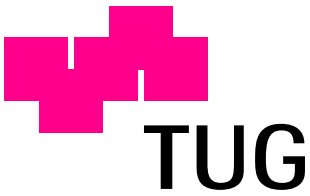
Doctoral Thesis

Multipath Tracking and Prediction for Multiple-Input Multiple-Output Wireless Channels

Dmitriy Shutin

Faculty of Electrical Engineering and Information Technology
Graz University of Technology, Austria



First Advisor:
Prof. Dr. Gernot Kubin, Graz University of Technology, Austria
Second Advisor:
Prof. Dr. Bernard Fleury, Aalborg University, Denmark

Graz, December 12, 2006

Time-series forecasting is like driving a car blindfolded by following the instructions from someone looking back through the rear window

Abstract

In this work we develop and study a framework for tracking and prediction of multipath components for wireless MIMO channels. The proposed methodology is a multi-stage procedure that relies on the concept of hypermodels, which capture the dynamics for each multipath.

First the individual multipaths are resolved and extracted. In this work we also develop a new estimation algorithm based on the Evidence Procedure and the SAGE algorithm that allows to determine the number of multipath components. The extracted components are then tracked and predicted over time using hypermodels, which are build iteratively, as the tracking proceeds. For prediction we use linear as well as nonlinear hypermodels.

We find that linear predictors are more efficient since they are adapted faster. With only 3 coefficients we achieve prediction horizons up to 3 times the wavelength λ for real-world measured data, as compared to 1.5λ reported so far in the literature.

Zusammenfassung

In dieser Arbeit untersuchen und entwickeln wir ein System für das Verfolgen und die Vorhersage von Komponenten der Mehrwegeausbreitung auf MIMO Funkkanälen. Das vorgeschlagene System ist ein Mehrstufenverfahren, das sich auf das Hypermodellkonzept stützt, um die Dynamik für jede Komponente zu repräsentieren.

Zuerst werden die individuellen Ausbreitungspfade geschätzt und extrahiert. Wir entwickeln auch einen neuen Schätzalgorithmus, der auf dem Evidenzverfahren und dem SAGE Algorithmus basiert, was auch erlaubt die Anzahl von Ausbreitungspfaden zu bestimmen. Die extrahierten Komponenten werden dann mittels Hypermodellen, die iterativ gelernt werden, verfolgt und vorhergesagt. Für die Vorhersage verwenden wir lineare sowie nichtlineare Hypermodelle.

Es wurde festgestellt, dass die linearen Hypermodelle effizienter sind, weil sie schneller angepaßt werden können. Mit nur 3 Modellkoeffizienten erreichen wir einen Vorhersagehorizont von bis zum Dreichfachen der Wellenlänge λ für real gemessene Daten im Vergleich zu 1.5λ, was bisher in der Literatur berichtet wurde.

Acknowledgment

I would very much like to express my gratitude to all those who have helped and influenced me along this winding road to the completion of the thesis.

First of all, I would like to thank my dear friend, Vyacheslav Vinogradov, for his insights into the philosophy of science and scientific methods. Although our discussions were often quite abstract, it is only now, when this work is done, I can really appreciate the importance of questions raised during our discussions.

I thank my supervisor Prof. Gernot Kubin for his giving me a chance of becoming a researcher. I always admired your ability to explain complicated concepts in a simple and very logical way. Your style has greatly influenced me in many respects and formed my understanding of the Signal Processing field.

I am also very grateful to Prof. Bernard H. Fleury from Aalborg University, with whose acquaintance I can boast for almost two years. The experience I got while working together with him in his group in Aalborg is indispensable. I hope that one day I will be able to master his scientific working style.

I should also thank the whole Signal Processing and Speech Communication laboratory that over these years has become the second home for me, and this home is difficult, if not impossible, to forget.

Also, I would like mention many colleagues from the Forschungszentrum Telekommunikation Wien (FTW), with whom I had a great pleasure to discuss my (as well as their) ongoing research during my numerous visits.

Graz, Dmitriy Shutin
December 12, 2006

to my parents, Vladimir and Zhanna,
and to my beloved wife Olga and our beautiful daughter Vera.

Contents

List of Figures xvii

Abbreviations xxi

1 Introduction 1
- 1.1 Fading phenomena in wireless channels . . . 1
  - 1.1.1 Large-scale fading . . . 2
  - 1.1.2 Small-scale fading . . . 3
- 1.2 Mitigating fading effects . . . 4
  - 1.2.1 Channel prediction and hypermodel idea . . . 5
  - 1.2.2 What can MIMO channels offer? . . . 7
  - 1.2.3 Multipath-based channel prediction . . . 9
- 1.3 Outline of the thesis . . . 9
- 1.4 Work contributions . . . 12

2 Understanding MIMO channels 15
- 2.1 Wireless channel impulse response . . . 15
- 2.2 SIMO channel . . . 20
- 2.3 MIMO channel representation . . . 22
- 2.4 Discussion . . . 24

3 MIMO channel estimation 27
- 3.1 Channel sounding . . . 27
  - 3.1.1 Channel sounding using pulse-compression techniques . . . 28
  - 3.1.2 Frequency domain channel sounding . . . 29
  - 3.1.3 Signal model in a plane waves scenario . . . 30
  - 3.1.4 Sampling wireless channels . . . 33
- 3.2 Space-Alternating Generalized Expectation-Maximization . . . 34
  - 3.2.1 Initializing SAGE with Matching Pursuit algorithm . . . 38
  - 3.2.2 Some application examples . . . 41
- 3.3 Conclusions and discussion . . . 46

4 Evidence Procedure and channel estimation 49
4.1 Signal model . . . 51
4.2 Evidence maximization, Relevance Vector Machines and wireless channels . . . 52
4.2.1 Learning algorithm . . . 53
4.2.2 Extensions to multiple channel observations . . . 56
4.3 Model selection and basis pruning . . . 58
4.3.1 Statistical analysis of the hyperparameters in the stationary point . . . 59
4.3.2 Improving the learning algorithm to cope with the model selection . . . 70
4.3.3 MDL principle and Evidence Procedure . . . 71
4.4 Application of the RVM to wireless channels . . . 75
4.4.1 Simulation setup . . . 76
4.4.2 Numerical simulations . . . 77
4.4.3 Results for measured channels . . . 81
4.5 SAGE iterations and SAGE-RVM algorithm . . . 83
4.5.1 Basic steps of the SAGE-RVM algorithm . . . 83
4.5.2 Some application examples . . . 86
4.6 Discussion and conclusions . . . 88
4.6.1 Evidence Procedure . . . 88
4.6.2 SAGE-RVM algorithm . . . 90

5 Channel tracking 91
5.1 Multipath tracking . . . 92
5.1.1 Dynamic programming and assignment problem . . . 93
5.1.2 Selecting the cost function . . . 96
5.2 Structure hypermodel $\mathcal{S}_k$ for channel tracking . . . 99
5.2.1 Damped local linear trend . . . 99
5.3 Hypermodels $\mathcal{A}_k$ . . . 100
5.3.1 Adaptive Linear Predictor (ALP) . . . 101
5.3.2 Iterated Adaptive Linear Predictor (IALP) . . . 102
5.3.3 Nonlinear predictor based on Volterra models (AVNP) . . . 103
5.3.4 Nonlinear predictor based on Neural Networks (IANNP) . . . 106
5.4 Discussion and conclusions . . . 107

6 Multipath forecasting 111
6.1 Choosing simulation parameters . . . 111
6.2 Measuring the prediction quality . . . 115
6.3 SAGE-based multipath prediction . . . 117
6.3.1 Tracking example 1: SIMO channel with a single track . . . 117
6.3.2 Tracking example 2: Extending tracking time . . . 127
6.3.3 Tracking example 3: Tracking multiple components . . . 130
6.4 Evidence Procedure-based multipath extraction and prediction . . . 137

6.4.1 Tracking example 4: Single component tracking 137
6.4.2 Tracking example 5: Tracking several components 139
6.5 Discussion of the obtained tracking and prediction results 144
6.5.1 Tracking 144
6.5.2 Gain prediction 145

7 Discussion and conclusions **149**

A Taylor approximation to the electrical distance (SIMO case) **153**

B Taylor approximation to the electrical distance (MIMO case) **157**

C Description of the channel data (FTW) **161**
C.1 General data description 161
C.1.1 Sample impulse response 163
C.1.2 Doppler-Delay profile 163

D Description of the channel data (Elektrobit) **165**

E Evidence update expressions **167**

Bibliography **171**

List of Figures

1.1 Multipath propagation of electromagnetic waves. 2
1.2 Large-scale fading in an outdoor environment. 3
1.3 Small-scale fading of the received power as the mobile moves a distance of several wavelengths in an outdoor environment. In this example, the wavelength $\lambda \approx 0.15$m. 4
1.4 Hypermodel approach to modeling the channel dynamics. 6
1.5 Predicting the impulse responses of the wireless channel. 8
1.6 Predicting multipath components in the impulse responses of the wireless channel. 10

2.1 The physical distance with a reference sensor. 18
2.2 Decomposition of the path delay into time-varying and time-invariant parts. 19
2.3 Geometrical situation considered in the SIMO case with the moving effective source and P-sensor array $D(P)$. 21
2.4 Geometrical situation considered in the MIMO case with the moving effective source. 23
2.5 Approximation of the effective source trajectory by a sequence of linear displacements. 25

3.1 An equivalent baseband model of radio channel sounding with receiver matched filter (MF) front-end. 28
3.2 Sounding signal $s(t)$. 28
3.3 Sequential SIMO channel acquisition and processing. 34
3.4 Matching Pursuit greedy signal approximation. 39
3.5 Structure of the coefficients w_{lj} for a single basis $\boldsymbol{r}_l$. 40
3.6 Structure of the matrix $\boldsymbol{W}_l$. 40
3.7 Evolution of the measured channel power-delay profile. 41
3.8 Goodness-of-fit for the SAGE approximation with $L = 1$. 42
3.9 Goodness-of-fit for the SAGE approximation with $L = 3$. 43
3.10 Goodness-of-fit for the SAGE approximation with $L = 15$. 44
3.11 Goodness-of-fit for the SAGE approximation with $L = 30$. 45
3.12 Approximation of a single component with delay τ' by three discrete components with delays τ_1, τ_2, and τ_3. 47

4.1 Graphical model representing the dependence structure of the discrete-time model of the wireless channel. 54
4.2 Iterative learning of the parameters; The superscript $[j]$ denotes the iteration index. 56
4.3 Usage of α_l in a multiple-observation discrete-time wireless channel model to represent P coherent channel measurements. 57
4.4 Evolution of the two representative solution trajectories for two cases: (a) $\{\alpha^{[j]}\}$ converges, (b) $\{\alpha^{[j]}\}$ diverges. 61
4.5 Model mismatch. 65
4.6 Evaluated correlation functions a)$R_{uu}(t)$ and b) $\gamma(\tau)$ 66
4.7 Comparison between the empirical and theoretical pdf's of the a)$\gamma(\tau)$ and b)$|\gamma(\tau)|^2$ for the linear approximation case. 67
4.8 Evaluated correlation functions a)$R_{uu}(t)$ and b) $\gamma(\tau)$ for the cosine approximation case. 68
4.9 Comparison between the empirical and theoretical pdf's of the a)$\gamma(\tau)$ and b)$|\gamma(\tau)|^2$ for the cosine approximation case. 68
4.10 Model selection by evidence evaluation. 74
4.11 Model selection by evidence evaluation. 74
4.12 Evidence-based model selection criteria. a) Empirical (bar plot) and theoretical (solid line) pdf's of hyperparameters α_{n}^{-1} (SNR = 10dB, and $P = 10$), b) Negative log-evidence as a function of the model order (number of paths) for different SNR values ($P = 5$, and $L = 20$). 78
4.13 Multipath detection rates based on the EP. (a) Quantile-based model selection versus P: $\rho = 1 - 10^{-6}$, $L = 5$; (b) Quantile-based model selection versus ρ: $P = 5$, $L = 5$; (c) Negative log-evidence-based detection versus P. 79
4.14 Comparison of the model selection schemes in a single path scenario. (a,c,e) path detection probability, and (b,d,f) probability of correct path extraction for $P = 5$, and (a,b) $N_s = 1$; (c,d) $N_s = 2$; and (e,f) $N_s = 4$. 82
4.15 Multipath detection results for quantile-based method with $\rho = 1 - 10^{-6}$. 84
4.16 (a,b,c) Goodness-of-fit for the SAGE-RVM algorithm. The number of estimated components is $L = 14$. 87
4.17 Evolution of the estimated multipath parameters (Delays, Doppler frequency, DoA, and number of wavefronts L). 88

5.1 Iterative multipath tracking and adaptation of the track hypermodels $\mathcal{H}_k$. 92
5.2 Possible track continuation scenarios. 94
5.3 Augmented graphs for balancing the assignment problem. 95
5.4 Geometrical definition of the spatial component $MCD_{DoA,kl}$. 97
5.5 The form of the distance function $f(\cdot,\cdot)$ for a single parameter. . . . 98

5.6 Structure of the ALP with RLS-based adaptation of predictor coefficients for $\mathcal{L} = 2$. 101
5.7 Signal flow diagram of the Volterra filter. 104
5.8 Structure of the Volterra model-based Nonlinear Predictor with RLS-based adaptation of predictor coefficients. 106
5.9 Structure of the Neural Network used for hypermodel approximation. 107

6.1 A sample measured prediction error for a one-step-ahead ALP hypermodel. 116
6.2 (Example 1) Reconstructed trajectories of the track structure parameters. 118
6.3 (Example 1) Evolution of the real and imaginary parts of the gain and of the power of the estimated track. 119
6.4 (Example 1) Spectrogram of the complex gain variation of the estimated track. 120
6.5 (Example 1) Complex gain prediction using the ALP hypermodel. $\mathcal{L} = 1$, $Q = 3$. 121
6.6 (Example 1) Prediction gain for the ALP hypermodel with different model orders. 121
6.7 (Example 1) Complex gain prediction using the IALP hypermodel. $\mathcal{L} = 1$, $Q = 3$. 123
6.8 (Example 1) Prediction gain for the IALP hypermodel with different model orders. 123
6.9 (Example 1) Complex gain prediction using the AVNP1 (Table 6.2) hypermodel. $\mathcal{L} = 1$. 124
6.10 (Example 1) Prediction gain for the AVNP hypermodel with different model structures. 125
6.11 Complex gain prediction using the IANNP1 (Table 6.3) hypermodel. $\mathcal{L} = 1$. 126
6.12 (Example 1) Prediction gain for the IANNP hypermodel with different network structures. 127
6.13 (Example 2) Reconstructed trajectories of the track structure parameters. 129
6.14 Example 2: Evolution of the real and imaginary parts of the gain and of the power of the estimated track. 130
6.15 (Example 2) Spectrogram of the complex gain variation of the estimated track. 130
6.16 (Example 2) Prediction gain for a single track evaluated over the distance of 71λ (10m). 131
6.17 (Example 3) Reconstructed multipath trajectories. $K = 5$. 132
6.18 Virtual reconstructed geometry of wavesources distribution. 133
6.19 (Example 3) Evolution of the track powers. 133
6.20 (Example 3) Evolution of the real and imaginary parts of the gain for the estimated tracks. 134

6.21 (Example 3)PG evaluated for $K = 5$ reconstructed tracks. 135
6.22 (Example 4) Reconstructed multipath trajectories. $K = 5$. (SAGE-RVM). 138
6.23 (Example 4) Evolution of the real and imaginary parts of the gain and of the power of the estimated track. 139
6.24 (Example 4) Spectrogram of the complex gain variation of the estimated track. 139
6.25 (Example 4) Prediction gain for a single track. 140
6.26 (Example 5) Reconstructed multipath tracks. (SAGE-RVM). 141
6.27 Evidence of the tracked components. 142
6.28 Evolution of the real and imaginary parts of the gain for the estimated tracks. 142
6.29 (Example 5) PG evaluated for $K = 5$ reconstructed tracks. 143

A.1 Computing the electrical distance term for the SIMO case. 153

B.1 Computing the electrical distance term for the MIMO case. 157

C.1 Transmitter and receiver array configurations. 162
C.2 Relationship between some of the sounding parameters and the structure of the impulse response. 162
C.3 A sample impulse response of the wireless SIMO channel. 163
C.4 Estimated Doppler bandwidth. 164

D.1 Evaluated normalized autocorrelation sequence of the sounding signal $u(t)$. a) autocorrelation $R_{uu}(t)$, b) close-up on the main lobe of the $R_{uu}(t)$. 165
D.2 Computed Power Delay Profile for the PropSound data 166

Abbreviations

ALP	Adaptive Linear Predictor
AR	Autoregression, autoregressive
AVNP	Adaptive Volterra Nonlinear Predictor
CSI	Channel State Information
DLLT	Damped Local Linear Trend
DoA	Direction-of-Arrival
DoD	Direction-of-Departure
EP	Evidence Procedure
IALP	Iterated Adaptive Linear Predictor
IANNP	Iterated Adaptive Neural Network Predictor
IR	Impulse responce
KF	Kalman Filter
LLT	Local Linear Trend
LOS	Line-Of-Sight
MCD	Multipath Component Distance
MF	Matched Filter
MIMO	Multiple-Input Multiple-Output
MISO	Multiple-Input Single-Output
MP	Matching Pursuit algorithm
NAR	Nonlinear Autoregression
NN	Neural Network

PDP	Power-Delay Profile
PG	Prediction Gain
PSD	Power Spectral Density
RX	Receiver
SAGE	Space-Alternating Generalized Expectation-maximization
SIMO	Single-Input Multiple-Output
SISO	Single-Input Single-Output
TX	Transmitter

Chapter 1

Introduction

Today it is difficult to overestimate the importance of telecommunications in our everyday life. The recent advances in many areas of computer science and electronics technology, coupled with the current economic globalization trends, have stimulated the booming increase of the global information production. It has been said that information is the new "gold" of the XXI century. Thus, it is extremely important to be able to access information at any given time, at any given place.

In comparison with other communication systems, mobile wireless communication systems seem to outpace other means of information exchange mainly due to the ubiquity of the electromagnetic waves and necessity to access information "anytime and anywhere". The new generations of wireless systems bring new requirements [ZAB99, Oli99] to satisfy ever increasing consumer demands for high speed data transfers (up to tens of Mbits/s), video, multimedia, as well as voice traffic to mobile users. This all creates a need for more powerful and efficient algorithms for modulation schemes, coding, power control, and detection techniques.

In the heart of any wireless communication systems lies the mobile channel – a system that ultimately characterizes the properties of the transmission media, and thus the achievable communication performance. The channel is both "a curse" and "a blessing" of wireless communication. The "blessing" comes simply from the ubiquitous nature of the transmission medium, that, in principle, allows the reception in almost any place where the electromagnetic field can be detected. This gives the user freedom of movement, and thus mobility. However the dear price paid for that freedom is channel variability. Channel properties vary depending on where the user is, putting more requirements on the design of the actual device. Mitigating the effects of channel variability, also known as *fading*, is the major concern in the present work.

