C_k H_k \mathbf{b}_k(n)\right) + S_1 \mathbf{w}(n) \tag{B.3}$$

The matched filtering matrix S_1 can be formulated as follows:

$$S_1^T(n) = \begin{pmatrix} C_{1,M}^1(n) & & & & \\ C_{1,M}^2(n) & C_{1,M}^1(n+N) & & 0 & \\ & C_{1,M}^2(n+N) & C_{1,M}^1(n+2N) & & \\ & & \ddots & \ddots & \\ 0 & & & & C_{1,M}^1(n+aN) \\ & & & & C_{1,M}^2(n+aN) \end{pmatrix} \tag{B.4}$$

B.3 Principle of Subspace-based path searcher

Now that the matched filtered signal $\mathbf{y}(n)$ (see (B.3)) can be observed, a covariance matrix can be calculated as follows:

$$R_y = E\left[\mathbf{y}(n)\mathbf{y}(n)^H\right] = \sigma_1^2 S_1(n)C_1(n)H_1H_1^H C_1(n)^H S_1(n)^H + R_I(n) + R_w(n)$$

with $\sigma_1^2 = E\left\{\alpha_1^2 b_1(n)^2\right\}$, $R_w(n)$ as the noise autocorrelation matrix and $R_I(n)$ as the multiple access interference from the other users with $k \neq 1$.

Averaging $R_y(n)$ over time leads to:

$$\begin{aligned} \overline{R_y} &= \frac{1}{N_s}\sum_{n=1}^{N_s} R_y(n) && \text{(B.5)} \\ &= \frac{\sigma_1^2}{N_s}\sum_{n=1}^{N_s} S_1(n)C_1(n)H_1H_1^H C_1(n)^H S_1(n)^H + \overline{R}_I + \overline{R}_w && \text{(B.6)} \\ &= \overset{N,N_s\to\infty}{\approx} \sigma_1^2 \overline{SC}_1 H_1 H_1^H \overline{SC}_1^H + \sum_{k=2}^{K}\sigma_k^2 R(\mathbf{h}_k) + \sigma_w^2 I && \text{(B.7)} \end{aligned}$$

With $\overline{SC}_1 = \frac{1}{N_s}\sum_{n=1}^{N_s} S_1(n)C_1(n)$ and N_s as the number of samples used to estimate the covariance matrix.

Using a Toeplitz Displacement, the channel can be estimated by removing the contribution of R_w and R_I from R_y as follows:

$$\begin{aligned} R_h &= \overline{R}_y\left(2:aM, 2:aM\right) - \overline{R}_y\left(1:aM-1, 1:aM-1\right) && \text{(B.8)} \\ &= \overline{R}_y^+ - \overline{R}_y^- && \text{(B.9)} \\ &= \left(\overline{SC}_1 H_1\right)^+ \left(\overline{SC}_1 H_1\right)^{+H} - \left(\overline{SC}_1 H_1\right)^- \left(\overline{SC}_1 H_1\right)^{-H} && \text{(B.10)} \\ &= \overline{SC}_1^+ H_1 H_1^H \overline{SC}_1^{+H} - \overline{SC}_1^- H_1 H_1^H \overline{SC}_1^{-H} && \text{(B.11)} \end{aligned}$$

In (B.8) $\overline{SC}_1^- = \overline{SC}_1(2:M, 1:M)$ and $\overline{SC}_1^+ = \overline{SC}_1(1:M-1, 1:M)$.

The matrix R_h should now be a function of the channel of user 1:

$$R_h \approx \begin{pmatrix} \mathbf{h}_1^+\mathbf{h}_1^{+H} & & & 0 \\ & \mathbf{h}_1\mathbf{h}_1^H & & \\ & & \ddots & \\ 0 & & & \mathbf{h}_1\mathbf{h}_1^H \end{pmatrix} - \begin{pmatrix} \mathbf{h}_1\mathbf{h}_1^H & & & 0 \\ & \mathbf{h}_1\mathbf{h}_1^H & & \\ & & \ddots & \\ 0 & & & \mathbf{h}_1^-\mathbf{h}_1^{-H} \end{pmatrix}$$

Eigendecomposition of R_h gives:

$$R_h = \sum_{i=1}^{aM-1} \lambda_i \mathbf{v}_i \mathbf{v}_i^H \tag{B.12}$$

Finally the channel can be estimated by solving the following subspace fitting problem:

$$\hat{\mathbf{h}}_1 = \arg\min_{\mathbf{h},W} \|H - VW\|_F^2 \tag{B.13}$$

With $H = \left[\left(\overline{SC}_1 H_1\right)^+, \left(\overline{SC}_1 H_1\right)^-\right]$ and $V = [\mathbf{v}_1, \ldots, \mathbf{v}_{a+1}, \mathbf{v}_{aM-a-1}, \ldots, \mathbf{v}_{aM-1}]$ containing all the eigenvectors of R_h.

C

A Power Delay Profile-based path searcher in C++

C.1 path searcher TOP h

```
#ifndef __Path_Searcher_TOP_h
#define __Path_Searcher_TOP_h
#include "itcomm.h"

//! Abstract base class for Path_Searcher_TOP classes.
class Path_Searcher_TOP {
public:
    Path_Searcher_TOP() { };

    //! Enable or disable messages.
    virtual void set_verbose(bool in_verbose) = 0;
    //! Returns message status.
    virtual bool get_verbose() = 0;

    //! Enable logging and set log file name.
    virtual void set_status_filename(char *in_status_filename) = 0;
    //! Returns logging status.
    virtual bool get_save_status() = 0;
    //! Reset the frame counter that is used by logging.
    virtual void reset_frame_counter() = 0;

    //! Set the oversample factor.
    virtual void set_oversample_factor(int in_oversample_factor) = 0;
    //! Returns the oversample factor.
    virtual int get_oversample_factor() = 0;
    //! Set the number of frames that the path searcher averages
    //! to estimate a multipath profile.
    virtual void set_no_averaged_frames(int in_auto_corr_hist_length) = 0;
    //! Returns the number of frames that the path searcher averages
    //! to estimate a multipath profile.
    virtual int get_no_averaged_frames() = 0;
    //! Set the length of the autocorrelation that is used.
    virtual void set_auto_corr_length(int in_auto_corr_length) = 0;
    //! Returns the length of the autocorrelation that is used.
    virtual int get_auto_corr_length() = 0;
    //! Set the length of the autocorrelation window.
    virtual void set_auto_corr_window(int in_auto_corr_window) = 0;
```

```
    //! Returns the length of the autocorrelation window.
    virtual int get_auto_corr_window() = 0;

    //! Give the path searcher the conjugate of the primary scramble code
    //! that is used to scramble the received signal.
    virtual void set_conj_scrmbl_code(cvec &in_conj_scrmbl_code) = 0;
    //! Returns the found delay profile.
    virtual ivec &get_del_vec() = 0;
    //! Returns the found power profile.
    virtual vec &get_pow_vec() = 0;
    //! Returns the found number of paths.
    virtual int get_num_paths() = 0;

    //! Perform path search operation.
    virtual void operator()(const cvec &in_chips) = 0;
};
#endif
```