1.1 Fading phenomena in wireless channels

The mobile channel places some fundamental limitations on the performance of the wireless communication systems. In a modern urban environment, a transmission path between the transmitter and the receiver may vary from a simple line-of-sight scenario, to a path completely obstructed by buildings, natural objects, or foliage. To put it shortly, the channel is constituted of all the objects that directly or in-

directly interact with the electromagnetic field created by the transmitter (see Fig. 1.1). The mechanisms behind electromagnetic wave propagation are diverse, but can generally be attributed to *reflection*, *diffraction*, and *scattering* [Rap02].

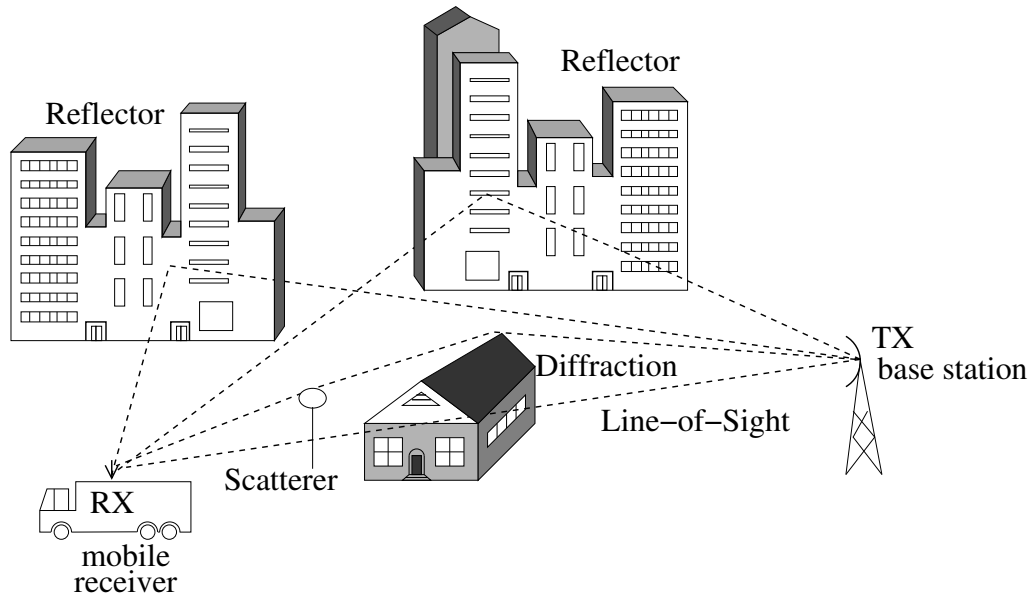


Figure 1.1: Multipath propagation of electromagnetic waves.

Due to the motion of either the transmitter/receiver or the objects that interact with the emitted signal, the electromagnetic waves travel along different paths of varying length. Interaction between all these waves causes the received power at a certain location to vary. These variations are called *fading*. Depending on the phases and amplitudes of the interacting waves, their total effect could be *constructive* (i.e., they sum up so that the total power increases), or *destructive*, when their interaction results in the drop of power. The difficulty in dealing with fading is its nonstationary behavior that strongly depends on the actual environment, i.e., the geometrical distribution of the interacting objects. The latter is itself subject to time variation, especially in mobile communications. This in turns means very complex and nonstationary behavior of the corresponding channel.

It is convenient to distinguish fading caused by slowly changing factors, such as moving away from the transmitter, causes slow power drop, and those that vary fast on top of them, mostly caused by the phase variations of multipath components arriving via different paths. These two types of fading are known as *large-scale fading* and *small-scale fading*, respectively.

1.1.1 Large-scale fading

The name of this type of fading speaks for itself. It describes the variations of the received power over relatively large distances, usually from tens to thousands of meters. Large-scale fading effects are mainly caused by the particularities of

the terrain profiles, e.g., suburban areas, mountain areas, cities, etc. A significant amount of efforts has been invested in the development of propagation models that accurately reflect the variation of the received power over large distances, which is an important factor in the design of the cellular networks [Rap02]. Large-scale propagation models are constructed to predict the mean power for an arbitrary transmitter-receiver separation and to estimate the coverage area of a transmitter in a certain environment. However, a particular model might include some additional constraints regulating the RX-TX separations over which it can be used. In Figure 1.2 one can see the variation of the instantaneous power of a measured wireless mobile channel over a distance of several tens of meters. The dashed line on the plot shows a more gradual power variation corresponding to the large-scale fading.

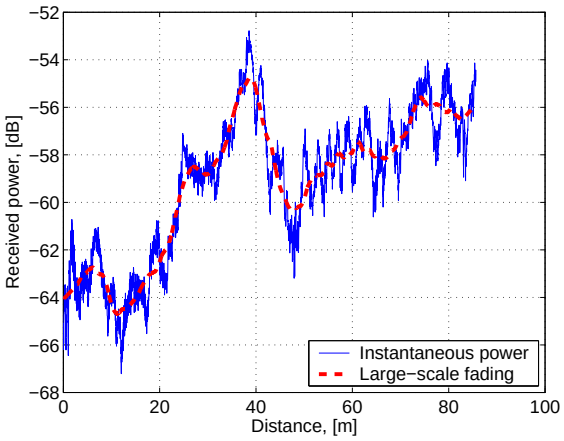


Figure 1.2: Large-scale fading in an outdoor environment.

It can be seen that as the instantaneous received power varies fast, the large-scale fading evolves at much lower rate.

1.1.2 Small-scale fading

Small-scale fading, on the other hand, stems from the rapid fluctuations of the phases of a radio signal over very short distances (on the order of several wavelengths, i.e., centimeter scale for a typical wireless communication system operating in the GHz frequency range). The cause of such rapid fluctuations is the interference between the multipath waves that arrive at the receiver at slightly different times. As the result, depending on the phases of the incoming wavefronts, the resulting power is either increased (maxima of the resulting interference pattern), or reduced (minima of the interference pattern). It is the movement through this pattern that creates the small-scale fading (see Fig. 1.3).

There are several physical factors in the radio propagation channel that influence small-scale fading. Some of the most dominant factors are:

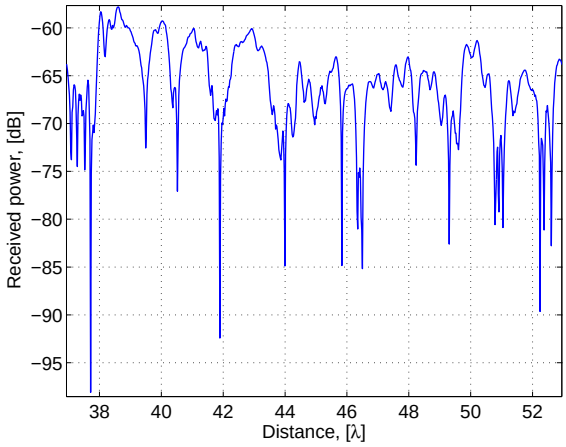


Figure 1.3: Small-scale fading of the received power as the mobile moves a distance of several wavelengths in an outdoor environment. In this example, the wavelength $\lambda \approx 0.15$m.

- **Multipath propagation** – The presence of reflecting and scattering objects that spread the signal energy in the amplitude, phase and time. These effects produce multiple copies of the transmitted signal that arrive at the receiving antenna causing interference (see, for example, Fig. 1.1, where 3 paths of different length are superimposed at the receiving antenna). Multipath propagation causes *delay spread* – the time duration needed for all of the replicas of the emitted signal, or in other words echoes, to die out. If we are talking about multiple antennas, then in addition to the latter, the multipath propagation also induces *angular spread* – the spread of the angles of the waves that impinge on the antenna array at a certain instant of time t.

- **Speed of the mobile** – The motion of the transceiver through the interference field pattern results is time-dependent phase variations. These variations cause a specific modulation of the transmitted signal, also known as the Doppler shift.

The large-scale effects are the key factors that govern the design and planning of the cellular network. The small-scale fading, on the other hand, directly impacts the design of the actual transceiver, since this is where the knowledge of the instantaneous power is mostly needed.

1.2 Mitigating fading effects

Fading effects are among the most critical factors affecting the quality of the communication link. To ensure reliable communication the transceiver should take actions to mitigate fading effects, and different strategies exist that try to accomplish this task. In general, two main approaches can be taken – either try to come up with a

clever coding scheme that allows a distribution of data bits that turns out to be very robust against fading (typical examples include space-time codes [TSA98, PNG03]), or try to construct a more intelligent transmitter that knows how to "invert" the distortions introduced by the channel. The first approach is based mainly on information theory and code design. The latter approach, on the other hand, relies on the results of system theory and signal processing.

In the present work, we are mainly investigating the second approach, i.e., we are trying to a find a system that, by exploiting knowledge about the state of the propagation environment, tries to counteract fading.

The small-scale variations of a mobile radio channel can be directly observed as the temporal variations of the impulse response of the channel. The channel impulse response (IR) is the key characteristics of the wireless transmission medium, and it contains all information about the local propagation environment necessary to simulate or analyze any type of transmission through the channel. Thus, in order to find methods counteracting fading, it is imperative to somehow observe and represent the actual channel dynamics.

Here, again two concurrent approaches exist – one can endeavor a statistical approach that lies in finding statistical models that approximate channel behavior. For instance, the channel parameters are treated as random variables described with the appropriate density functions. This allows to describe the channel behavior in terms of the statistical moments, which are further used to optimally design the communication system [KA00].

Alternatively, one can think of a deterministic approach, treating channel observations as samples from a certain multidimensional complex dynamical process that can be learned and represented accurately with some deterministic models. Channel prediction is a method that falls under this category and the one that we present in this work. Below we discuss the specifics of this approach in more details.

1.2.1 Channel prediction and hypermodel idea

The study presented here relies on a deterministic approach to fading compensation. In other words, we are looking for a model that deterministically represents the local (i.e., over a certain time frame) dynamics of the mobile channel. The required model should capture the evolution of the propagation environment during this frame as closely as possible. We will call this model *a hypermodel*.

Let us consider the following example:

Example

The mobile transmitter, moving with a velocity of $v = 30$m/s emits a narrowband signal with the center frequency $f_c = 2$GHz. The corresponding wavelength is $\lambda \approx 0.15$m. The receiver is a fixed linear antenna array with sensors spaced at a distance $d = \lambda/2$.

At the receiver, each sensor in the antenna array will receive an incident plane wave with a delay $\Delta = d/c = (\lambda/2)/3 \cdot 10^8 = 0.5$ns, which is equal to the time it

takes for an electromagnetic wave to travel from one sensor to the neighboring one. Here, $c = 3 \cdot 10^8$m/s is the velocity of light.

Lets us further assume that multipath propagation occurs with the length of the shortest (line of sight) and the longest propagation path being $r_1 = 1000$m and $r_2 = 10000$m, respectively. Thus, we are expecting multipath components arriving with delays varying from $r_1/c = 3.3 \cdot 10^{-6} = 3.3\mu$sec to $r_2/c = 33\mu$sec.

Due to the motion, the incident waves will experience a Doppler shift. The maximum Doppler shift induced by the moving transmitter in this case is $f_c \frac{v}{c} = 200$Hz. We can further conclude that in this case the time interval over which this channel might be considered time-invariant is upper bounded by $1/(2 \cdot 200)$Hz$= 2.5$msec.

From this example, we readily see that several different time scales are involved in the channel dynamics. It is reasonable to assume that the propagation delay between the sensors Δ is much smaller than the arrival time differences between different multipath components. Likewise, the time of multipath component arrivals are much smaller than the variation time constraints induced by Doppler effects. This temporal layering of the physics behind multipath propagation can be exploited in modeling the dynamics of the channel.

Let us consider the diagram shown in Fig. 1.4. The lower level in Fig. 1.4 represents

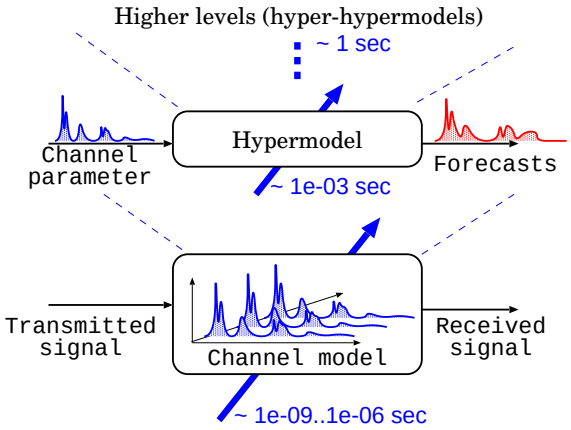


Figure 1.4: Hypermodel approach to modeling the channel dynamics.

the transmission over the wireless channel. As we see from the example, this layer finds itself in the nano and microsecond range. The variation of the channel due to the motion (the source of the Doppler effects) happens on the millisecond time-scale. Thus, there exists a factor of 10^3 more time to learn the dynamics of the arriving reflections in order to use it for mitigating fading. This model that captures this dynamics we call a *hypermodel*.

Clearly, the hypermodel itself is time-varying, since we aim at capturing the local dynamics. The parameters of the hypermodel will also evolve with time. However,

these variations will occur with an even lower rate. In theory, we could have similarly introduced the "hyper-hypermodel", that would govern the variations of the hypermodel. This "hyper-hypermodel" would then reflect changes that happen on the scale of seconds, e.g., when a car drives around a corner of a city block.

The hypermodel can be used to determine the *channel state information* (CSI) – a set of parameters that characterize (up to an application-dependent accuracy) an instantaneous wireless propagation environment. But how do we mitigate the fading assuming we do have the hypermodel? Should the current CSI be known in advance, the transceiver could re-allocate internal resources in a better way or alter the transmission scheme in anticipation of the future conditions. This can be accomplished by forecasting the mobile channel into the future.

Fading mitigation by means of channel prediction has been studied and proved viable in a number of works [Sem03, DH00, HW98, EK99, Ekm02, EDHH98, AJJF99, ADX02, VTR00]. These techniques are used to aid power control and resource allocation [Ekm02, ADX02], downlink diversity and adaptive modulation [DH00, HHDH99].

It is often assumed that fading can be modeled as a deterministic sum of sinusoidal processes. The time variations of the process parameters can be captured with either linear hypermodels (based on autoregressive models) [HW98, VTR00], or nonlinear ones [EK99, Ekm02]. In the latter, the authors treat temporal variations of the CSI as a nonlinear dynamical process.

Once the hypermodel is learned, the predictions are then made by propagating the learned models into the future. These methods are studied for Single-Input-Single-Output (SISO) narrow-band[EDHH98, HW98, HHDH99] as well as for wide-band channels [DXL01, Sem03]. In [ADX02] it has recently been proposed to combine different channels in a smart-antenna system for prediction of the downlink received power. However, the authors only consider the narrowband case.

In general, the proposed strategies differ in the way the hypermodel is built, but they all are very similar in the approach taken to channel prediction (Fig. 1.5).

The channel dynamics is learned from the successive channel observations. Usually, a sampled channel IR is observed for some period. The measured channel IR coefficient sequence is then used to estimate the hypermodel parameters. The diversity of the approaches varies with the structure of the models fitted to the observations of the channel. Once the hypermodel is found, it can be used to forecast the channel taps into the future by simply extrapolating the captured dynamics beyond the observation window. Note that, in mobile environments, the learned hypermodel must be continuously updated.

1.2.2 What can MIMO channels offer?

The approaches considered in the literature so far mostly focus on the analysis of the Single-Input-Single-Output (SISO) channels, thus omitting aspects arising when multiple-sensor antennas are employed. SISO channels are "blind" to directional information. In case of Multiple-Input-Multiple-Output (MIMO) mobile channels,

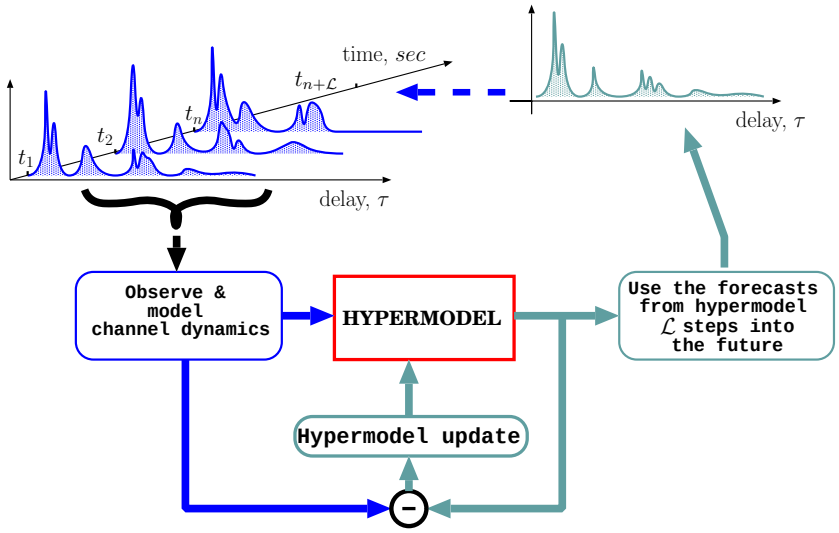


Figure 1.5: Predicting the impulse responses of the wireless channel.

the impulse response contains information about the angular distribution of the incoming and emitted wavefronts. This gives additional degrees of freedom in dealing with channel prediction.

In principle, the MIMO system with F transmitting and P receiving antennas, consists of $F \times P$ SISO channels[1]. Thus, the additional degrees of freedom come at the price of higher dimensional data.

Of course, we can exploit the extra data we gain due to the increased dimensionality of the problem to better estimate the hypermodel parameters. But such approach would completely ignore the rich internal structure offered by MIMO channels. Consider this: in case of a SISO channel, two multipath components can be separated by either their delays or their Doppler frequencies. In the case of a MIMO channel, we can separate the components not only by their delays and Doppler shifts, but also by their Direction-of-Arrival (DoA) and Direction-of-Departure (DoD). Why would we want to do that?

It is known that time-, frequency- and space-selective fading results from interference and temporal variations of multipath components in the corresponding domains. The statistical measures that assess the immunity to fading are known as coherence parameters: coherence time T_{coh}, coherence bandwidth B_{coh}, and space coherence S_{coh} [PNG03] and they tell us how long fading will not affect our system. Increasing coherence parameters means decreasing the effect of fading. The question is how this can be achieved?

[1] As a matter of fact, an $F \times P$ MIMO system is more than simply a collection of $F \times P$ SISO channels, since these individual subchannels are usually dependent

1.2.3 Multipath-based channel prediction

To find an answer to this question, we appeal to the well-known *divide et impera* principle, which translates from Latin to "divide and conquer". The multipath channel contains information that describes how the waves are interfering with each other. It then makes sense to extract the multipath components from the channel and treat each individual multipath component as a separate transmission line. What could that bring us in theory?

First of all, by separating the multipath components arriving at different time instances (i.e., having different delays) we increase the coherence bandwidth B_{coh} of the resulting individual channels. Similarly, by separating the waves arriving from different directions we decrease the angular spread of the resulting subchannels, thus increasing the space coherence S_{coh}. The Doppler bandwidth for individual channels is also decreased, since fewer waves are overlapping, meaning an increased coherence time T_{coh}. All in one, this approach creates parallel "multipath channels", each having better coherence characteristics than their mixture in the channel .

The description of the multipath channel in terms of the multipath components has also one very important consequence. Since each multipath component can be described by a relatively small set of parameters, such decomposition allows to represent the channel very compactly.

In the light of what we propose, the paradigm of channel forecasting is reformulated as follows:

> *Decompose the wireless channel in the contributing multipath components and learn the dynamics of these components. Make prediction of the channel evolution by extrapolating the dynamics of individual components into the future.*

This new paradigm fundamentally differs from the previous approaches. Instead of modelling the dynamics of the measured channel coefficients, the hypermodel is used to model the dynamics of the individual multipath components (Fig. 1.6).

The multipath parameter estimation block in Figure 1.6 plays the role of data compression: the channels representation is reduced to the set of several contributing multipath components, each determined by an N-tuple vector. Note also that this approach can equally be applied to SISO channels as well.

Now, let us outline the channel prediction approach discussed in this thesis.

1.3 Outline of the thesis

The proposed multipath-based channel prediction requires several crucial steps that we discuss in detail in the following chapters.

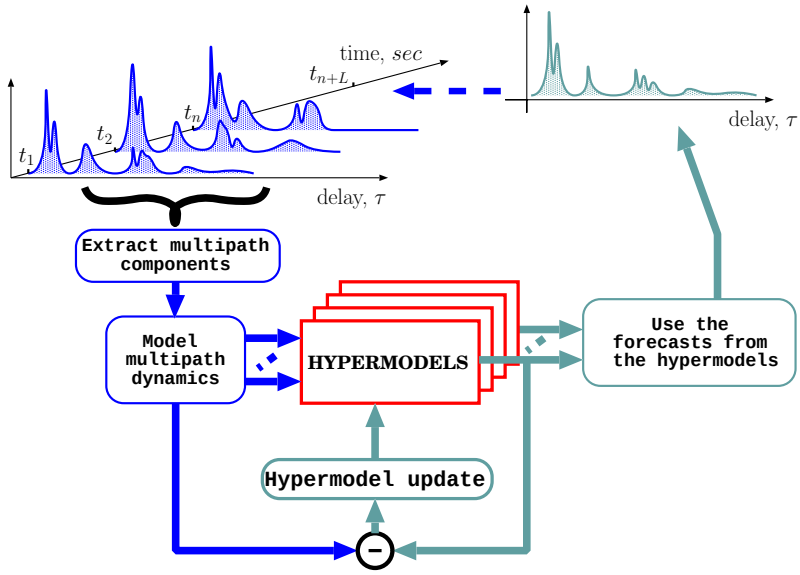


Figure 1.6: Predicting multipath components in the impulse responses of the wireless channel.

Understanding MIMO channels

In Chapter 2 we review some common models of the wireless MIMO channel impulse response. These models will allow us to understand what physical parameters are involved in shaping the dynamics of the multipath components. In order to do so we study simplified scenarios in which the dynamics of the multipath components can be easily analyzed as a function of the number of elements in the antenna array, the velocity of the objects along the propagation path, their geometrical distribution, etc. This study allows to select the proper parametrization for multipath components.

Multipath extraction

In a practical case, however, decomposition of a channel into its contributing wavefronts is not so straightforward. In this work we assume that channel information is obtained using special measurement equipment, also known as channel sounding equipment. The sounding equipment does not deliver any information on how many components are present in the measured channel characteristic, nor does it estimate the multipath components and their parameters. Thus, multipath components must be estimated from the measured data using channel estimation algorithms.

The goal of channel estimation is to extract the multipath components (by estimating the corresponding parameters) from the measured data. In general, estimation of multipath components requires multidimensional optimization, since each component is described by a vector of parameters. In Chapter 3 and Chapter 4 we consider two methods that solve this estimation task. Both methods are somewhat

similar in spirit: they both exploit channel model to find multipath parameters. The first method is known as the SAGE algorithm. The SAGE algorithm is an approximation to the Maximum Likelihood method that allows to replace a multidimensional optimization procedure by a series of one-dimensional optimizations. The SAGE algorithm, however, lacks the ability to find the number of components present in the measurement data, and as a result might estimate wrong components.

The second algorithm considered here is known as the Evidence Procedure. The Evidence Procedure was originally developed in Learning Theory for solving regression problems. We have extended it further to apply it to our problem. Similarly to SAGE it relies on the channel model to solve the estimation problem, however, unlike SAGE, it is developed within the Bayesian framework. The major advantage of the Evidence Procedure is its ability to estimate the model order, i.e., the number of present multipath components, along with the other multipath parameters.

Tracking multipath components

Once the parameters have been estimated, the next step is to reconstruct their dynamical behavior. The used estimation algorithm outputs a set of parameter estimates for each multipath component. Clearly, once we consider channels measured at successive time intervals we obtain such estimates for each interval. However, the estimation algorithm does not tell us how these successive estimates are to be associated over time. In other words, which of the obtained successive estimates correspond to the same physical multipath component. Thus, the estimates must be ordered in time so as to correspond to their respective multipath components.

In Chapter 5 we propose a tracking algorithm that solves this association problem. The algorithm exploits predictive properties of the hypermodel to anticipate the possible track evolution. Dynamic Programming is used to assign the estimates to the physical components to be tracked.

Hypermodel Learning

In Chapter 5 we also propose to associate two sub-hypermodels with each multipath: a *structure hypermodel* that is used for modeling the changes in the track structure and assists tracking, and a *gain hypermodel* that reflect the variations of the multipath gain. This is a simple split of one hypermodel into two sub-structures. For structure hypermodels, which are simpler, we use linear models. To accurately represent the gain hypermodels, we use linear as well as nonlinear structures.

The tracking algorithm we use relies on the existence of the learned hypermodels for individual multipath components. Since initially the hypermodels are not available, we propose to estimate them recursively. In Chapter 5 we propose four recursive algorithms that perform on-line estimation of the hypermodel parameters, for both structure and gain hypermodels.

Predicting multipath components

In Chapter 6 we consider an application of the described methodology to measured wireless MIMO channels, as well as the results of using the learned hypermodels for prediction. We discuss how to select some of the simulation parameters and what effect they have on the resulting prediction performance.

We also discuss how to assess the prediction performance. The major challenge here is the nonstationarity of the prediction error. As we will show, since the hypermodels are constructed on-line, we might expect some transient behavior. Also, in realistic scenarios, multipath components have a finite life time, which means they are constantly appearing and disappearing. This again constitutes a source of nonstationarities in the prediction.

Discussion of the results and conclusions

Finally, in Chapter 7 we will discuss the presented approaches to the channel prediction based on the multipath components and draw some conclusions regarding the proposed approach.

In this chapter we also consider some open issues that should be addressed further as well as possible applications of the proposed prediction methodology in the design of future generations of wireless communication systems.

1.4 Work contributions

Let us list here the scientific contributions that appear in the presented work.

During the beginning of the research work a significant amount of efforts was invested in the development of the channel clustering algorithms. These ideas are reflected in the publications [SG04, Shu04b, Shu04a]. The main goal of these works was to extract the clusters from the measured channel impulse responses for the purpose of modeling and predicting the channel dynamics based on the clusters. Although in the presented work these ideas were not exploited, we definitely think that incorporating clusters in the whole framework is beneficial. Actually, one of the conclusions we made after this work had been done, was that clusters can actually resolve many problems in the tracking algorithms, and can in fact be beneficial for multipath parameter estimation.

We also invested a lot of efforts into the research and development of the multipath estimation algorithms, namely Evidence Procedure and SAGE-RVM discussed in Chapter 4. The development of these ideas were published in [SK04, SF05]. We also prepared and submitted a journal article that summarizes the results on the application of the Evidence Procedure specifically to the estimation of wireless channels [SKF].

Some intermediate tracking and prediction results were also presented at the International Conference on Information, Communications and Signal Processing

(ICICS'05). The paper [SG05], which appeared in the conference proceedings, received the "Best Student Paper Award". The extension of the obtained results, which also summarizes the results we obtained in this work, was recently accepted for publication in the proceedings of the VTC'07 conference.

Chapter 2

Understanding MIMO channels

We begin this chapter with an overview of some common channel models used to describe the terrestrial radio communication channels. In many cases such models are based on the planar wave fronts assumption, constant vehicle velocities, and propagation via LOS, reflectors, and scatterers. These simplifications, though quite realistic, make the analysis of the channel structure analytically tractable. In the following we analyze which physical parameters constitute the wireless channel, and how these parameters vary with time. This analysis will help us understand which parameters are needed to describe a multipath component.

Although propagation of the electormagnetic waves in space is perfectly described by linear differential Maxwell's equations, the dynamics of the channel variation might be nonlinear (for instance in case of accelerated motion).

Linearizing this dynamics is often equivalent to taking the classical planar wave assumption. It will be shown that in the case of planar wave fronts the channel can be described as a weighted sum of the complex exponentials. From linear system theory it is known that such representation can be perfectly extrapolated in time, provided the number of components, as well as their weights and frequencies are known and constant. In Sections 2.2 and 2.3 we derive the corresponding channel representation for Single-Input-Multiple-Output (SIMO) and Multiple-Input-Multiple-Output (MIMO) wireless channels. It will be shown that in those cases each multipath component can be described by a set of parameters that can be estimated and tracked individually.

2.1 Wireless channel impulse response

The small-scale variations of a mobile radio channel can be directly related to the variations of the impulse response of the channel. The multipath channel impulse response (IR) is the key characteristics of the wireless transmission medium, and it contains all information about the local propagation environment necessary to simulate or analyze any type of transmission through the channel.

Generally, a signal $y(t)$ received at an antenna can be modeled as a linear combi-

nation of scaled and shifted versions of an arbitrary transmitted signal $x(t)$, i.e.,

$$y(t) = \sum_{l=1}^{L} a_l(t)x(t - \tau_l(t)). \tag{2.1}$$

In principle, the validity of eq. (2.1) follows from the assumption that the attenuations $a_l(t)$ and the propagation delays $\tau_l(t)$ are frequency-independent. It is a reasonable to assume so when the bandwidth B_x of the transmitted signal is narrow relative to the carrier frequency ω_c.

The linearity of (2.1) allows to introduce the impulse response $h(t, \tau)$ that describes the response of the channel at time instant t to an impulse at instant $t - \tau$:

$$h(t, \tau) = \sum_{l=1}^{L} a_l(t)\delta(\tau - \tau_l(t)), \tag{2.2}$$

where

$$y(t) = \int_0^{\infty} h(t, \tau)x(t - \tau)d\tau. \tag{2.3}$$

Equation (2.2) is a classical passband model of a wireless multipath channel [Mol05, ch. 6].

In a typical wireless application, information transmission occurs in a passband $[\omega_c - 0.5B_x, \omega_c + 0.5B_x]$ centered at a carrier frequency ω_c. Bandlimitation occurs due to multiple factors, such as finite bandwidth of the transceiver hardware, as well as different legal restrictions. Most of the processing (e.g., coding/decoding, modulation/demodulation, etc.) usually takes place at the baseband. Thus, from a communication system design/analysis point of view, it is most useful to consider a *baseband equivalent* description of the system.

It is known [Mol05, Pro95] that the transmitted signal $x(t)$ is related to its baseband description $x_b(t)$ as $x(t) = \mathrm{Re}\{x_b(t)\mathrm{e}^{j\omega_c t}\}$. Similarly, $y(t) = \mathrm{Re}\{y_b(t)\mathrm{e}^{j\omega_c t}\}$, where $y_b(t)$ is the baseband description of the received signal $y(t)$. It then follows that [Pro95, ch. 14]

$$y_b(t) = \sum_{l=1}^{L} a_l(t)x_b(t - \tau_l(t))\mathrm{e}^{-j\omega_c\tau_l(t)}. \tag{2.4}$$

The corresponding baseband equivalent description of the channel $h_b(t, \tau)$ then follows directly from (2.4):

$$h_b(t, \tau) = \sum_{l=1}^{L} a_l(t)\mathrm{e}^{-j\omega_c\tau_l(t)}\delta(\tau - \tau_l(t)). \tag{2.5}$$

Thus, the baseband channel is equivalent to the passband channel $h_p(t, \tau)$ within the bandwidth of the system. Further on we will always assume, unless stated

otherwise, that we are working in the baseband. Thus, we will use $h(t,\tau)$ to denote the baseband channel description.

Since the bandwidth of the system is finite, it is possible to sample the corresponding channel representation. Let $p(t)$ be a time invariant impulse response of the concatenated system consisting of the pulse shaping filter, transceiver RF blocks, and the receiver baseband (matched) filter. Also let the symbol interval be T_c. Then a discrete-time channel impulse response can be described by an FIR-filter with the kth time-varying tap given by [NCP97, Ekm02]

$$\begin{aligned} h(t,k) &= \int_0^{KT_c} p(T_c k-\tau)h(t,\tau)d\tau = \\ &= \sum_{l=1}^{L} p(T_c k-\tau_l(t))a_l(t)\mathrm{e}^{-j\omega_c\tau_l(t)}, \end{aligned} \tag{2.6}$$

where KT_c covers the duration of the continuous-time impulse response in the delay domain τ. However, assuming that $p(\cdot)$ has its effective support on the closed interval $[-\Delta T_c, \Delta T_c]$, the number of reflectors and scatterers contributing to the kth tap will be limited to the path with delays in the interval $[T_c(k-\Delta), T_c(k+\Delta)]$. Note that L may be arbitrary large, but finite. A limited number of contributions is, of course, an advantage when the channel tap is to be predicted – a finite number of contributions allows to process them individually.

In an ideal noiseless and lossless environment the radio waves might interact with objects forever, resulting in paths of unbounded delays, requiring an IIR description of the channel. In practice we can, however, assume that the impulse response $h(t,\tau)$ will be of finite length, as multipaths with large delays are sufficiently attenuated, e.g., through propagation losses and losses at the reflecting/scattering surfaces, to fall below the background noise level.

As we shortly mentioned before, Doppler effects influence the rate of the channel variation. From (2.6) or (2.2) the Doppler effects are not yet explicitly visible. In fact they are 'hidden' in the time-varying delay $\tau_l(t) = r_l(t)/c$, where $c = 3\times10^8$m/s is the speed of light and $r_l(t)$ is the distance traveled by the wave generating the multipath component. Clearly, in the mobile environment the variation of $r_l(t)$ is related to the displacement of the mobile terminal. The latter is exactly the source of the Doppler effects. Later in this chapter, we will see in more details how the velocity of the mobile transceiver influences the variation of the distance $r_l(t)$.

What is missing in the presented channel description is the MIMO aspect. Let us illustrate the impact of multiple antennas based on the SIMO system. We assume a communication system with a single transmit antenna and a receiving antenna array with P elements. Thus, there will be P SISO links between the transmit and the receive antennas, each having an impulse response similar to eq. (2.2). However, each receive antenna will see the impinging waves with a slightly different delay $\tau_{l,p}(t)$, $p = 0,\ldots,P-1$, thus making the impulse response sensor-dependent.

The physical distance from the wave source to each of the antenna sensors can be represented with respect to a reference sensor as $r_{l,p}(t) = r_{l,0}(t) + \Delta r_{l,p}(t)$ (Fig.

2.1). Then, the time-varying impulse response $h_p(t,\tau)$ of the pth baseband channel

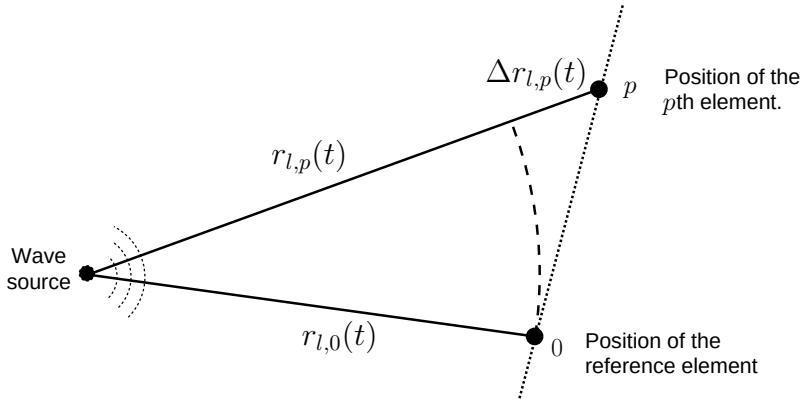


Figure 2.1: The physical distance with a reference sensor.

can be described as:

$$h_p(t,\tau) = \sum_{l=1}^{L} a_l(t) \mathrm{e}^{\left(-j\omega_c \tau_{l,0}(t) - j\omega_c \Delta\tau_{l,p}(t)\right)} \delta(\tau - \tau_{l,0}(t) - \Delta\tau_{l,p}(t)), \qquad (2.7)$$

where $\tau_{l,p}(t) = r_{l,p}(t)/c = \tau_{l,0}(t) + \Delta\tau_{l,p}(t)$ is the corresponding path delay. The term $\Delta\tau_{l,p}(t) = \Delta r_{l,p}(t)/c$ in (2.7) stands for the propagation delay between the reference sensor and the sensor p.

For many practical systems the term $\Delta r_{l,p}(t)/c$ appearing in (2.7) in the argument of the $\delta(\cdot)$ function is sufficiently small and can be safely neglected. This is equivalent to assuming that the wave front reaches all the sensors simultaneously and that all sensors "see" the same received signal. The following example illustrates why such an approximation is reasonable.

Example

Let us consider a signal with an absolute bandwidth of 200MHz. The passband signal is formed by modulating the baseband representation with the carrier frequency $f_c = \omega_c/(2\pi) = 2$GHz.

The fastest variation of the passband signal will occur with the period of the highest harmonics of the baseband representation, i.e., 1/100MHz= 10nsec.

Let us further assume an antenna array with the spacing between the sensors to be $\lambda/2$, where $\lambda = 0.15$m for the assumed carrier frequency. From simple geometrical considerations it follows that $\Delta r_{l,p}(t) \leq \lambda/2$. Thus, the maximum propagation delay between sensors is upper-bounded as $\Delta\tau_{l,p}(t) \leq 0.25$nsec. In other words the propagation time between the sensors is several orders of magnitude smaller than the fastest frequency variation of the transmitted signal.

From this example we readily see that we can safely neglect $\Delta\tau_{l,p}(t)$ as long as the bandwidth of the transmitted signal is small compared to the carrier frequency.

Neglecting the $\Delta r_{l,p}(t)/c$ term in (2.7) results in

$$h_p(t,\tau) = \sum_{l=1}^{L} a_l(t) \mathrm{e}^{\left(-j\omega_c \tau_{l,0}(t) - j\omega_c \Delta\tau_{l,p}(t)\right)} \delta(\tau - \tau_{l,0}(t)). \tag{2.8}$$

Effective source

Let us now consider a single lth multipath component in the representation (2.8). If we base our modeling on ray optics and omit the effects of diffraction and Fresnel optics, a scatterer can be modeled as an effective source induced by a wave front, whereas a reflector generates an effective source as the mirror image of the emitting source [Ekm02]. Thus both scatterers and the mirror source can be viewed as effective sources emitting spherical or cylindrical wavefronts. This enables us to simplify the expression (2.8) by separating "time-invariant" and "time-dependent" factors.