C.2 downlink receiver h

```
#ifndef __downlink_receiver_h
#define __downlink_receiver_h

#include "itcomm.h"
#include "cell_searcher.h"
#include "channel_estimator.h"
#include "my_pulse_shape.h"
#include "path_searcher_TOP.h"
#include "path_searcher.h"
#include "pdp_path_searcher.h"
#include "wcdma_modulator.h"

//! Downlink receiver class.
class Downlink_Receiver {
public:
    Downlink_Receiver(void);
    ~Downlink_Receiver(void);

    //! Enable logging and set log filename.
    void set_status_filename(char *in_status_filename);
    //! Returns logging status.
    bool get_save_status() { return save_status; };
    //! Reset the frame counter that is used by logging.
    void reset_frame_counter() { frame_counter = 1; };

    //! Enable or disable messages.
    void set_verbose(bool in_verbose);
    //! Returns message status.
    bool get_verbose() { return verbose; };

    //! Set the oversample factor.
    void set_oversample_factor(int in_oversample_factor);
    //! Returns the oversample factor.
    int get_oversample_factor(void) { return oversample_factor; };
    //! Enable or disable pulse shaping.
    void set_pulse_shape(bool in_pulse_shape) { pulse_shape = in_pulse_shape; };
    //! Get pulse shaping status.
    bool get_pulse_shape(void) { return pulse_shape; };
    //! Set the length (number of taps) of the pulse shape filter.
    void set_pulse_shape_filter_length(int in_pulse_shape_filter_length);
    //! Returns the length (number of taps) of the pulse shape filter.
    int get_pulse_shape_filter_length(void) { return pulse_shape_filter_length; };

    //! Get cell search status.
    bool get_search_cell() { return search_cell; };
    //! Enable or disable cell searcher messages.
    void set_cell_search_verbose(bool in_verbose) { cell_searcher.set_verbose(in_verbose); };
    //! Returns cell searcher message status.
    bool get_cell_search_verbose() { return cell_searcher.get_verbose(); };
    //! Enable cell searcher logging and set cell searcher log file name.
    void set_cell_search_status_filename(char *in_status_filename)
    { cell_searcher.set_status_filename(in_status_filename); };
    //! Set the number of slots the cell searcher uses to identify
    //! the primary scrambling code of the cell.
    void set_num_cell_search_slots(int in_num_search_slots)
    { cell_searcher.set_num_search_slots(in_num_search_slots); };
    //! Returns the number of slots the cell searcher uses to identify
    //! the primary scrambling code of the cell.
    int get_num_cell_search_slots() { return cell_searcher.get_num_search_slots(); };
    //! Set the threshold for the number of votes a scrambling code should receive
```

```
//! before it is identified as the primary scrambling code of the cell.
void set_cell_search_votes_treshold(int in_votes_treshold)
{ cell_searcher.set_votes_treshold(in_votes_treshold); };
//! Returns the threshold for the number of votes a scrambling code should receive
//! before it is identified as the primary scrambling code of the cell.
int get_cell_search_votes_treshold() { return cell_searcher.get_votes_treshold(); };
//! Set the primary scramble code and disable cell search.
void set_p_scrmbl_code(int in_scrmbl_code_group, int in_p_scrmbl_code_index)
{ cell_searcher.set_p_scrmbl_code(in_scrmbl_code_group, in_p_scrmbl_code_index);
search_cell = false; };
//! Returns the complex conjugate of the primary scramble code that the cell searcher found.
cvec &get_conj_p_scrmbl_code() { return cell_searcher.get_conj_p_scrmbl_code(); };
//! Returns the code group of the primary scramble code that the cell searcher found.
int get_scrmbl_code_group() { return cell_searcher.get_scrmbl_code_group(); };
//! Returns the index of the primary scramble code that the cell searcher found.
int get_p_scrmbl_code_index() { return cell_searcher.get_p_scrmbl_code_index(); };
//! Give the receiver the delay profile of the channel so no path search is required.
void set_del_vec(ivec in_delay_vec) { del_vec = in_delay_vec; search_paths = false; };
//! Returns the delay profile that is known to the receiver.
ivec &get_del_vec() { return del_vec; };
//! Get path search status.
bool get_search_paths() { return search_paths; };

//! Enable or disable path searcher messages.
void set_path_search_verbose(bool in_verbose) { path_searcher -> set_verbose(in_verbose); };

//! Returns path searcher message status.
bool get_path_search_verbose() { return path_searcher -> get_verbose(); };

//! Enable path searcher logging and set path searcher log file name.
void set_path_search_status_filename(char *in_status_filename)
{ path_searcher -> set_status_filename(in_status_filename); };

//! Set the number of frames that the path searcher averages to estimate a multipath profile.
void set_path_search_no_averaged_frames(int in_no_averaged_frames)
{ path_searcher -> set_no_averaged_frames(in_no_averaged_frames); };

//! Returns the number of frames that the path searcher averages to estimate a multipath profile.
int get_path_search_no_averaged_frames() { return path_searcher -> get_no_averaged_frames(); };

//! Set the maximum delay in samples that the path searcher should be able to resolve.
void set_path_search_max_delay(int in_auto_corr_length)
{ path_searcher -> set_auto_corr_length(in_auto_corr_length); };

//! Returns the maximum delay in samples that the path searcher is able to resolve.
int get_path_search_max_delay() { return path_searcher -> get_auto_corr_length(); };

//! Set the length of the autocorrelation window that the path searcher uses.
void set_path_search_auto_corr_window(int in_auto_corr_window)
{ path_searcher -> set_auto_corr_window(in_auto_corr_window); };

//! Returns the length of the autocorrelation window that the path searcher uses.
int get_path_search_auto_corr_window() { return path_searcher -> get_auto_corr_window(); };

//! Returns the delay profile that the path searcher found.
ivec &get_path_search_del_vec() { return path_searcher -> get_del_vec(); };

//! Returns the power profile that the path searcher found.
vec &get_path_search_pow_vec() { return path_searcher -> get_pow_vec(); };

//! Return the number of multipaths that the path searcher found.
int get_path_search_num_paths() { return path_searcher -> get_num_paths(); };
```

```
    //! Set the number of rake fingers of the receiver.
    void set_num_rake_fingers(int in_num_rake_fingers);
    //! Returns the number of rake fingers of the receiver.
    int get_num_rake_fingers(void) { return num_rake_fingers; };

    //! Enable or disable channel estimation.
    void set_estimate_channel(bool in_estimate_channel)
    { estimate_channel = in_estimate_channel; };
    //! Returns channel estimation enabled status.
    bool get_estimate_channel() { return estimate_channel; };
    //! Enable channel estimator logging and set channel estimator log file name.
    void set_channel_estimator_status_filename(char *in_status_filename);
    //! Set the length of the channel estimator filter.
    void set_channel_estimator_filter_length(int filter_length);
    //! Returns the length of the channel estimator filter.
    int get_channel_estimator_filter_length();

    //! Give the receiver the list of UMTS channels that are present in the received signal.
    void set_ch_settings(int in_num_ch, Downlink_Ch_Setting in_ch_settings[]);

    //! Perform receive operation.
    void operator()(const cvec &received_chips, bvec received_data_bits[],
    const double transmitter_power = 0.0, const cmat &channel_realization = "0");

private:
    bool                save_status;
    char                *status_filename, variable_name[256];
    it_file             status_file;
    long                frame_counter;
    bool                verbose, pulse_shape, search_cell, search_paths, estimate_channel;
    int                 oversample_factor, pulse_shape_filter_length, num_rake_fingers,
                        num_allocated_rake_fingers, num_ch, pcpich,
                        channel_estimator_filter_length;
    Downlink_Ch_Setting *ch_settings;
    smat                ch_codes;
    ivec                del_vec;
    cvec                oversampled_chips, downsampled_chips, conj_scrmbl_code,
                        descrambled_chips, symbols, channel_estimates, mrc_symbols;
    bvec                received_bits;


    My_Root_Raised_Cosine<double_complex>   pulse_shape_filter;
    Cell_Searcher                           cell_searcher;
    Path_Searcher_TOP                       *path_searcher;
    Channel_Estimator                       *channel_estimators;
    WCDMA_Spread_2d                         dpch_despreader;
    WCDMA_QPSK_Modulator                    qpsk_demodulator;
    WCDMA_QAM16_Modulator                   qam16_demodulator;

};

#endif // __downlink_receiver_h
```

C.3 downlink receiver cpp

```
#include "downlink_receiver.h"
#include "downlink_scrambling_code_generator.h"
#include "my_vec_func.h"
#include "wcdma_constants.h"

Downlink_Receiver::Downlink_Receiver(void)
{
    save_status = false;
    status_filename = NULL;
    reset_frame_counter();
    verbose = false;
    pulse_shape = true; search_cell = true; search_paths = true; estimate_channel = true;
    channel_estimators = NULL;
    oversample_factor = 4;
    set_pulse_shape_filter_length(10);
    set_num_rake_fingers(4);
    set_channel_estimator_filter_length(16);
    path_searcher = NULL;

    if (search_paths) {
        switch( 'PDP' )
        {
            case 'JOR': path_searcher = new     Path_Searcher; break;
            case 'PDP': path_searcher = new PDP_path_searcher; break;
            //case 'MLE':   path_searcher = new MLE_path_searcher; break;
            //case 'SUB':   path_searcher = new SUB_path_searcher; break;
        }
    }
}

Downlink_Receiver::~Downlink_Receiver(void)
{
    if (channel_estimators != NULL) {
        delete [] channel_estimators;
    }

    if (path_searcher != NULL) {
        delete path_searcher;
    }
}

void Downlink_Receiver::set_status_filename(char *in_status_filename)
{
    if (in_status_filename == NULL) {
        save_status = false;
    } else {
        status_filename = in_status_filename;
        save_status = true;
    }
}

void Downlink_Receiver::set_verbose(bool in_verbose)
{
    verbose = in_verbose;
    set_cell_search_verbose(verbose);
    set_path_search_verbose(verbose);
}

void Downlink_Receiver::set_oversample_factor(int in_oversample_factor)
{
```

```
    oversample_factor = in_oversample_factor;
    pulse_shape_filter.set_pulse_shape(PULSE_SHAPE_FILTER_ROLLOFF,
                        pulse_shape_filter_length, oversample_factor);
    path_searcher -> set_oversample_factor(oversample_factor);
}

void Downlink_Receiver::set_pulse_shape_filter_length(int in_pulse_shape_filter_length)
{
    pulse_shape_filter_length = in_pulse_shape_filter_length;
    pulse_shape_filter.set_pulse_shape(PULSE_SHAPE_FILTER_ROLLOFF,
                        pulse_shape_filter_length, oversample_factor);
}

void Downlink_Receiver::set_num_rake_fingers(int in_num_rake_fingers)
{
    if (channel_estimators != NULL) {
        delete [] channel_estimators;
        num_rake_fingers = 0;
        num_allocated_rake_fingers = 0;
        channel_estimators = NULL;
    }
    channel_estimators = new Channel_Estimator[in_num_rake_fingers];
    if (channel_estimators != NULL) {
        num_rake_fingers = in_num_rake_fingers;
        num_allocated_rake_fingers = 0;
        for (int i = 0; i < num_rake_fingers; i++) {
            channel_estimators[i].set_finger_number(i);
        }
    }
}

void Downlink_Receiver::set_channel_estimator_status_filename(char *in_status_filename)
{
    for (int i = 0; i < num_rake_fingers; i++) {
        channel_estimators[i].set_status_filename(in_status_filename);
    }
}

void Downlink_Receiver::set_channel_estimator_filter_length(int filter_length)
{
    channel_estimator_filter_length = filter_length;
    for (int i = 0; i < num_rake_fingers; i++) {
        channel_estimators[i].set_ma_filter_length(filter_length);
    }
}

int Downlink_Receiver::get_channel_estimator_filter_length()
{
    if (num_rake_fingers > 0) {
        return channel_estimators[0].get_ma_filter_length();
    } else {
        return 0;
    }
}

void Downlink_Receiver::set_ch_settings(int in_num_ch, Downlink_Ch_Setting in_ch_settings[])
{
    num_ch = in_num_ch; ch_settings = in_ch_settings;

    pcpich = -1;
    for (int ch = 0; ch < num_ch; ch++) {
        if (ch_settings[ch].ch_type == P_CPICH) {
```

```
            pcpich = ch;
        }
    }
};

void Downlink_Receiver::operator()(const cvec &received_chips, bvec received_data_bits[],
                            const double transmitter_power, const cmat &channel_realization)
{
    int     slot_format, sf, bits_per_slot, ch, finger, slot, data_bits_per_slot, received_bits_index;
    //cvec  scaled_symbols;
    double  p_cpich_power, channel_power;

    oversampled_chips.set_length(received_chips.length());
    downsampled_chips.set_length(received_chips.length() / oversample_factor);
    downsampled_chips.zeros();
    conj_scrmbl_code.set_length(CHIPS_PER_FRAME + channel_estimator_filter_length / 2 * P_CPICH_SF);
    descrambled_chips.set_length(CHIPS_PER_FRAME + channel_estimator_filter_length / 2 * P_CPICH_SF);

    if (pulse_shape) {
        pulse_shape_filter.shape_samples(received_chips, oversampled_chips);
    } else {
        oversampled_chips = received_chips;
    }
    if (search_cell) {
        //skip(oversampled_chips, oversample_factor, downsampled_chips);
        downsample_average(oversampled_chips, oversample_factor, downsampled_chips);
    }

    if (save_status) {
        status_file.open(status_filename, false);
        sprintf(variable_name, "frame%li_downsampled_chips", frame_counter);
        status_file << Name(variable_name) << downsampled_chips;
        status_file.close();
    }

    if (search_cell) {
        cell_searcher(downsampled_chips);
    }
    if (cell_searcher.cell_found) {
        if (cell_searcher.cell_changed) {
            path_searcher -> set_conj_scrmbl_code(cell_searcher.conj_p_scrmbl_code);
            conj_scrmbl_code = concat(cell_searcher.conj_p_scrmbl_code,
                    cell_searcher.conj_p_scrmbl_code.mid(0, channel_estimator_filter_length /
                    2 * P_CPICH_SF));
            cell_searcher.cell_changed = false;
        }
        if (search_paths) {
            path_searcher -> operator()(oversampled_chips);
            num_allocated_rake_fingers = min(path_searcher -> get_num_paths(), num_rake_fingers);
        } else {
            num_allocated_rake_fingers = min(del_vec.length(), num_rake_fingers);
        }
        for (ch = 0; ch < num_ch; ch++) {
            slot_format = ch_settings[ch].slot_format;
            if (ch_settings[ch].ch_type == DPCH) {
                sf = DPCH_Formats[slot_format].sf;
                bits_per_slot = DPCH_Formats[slot_format].bits_per_slot;
            } else if (ch_settings[ch].ch_type == HS_PDSCH) {
                sf = HS_PDSCH_Formats[slot_format].sf;
                bits_per_slot = HS_PDSCH_Formats[slot_format].bits_per_slot;
            }
            if ((ch_settings[ch].ch_type == DPCH) || (ch_settings[ch].ch_type == HS_PDSCH)) {
```

```
dpch_despreader.set_code(to_vec(wcdma_spreading_codes(sf).get_row(
    ch_settings[ch].ch_code)),
    to_vec(wcdma_spreading_codes(sf).get_row(ch_settings[ch].ch_code)));
symbols.set_length(CHIPS_PER_FRAME / sf);
channel_estimates.set_length(CHIPS_PER_FRAME / sf);
//scaled_symbols.set_length(CHIPS_PER_FRAME / sf);
mrc_symbols.set_length(CHIPS_PER_FRAME / sf);
mrc_symbols.zeros();
p_cpich_power = 0.0;
channel_power = 0.0;
received_bits.set_length(SLOTS_PER_FRAME * bits_per_slot);
for (finger = 0; finger < num_allocated_rake_fingers; finger++) {
    if (search_paths) {
        downsample_average_mid(oversampled_chips, path_searcher ->
            get_del_vec()[finger], -1, oversample_factor, downsampled_chips);
    } else {
        if (pulse_shape) {
            downsample_average_mid(oversampled_chips,
            del_vec(finger) + pulse_shape_filter_length * oversample_factor, -1,
            oversample_factor, downsampled_chips);
        } else {
            downsample_average_mid(oversampled_chips,
            del_vec(finger) + pulse_shape_filter_length * oversample_factor, -1,
            oversample_factor, downsampled_chips);
        }
    }

    mid_elem_mult(downsampled_chips, 0,
        CHIPS_PER_FRAME + channel_estimator_filter_length / 2 * P_CPICH_SF,
        conj_scrmbl_code, descrambled_chips);
    if (estimate_channel) {
        channel_estimators[finger].set_target_sf(sf);
        channel_estimators[finger](descrambled_chips);
    }

    dpch_despreader.despread_mid(descrambled_chips, 0, CHIPS_PER_FRAME, symbols, 0);

    if (save_status) {
        status_file.open(status_filename, false);
        sprintf(variable_name, "frame%li_ch_%i_finger_%i_ds_chips",
            frame_counter, ch, finger);
        status_file << Name(variable_name) << downsampled_chips;
        sprintf(variable_name, "frame%li_ch_%i_finger_%i_chips",
            frame_counter, ch, finger);
        status_file << Name(variable_name) << descrambled_chips;
        sprintf(variable_name, "frame%li_ch_%i_finger_%i_symbols",
            frame_counter, ch, finger);
        status_file << Name(variable_name) << symbols;
        status_file.close();
    }

    if (estimate_channel) {
        elem_mult_add(symbols, channel_estimators[finger].channel_estimates,
            mrc_symbols);
        p_cpich_power +=
            mean(sqr(abs(channel_estimators[finger].channel_estimates)));
    } else {
        downsample_average_mid(channel_realization.get_row(finger), 0,
            symbols.length() * oversample_factor * sf, oversample_factor * sf,
            channel_estimates);
            // TODO: Check if this works correctly.
        elem_mult_add(symbols, conj(channel_estimates), mrc_symbols);
```

```
                    channel_power += mean(sqr(abs(channel_estimates)));
                }
                if (save_status) {
                    status_file.open(status_filename, false);
                    sprintf(variable_name, "frame%li_ch_%i_finger_%i_channel_estimates",
                        frame_counter, ch, finger);
                    status_file << Name(variable_name) << channel_estimates;
                    //sprintf(variable_name, "frame%li_ch_%i_finger_%i_scaled_symbols",
                        frame_counter, ch, finger);
                    //status_file << Name(variable_name) << scaled_symbols;
                    status_file.close();
                }
            }
            if (estimate_channel) {
                mrc_symbols /= inv_dB(ch_settings[ch].level_setting -
                    ch_settings[pcpich].level_setting) *
                    ((double)sf) / ((double)P_CPICH_SF) * p_cpich_power;
            } else {
                // /= 2 * (double)sf because spreading and despreading multiplies with the SF
                mrc_symbols /= inv_dB(ch_settings[ch].level_setting) *
                    transmitter_power * 2.0 * (double)sf * channel_power;
            }
            if (save_status) {
                status_file.open(status_filename, false);
                sprintf(variable_name, "frame%li_ch_%i_mrc_symbols", frame_counter, ch);
                status_file << Name(variable_name) << mrc_symbols;
                status_file.close();
            }
        }
        if (ch_settings[ch].ch_type == DPCH) {
            qpsk_demodulator.demodulate_bits(mrc_symbols, received_bits);
            data_bits_per_slot = DPCH_Formats[slot_format].n_data1 +
                DPCH_Formats[slot_format].n_data2;
            for (slot = 0; slot < SLOTS_PER_FRAME; slot++) {
                received_bits_index = slot * DPCH_Formats[slot_format].bits_per_slot;
                replace_mid_mid(received_bits, received_bits_index,
                    DPCH_Formats[slot_format].n_data1,
                    received_data_bits[ch], slot * data_bits_per_slot);
                received_bits_index +=
                    DPCH_Formats[slot_format].n_data1 + DPCH_Formats[slot_format].n_tpc +
                    DPCH_Formats[slot_format].n_tfci;
                replace_mid_mid(received_bits, received_bits_index,
                    DPCH_Formats[slot_format].n_data2,
                    received_data_bits[ch], slot * data_bits_per_slot +
                    DPCH_Formats[slot_format].n_data1);
                }
            } else if (ch_settings[ch].ch_type == HS_PDSCH) {
                if (slot_format == HS_PDSCH_SLOT_FORMAT_0) {
                    qpsk_demodulator.demodulate_bits(mrc_symbols, received_data_bits[ch]);
                } else if (slot_format == HS_PDSCH_SLOT_FORMAT_1) {
                    qam16_demodulator.demodulate_bits(mrc_symbols, received_data_bits[ch]);
                }
            }
        }
    }
}
```