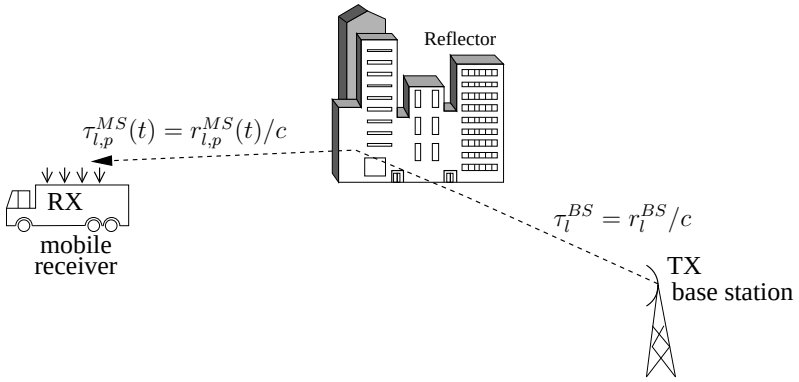


Figure 2.2: Decomposition of the path delay into time-varying and time-invariant parts.

The path delay $\tau_{l,p}(t)$ can be decomposed into the sum of a time-varying delay from the effective source to the mobile $\tau_{l,p}^{MS}(t)$ and a time-invariant and sensor-invariant delay from the base station to the effective source, τ_l^{BS}:

$$\begin{aligned} \tau_{l,p}(t) &= \tau_{l,p}^{MS}(t) + \tau_l^{BS}, \quad \text{and} \\ \tau_{l,p}^{MS}(t) &= \tau_{l,0}^{MS}(t) + \Delta\tau_{l,p}^{MS}(t) \end{aligned} \tag{2.9}$$

For a reflector, the secondary source is the mirrored image of the primary source. Thus, in general τ_l^{BS} will stay almost time invariant, or will vary significantly slower as compared to the change in the path delay from the effective source to the mobile station $\tau_{l,p}^{MS}(t)$. This allows to separate factors changing on different time scales, namely changes due to the fast fading and those attributed to slow fading.

Let us now define

$$\alpha_l(t,\tau) = a_l(t) \mathrm{e}^{-jw_c\tau_l^{BS}} \delta(\tau - \tau_{l,0}(t)), \tag{2.10}$$

and

$$\zeta_{l,p}(t) = -w_c(\tau_{l,0}^{MS}(t) + \Delta\tau_{l,p}^{MS}(t)) = -\kappa r_{l,p}^{MS}(t) = -\kappa\|\boldsymbol{r}_{l,p}^{MS}(t)\| \tag{2.11}$$

In (2.11) $\boldsymbol{r}_l^{MS}(t)$ is a vector in space pointing from the lth effective source to the pth sensor of the mobile antenna and $\kappa = \omega_c/c$ is the wave number. The term $\kappa\|\boldsymbol{r}_{l,p}^{MS}(t)\|$ is also known as the electrical distance [Ekm02]. Now, by combining (2.8) with (2.9), and making use of (2.10) and (2.11), we arrive at the resulting bandpass channel impulse response

$$h_p(t,\tau) = \sum_{l=1}^{L} \alpha_l(t,\tau)\mathrm{e}^{j\zeta_{l,p}(t)}. \tag{2.12}$$

Splitting parameters in this way exemplifies different time-scales of the parameter variations. The major variations of the first term $\alpha_l(t,\tau)$ are attributed to the variations of the magnitude of $a_l(t)$, and thus it accounts for the large-scale fading. The phase term $\zeta_{l,p}(t)$, on the other hand, varies much faster. In fact, the change of $r_{l,p}^{MS}(t)$ by as much as one wavelength causes $\zeta_{l,p}(t)$ to undergo a phase rotation of 2π. As the result, the superposition of the components in (2.12) causes the channel to undergo a small-scale fading.

To gain a deeper understanding of the source of the small-scale fading, we analyze the electrical distance term $-\kappa\|\boldsymbol{r}_{l,p}^{MS}(t)\|$ in more detail as the receiving sensor array moves.

2.2 SIMO channel

Let us consider more closely the electrical distance term $-\kappa\|\boldsymbol{r}_{l,p}^{MS}(t)\|$ that enters the phase in eq. (2.12). In the sequel we drop the superscript notation $(\cdot)^{MS}$ for simplicity. To simplify the derivations we also restrict ourselves to a single effective source and consider a linear sensor array $D(P)$ with P sensors. The extension to other array geometries is straight-forward. We will further assume that the effective source is mobile while the sensor array is fixed[1].

The distance from the lth effective source to the mobile can be defined as (see Figure 2.3):

$$\boldsymbol{r}_{l,p}(t) = \boldsymbol{r}_{l,p}(0) - \boldsymbol{x} = \boldsymbol{r}_{l,0}(0) + \boldsymbol{d}_p - \boldsymbol{x},$$

where $\boldsymbol{x}$ is the displacement of the effective source relative to the origin O and the fixed sensor array. The subscript indices $(\cdot)_{l,p}$ refer to the lth multipath ray received by the pth sensor in the antenna array, respectively. Vector $\boldsymbol{d}_p$ points from a reference sensor $p = 0$ to another sensor p in the array. Using straight-forward

[1]The roles of transmitter and receiver in this setup could always be interchanged due to the reciprocity of the channel.

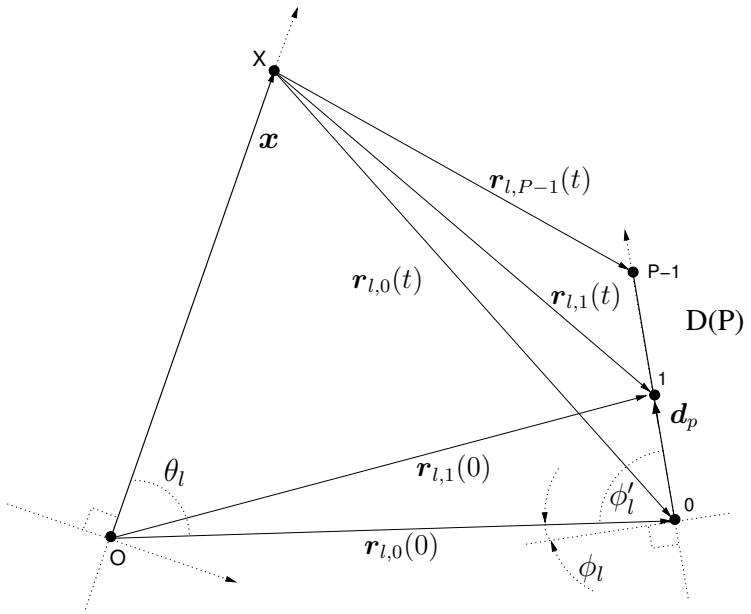


Figure 2.3: Geometrical situation considered in the SIMO case with the moving effective source and P-sensor array $D(P)$.

geometrical rules, the length of the lth path $\boldsymbol{r}_{l,p}(t)$ can be expressed in the following way:

$$\begin{aligned}\|\boldsymbol{r}_{l,p}(t)\| = \Big[&\|\boldsymbol{r}_{l,0}(0)\|^2 + \|\boldsymbol{d}_p\|^2 + \|\boldsymbol{x}\|^2 - \\ &- 2\langle \boldsymbol{r}_{l,0}(0), \boldsymbol{d}_p\rangle - 2\langle \boldsymbol{r}_{l,0}(0), \boldsymbol{x}\rangle - 2\langle \boldsymbol{d}_p, \boldsymbol{x}\rangle\Big]^{1/2}.\end{aligned} \tag{2.13}$$

Now, let us consider eq. (2.13) in more details. To simplify the analysis of the term $\|\boldsymbol{r}_{l,p}(t)\|$ that governs the variation of the electrical distance, we expand the square root of the right-hand side of (2.13) into a second order Taylor series around zero. For the case of linear antenna array this term is given as:

$$\begin{aligned}\|\boldsymbol{r}_{l,p}(t)\| \approx\ &\|\boldsymbol{r}_{l,0}(0)\| + \|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l) - \\ &- \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2 - (\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))^2}{\|\boldsymbol{r}_{l,0}(0)\|} - \\ &- \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2(\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))}{\|\boldsymbol{r}_{l,0}(0)\|^2} - \frac{1}{8}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^4}{\|\boldsymbol{r}_{l,0}(0)\|^3}.\end{aligned} \tag{2.14}$$

The details of this expansion are summarized in Appendix A. In (2.14) ϕ_l is the angle of incidence (Direction-of-Arrival), and θ_l is the direction of the effective source movement, as shown in Fig. 2.3.

There are several important observations that can be made based on (2.14). First of all, we see that $\|\boldsymbol{r}_{l,p}(t)\|$ depends nonlinearly on the displacement vector $\boldsymbol{x}$. In particular, the higher order terms in the expansion exemplify the dependency of $\|\boldsymbol{r}_{l,p}(t)\|$

on different setup parameters, like angles, displacement vector $\boldsymbol{x}$, etc. However, the linear terms in the expansion are straightforward to analyze.

If the initial distance $\|\boldsymbol{r}_{l,0}(0)\|$ is much larger than the antenna array dimension (represented by $\|\boldsymbol{d}_p\|$) and the traveled distance $\|\boldsymbol{x}\|$, then higher-order terms in the Taylor approximation can be safely discarded. This assumption will culminate in the widely used plane wave assumption, since all vectors $\boldsymbol{r}_{l,p}(t)$ for $p = 0 \dots P-1$ can then be assumed to be co-linear. Thus, the simplified expression for the path distance is then computed as

$$\|\boldsymbol{r}_{l,p}(t)\| \approx \|\boldsymbol{r}_{l,0}(0)\| + \|\boldsymbol{d}_p\| \sin(\phi_l) - \|\boldsymbol{x}\| \cos(\theta_l).$$

The latter expression allows us to approximate the phase term $\zeta_{l,p}(t)$ in (2.12) as

$$\zeta_{l,p}(t) \approx -\kappa\Big(\|\boldsymbol{r}_{l,0}(0)\| + \|\boldsymbol{d}_p\| \sin(\phi_l) - \|\boldsymbol{x}\| \cos(\theta_l)\Big). \tag{2.15}$$

To make this result a bit more pictorial, we assume that the wave source is moving with a constant velocity v m/s, and thus $\|\boldsymbol{x}\| = vt$. Also, for a linear sensor array $\|\boldsymbol{d}_p\| = pd$, where d is the distance between sensors. By noting that $\kappa = \omega_c/c = 2\pi/\lambda$, (2.15) can be represented as

$$\zeta_{l,p}(t) \approx -\frac{2\pi}{\lambda}\|\boldsymbol{r}_{l,0}(0)\| - \frac{2\pi}{\lambda} pd \sin(\phi_l) + \frac{2\pi}{\lambda} vt \cos(\theta_l). \tag{2.16}$$

The complete linearized representation of the multipath channel can be thus represented as

$$h_p(t,\tau) = \sum_{l=1}^{L} \alpha_l(t,\tau) \mathrm{e}^{-j\frac{2\pi}{\lambda}\|\boldsymbol{r}_{l,0}(0)\|} \mathrm{e}^{-j2\pi\frac{d}{\lambda}p\sin(\phi_l)} \mathrm{e}^{j2\pi\nu_l t}, \tag{2.17}$$

where $\nu_l = v\cos(\theta_l)/\lambda$ is the Doppler shift induced by the lth moving source. Notice that $\frac{2\pi}{\lambda} pd \sin(\phi_l)$ is a phase shift across sensors due to the nonzero angle of incidence.

It is clear, that in the ideal case (where the plane wave assumption is valid, i.e., (2.17) holds), and the Doppler shifts as well as the corresponding angles of incidence are known, channel prediction is equivalent to the extrapolation of (2.17) into the future.

2.3 MIMO channel representation

Similarly, the analysis done for SIMO systems in Section 2.2 can be performed for MIMO systems. The corresponding propagation scenario is depicted in Figure 2.4. Let M denote the number of transmit elements. Generally, a MIMO scenario is equivalent to M individual SIMO (MISO) cases. The lower indices l, m, p refer to the lth wave travelling between the mth element of the transmit array $F(M)$, $m = 0 \dots M-1$, and pth element of the receive array $D(P)$, $p = 0 \dots P-1$, respectively.

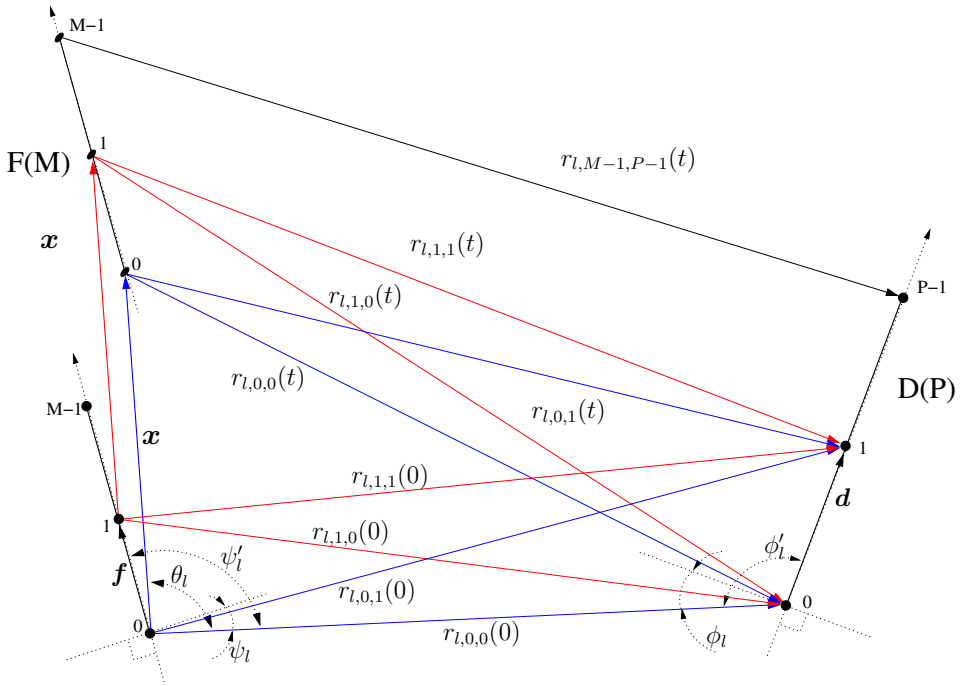


Figure 2.4: Geometrical situation considered in the MIMO case with the moving effective source.

The corresponding distances between sensors are expressed as

$$\begin{aligned}
\boldsymbol{r}_{l,0,0}(t) &= \boldsymbol{r}_{l,0,0}(0) - \boldsymbol{x}, \\
\boldsymbol{r}_{l,0,p}(t) &= \boldsymbol{r}_{l,0,p}(0) - \boldsymbol{x} = \boldsymbol{r}_{l,0,0}(0) + \boldsymbol{d}_p - \boldsymbol{x}, \\
\boldsymbol{r}_{l,m,p}(t) &= \boldsymbol{r}_{l,m,p}(0) - \boldsymbol{x} = \boldsymbol{r}_{l,m,0}(0) + \boldsymbol{d}_p - \boldsymbol{x} = \boldsymbol{r}_{l,0,0}(0) - \boldsymbol{f}_m + \boldsymbol{d}_p - \boldsymbol{x}.
\end{aligned}$$

Similarly to the SIMO case, we consider the expansion of the $\|\boldsymbol{r}_{l,m,p}(t)\|$ into the Taylor series to make the analysis of the resulting electrical distance tractable. For the details of computing the Taylor expansion in this case the reader is referred to Appendix B. The final expression is given as

$$\begin{aligned}
\|\boldsymbol{r}_{l,m,p}(t)\| \approx \|\boldsymbol{r}_{l,0,0}(0)\| \Bigg(& 1 + \frac{\langle \boldsymbol{r}_{l,0,0}(0), \boldsymbol{d}_p \rangle}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \frac{\|\boldsymbol{d}_p\|^2}{2\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \frac{\|\boldsymbol{x}\|^2}{2\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \\
& + \frac{\|\boldsymbol{f}_m\|^2}{2\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \frac{\langle \boldsymbol{x}, \boldsymbol{f}_m \rangle}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} - \frac{\langle \boldsymbol{r}_{l,0,0}(0), \boldsymbol{f}_m \rangle}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} - \frac{\langle \boldsymbol{r}_{l,0,0}(0), \boldsymbol{x} \rangle}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} - \\
& - \frac{\langle \boldsymbol{f}_m, \boldsymbol{d}_p \rangle}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} - \frac{\langle \boldsymbol{x}, \boldsymbol{d}_p \rangle}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} \Bigg).
\end{aligned} \tag{2.18}$$

As compared to (2.14) we see a lot of similarities. Here again $\|\boldsymbol{r}_{l,m,p}(t)\|$ depends nonlinearly on the angles and array parameters (see (B.3) in Appendix B). As the initial separation $\|\boldsymbol{r}_{l,0,0}(0)\|$ between the antenna arrays grows, it drives the higher

order terms of the Taylor expansion to zero, thus culminating in the plane wave scenario. In the limit, as $\|\boldsymbol{r}_{l,0,0}(0)\| \to \infty$, all the paths between $F(M)$ and $D(P)$ become parallel to each other, and the resulting linear approximation takes the form

$$\|\boldsymbol{r}_{l,m,p}(t)\| = \|\boldsymbol{r}_{l,0,0}(0)\| + \|\boldsymbol{d}_p\| \sin(\phi_l) - \|\boldsymbol{f}_m\| \sin(\psi_l) - \|\boldsymbol{x}\| \cos(\theta_l).$$

Here ψ_l is the Direction-of-Departure (DoD), and ϕ_l and θ_l are again the DoA and the direction of array movement, respectively.

Consequently, the corresponding electrical distance $\zeta_{l,m,p}(t)$, which now depends on both on the transmit sensor m and the receive sensor p, can be approximated as

$$\zeta_{l,m,p}(t) \approx -\kappa(\|\boldsymbol{r}_{l,0,0}(0)\| + \|\boldsymbol{d}_p\| \sin(\phi_l) - \|\boldsymbol{f}_m\| \sin(\psi_l) - \|\boldsymbol{x}\| \cos(\theta_l)). \quad (2.19)$$

Again, assuming a constant velocity v m/s for the mobile terminal and movements without rotation or acceleration, we write $\|\boldsymbol{x}\| = vt$. Also, for a linear sensor array $\|\boldsymbol{d}_p\| = pd$, and $\|\boldsymbol{f}_m\| = mf$, where d and f are the distances between neighboring elements in the receive and transmit arrays, respectively. By noting, that $\kappa = 2\pi/\lambda$ we rewrite $\zeta_{l,m,p}(t)$ as

$$\zeta_{l,m,p}(t) \approx -\frac{2\pi}{\lambda}\|\boldsymbol{r}_{l,0,0}(0)\| - \frac{2\pi}{\lambda} pd \sin(\phi_l) + \frac{2\pi}{\lambda} mf \sin(\psi_l) + \frac{2\pi}{\lambda} vt \cos(\theta_l), \quad (2.20)$$

The complete linearized representation of the MIMO channel with linear sensor arrays on both sides can thus be approximated as

$$h_{m,p}(t,\tau) = \sum_{l=1}^{L} \alpha_l(t) \mathrm{e}^{-j\frac{2\pi}{\lambda}\|\boldsymbol{r}_{l,0,0}(0)\|} \mathrm{e}^{-jp2\pi\frac{d}{\lambda}\sin(\phi_l)} \mathrm{e}^{jm2\pi\frac{f}{\lambda}\sin(\psi_l)} \mathrm{e}^{j2\pi\nu_l t}, \quad (2.21)$$

where $v\cos(\theta_l)/\lambda = \nu_l$ is Doppler shift induced by the lth moving source, and $\frac{2\pi}{\lambda} pd \sin(\phi_l)$ and $\frac{2\pi}{\lambda} mf \sin(\psi_l)$ are phase shifts across sensors due to the nonzero angle of incidence and angle of departure, respectively. Again, should we know all the required parameters, the dynamics of the channel can be accordingly extrapolated into the future, at least as long as the plane wave assumption holds.

2.4 Discussion

Let us summarize the results we obtained so far. The whole analysis we have presented here is based on a few fundamental concepts.

- First of all, it is assumed that the received signal $y(t)$ is a linear combination of the scaled and delayed versions of the transmitted signal $x(t)$. It is exactly this superposition that causes the received signal to undergo fading.

- The linear dependency between the input and the output of the channel allows us to introduce a time-varying impulse response $h(t,\tau)$ that describes how the copies of $x(t)$ are dispersed in time and how they interfere.

- A MIMO wireless channel impulse response is given by individual impulse responses between each transmitting and each receiving sensor in the antenna arrays.

The channel impulse response is thus a crucial concept in the whole analysis. It is the key to counteracting fading, since it contains all the information of the arriving multipath components and their variation. In order to study what constitutes the dynamics of multipath components, we study a single component in a simplified scenario, assuming linear antenna arrays, and straight-line motion. The following observations can be made:

- The dynamics of a single multipath component is respresented by the interaction between the induced Doppler shift, governed by the $v\cos(\theta_l)$ term, DoA and DoD, captured in the $\sin(\phi_l)$ and $\sin(\psi_l)$ terms, respectively. In general this interaction is nonlinear and, as the result, the above parameters contribute nonlinearly into the change of the corresponding electrical distance terms $\zeta_{l,p}(t)$ and $\zeta_{l,m,p}(t)$.

- A linear approximation to the multipath dynamics is possible, resulting in the plane wave assumption. It naturally follows when the physical distance between the transmit and receive arrays grows much larger than their characteristic dimension and displacement length $\|\boldsymbol{x}\|$.

- In the linear approximation, the multipath parameters contribute linearly to the change of the electrical distance $\zeta_{l,m,p}(t)$. This simplifies the dynamics of the channel significantly and motivates the application of the linear models to represent the channel dynamics.

- In general, all multipath parameters are functions of time. Basically, any curved motion can be approximated by a series of linear displacements $\boldsymbol{x}_i$, as shown in Fig. 2.5. Clearly, this makes DoA, DoD, as well as Doppler frequency

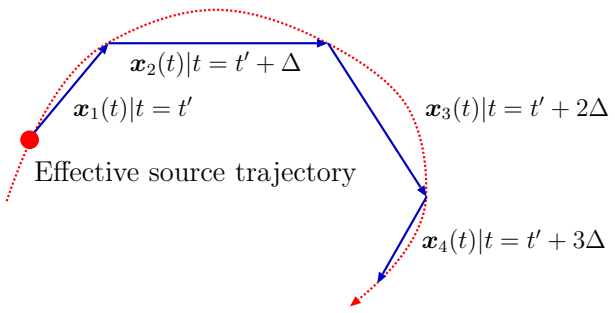


Figure 2.5: Approximation of the effective source trajectory by a sequence of linear displacements.

to be a function of i, which immediately translates into their dependency on time t as $\Delta \to 0$. The rate of these variations depends on the proximity to

the effective source, complexity of movement, i.e., curved/linear trajectory, constant velocity or movement with acceleration, etc.

- It should be stressed that the linear approximation can also be used in the non-plane wave case. However, by doing so we introduce an irreducible error in the representation of the multipath component and, as the consequence, in the true underlying dynamics of this component. We expect that such errors might necessitate faster change of the hypermodel parametrization, thus requiring hypermodels to be more agile.

Now, as we see how the multipath parameters influence the dynamics of the multipath components, we can develop algorithms to estimate these parameters from channel measurement data and, using hypermodels, to learn the underlying dynamics.

Chapter 3

MIMO channel estimation

In the previous chapter we considered a model of a multipath wireless channel. Prior to discussing how to estimate multipath components from the channel data, we need to answer the question how these channels are to be measured.

To be able to verify the performance of the multipath-based prediction proposed here, it would be best to use channel data collected with channel sounding equipment. The major advantage of the resulting channel responses is their high resolution. In Section 3.1 we give an overview of the most common channel sounding approaches, namely sounding in the time and frequency domains.

The resulting channel representations are then used to estimate multipath parameters by exploiting channel models we considered in Chapter 2. In our work we use two algorithms to estimate multipath parameters: the first one is known as Space-Alternating Generalized Expectation-maximization (SAGE) algorithm, discussed in Section 3.2. The other algorithm is known as the Evidence Procedure. Like SAGE it is a model-based parameter estimation technique, but unlike SAGE, the Evidence Procedure also allows to estimate the number of multipath components. We develop and apply this algorithm to the estimation of multipath parameters in Chapter 4.

3.1 Channel sounding

The goals of channel sounding are manifold and include: obtaining high-resolution channel characteristics for constructing realistic channel models, studying particular propagation environments for positioning base-stations, etc. Channel sounding usually consists of two steps: 1) sending a specific sounding/training signal $s(t)$ through the channel, and 2) measuring the response $y(t)$ at the other end of the transmission line. Depending on the particular sounding method, the obtained signal $y(t)$ might also be filtered with a specific receive filter, or matched filter (MF). The output signal ($y(t)$, or MF output) is then later used as the input data for the multipath parameter estimation algorithms.

In the sequel we give a short overview of the two sounding methods and the corresponding signal models that are used to obtain the measured channel data we exploit in our work. In [Mol05, ch. 8] an interested reader can find more details on different channel sounding methods, including those we summarize here. For

simplicity, we will consider SIMO setups. In case of MISO, as well as SISO and MIMO systems, the concepts of channel sounding remain unchanged.

3.1.1 Channel sounding using pulse-compression techniques

Let us consider an equivalent baseband channel sounding scheme shown in Fig. 3.1. The transmitter (Tx) emits a sounding signal $s(t)$ (Fig. 3.2) that consists of periodically repeated burst waveforms $u(t)$, i.e., $s(t) = \sum_{i=0}^{I-1} u(t - iT_f)$, where $u(t)$ has duration $T_u \leq T_f$ and is formed as $u(t) = \sum_{m=0}^{M-1} b_m p(t - mT_p)$. The sequence $b_0 \dots b_{M-1}$ is the known sounding sequence consisting of M chips, and $p(t)$ is the shaping pulse of duration T_p, $MT_p = T_u$. Furthermore, we assume that the receiver

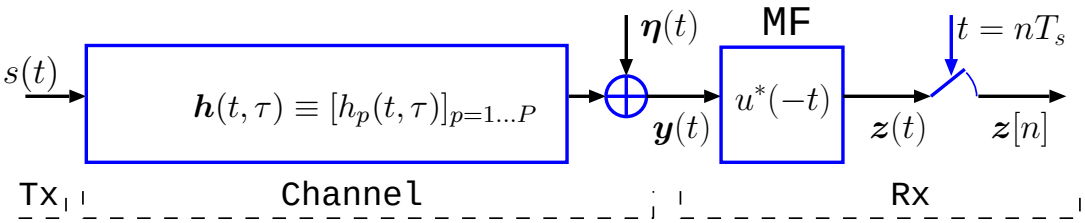


Figure 3.1: An equivalent baseband model of radio channel sounding with receiver matched filter (MF) front-end.

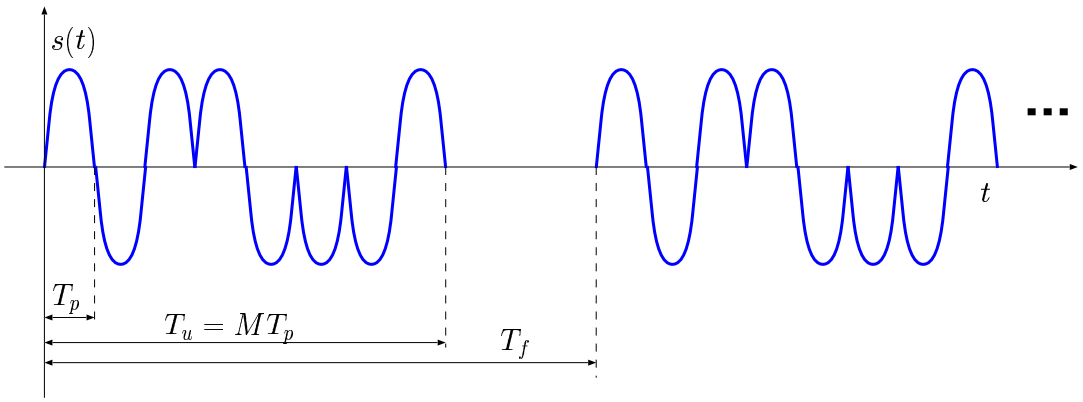


Figure 3.2: Sounding signal $s(t)$.

(Rx) is equipped with a planar antenna array consisting of P sensors. Thus, there are P SISO channels $h_p(t, \tau)$, which we collect into a time-varying P-component vector $\boldsymbol{h}(t, \tau)$.

The received signal $\boldsymbol{y}(t) \in \mathbb{C}^{P \times 1}$ is measured over the observation interval

$$\mathcal{O} = \bigcup_{i=0}^{I-1} O_i = \bigcup_{i=0}^{I-1} \left[\left(i - \frac{I-1}{2} \right) T_f - T_u/2, \left(i - \frac{I-1}{2} \right) T_f + T_u/2 \right]$$

that consists of I periods of the burst waveform $u(t)$. We will generally assume that the multipath parametrization stays time-invariant over the observation window $\mathcal{O}$. For a single burst of duration T_u, the received signal $y(t)$ is simply computed as

$$\boldsymbol{y}(t) = \int \boldsymbol{h}(t,\tau)u(t-\tau)d\tau.$$

The receiver front-end consists of a matched filter (MF) matched to the transmitted burst waveform $u(t)$. The output $\boldsymbol{z}(t) \in \mathbb{C}^{P\times 1}$ of the MF to a single burst input in the interval $\mathcal{O}_i$ the is then given as

$$\boldsymbol{z}(t) = u^*(-t) \star \int \boldsymbol{h}(t,\tau)u(t-\tau)d\tau = \int\int \boldsymbol{h}(t+t',\tau)u(t+t'-\tau)u^*(t')dt'd\tau, \tag{3.1}$$

where $\star$ denotes the convolution operation. If within the measurement interval $\mathcal{O}_i$ it can be assumed that $\boldsymbol{h}(t+t',\tau) = \boldsymbol{h}(t,\tau)$, then (3.1) simplifies to

$$\begin{aligned}\boldsymbol{z}(t) = \int \boldsymbol{h}(t,\tau) \int u(t+t'-\tau)u^*(t')dt'd\tau = \\ \int \boldsymbol{h}(t,\tau)R_{uu}(t-\tau)d\tau,\end{aligned} \tag{3.2}$$

where $R_{uu}(t) = \int u^*(t')u(t+t')dt'$ is the autocorrelation function of the burst waveform $u(t)$.

The sounding burst $u(t)$ is specifically designed so as to make sure that $R_{uu}(t)$ closely approximates a delta pulse. This in turn ensures that (3.2) closely approximates the channel impulse response $\boldsymbol{h}(t,\tau)$.

Clearly, the longer the observation window $\mathcal{O}$ the better parameter estimates we expect. However, due to the time-varying nature of the channel the interval $\mathcal{O}$ can not be infinite. By the Sampling Theorem we known that $1/T_f$ must be at least two times larger than the maximum occurring Doppler frequency. By setting $T_f > T_u$ we can also increase the observation span, and thus improve the Doppler resolution, while in the same time limit the amount of measured data to be stored. This is possible since the absolute Doppler frequency of the impinging waves is considerably smaller than the inverse of the burst duration T_u (why is that we show later in Section 3.1.3 when we talk about the resulting channel model). On the other hand, I must be chosen in such a way that over the observation window $\mathcal{O}$ the multipath parametrization stays time-invariant. The choice of I (for a fixed T_f) essentially upperbounds the observation window $\mathcal{O}$.

3.1.2 Frequency domain channel sounding

Due to the dual relationship between time and frequency it is possible to perform similar channel measurement in the frequency domain [Rap02, Mol05].

The difference between the frequency and time domain measurements lie in the form of the burst signal $u(t)$. In frequency domain sounding the main criterion for its design is the that it has the flat power spectrum $|U(j\omega)|^2$ over the frequency band of interest $[f_0, f_0 + \Delta f]$.

One method of frequency domain sounding is based on chipring. The transmit waveform is given as [Mol05, ch. 8]

$$u(t) = \exp\Big[j2\pi\Big(f_0 t + \Delta f \frac{t^2}{2T_u}\Big)\Big], \quad 0 \le t \le T_u.$$

The instantaneous frequency changes linearly with time, covering the whole bandwidth of interest $[f_0, f_0 + \Delta f]$. The receiver filter is again a matched filter with the frequency response $U^*(j\omega)$. Thus, the frequency response of the MF output over the frequency range $[f_0, f_0 + \Delta f]$ for $t \in \mathcal{O}_i$ is readily given as

$$Z(j\omega) = H_p(t, j\omega)U(j\omega)U^*(j\omega) = H_p(t, j\omega) \times \text{const}$$

This method is also known as the frequency domain correlation processing. Again, it is assumed that the channel $H_p(t, j\omega)$, $p = 0 \dots P-1$ stays time-invariant over the measurement window $t \in O_i$.

Since it makes sense to consider only the channel bandwith equal to or smaller than that where $|U(j\omega)|^2$ is constant, the MF output is bandlimited with a receive filter $R(j\omega)$, resulting in the signal

$$z_p(t) = \mathrm{FT}^{-1}\{H_p(t, j\omega)R(j\omega) \times \text{const}\} = \int h_p(t, \tau)r(t - \tau)d\tau$$

received at a single antenna element p. We see that $r(t)$ is equivalent to the autocorrelation function $R_{uu}(t)$ of the burst waveform in equ. (3.2).

3.1.3 Signal model in a plane waves scenario

Now we are ready to introduce the model that is exploited in the parameter estimation algorithms. To do that, we restrict ourselves to the SIMO case and review the plane wave channel model (2.17) in the light of the channel sounding considered above.

We will assume that the receiver (Rx) is equipped with an antenna array consisting of P sensors located at $\boldsymbol{x}_0, \dots, \boldsymbol{x}_{P-1} \in \mathbb{R}^2$ with respect an arbitrary reference point. We also assume this array to be linear, with the spacing between the antenna elements equal d. Provided the electromagnetic coupling between the antenna elements can be neglected, the components of the P-dimensional complex vector $\boldsymbol{c}(\phi_l) = [c_0(\phi_l), \dots, c_{P-1}(\phi_l)]^T$, also known as the steering vector of the array, are defined as

$$c_p(\phi_l) = f_p(\phi_l) \exp(j2\pi\lambda^{-1}\langle \boldsymbol{e}(\phi_l), \boldsymbol{x}_p\rangle), \tag{3.3}$$

with λ, $\boldsymbol{e}(\phi_l)$ and $f_p(\phi_l)$ denoting the wavelength, the unit vector in $\mathbb{R}^2$ pointing in the direction determined by ϕ_l and the complex electric field pattern of the pth sensor, respectively.

Further, let us assume that $\boldsymbol{h}(t,\tau) = \boldsymbol{h}(\tau)$ over the observation interval $\mathcal{O}_i$. Now, by combining (2.10) and (2.17), we note that

$$\delta(\tau - \tau_l) \equiv \delta(\tau - \tau_{l,0}(t)), \tag{3.4}$$

$$\tilde{a}_l \equiv a_l \mathrm{e}^{-j\omega_c \tau_l^{BS}} \mathrm{e}^{-j\frac{2\pi}{\lambda}\|\boldsymbol{r}_{l,0}(0)\|}, \tag{3.5}$$

$$c_p(\phi_l) \equiv \mathrm{e}^{-j2\pi\frac{d}{\lambda}p\sin(\phi_l)}, \tag{3.6}$$

where (3.6) follows immediately for the linear antenna array and constant electric field pattern[1] $f_p(\phi_l)$, i.e., for $f_p(\phi_l) = \text{const}$, $\forall p, \phi_l$.

Under these assumptions, a model of the impulse response of the wireless SIMO channel over the measurement interval $\mathcal{O}_i$ can be represented as

$$\boldsymbol{h}(\tau) = \sum_{l=1}^{L} \tilde{a}_l \boldsymbol{c}(\phi_l) \mathrm{e}^{j2\pi\nu_l t} \delta(\tau - \tau_l). \tag{3.7}$$

Here, $\tilde{a}_l$, τ_l and ν_l are the compound complex gain as defined in (3.5), the delay, and the Doppler shift of the lth multipath component, respectively. In the following text we will denote the compound gain of the multipath component as a_l, rather than $\tilde{a}_l$, to simplify the notations.

Note that although we explicitly specify the dependency on the Doppler shift ν_l, it is reasonable to assume that for $\mathrm{e}^{j2\pi\nu_l t} = \text{const}$ for $t \in \mathcal{O}_i$, which also follows from the assumption of channel time-invariance. In many cases it is possible to assume that the maximum absolute Doppler frequency of the impinging waves is much smaller than the inverse of a single burst duration $1/T_u$. Let us consider the following example.

Example

Taking the pulse compression channel sounding as an example, let us assume that the shaping pulse width equals $T_p = 10\text{nsec}$ and that the burst waveform consists of $M = 512$ pulses. Then, $T_u = 5.12\mu\text{sec}$, and thus $1/5.12\mu\text{sec}$=195312.5Hz.

Assuming the carrier frequency $f_c = \omega_c/2\pi = 2\text{GHz}$, we easily conclude that the velocity of an object generating such a Doppler shift must be $195312.5\text{Hz} \times 3 \cdot 10^8 \frac{\text{m}}{\text{s}} / 2 \cdot 10^9\text{Hz} \approx 29296\text{m/s}$, which is more than two times larger than the Earth escape velocity.

This low Doppler frequency assumption is equivalent to assuming that, within a single observation window O_i we can safely neglect the influence of the Doppler shifts.