C.4 pdp path searcher h

```
#ifndef __pdp_path_searcher_h
#define __pdp_path_searcher_h
#include "itcomm.h"
#include "path_searcher_TOP.h"

// Path searcher class
class PDP_path_searcher : public Path_Searcher_TOP {
public:
    PDP_path_searcher(void);

    //! Enable or disable messages.
    void set_verbose(bool in_verbose) { verbose = in_verbose; };
    //! Returns message status.
    bool get_verbose() { return verbose; };

    //! Enable logging and set log file name.
    void set_status_filename(char *in_status_filename);
    //! Returns logging status.
    bool get_save_status() { return save_status; };
    //! Reset the frame counter that is used by logging.
    void reset_frame_counter() { frame_counter = 1; };

    //! Set the oversample factor.
    void set_oversample_factor(int in_oversample_factor)
    {oversample_factor = in_oversample_factor; };
    //! Returns the oversample factor.
    int get_oversample_factor() { return oversample_factor; };
    //! Set the number of frames that the path searcher averages
    //! to estimate a multipath profile.
    void set_no_averaged_frames(int in_auto_corr_hist_length);
    //! Returns the number of frames that the path searcher averages
    //! to estimate a multipath profile.
    int get_no_averaged_frames() { return averaging_length; };
    //! Set the length of the autocorrelation that is used.
    void set_auto_corr_length(int in_auto_corr_length);
    //! Returns the length of the autocorrelation that is used.
    int get_auto_corr_length() { return autocorr_length; };
    //! Set the length of the autocorrelation window.
    void set_auto_corr_window(int in_auto_corr_window)
    { autocorr_window_size = in_auto_corr_window; };
    //! Returns the length of the autocorrelation window.
    int get_auto_corr_window() { return autocorr_window_size; };

    //! Give the path searcher the conjugate of the primary scramble code
    //! that is used to scramble the received signal.
    void set_conj_scrmbl_code(cvec &in_conj_scrmbl_code);
    //! Returns the found delay profile.
    ivec &get_del_vec() { return path_delay; };
    //! Returns the found power profile.
    vec &get_pow_vec() { return path_gain; };
    //! Returns the found number of paths.
    int get_num_paths() { return NOF_paths; };

    //! Perform path search operation.
    void operator()(const cvec &in_chips);

private:
    int     autocorr_length, autocorr_window_size, averaging_length, autocorr_new, autocorr_old;
    int     NOF_paths, frame_counter;
    cvec    received_signal, conjugate_signal;
```

```
    vec     pdp;
    cmat    autocorr_signal;
    ivec    path_delay;
    vec     path_gain;
    ivec    peak_delay;
    vec     peak_gain;
    int     oversample_factor;
    bool    verbose, save_status;
    char    *status_filename, variable_name[256];
    it_file status_file;
};
#endif
```

C.5 pdp path searcher cpp

```
#include "pdp_path_searcher.h"

/*****************************************************************/
/***                                                           ***/
/***                   Initialize path searcher                ***/
/***                                                           ***/
/*****************************************************************/

PDP_path_searcher::PDP_path_searcher(void)
{
    // Initialize
    set_oversample_factor(4);
    save_status = false;
    status_filename = NULL;
    frame_counter = 1;
    autocorr_window_size = 2560;
    autocorr_length = 340;

    // Initialize PDP calculator
    averaging_length = 8;
    pdp.set_length(autocorr_length);
    pdp.zeros();

    // Initialize correlator
    autocorr_new = 0;
    autocorr_old = 1;
    autocorr_signal.set_size(averaging_length + 1, autocorr_length);
    autocorr_signal.zeros();

    // Initialize peak selector
    peak_gain.set_length(autocorr_length);
    peak_delay.set_length(autocorr_length);

    // Initialize path selector
    path_gain.set_length(autocorr_length);
    path_delay.set_length(autocorr_length);
    NOF_paths = 0;
}

void PDP_path_searcher::set_conj_scrmbl_code(cvec &in_conj_scrmbl_code)
{
        conjugate_signal = repeat(in_conj_scrmbl_code.left(
        autocorr_window_size / oversample_factor), oversample_factor);
}

void PDP_path_searcher::set_status_filename(char *in_status_filename)
{
    if (in_status_filename == NULL) {
        save_status = false;
    }
    else {
        status_filename = in_status_filename;
        save_status = true;
    }
}

void PDP_path_searcher::set_no_averaged_frames(int in_auto_corr_hist_length)
{/*
    averaging_length = in_auto_corr_hist_length;
    autocorr_signal.set_size(averaging_length + 1, autocorr_length);
```

```
    autocorr_signal.zeros();
*/}

void PDP_path_searcher::set_auto_corr_length(int in_auto_corr_length)
{/*
    autocorr_length = in_auto_corr_length;
    autocorr_signal.set_size(averaging_length + 1, autocorr_length);
    autocorr_signal.zeros();
    pdp.set_size(autocorr_length);
    pdp.zeros();
    peak_delay.set_length(autocorr_length);
    peak_gain.set_length(autocorr_length);
*/}

void PDP_path_searcher::operator()(const cvec &received_signal)
{

    /*****************************************************************/
    /***                                                           ***/
    /***                Correlate the received signal              ***/
    /***                                                           ***/
    /*****************************************************************/

    int i, j, k, l, m, NOF_peaks;
    cvec zero_padded_signal, autocorr_signal_temp;
    double threshold;

    zero_padded_signal.set_length(autocorr_window_size + autocorr_length);
    zero_padded_signal.zeros();
    autocorr_signal_temp.set_length(autocorr_window_size);
    autocorr_signal_temp.zeros();

    peak_gain.zeros();
    peak_delay.zeros();
    path_gain.zeros();
    path_delay.zeros();

    for (k = 0; k < autocorr_window_size; k++) {
        zero_padded_signal(k) = received_signal(k);
    }

    for (j = 0; j < autocorr_length; j++) {
        for (i = 0; i < autocorr_window_size; i++)  {
            autocorr_signal_temp(i) = zero_padded_signal(i+j) * conjugate_signal(i);
        }
        autocorr_signal(autocorr_new, j) = 1.0 / autocorr_window_size * sum(autocorr_signal_temp);
    }

    /*****************************************************************/
    /***                                                           ***/
    /***                    Calculate the PDP                      ***/
    /***                                                           ***/
    /*****************************************************************/

    for (m = 0; m < autocorr_length; m++) {

        pdp(m) -= (abs(autocorr_signal(autocorr_old,m))) *
        (abs(autocorr_signal(autocorr_old,m))) / averaging_length;

        pdp(m) += (abs(autocorr_signal(autocorr_new,m))) *
        (abs(autocorr_signal(autocorr_new,m))) / averaging_length;
    }
```

```
autocorr_new = ((autocorr_new + 1) % (averaging_length + 1));
autocorr_old = ((autocorr_old + 1) % (averaging_length + 1));

/*****************************************************************/
/***                                                           ***/
/***                Set a threshold for the PDP                ***/
/***                                                           ***/
/*****************************************************************/

threshold = 1.0 / autocorr_length * sum(pdp) * (2.0 + 4.0 / sqrt(averaging_length));

/*****************************************************************/
/***                                                           ***/
/***            Search the local peaks in the PDP              ***/
/***                                                           ***/
/*****************************************************************/

l = 0;

if (pdp(0) > pdp(1)) {
    peak_gain(0) = pdp(0);
    peak_delay(0) = 0;
    l = 1;
}

for (k = 1; k < autocorr_length - 1; k++) {
    if (pdp(k) > pdp(k-1)) {
        if (pdp(k) > pdp(k+1)) {
            peak_gain(l) = pdp(k);
            peak_delay(l) = k;
            l++;
        }
    }
}

NOF_peaks = l;

/*****************************************************************/
/***                                                           ***/
/***        Select paths from the vector with the local peaks  ***/
/***                                                           ***/
/*****************************************************************/

j = 0;
for (i = 0; i < NOF_peaks; i++) {
    if (peak_gain(i) > threshold) {
        path_gain(j) = peak_gain(i);
        path_delay(j) = peak_delay(i);
        j++;
    }
}

NOF_paths = j;

/*****************************************************************/
/***                                                           ***/
/***    Store signal values from current frame in status file  ***/
/***                                                           ***/
/*****************************************************************/

if (save_status) {
```

```
        status_file.open(status_filename, false);

            //sprintf(variable_name, "frame%li_received_signal", frame_counter);
            //status_file << Name(variable_name) << (received_signal);

            //sprintf(variable_name, "frame%li_oversampled_scrmbl_code", frame_counter);
            //status_file << Name(variable_name) << (conj(conjugate_signal));

            sprintf(variable_name, "frame%li_avg_auto_corr", frame_counter);
            status_file << Name(variable_name) << pdp;

            sprintf(variable_name, "frame%li_peak_del_vec", frame_counter);
            status_file << Name(variable_name) << peak_delay;

            sprintf(variable_name, "frame%li_peak_pow_vec", frame_counter);
            status_file << Name(variable_name) << (peak_gain);

            sprintf(variable_name, "frame%li_num_peaks", frame_counter);
            status_file << Name(variable_name) << NOF_peaks;

            sprintf(variable_name, "frame%li_threshold", frame_counter);
            status_file << Name(variable_name) << threshold;

            sprintf(variable_name, "frame%li_detected_del_vec", frame_counter);
            status_file << Name(variable_name) << path_delay;

            sprintf(variable_name, "frame%li_detected_pow_vec", frame_counter);
            status_file << Name(variable_name) << (path_gain);

            sprintf(variable_name, "frame%li_num_paths", frame_counter);
            status_file << Name(variable_name) << NOF_paths;

        status_file.close();
    }

    frame_counter++;
}
```