[1] This is implicitly assumed in the Chapter 2, since the sensor field pattern is consumed in the factor $a_l(t)$ in (2.10), which is "sensor and direction independent", or, equivalently, constant for all sensors and directions ϕ. Accounting for this factor explicitly leads to the definition of the steering vector given in (3.3)

The resulting received signal $\boldsymbol{y}(t) \in \mathbb{C}^{P\times 1}$ is then given in the time domain as [FTH+99]

$$\boldsymbol{y}(t) = \sum_{l=1}^{L} a_l \boldsymbol{c}(\phi_l) \mathrm{e}^{j2\pi\nu_l t} u(t-\tau_l) + \boldsymbol{\eta}(t). \tag{3.8}$$

The additive term $\boldsymbol{\eta}(t) \in \mathbb{C}^{P\times 1}$ is a vector-valued complex white Gaussian noise process, i.e., the components of $\boldsymbol{\eta}(t)$ are independent complex Gaussian processes with double-sided spectral density N_0.

The low Doppler frequency assumption, however, does not prohibit the estimation of the Doppler frequencies. Assuming that $1/T_f > 2\max_l\{\nu_l\}$, which is dictated by the Sampling Theorem, we can approximate the Doppler shift over the observation interval $\mathcal{O}$ consisting of I periods of the sounding signal $u(t)$ as

$$\exp(j2\pi\nu_l t) \approx \exp\left(j2\pi(i - \frac{I-1}{2})\nu_l T_f\right) \tag{3.9}$$

for t in the time interval $O_i \subset \mathcal{O}$, $i = 0 \ldots I-1$. Taking approximation (3.9) into account, the signal $\boldsymbol{z}(t)$ at the output of the MF for the time $t \in O_i$ is computed as

$$\boldsymbol{z}(t)|_{t\in O_i} = \sum_{l=1}^{L} a_l \boldsymbol{c}(\phi_l) \exp\left(j2\pi(i - \frac{I-1}{2})\nu_l T_f\right) R_{uu}(t-\tau_l) + \boldsymbol{\xi}(t), \tag{3.10}$$

where $\boldsymbol{\xi}(t) = \int \boldsymbol{\eta}^*(t')u(t+t')dt'$ is a spatially white (i.e., uncorrelated) P-dimensional vector with each element being a zero-mean wide-sense stationary (WSS) Gaussian noise with autocorrelation function

$$\begin{aligned} R_{\xi\xi}(t) =& E\{\xi_p^*(t')\xi_p(t+t')\} = N_0 R_{uu}(t), \quad \text{and} \\ & E\{\xi_p(t')\xi_p(t+t')\} = 0. \end{aligned} \tag{3.11}$$

Equation (3.10) states that the channel response is a linear combination of L scaled and delayed kernel functions $R_{uu}(t-\tau_l)$, weighted across sensors as given by the components of $\boldsymbol{c}(\phi_l)$, and across time according to the Doppler frequency ν_l, and observed in the presence of the colored noise $\boldsymbol{\xi}(t)$.

The model of the received signal (3.10) can be used for different model-based channel estimation algorithms. In general, the channel estimation problem is posed as follows: given the measured signals $\boldsymbol{z}_p(t), p = 0, \ldots, P-1$, determine the order L of the model and estimate optimally (with respect to some quality criterion) all multipath parameters a_l, τ_l, ν_l ,and ϕ_l, for $l = 1 \ldots L$.

Should $R_{uu}(t)$ be an ideal delta impulse, the estimation of channel parameters would be a relatively simple task due to the sparse structure of the channel IR. However, a "non-ideal" form of $R_{uu}(t)$ and additive noise at the receive antenna necessitates the usage of high-resolution algorithms [Mol05, ch. 8] able to estimate multipath parameters.

We also would like to add that in the case of frequency-domain sounding, the resulting mathematical description of the channel model for the plane wave case will be functionally identical to the results obtained for the time-domain sounding due to the dual relationship between the time and frequency domains.

3.1.4 Sampling wireless channels

As it was shortly mentioned in Chapter 2, the wireless channel is bandlimited. Thus, it is possible to represent (3.10) by discrete samples.

In practice the output of the MF is sampled with the sampling period $T_s \leq T_p$, resulting in N P-tuples of the MF output, where N is the number of MF output samples. This means that for the time duration $t \in O_i$, the output of each sensor can be collected into a vector and can be rewritten (3.10) in a vector form:

$$\boldsymbol{z}_p|_{t\in O_i} = \boldsymbol{K}\boldsymbol{w}_p + \boldsymbol{\xi}_p, \quad p = 0 \dots P-1, \tag{3.12}$$

where we have defined

$$\begin{aligned} \boldsymbol{z}_p =& [z_p[0], z_p[1], \dots, z_p[N-1]]^T, \\ \boldsymbol{w}_p =& [a_1 c_p(\phi_1) \mathrm{e}^{j2\pi(i-\frac{I-1}{2})\nu_1 T_f}, \dots, a_L c_p(\phi_L) \mathrm{e}^{j2\pi(i-\frac{I-1}{2})\nu_L T_f}]^T, \\ \boldsymbol{\xi}_p =& [\xi_p[0], \xi_p[1], \dots, \xi_p[N-1]]^T. \end{aligned} \tag{3.13}$$

The additive noise vector $\boldsymbol{\xi}_p$ possesses some useful properties that can be exploited in different estimation algorithms:

$$E\{\boldsymbol{\xi}_p\} = \boldsymbol{0}, E\{\boldsymbol{\xi}_m \boldsymbol{\xi}_k^H\} = \boldsymbol{0}, \text{for } m \neq k, \text{ and} \tag{3.14}$$

$$E\{\boldsymbol{\xi}_p \boldsymbol{\xi}_p^H\} = \boldsymbol{\Sigma} = N_0 \boldsymbol{\Lambda}, \text{where } \Lambda_{q,k} = R_{uu}((q-k)T_s). \tag{3.15}$$

Here $E\{\cdot\}$ denotes the expectation operator. Note that (3.15) follows directly from (3.11). The matrix $\boldsymbol{K}$, also called the design matrix, accumulates the shifted and sampled versions of the kernel function $R_{uu}(t)$. It is constructed as follows: $\boldsymbol{K} = [\boldsymbol{r}_1, \dots, \boldsymbol{r}_L]$, with $\boldsymbol{r}_l = [R_{uu}(-\tau_l), R_{uu}(T_s - \tau_l), \dots, R_{uu}((N-1)T_s - \tau_l)]^T$.

It important to stress that this sampled representation is valid assuming: a) the Doppler frequency can be approximated as in (3.9), and b) the multipath parameters stay time-invariant over the time interval $\mathcal{O}$.

Thus, the time-varying multipath channel $h_p(t, \tau)$ is represented by discrete samples, spaced equidistantly along the delay τ, $\tau_n = nT_s$, $n = 0, \dots, N-1$, and by equidistant discrete samples in time t, with the spacing equal to the repetition period T_f of the sounding waveform, i.e., $t_i = iT_f$, $i = 0, \dots, I-1$, as shown in Fig. 3.3.

Multipath parameters are usually estimated over a window consisting of I periods of the burst waveform $u(t)$. The resulting duration of the estimation window is thus $(I-1)T_f + T_u$. In order to be able to capture the dynamics of the parameter variations, the channel measurements and parameter estimation are repeated with the period $T_e \geq IT_f$. We can thus say that the new SIMO channel representation is obtained at $t_q = qT_e$, where $q = 0, 1, 2...$ corresponds to the samples of the SIMO estimation window (see Fig. 3.3).

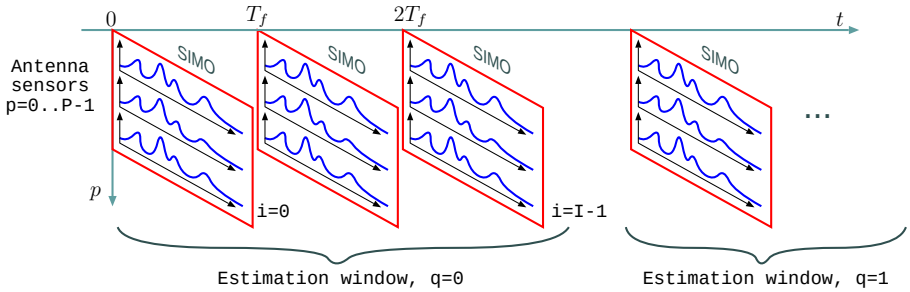


Figure 3.3: Sequential SIMO channel acquisition and processing.

3.2 Space-Alternating Generalized Expectation-Maximization

As we have seen in Section 3.1, under the plane wave assumption the impulse response of a wireless multipath channel can be represented as the sum of delayed and weighted Dirac impulses, each representing one individual multipath component. Such special structure of the channel impulse response implies that the filtered signal $z(t)$ should have sparse structure, which would in turn imply simple estimation of the channel parameters. Unfortunately, this sparsity is often obscured by additive noise and temporal dispersion due to the finite bandwidth of the transmitter and receiver hardware. This motivates the application of algorithms capable of recovering this sparse structure from the measurements.

Various algorithms have been proposed for estimating multipath parameters from measurement data. The used methods can be grouped into three categories as defined in [KV96]: spectral estimation (MUSIC)[Sch86], parametric subspace methods [RK89, HN95], and deterministic methods. The Expectation-Maximization (EM) algorithm [DLR77], as well as SAGE [FTH+99, FDHT96] belongs to the latter category.

SAGE is a generalization of the EM algorithm that is used to replace the high-dimensional optimization procedure, necessary to compute the joint maximum likelihood estimates, with several separate maximization processes, which can be performed sequentially. This property makes SAGE particularly suitable for joint estimation of the multipath parameters. Like any maximum likelihood method, SAGE relies on the assumed data model, which in the case of wireless channels is specified as (3.7), i.e., it assumes the plane wave scenario.

Now, we summarize the major steps of the SAGE algorithm for multipath parameter estimation. For more details on the SAGE algorithm the interested reader is referred to [FH94, FTH+99, FDHT96, PFM97]. Again, for the sake of simplicity we will consider the SIMO case only. Extension of the SAGE algorithm to MISO, MIMO, and SISO channel IR's does not pose any significant difficulty and thus it is not discussed.

We see from (3.8) that the wireless channel can be modeled as a sum of L con-

tributing wavefronts, where each wave is described by the corresponding multipath delay τ_l, Doppler shift ν_l, DoA ϕ_l, as well as path gains a_l. Let us denote a set of parameters describing each multipath as[2]

$$\boldsymbol{\theta}_l = \{a_l, \tau_l, \nu_l, \phi_l\}.$$

The contribution of each of the wavefronts to the signal at the output of the MF for $t \in O$ can be represented as

$$\boldsymbol{s}(t; \boldsymbol{\theta}_l) = a_l \boldsymbol{c}(\phi_l) \exp\left(j2\pi(i - \frac{I-1}{2})\nu_l T_f\right) R_{uu}(t - \tau_l)$$

Now, given the matched filter output $\boldsymbol{z}(t)$, SAGE solves the following optimization problem:

$$\boldsymbol{\Theta}^{ML} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \left\| \boldsymbol{z}(t) - \boldsymbol{S}(t; \boldsymbol{\Theta}) \right\|^2, \tag{3.16}$$

where

$$\boldsymbol{S}(t; \boldsymbol{\Theta}) = \sum_{l=1}^{L} \boldsymbol{s}(t; \boldsymbol{\theta}_l),$$

and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_L\}$ is the union of all multipath parameter sets.

Assuming that additive noise term $\boldsymbol{\xi}(t)$ in (3.10) is a stationary complex zero-mean Gaussian process with the covariance matrix $\boldsymbol{\Sigma} = E\{\boldsymbol{\xi}(t)\boldsymbol{\xi}(t)^H\}$, the minimization of (3.16) is equivalent to maximization of the likelihood function $\Lambda(\boldsymbol{\Theta}; \boldsymbol{z})$ defined as [Poo96]

$$\Lambda(\boldsymbol{\Theta}; \boldsymbol{z}) \propto \int_{\mathcal{O}} 2\mathrm{Re}\left\{\boldsymbol{S}(t'; \boldsymbol{\Theta})^H \boldsymbol{\Sigma}^{-1} \boldsymbol{z}(t')\right\} dt' - \int_{\mathcal{O}} \boldsymbol{S}(t'; \boldsymbol{\Theta})^H \boldsymbol{\Sigma}^{-1} \boldsymbol{S}(t'; \boldsymbol{\Theta}) dt', \tag{3.17}$$

where $\mathrm{Re}\{\cdot\}$ denotes the real part, and $(\cdot)^H$ denotes the Hermitian transpose of the argument, respectively.

The brute force approach to find the optimum $\boldsymbol{\Theta}^{ML} = \mathrm{argmax}_{\boldsymbol{\Theta}}\{\Lambda(\boldsymbol{\Theta}; \boldsymbol{z})\}$ is computationally prohibitive since it results in an intractable high-dimensional optimization procedure[3]. One possible solution to this problem can be found within the Expectation-Maximization (EM) framework. The EM algorithm is an iterative optimization scheme that relies on the two key concepts of the complete (unobserved) and incomplete (observed) data. In some cases it might be easier to estimate the required parameters based on the complete data rather than directly from the observed data. The incomplete data is then used to estimate the complete data, which constitutes the E-step of the algorithm. The latter is then used to obtain the refined parameter estimates, which is the M-step of the algorithm. Iteration between E-

[2]The set of parameters might be extended to include wave polarization, elevation angles, etc., if the channel measurements allow to identify them.

[3]In the considered SIMO case this will require searching simultaneously $L \times (3 + 2)$ dimensions – 3 real-valued parameters and 1 complex-valued path gain.

and M-steps form the basis for the EM algorithm(see [DLR77, Moo96]). The iterative scheme requires a good, i.e., close to the optimum, initialization of the sought parameters.

In case of wireless channels the incomplete data is basically the output of the MF (3.10). In (3.16) the individual signals $\boldsymbol{s}(t;\boldsymbol{\theta}_l)$ corrupted by the additive observation noise form the natural complete data [FW88]:

$$\boldsymbol{x}_l(t) = \boldsymbol{s}(t;\boldsymbol{\theta}_l) + \boldsymbol{\xi}_l(t), \quad l = 1 \dots L, \tag{3.18}$$

where

$$\boldsymbol{\xi}(t) = \sum_{l=1}^{L} \sqrt{\beta_l}\boldsymbol{\xi}_l(t), \text{and} \quad \sum_{l=1}^{L} \beta_l = 1. \tag{3.19}$$

Note that $\boldsymbol{\xi}_l(t)$ are independent complex white Gaussian noises. The factors β_l are free design parameters [FW88]. However, it was shown [FH94, FTH+99] that by setting $\beta_l = 1$ the conditional Fisher information of $\boldsymbol{x}_l(t)$ given $\boldsymbol{z}(t)$ is maximized, what in turn maximizes the asymptotic convergence rate of the EM algorithm [DLR77].

At each iteration of the algorithm, we estimate the complete data $\boldsymbol{x}_l(t)$ based on the observation $\boldsymbol{z}(t)$ and some previous parameter estimates $\hat{\boldsymbol{\Theta}}'$:

$$\hat{\boldsymbol{x}}_l(t;\hat{\boldsymbol{\theta}}_l') = E\{\boldsymbol{x}_l(t)|\boldsymbol{z}(t), \hat{\boldsymbol{\Theta}}'\} \quad l = 1, \dots, L.$$

This forms the Expectation step of the EM-algorithm. Having estimated the complete data, we can then use it to refine our parameter estimates. The likelihood (3.17) can then be reformulated as a function of $\boldsymbol{\theta}_l$ and complete data $\boldsymbol{x}_l(t)$:

$$\Lambda(\boldsymbol{\theta}_l;\boldsymbol{x}_l(t)) \propto \int_{\mathcal{O}} 2\mathrm{Re}\left\{\boldsymbol{s}(t';\boldsymbol{\theta}_l)^H \boldsymbol{\Sigma}_l^{-1} \boldsymbol{x}_l(t')\right\} dt' - \int_{\mathcal{O}} \boldsymbol{s}(t';\boldsymbol{\theta}_l)^H \boldsymbol{\Sigma}_l^{-1} \boldsymbol{s}(t';\boldsymbol{\theta}_l) dt', \tag{3.20}$$

where $\boldsymbol{\Sigma}_l = E\{\boldsymbol{\xi}_l(t)\boldsymbol{\xi}_l(t)^H\}$. Then, the new refined parameter estimate $\hat{\boldsymbol{\theta}}_l''$ can be obtained by solving

$$\hat{\boldsymbol{\theta}}_l'' = \underset{\boldsymbol{\theta}_l}{\operatorname{argmax}} \Lambda(\boldsymbol{\theta}_l; \hat{\boldsymbol{x}}_l(t;\hat{\boldsymbol{\theta}}_l')), \quad l = 1 \dots L. \tag{3.21}$$

Expression (3.21) forms the Maximization step of the EM-algorithm. It can be seen that instead of an $L \times 5$-D optimization problem, we end up having L separate 5-D optimizations – three real-valued parameters (delay, Doppler shift, and DoA), and one complex-valued multipath gain.

However, 5-D optimization is also not trivial. To further simplify the optimization procedure, the SAGE algorithm is introduced. The SAGE algorithm is used to update not all, but a subset of the parameters, while keeping the others fixed [FH94]. Basically, SAGE is a grouped coordinate descent method that, in case of wireless channels, allows to exchange a 5-D optimization by a sequence of 5 separate 1-D searches, thus significantly reducing the computational load. The SAGE iterations are guaranteed to converge to a maximum of the corresponding likelihood function,

but it might be a local, rather than global optimum. Thus it is important to provide a good initialization for the multipath parameter values to make sure that the solution converges to an global optimum.

Algorithm 3.1 presented below outlines the major steps of applying the SAGE algorithm to multipath parameter estimation.