D

A MLE-EM path searcher in C++

D.1 mle path searcher h

```
#ifndef __mle_path_searcher_h
#define __mle_path_searcher_h
#include "itcomm.h"
#include "path_searcher_TOP.h"

// Path searcher class
class MLE_path_searcher : public Path_Searcher_TOP {
public:
    MLE_path_searcher(void);

    //! Enable or disable messages.
    void set_verbose(bool in_verbose) { verbose = in_verbose; };
    //! Returns message status.
    bool get_verbose() { return verbose; };

    //! Enable logging and set log file name.
    void set_status_filename(char *in_status_filename);
    //! Returns logging status.
    bool get_save_status() { return save_status; };
    //! Reset the frame counter that is used by logging.
    void reset_frame_counter() { frame_counter = 1; };

    //! Set the oversample factor.
    void set_oversample_factor(int in_oversample_factor) {oversample_factor = in_oversample_factor; };
    //! Returns the oversample factor.
    int get_oversample_factor() { return oversample_factor; };
    //! Set the number of frames that the path searcher averages to estimate a multipath profile.
    void set_no_averaged_frames(int in_auto_corr_hist_length);
    //! Returns the number of frames that the path searcher averages to estimate a multipath profile.
    int get_no_averaged_frames() { return 0; };
    //! Set the length of the autocorrelation that is used.
    void set_auto_corr_length(int in_auto_corr_length);
    //! Returns the length of the autocorrelation that is used.
    int get_auto_corr_length() { return corr_length; };
    //! Set the length of the autocorrelation window.
    void set_auto_corr_window(int in_auto_corr_window) { corr_window = in_auto_corr_window; };
    //! Returns the length of the autocorrelation window.
```

```
    int get_auto_corr_window() { return corr_window; };

    //! Give the path searcher the conjugate of the primary scramble code
    //! that is used to scramble the received signal.
    void set_conj_scrmbl_code(cvec &in_conj_scrmbl_code);
    //! Returns the found delay profile.
    ivec &get_del_vec() { return path_delays; };
    //! Returns the found power profile.
    vec &get_pow_vec() { return abs_path_gains; };
    //! Returns the found number of paths.
    int get_num_paths() { return NOF_paths; };

    //! Perform path search operation.
    void operator()(const cvec &in_chips);

private:

    // Algorithm parameters
    bool    test, verbose, save_status;
    char    *status_filename, variable_name[256];
    it_file status_file;
    int     frame_counter;
    int     corr_window;
    int     corr_length;
    int     NOF_paths;                  // L
    vec     EM_params;                  // beta
    int     mu_max;                     // mu_ML

    // Variables
    int     oversample_factor;
    int     pilot_duration;             // T
    float   pilot_power;                // P

    vec     abs_path_gains;             // abs_alpha

    cvec    path_gains;                 // alpha
    ivec    path_delays;                // tau

    // Signals
    cvec    conjugated_scrambling_code; // u*
    cvec    scrambling_code;            // u
    cmat    shifted_scrambling_code;    // s
    cmat    shifted_conj_scrmbl_code;   //
    cvec    received_signal;            // r
    cvec    noise_estimate;             // n
    cmat    signal_estimate;            // x
    cmat    correlated_signal;          // z

};

#endif
```