Algorithm 3.1 SAGE algorithm for estimating channel parameters

Initialize algorithm: L, $\boldsymbol{\theta}_l^{[0]} = \{a_l^{[0]}, \tau_l^{[0]}, \phi_l^{[0]}, \nu_l^{[0]}\}, \quad l = 1, \ldots, L$
% — begin SAGE iterations —%
for each $l = 1, \ldots, L$

E-Step: Estimate the complete data

$$\hat{\boldsymbol{x}}_l(t; \boldsymbol{\theta}_l^{[k]}) = \boldsymbol{z}(t) - \sum_{l'=1, l' \neq l}^{L} s(t; \boldsymbol{\theta}_{l'}^{[k]}) \tag{3.22}$$

M-Step: Find new parameters as:

$$\tau_l^{[k+1]} = \underset{\tau_l}{\operatorname{argmax}} \left| Z(\tau_l, \phi_l^{[k]}, \nu_l^{[k]}; \hat{\boldsymbol{x}}_l(t; \boldsymbol{\theta}_l^{[k]})) \right| \tag{3.23}$$

$$\phi_l^{[k+1]} = \underset{\phi_l}{\operatorname{argmax}} \left| Z(\tau_l^{[k+1]}, \phi_l, \nu_l^{[k]}; \hat{\boldsymbol{x}}_l(t; \boldsymbol{\theta}_l^{[k]})) \right| \tag{3.24}$$

$$\nu_l^{[k+1]} = \underset{\nu_l}{\operatorname{argmax}} \left| Z(\tau_l^{[k+1]}, \phi_l^{[k+1]}, \nu_l; \hat{\boldsymbol{x}}_l(t; \boldsymbol{\theta}_l^{[k]})) \right| \tag{3.25}$$

$$a_l^{[k+1]} = \frac{1}{I \|\boldsymbol{c}(\phi_l^{[k+1]})\|^2 T_u E_s} Z(\tau_l^{[k+1]}, \phi_l^{[k+1]}, \nu_l^{[k+1]}; \boldsymbol{x}_l(t; \hat{\boldsymbol{\theta}}_l^{[k]})) \tag{3.26}$$

Here E_s is the power of the sounding signal. (3.27)

$$\boldsymbol{\theta}_l^{[k+1]} = \{a_l^{[k+1]}, \tau_l^{[k+1]}, \phi_l^{[k+1]}, \nu_l^{[k+1]}\}, \quad l = 1, \ldots, L \tag{3.28}$$

where

$$\begin{aligned} Z(\tau, \phi, \nu; \boldsymbol{x}_l(t)) = \sum_{i=0}^{I-1} \int_{O_i} & R_{uu}^*(t' - \tau) \boldsymbol{c}^H(\phi) \times \\ & \exp\left(-j2\pi (i - \frac{I-1}{2}) \nu T_f \right) \boldsymbol{x}_l(t') dt' \end{aligned} \tag{3.29}$$

end

It can be inferred that $Z(\tau, \phi, \nu; \boldsymbol{x}_l(t))$ acts as a beamformer in the corresponding domain, reaching its maximum only when the values of the free parameters coincide with the true ones that parametrize the multipath $s(t; \boldsymbol{\theta}_l)$.

The iterations of the Algorithm 3.1 are repeated until some suitable convergence

criterion for the parameters of interest is met. In our simulations we stopped the iterations when the relative changed of the parameter values was less than 1%.

3.2.1 Initializing SAGE with Matching Pursuit algorithm

SAGE is an iterative technique that requires a proper initialization. In general, this initialization has to be derived directly from the measured data, and/or from any available *a priori* knowledge. In the former case, the multipath parameters are initialized incoherently: first, the multipath delay is found, and then the corresponding DoA, Doppler frequency, and finally, the multipath gain. To find the initial values of the multipath delays we use the Matching Pursuit (MP) technique.

Matching Pursuit (MP) is a greedy iterative algorithm for deriving a signal decomposition in terms of expansion functions (also called atoms) chosen from a dictionary. MP was first introduced in [MZ93] for time-frequency representations, and has been later extended into Orthogonal Matching Pursuit (OMP) [DMA97, PRK93] for general sparse approximate solutions to signal representation problems.

From (3.12) we notice that, within the MP framework, the design matrix $\boldsymbol{K}$ can be treated as approximation dictionary, and the columns in the matrix as atoms. Of course, each column in the matrix corresponds to a certain multipath component delay. Since these delays are initially unknown, we come up with an overcomplete representation of the data, by quantizing the range of possible multipath delay values τ_l's. This will effectively generate the corresponding design matrix $\boldsymbol{K}$, or in terms of the sparse approximation, a dictionary, where each column represents a basis function $\boldsymbol{r}_l = [R_{uu}[-\tau_l], R_{uu}[T_s - \tau_l], \ldots, R_{uu}[(N-1)T_s - \tau_l]]^T$.

Now, let us assume that we have an antenna array with P receive elements where each estimation window consists of I SIMO blocks. Since the delays are initialized incoherently, we may neglect the channel structure along the time samples t_i and space p (antenna sensors) dimensions and consider each measured channel as an independent realization. Thus we can say that we have $J = I \times P$ statistically independent SISO channel realizations $\boldsymbol{z}_j$, $j = 0, \ldots, J-1$.

The greedy iteration of the MP is carried out as follows: first, the atom $\boldsymbol{r}_l$ from the dictionary $\boldsymbol{K}$ that best approximates the measured signal $\boldsymbol{z}_j^{[0]}$ is selected. The squared norm L_2 is often used as a metric to measure the quality of approximation simply because of its mathematical convenience, although other criteria can be imagined. A graphical representation of this procedure is given in Fig. 3.4.

The projection of the $\boldsymbol{z}_j^{[0]}$ on the selected basis (e.g., $\boldsymbol{r}_1$ as in Fig. 3.4) is then subtracted from signal $\boldsymbol{z}_j^{[0]}$ and the process is iterated on the residual $\boldsymbol{z}_j^{[1]}$.

The MP algorithm is in some sense equivalent the the Gramm-Schmidt orthogonalization procedure, since the obtained residual is always orthogonal to the selected vectors in the expansion[4]. It also can be thought as a successive interference cancellation approach to initialize multipath parameters as proposed in [FTH+99].

[4]However, the basis is constrained to be neither orthogonal, nor normalized.

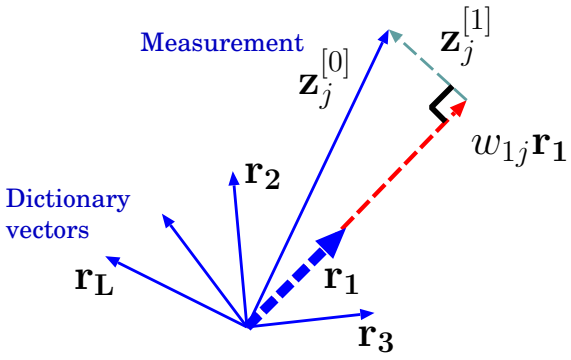


Figure 3.4: Matching Pursuit greedy signal approximation.

The MP algorithm with a fixed number of components is presented in Algorithm 3.2.

Algorithm 3.2 MP algorithm for delay initialization

Initialize the dictionary $\mathcal{D} \equiv \boldsymbol{K}$.
Initial residual $\boldsymbol{d}_j^{[0]} = \boldsymbol{z}_j$
% — begin MP iterations —%
for $l = 1, \ldots, L$

Find the best matching atom

$$\boldsymbol{r}_l = \underset{\boldsymbol{r}\in\mathcal{D}}{\operatorname{argmax}} \sum_{j=0}^{J-1} |\boldsymbol{r}^H, \boldsymbol{d}_j^{[l-1]}|$$

Compute:

$$w_{lj} = \frac{\boldsymbol{r}_l^H \boldsymbol{d}_j^{[l-1]}}{\|\boldsymbol{r}_l\|^2}, \quad \boldsymbol{d}_j^{[l]} = \boldsymbol{d}_j^{[l-1]} - w_{lj}\boldsymbol{r}_l$$

end

Resulting approximation

$$\boldsymbol{z}_j \approx \sum_{l=1}^{L} w_{lj}\boldsymbol{r}_l$$

The presented algorithm constrains L to a certain predefined number, just as we need in the SAGE algorithm. Alternatively, one can proceed with the MP iterations until the energy of the residual $\boldsymbol{z}_j^{[l]}$ falls below the certain threshold. In this case the MP iterations guarantee to produce the representation that capture the desired portion of the total impulse response power. However, finding objective rules to select this threshold is not a trivial task. A threshold-based model order selection procedure will be discussed later in Chapter 4. Algorithm 3.2 also accounts for the

multiple channels specific to SIMO/MISO and MIMO systems: the best matching atom is found jointly over the $J = I \times P$ channel realizations, following the idea of simultaneous matching pursuit presented in [TGS05]. Of course, we assume that the structure of the channel stays invariant over the estimation window $\mathcal{O}$.

Once the delays have been found, we can initialize other multipath parameters. We initialize them using the coefficients w_{lj}. The obtained coefficients are organized in the structure shown in Fig. 3.5.

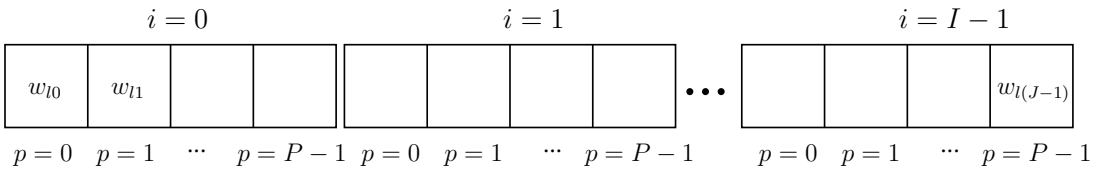


Figure 3.5: Structure of the coefficients w_{lj} for a single basis $\boldsymbol{r}_l$.

We can easily transform coefficients w_{lj} into the matrix $\boldsymbol{W}_l$ with columns and rows corresponding to the time and space, respectively, as shown in Fig. 3.6.

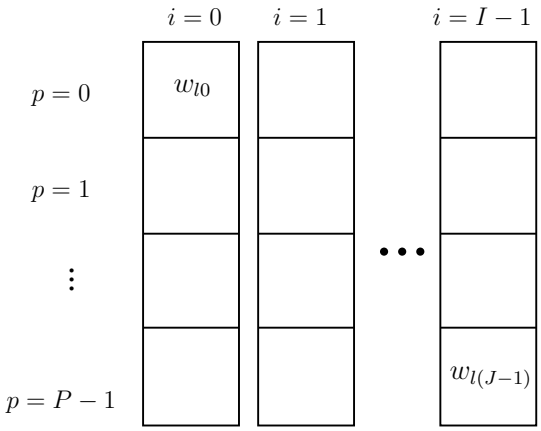


Figure 3.6: Structure of the matrix $\boldsymbol{W}_l$.

Coefficients in the matrix $\boldsymbol{W}_l$ are then used to initialize the angular information. Let as define $\boldsymbol{\gamma}_i$ as the ith column of the matrix $\boldsymbol{W}_l$, so that $\boldsymbol{W}_l = [\boldsymbol{\gamma}_0 \dots \boldsymbol{\gamma}_{I-1}]$. Then, the initial value of the DoA ϕ_l is found as the maximizer of the following function

$$\phi_l = \underset{\phi}{\operatorname{argmax}} \sum_{i=0}^{I-1} |\boldsymbol{\gamma}_i^H \boldsymbol{c}(\phi)|. \tag{3.30}$$

Similarly, we can initialize the Doppler frequency. Let as define $\boldsymbol{\delta}_p$ as the pth row of the matrix $\boldsymbol{W}_l$, so that $\boldsymbol{W}_l = [\boldsymbol{\delta}_0^T \dots \boldsymbol{\delta}_{P-1}^T]^T$. Then, the incoherent initialization of the Doppler frequency for the lth component is found as the value that maximizes

$$\nu_l = \underset{\phi}{\operatorname{argmax}} \sum_{p=0}^{P-1} |\boldsymbol{d}(\nu) \boldsymbol{\delta}_p^H|, \tag{3.31}$$

where $\boldsymbol{d}(\nu) = [d_0(\nu) \dots d_{I-1}(\nu)]$, and

$$d_i(\nu) = \exp(j2\pi(i - (I-1)/2)\nu T_f), i = 0, \dots, I-1. \tag{3.32}$$

Finally, the initialization of the multipath gain a_l is found as

$$a_l = \frac{\boldsymbol{c}^H(\phi_l)\boldsymbol{W}_l\boldsymbol{d}^H(\nu_l)}{||\boldsymbol{c}(\phi_l)||^2||\boldsymbol{d}(\nu_l)||^2}, \tag{3.33}$$

where ϕ_l and ν_l are solutions to the maximization problems (3.30) and (3.31), respectively. This finalizes the initialization of the SAGE algorithm.

3.2.2 Some application examples

To demonstrate the resulting multipath parameter estimation we consider application of the SAGE algorithm to measured channel data from the FTW database described in Appendix C. Since the SAGE algorithm minimizes the functional (3.16), we thus consider the resulting goodness-of-fit between the real power profiles and those obtained based on the SAGE estimates assuming different numbers of the multipath components L (Figures 3.8-3.11).

In all simulations $I = 5$, with $T_e = 20$msec, which for the FTW database corresponds to the spatial resolution of $\approx \lambda/7$. The receive antenna is a linear array with $P = 8$ elements. The multipath parameter estimation was done sequentially over 560msec of measurement time, which corresponds to $\approx 4\lambda$ of walked distance, or 29 consecutive estimation windows. Evolution of the corresponding measured power delay profile is shown in Fig. 3.7.

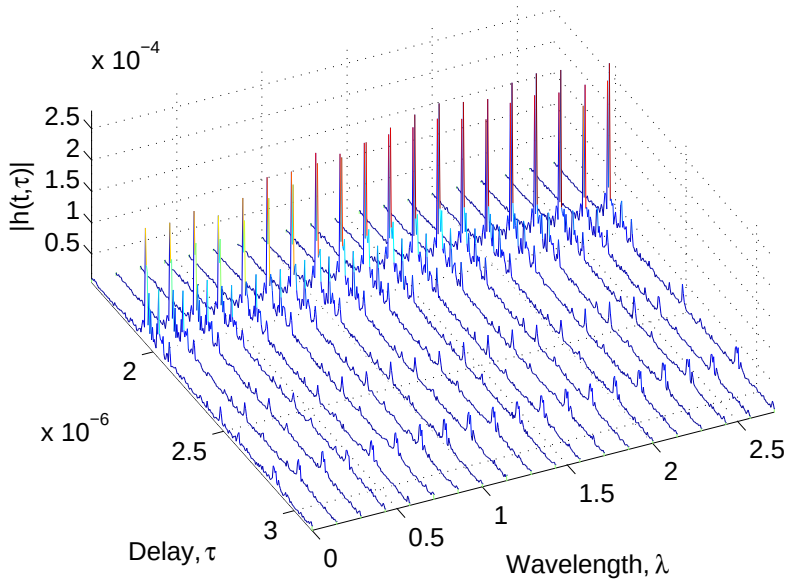
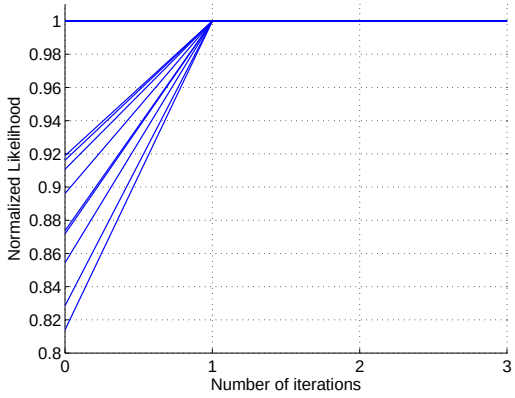
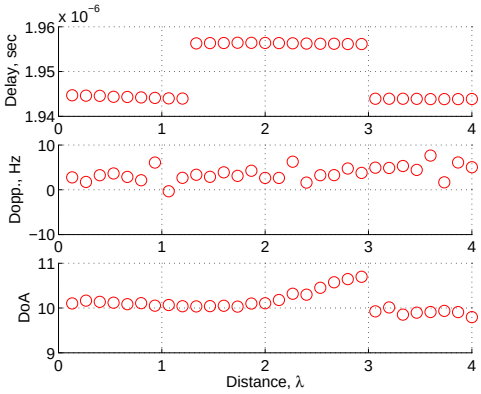


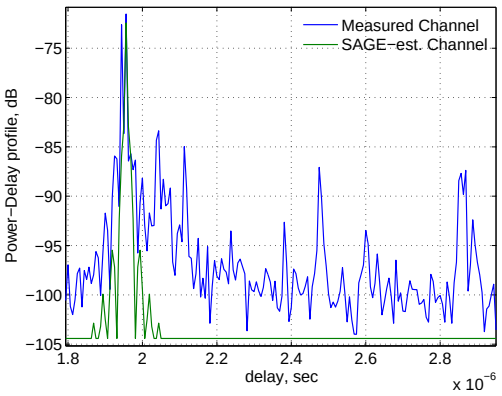
Figure 3.7: Evolution of the measured channel power-delay profile.



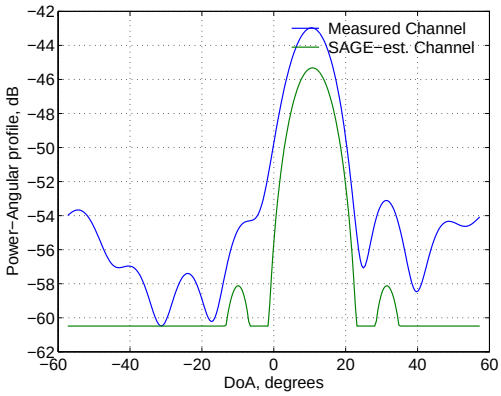
(a) Convergence in likelihood (normalized to 1) for different estimation windows.



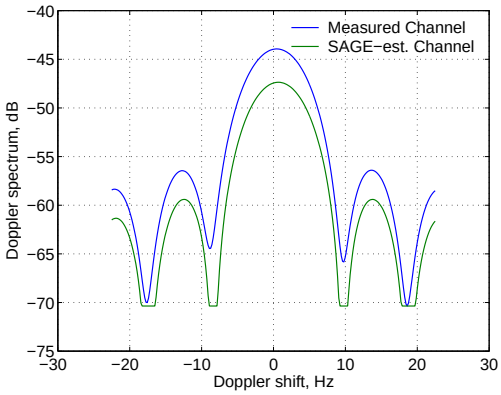
(b) Evolution of the estimated parameters.



(c) Estimated Power-Delay Profile.

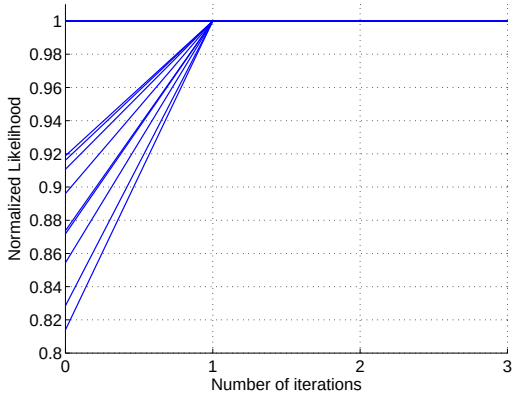


(d) Estimated Power-Angular Profile.

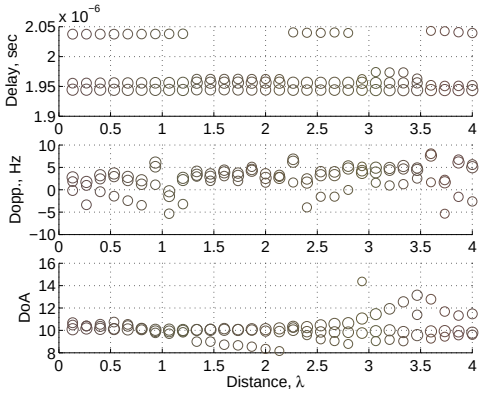


(e) Estimated Doppler spectrum.