D.2 mle path searcher cpp

```
#include "mle_path_searcher.h"

/*****************************************************************/
/***                                                           ***/
/***                   Initialize path searcher                ***/
/***                                                           ***/
/*****************************************************************/

MLE_path_searcher::MLE_path_searcher(void)
{
    // Algorithm parameters
    save_status = false;
    status_filename = NULL;
    frame_counter = 1;
    corr_window = 2560;
    corr_length = 340;
    NOF_paths = 1;
    EM_params.set_length(NOF_paths);
    EM_params.ones();
    EM_params = EM_params/NOF_paths;
    mu_max = 40;

    // Variables
    set_oversample_factor(4);
    pilot_duration = 1;
    pilot_power = 1;

    abs_path_gains.set_length(NOF_paths);
    abs_path_gains.zeros();

    path_gains.set_length(NOF_paths);
    path_gains.zeros();
    path_delays.set_length(NOF_paths);
    path_delays.zeros();

    // Signals
    scrambling_code.set_length(corr_window);
    scrambling_code.zeros();
    shifted_scrambling_code.set_size(NOF_paths,corr_window);
    shifted_scrambling_code.zeros();
    shifted_conj_scrmbl_code.set_size(NOF_paths,corr_window);
    shifted_conj_scrmbl_code.zeros();
    noise_estimate.set_size(corr_window);
    noise_estimate.zeros();
    signal_estimate.set_size(NOF_paths,corr_window);
    signal_estimate.zeros();
    correlated_signal.set_size(NOF_paths,corr_length);
    correlated_signal.zeros();
}

void MLE_path_searcher::set_conj_scrmbl_code(cvec &in_conj_scrmbl_code)
{
        conjugated_scrambling_code =
        repeat(in_conj_scrmbl_code.left(corr_window / oversample_factor), oversample_factor);

        scrambling_code = conj(conjugated_scrambling_code);
}

void MLE_path_searcher::set_status_filename(char *in_status_filename)
{
```

```
    if (in_status_filename == NULL) {
        save_status = false;
    } else {
        status_filename = in_status_filename;
        save_status = true;
    }
}

void MLE_path_searcher::set_no_averaged_frames(int in_auto_corr_hist_length){}

void MLE_path_searcher::set_auto_corr_length(int in_auto_corr_length){}

void MLE_path_searcher::operator()(const cvec &received_signal)
{
int mu;

cvec new_path_gains;
new_path_gains.set_length(NOF_paths);
new_path_gains.zeros();

ivec new_path_delays;
new_path_delays.set_length(NOF_paths);
new_path_delays.zeros();

/*****************************************************************/
/***                                                           ***/
/***            Start iteration mu of EM algorithm             ***/
/***                                                           ***/
/*****************************************************************/

for(mu = 0; mu < mu_max; mu++){
    if(mu==0){
        cout << "Initial path gains = " << abs(path_gains) << endl;
        cout << "Initial path delays = " << path_delays << endl;
    }

    /*****************************************************************/
    /***                                                           ***/
    /***            Form shifted scrambling code matrix            ***/
    /***                                                           ***/
    /*****************************************************************/

    int j;
    for (i = 0; i < NOF_paths; i++){
        for (j = 0; j < corr_window; j++){
            shifted_scrambling_code(i,j) =
            path_gains(i) * scrambling_code(mod(j - path_delays(i),corr_window));
        }
    }

    /*****************************************************************/
    /***                                                           ***/
    /***                Determine noise estimate                   ***/
    /***                                                           ***/
    /*****************************************************************/

    cvec temp_sum;
    temp_sum.set_length(corr_window);
    temp_sum.zeros();
    for (j = 0; j < corr_window; j++){
        for (i = 0; i < NOF_paths; i++){
            temp_sum(j) += shifted_scrambling_code(i,j);
```

```
    }
    noise_estimate(j) = received_signal(j) - temp_sum(j);
}

/*****************************************************************/
/***                                                           ***/
/***                Determine signal estimates                 ***/
/***                                                           ***/
/*****************************************************************/

for (i = 0; i < NOF_paths; i++){
    for (j = 0; j < corr_window; j++){
        signal_estimate(i,j) = shifted_scrambling_code(i,j) + EM_params(i) * noise_estimate(j);
    }
}

/*****************************************************************/
/***                                                           ***/
/***                Correlate the signal estimates             ***/
/***                                                           ***/
/*****************************************************************/

cvec temp_corr_signal;
temp_corr_signal.set_length(corr_window);
temp_corr_signal.zeros();
int k;

for (k = 0; k < NOF_paths; k++) {
    for (i = 0; i < corr_length; i++) {
        for (j = 0; j < corr_window; j++)   {
            temp_corr_signal(j) = signal_estimate(k,j) * conj(scrambling_code(mod(j-i,corr_window)));
        }
        correlated_signal(k,i) = 1.0 / corr_window * sum(temp_corr_signal);
    }
}

/*****************************************************************/
/***                                                           ***/
/***                Calculate path parameters                  ***/
/***                                                           ***/
/*****************************************************************/

for (i = 0; i < corr_length; i++) {
    for (k = 0; k < NOF_paths; k++) {
        if (abs(correlated_signal(k,i)) > abs(new_path_gains(k))){
            new_path_gains(k) = correlated_signal(k,i);
            new_path_delays(k) = i;

            for (int q=0; q<NOF_paths-1; q++){
                correlated_signal(mod(k+q+1,NOF_paths),i) = 0;
            }
        }
    }
}

new_path_gains = new_path_gains / (pilot_duration * pilot_power);

/*****************************************************************/
/***                                                           ***/
/***                Determine convergence rate                 ***/
/***                                                           ***/
/*****************************************************************/
```

```
    cvec diff_path_gains;
    diff_path_gains.set_length(NOF_paths);
    diff_path_gains.zeros();

    ivec diff_path_delays;
    diff_path_delays.set_length(NOF_paths);
    diff_path_delays.zeros();

    for (i = 0; i < NOF_paths; i++) {
        diff_path_gains(i) = new_path_gains(i) - path_gains(i);
        diff_path_delays(i) = new_path_delays(i) - path_delays(i);
    }

    /*****************************************************************/
    /***                                                           ***/
    /***                   Start next iteration                    ***/
    /***                                                           ***/
    /*****************************************************************/

    double metric;
    metric = 0;
    for (i = 0; i < NOF_paths; i++) {
        metric += (abs(diff_path_gains(i))/abs(path_gains(i)));
    }

    if(metric < 0.001){
        cout << "converged in " << mu+1 << " iteration(s)" << endl;
        mu = mu_max;
    }
    else{
        // Theta(mu) = Theta(mu + 1)
        path_gains = new_path_gains;
        path_delays = new_path_delays;

        if(mu == mu_max-1) {cout << "maximum of " << mu_max << " iterations reached" << endl;}
    }
}

cout << "Final channel parameter estimates: " << endl;
cout << "path_delays = " << path_delays << endl;
cout << "path_gains = " << abs(path_gains) << endl;

    /*****************************************************************/
    /***                                                           ***/
    /***    Store signal values from current frame in status file  ***/
    /***                                                           ***/
    /*****************************************************************/

    if (save_status) {

        status_file.open(status_filename, false);

            sprintf(variable_name, "frame%li_received_signal", frame_counter);
            status_file << Name(variable_name) << received_signal;

            sprintf(variable_name, "frame%li_oversampled_scrmbl_code", frame_counter);
            status_file << Name(variable_name) << scrambling_code;

            sprintf(variable_name, "frame%li_shifted_scrambling_code", frame_counter);
            status_file << Name(variable_name) << shifted_scrambling_code;
```

```
            sprintf(variable_name, "frame%li_noise_estimate", frame_counter);
            status_file << Name(variable_name) << noise_estimate;

            sprintf(variable_name, "frame%li_signal_estimate", frame_counter);
            status_file << Name(variable_name) << signal_estimate;

            sprintf(variable_name, "frame%li_correlated_signal", frame_counter);
            status_file << Name(variable_name) << correlated_signal;

            sprintf(variable_name, "frame%li_detected_del_vec", frame_counter);
            status_file << Name(variable_name) << path_delays;

            sprintf(variable_name, "frame%li_detected_pow_vec", frame_counter);
            status_file << Name(variable_name) << (path_gains);

            sprintf(variable_name, "frame%li_num_paths", frame_counter);
            status_file << Name(variable_name) << NOF_paths;

        status_file.close();
    }
    frame_counter++;
}
```

Bibliography

[1] 3GPP. "Universal mobile telecommunications system (UMTS); base station (BS) radio transmission and reception (FDD)". *ETSI TS*, vol. 125(no. 104):version 3.3.0, September 2003.

[2] S. Buzzi and H.V. Poor. "On parameter estimation in long-code DS/CDMA systems: Cramér-Rao bounds and least-squares algorithms". *IEEE transactions on signal processing*, vol. 51(no. 2):pp. 545–559, February 2003.

[3] G. Caire and U. Mitra. "Weighted RLS channel estimators for DS/CDMA signals multipath". *Military communications conference proceedings*, vol. 2(no. 1):pp. 1307–1311, October 1999.

[4] A.H. Ulusoy et al. "Multipath-decorrelating receiver using adaptive path selection for synchronous CDMA frequency-selective fading channels". *Electronics letters*, vol. 36(no. 22):pp. 1877–1878, October 2000.

[5] B.H. Fleury et al. "Channel parameter estimation in mobile radio environments using the SAGE algorithm". *IEEE journal on selected areas in communications*, vol. 17(no. 3):pp. 434–450, March 1999.

[6] C. Escudero et al. "A Toeplitz displacement method for blind multipath estimation for long code DS/CDMA signals". *IEEE transactions on signal processing*, vol. 49(no. 3):pp. 654–665, March 2001.

[7] C. Xiao et al. "Second-order statistical properties of the WSS Jakes' fading channel simulator". *IEEE transactions on communications*, vol. 50(no. 6):pp. 888–891, June 2002.

[8] D. Dahlhaus et al. "A sequential algorithm for joint parameter estimation and multiuser detection in DS/CDMA systems with multipath propagation". *Wireless personal communications: an international journal*, vol. 6(no. 1/2):pp. 161–178, January 1998.

[9] E. Ertin et al. "Iterative techniques for DS/CDMA multipath channel estimation". *Proceedings of the 1999 Allerton conference*, vol. 1(no. 1):pp. 772–781, September 1998.

[10] E. Ertin et al. "Maximum-likelihood based multipath channel estimation for code-division multiple access systems. *IEEE transactions on communications*, vol. 49(no. 2):pp. 290–302, February 2001.

[11] F. Bouchereau et al. "Multipath delay estimation using a superresolution PN-correlation method". *IEEE transactions on signal processing*, vol. 49(no. 5):pp. 938–949, May 2001.

[12] G.W. Rice et al. "A novel channel estimation algorithm applied to UTRA-FDD". *IEE 3rd international conference on 3G mobile communication technologies*, vol. 489(no. 1):pp. 267–271, May 2002.

[13] J. Mitsugi et al. "Path-search algorithm introducing path-management tables for a DS-CDMA mobile terminal". *IEEE 13th international symposium on personal, indoor and mobile radio communications*, vol. 2(no. 1):pp. 730–734, September 2002.

[14] J.K. Seok et al. "Performance of coherent DS-SS/QPSK for mobile communications in fast-fading multipath and high-frequency offset". *IEEE transactions on vehicular technology*, vol. 50(no. 1):pp. 250–266, January 2001.

[15] P. Dent et al. "Jakes model revisited". *Electronic letters*, vol. 29(no. 13):pp. 1162–1163, June 1993.

[16] P.P. Moghaddam et al. "A new time-delay estimation in multipath". *IEEE transactions on signal processing*, vol. 51(no. 5):pp. 1129– 1142, May 2003.

[17] R. Hamila et al. "Subchip multipath delay estimation for downlink WCDMA system based on Teager-Kaiser operator". *IEEE communications letters*, vol. 7(no. 1):pp. 1–3, January 2003.

[18] S. Fukumoto et al. "Path search performance and its parameter optimization of pilot symbol-assisted coherent Rake receiver for W-CDMA mobile radio". *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. E83-A(no. 11):2110–2119, November 2000.

[19] T. Eyceoz et al. "Deterministic channel modeling and long range prediction of fast fading mobile radio channels". *IEEE communications letters*, vol. 2(no. 9):pp. 254–256, September 1998.

[20] T.L. Fulghum et al. "The Jakes fading model for antenna arrays incorporating azimuth spread". *IEEE transactions on vehicular technology*, vol. 51(no. 5):pp. 968–977, September 2002.

[21] V. Tripathi et al. "Channel acquisition for wideband CDMA Signals". *IEEE journal on selected areas in communications*, vol. 18(no. 8):pp. 1483–1494, August 2000.

[22] Y. Baoguo et al. "Channel estimation for OFDM transmission in multipath fading channels based on parametric channel modeling". *IEEE transactions on communications*, vol. 49(no. 3):pp. 467–479, March 2001.

[23] ETSI. "Universal mobile telecommunications system (UMTS); selection procedures for the choice of radio transmission technologies of the UMTS". *ETSI TR*, vol. 101(no. 112):version 3.2.0, April 1998.

[24] J.J. Fuchs. "Multipath time-delay detection and estimation". *IEEE transactions on signal processing*, vol. 47(no. 1):pp. 237–243, January 1999.

[25] K.C. Gan. "Path searcher for a WCDMA Rake receiver". *Motorola application note*, vol. 1(no. AN2252/D):pp. 1–17, March 2002.

[26] M. Guenach. "Receiver design for wideband CDMA communication systems". *Université catholique de Louvain, Faculté des sciences appliquées, Laboratoire de télécommunnications et télédetection*, 2002.

[27] S. Haykin. *Digital communications*. John Wiley and Sons, 1989.

[28] B.J. Hwang. "Performance analysis of multipath searcher in WCDMA System".

[29] W.C. Jakes. *Microwave mobile communications*. John Wiley and Sons, 1975.

[30] T. Kanungo and R.M. Haralick. "Receiver operating characteristic curves and optimal Bayesian operating points". *Proceedings of the international conference on image processing*, vol. 3(no. 1):pp. 256–259, October 1995.

[31] A. Kocian and B.H. Fleury. "EM-based joint data detection and channel estimation of DS-CDMA signals". *IEEE transactions on communications*, vol. 51(no. 10):pp. 1709–1720, October 2003.

[32] M.F. Pop and N.C. Beaulieu. "Limitations of sum-of-sinusoids fading channel simulators". *IEEE transactions on communications*, vol. 49(no. 4):pp. 699–708, April 2001.

[33] J.G. Proakis. *Digital communications*. McGraw-Hill, 2001.

[34] T.S. Rappaport. *Wireless communications: principles and practices*. Prentice-Hall, 2001.

[35] J.H. Shapiro. "Extended version of Van Trees's receiver operating characteristic approximation". *IEEE transactions on aerospace and electronic systems*, vol. 35(no. 2):pp. 709–716, April 1999.

[36] S.H. Won and Y.J. Kim. "Performance analysis of multi-path searcher for mobile station in W-CDMA system employing transmit diversity". *Electronics letters*, vol. 39(no. 1):pp. 137–138, January 2003.

[37] S.H. Won and K. Seo. "Performance analysis of multi-path searcher for UE in W-CDMA systems". *The 12th joint conference on communications and informations*, 2003.