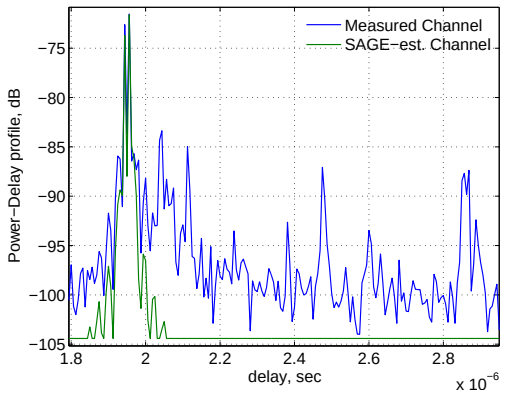
Figure 3.8: Goodness-of-fit for the SAGE approximation with $L = 1$.



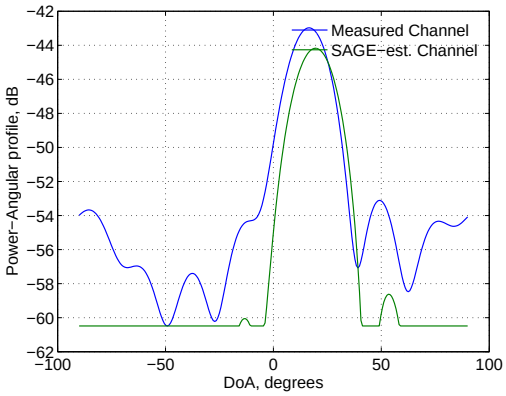
(a) Convergence in likelihood (normalized to 1) for different estimation windows.



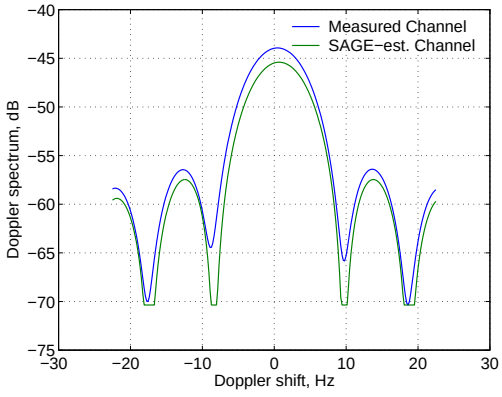
(b) Evolution of the estimated parameters.



(c) Estimated Power-Delay Profile.

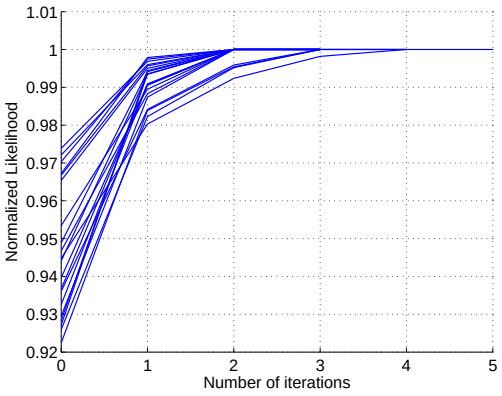


(d) Estimated Power-Angular Profile.

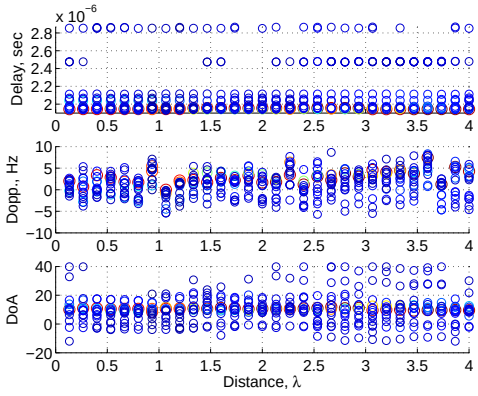


(e) Estimated Doppler spectrum.

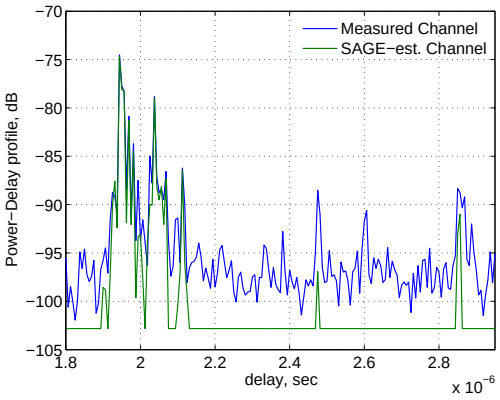
Figure 3.9: Goodness-of-fit for the SAGE approximation with $L = 3$.



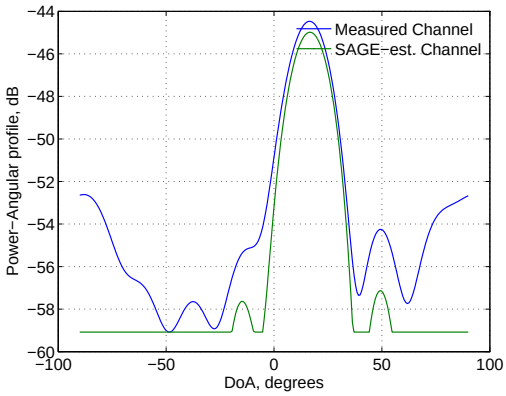
(a) Convergence in likelihood (normalized to 1) for different estimation windows.



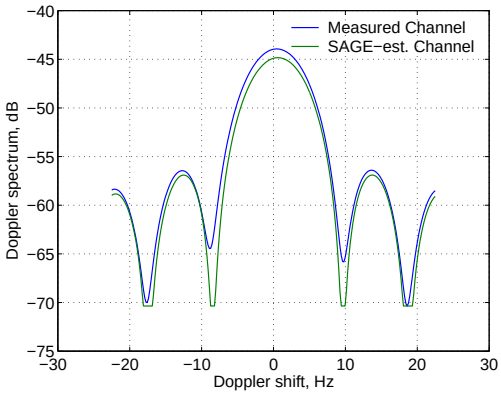
(b) Evolution of the estimated parameters.



(c) Estimated Power-Delay Profile.

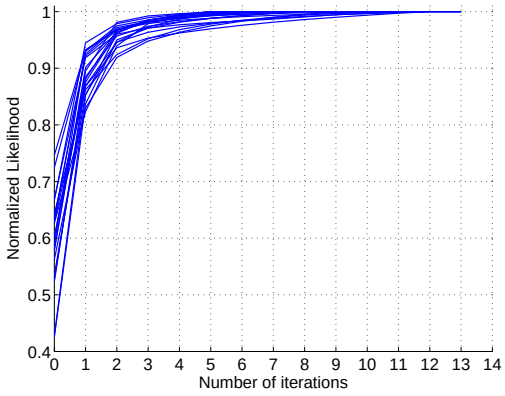


(d) Estimated Power-Angular Profile.

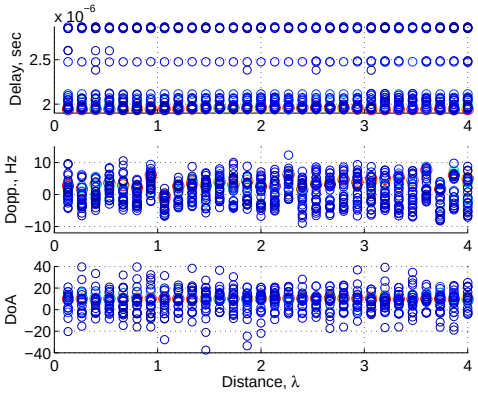


(e) Estimated Doppler spectrum.

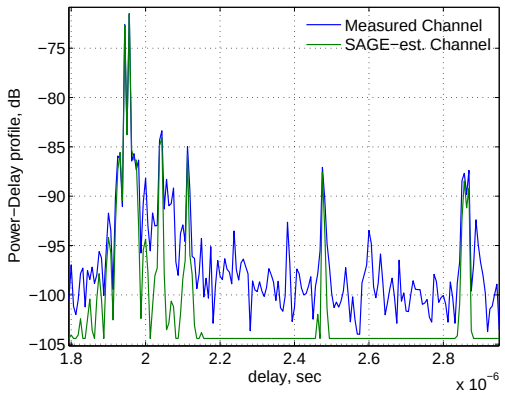
Figure 3.10: Goodness-of-fit for the SAGE approximation with $L = 15$.



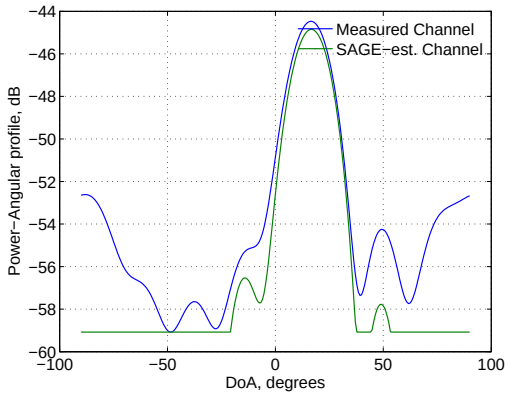
(a) Convergence in likelihood (normalized to 1) for different estimation windows.



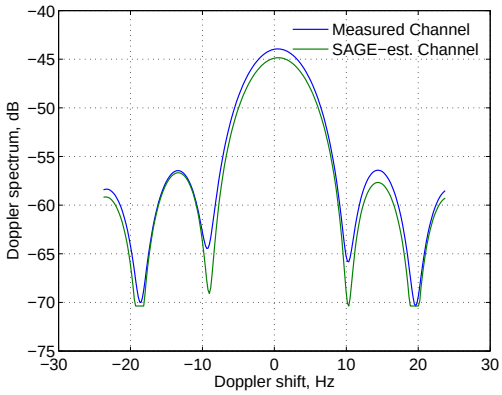
(b) Evolution of the estimated parameters.



(c) Estimated Power-Delay Profile.



(d) Estimated Power-Angular Profile.



(e) Estimated Doppler spectrum.

Figure 3.11: Goodness-of-fit for the SAGE approximation with $L = 30$.

The first estimation results in Fig. 3.8 are for $L = 1$.

Clearly in this case the algorithm converges after just a few iterations (Fig. 3.8(a)). From the results in Fig. 3.8(c) we can conclude that not all of the multipath components have been captured, since a significant part of the total measured energy was not captured.

By increasing the number of components we can improve the SAGE fit to the data (Fig. 3.9, 3.10, and 3.11), since with more components we can fit the data better. Of course the paid price is the increased estimation complexity: the algorithm requires more iterations to converge, and it is more prone to land in the local maximum of the likelihood. As the result, a proper initialization becomes a crucial aspect.

In the presented plots we also demonstrate how the estimated parameters (in this case delay, Doppler frequency, and DoA) vary with time (Fig. 3.8(b), 3.9(b), 3.10(b), and 3.11(b)). These plots partly show the necessity of multipath tracking, addressed in Chapter 5. To put it shortly, we need tracking to join the multipath estimations over time into multipath trajectories.

3.3 Conclusions and discussion

Let us summarize the results obtained in this chapter. In the beginning we considered channel sounding in the time and frequency domains. These are used to obtain channel measurement data that is used as the basis of our channel prediction approach.

In this work we consider channel data obtained using channel sounding equipment. Channel sounders are specifically designed for obtaining very accurate channel representations. Keep in mind that in the communication system, where the prediction algorithm would mostly be desired, the main goal is in delivering information to the recipient, rather than extracting channel data.

Clearly, working with high-resolution channel data obtained using a channel sounder is easier. However, such data might not always be available. Thus, we understand that in general the whole prediction framework should be adjusted so as to cope with the obtained channel data structure and resolution.

Parameter estimation

In order to be able to estimate the multipath parameters we take the plane wave channel model, developed in Chapter 2. In the general case we cannot say that the plane wave assumption always holds. If this assumption is not met, we would expect some performance degradation that mainly results in biased parameter estimates. This might result in additional multipath parameter noise that affects the tracking of the components over time. Thus, we would expect the best performance for the components that represent plane wavefronts.

The availability of the channel model allows to apply model-based parameter estimation algorithms within, for instance, the Maximum Likelihood framework. In

our work we exploit a particular instance of the Maximum Likelihood algorithm, known as SAGE. The SAGE algorithm is particularly useful for high dimensional optimizations, which is exactly the case for the joint estimation of the multipath parameters. This algorithm was shown to have good convergence properties, but due to iterative nature it requires a good initialization to avoid landing in local maxima of the likelihood function. It is common to derive the SAGE initialization directly from the measured data, as we explained in Section 3.2.1.

A particularly important aspect that arises when applying the SAGE algorithm to multipath parameter estimation is the estimation artifacts. The artifacts stem from the numerical optimization used in solving (3.16). This optimization involves quantizing the parameter search space, which effectively results in parameter value discretization. As the result, a single multipath component with parameters defined on the real line is approximated by several components with discretized parameters (Fig. 3.12). When the resolution of the discrete-time model is not fine enough,

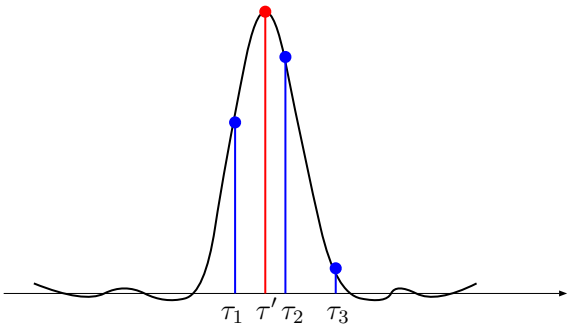


Figure 3.12: Approximation of a single component with delay τ' by three discrete components with delays τ_1, τ_2, and τ_3.

the estimation algorithm uses several discrete components to approximate a single continuous-time component. Since the number of components in the model is fixed, this might lead to some of them being used solely for approximation purposes and not for modelling individual propagation paths.

This problem is similar to the problem that occurs in fractional delay filters (FDF) [LVML96]. An FDF aims at approximating a delay that is not a multiple of the sampling period. As shown in [LVML96], such filters have infinite impulse response. Though FIR approximations exist, they require several samples to represent a single fractional delay. In our case these additional components are very undesirable. For example, when using SAGE with $L = 2$, we might estimate two components corresponding to the same multipath, rather than two physically different multipath components. These artifacts have correlated parameters, and therefore might cause difficulties for the tracking algorithm. They prevent other, possibly weaker, multipath components to be extracted from the measurement data, too.

Thus, it would be desired to introduce a mechanism that allows estimating the number of multipath components L from the data to minimize the chance of missing

some of the potential multipaths. In the following chapter, we consider an algorithm that implements this approach within the Bayesian framework.

Chapter 4

Evidence Procedure and channel estimation

In this chapter we consider a Bayesian approach to estimation of wireless channels. As it was shortly mentioned in Chapter 3, the complete channel estimation problem consists of finding the number L of impinging wavefronts and their corresponding parameters. Joint estimation of the model order L and model parameters is the most desired solution, but it is a particularly difficult task. It often leads to analytically intractable and computationally very expensive optimization procedures.

The problem is often relaxed by assuming that the number of components is fixed, which simplifies optimization in many cases like, for instance, in the SAGE algorithm [KV96, FTH$^+$99] discussed in Section 3.2.

There are however ways to estimate the model order from the measured data. Empirical methods, exemplified by cross-validation, can be employed (see, for example, [DHS00]). Cross-validation selects the optimal model by measuring its performance over a validation data set and selecting the one that performs the best. In case of practical multipath channels, such data sets are often not available due to the time-variability of the channel impulse responses.

Alternatively, one can employ model selection schemes in the spirit of Ockham's razor: simple models (in terms of the number of parameters involved) are preferred over more complex ones. It is clear that the simpler the model is, the worse it approximates the measured data. Thus, there is an optimum solution that balances the approximation quality with the number of involved parameters. This approach does not require the validation set, but instead relies on the data model, which makes the Ockham's razor particularly interesting for model-based estimation algorithms. Examples are the Akaike Information Criterion (AIC) and Minimum Description Length (MDL) principle [WK85, Ris78]. Since in our case the data model is readily available, we take this approach in the proposed algorithm.

Consider a certain class of parametric models (hypotheses) $\mathcal{H}_i$ defined as the collection of prior distributions $p(\boldsymbol{w}_i|\mathcal{H}_i)$ for the model parameters $\boldsymbol{w}_i$[1]. Given the measurement data $\mathcal{Z}$ and a family of conditional distributions $p(\mathcal{Z}|\boldsymbol{w}_i, \mathcal{H}_i)$, our goal is to infer the hypothesis $\hat{\mathcal{H}}$ and the corresponding parameters $\hat{\boldsymbol{w}}$ that maximize the

[1]Here the subscript i denotes different possible hypothesis rather than sounding sequence periods, as in Chapter 3.

posterior

$$\{\hat{\boldsymbol{w}}, \hat{\mathcal{H}}\} = \underset{\boldsymbol{w}_i, \mathcal{H}_i}{\operatorname{argmax}} \Big\{ p(\boldsymbol{w}_i, \mathcal{H}_i | \mathcal{Z}) \Big\}. \tag{4.1}$$

The key to solving (4.1) lies in inferring the corresponding parameters $\boldsymbol{w}_i$ and $\mathcal{H}_i$ from the data $\mathcal{Z}$, which is often a nontrivial task. As far as the Bayesian methodology is concerned, there are two ways this inference problem can be solved [Hay01, sec. 5]. In the *joint estimation method*, $p(\boldsymbol{w}_i, \mathcal{H}_i|\mathcal{Z})$ is maximized directly with respect to the quantities of interest $\boldsymbol{w}_i$ and $\mathcal{H}_i$. This often leads to computationally intractable optimization algorithms. Alternatively, one can rewrite the posterior $p(\boldsymbol{w}_i, \mathcal{H}_i|\mathcal{Z})$ as

$$p(\boldsymbol{w}_i, \mathcal{H}_i|\mathcal{Z}) = p(\boldsymbol{w}_i|\mathcal{Z}, \mathcal{H}_i) p(\mathcal{H}_i|\mathcal{Z}) \tag{4.2}$$

and maximize each term on the right-hand side sequentially from right to left. This approach is known as the *marginal estimation method*. Marginal estimation methods (MEM) are well exemplified by Expectation-Maximization (EM) algorithms and used in many different signal processing applications (see [DHS00, FW88, FTH+99]). MEMs are usually easier to compute, however they are prone to land in a local rather than global optimum.

We recognize the first factor on the right-hand side of (4.2) as a parameter posterior, while the other one is a posterior for different model hypotheses. It is the maximization of $p(\mathcal{H}_i|\mathcal{Z})$ that guides our model selection decision. Then, the data analysis consists of two steps [Mac03, ch. 28],[Fit98]:

1. Inferring the parameter posterior under the hypothesis $\mathcal{H}_i$

$$p(\boldsymbol{w}_i|\mathcal{Z}, \mathcal{H}_i) = \frac{p(\mathcal{Z}|\boldsymbol{w}_i, \mathcal{H}_i) p(\boldsymbol{w}_i|\mathcal{H}_i)}{p(\mathcal{Z}|\mathcal{H}_i)} \equiv \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}. \tag{4.3}$$

2. Comparing different model hypotheses using the model posterior

$$p(\mathcal{H}_i|\mathcal{Z}) \propto p(\mathcal{Z}|\mathcal{H}_i) p(\mathcal{H}_i) \equiv \text{Evidence} \times \text{Hypothesis Prior}. \tag{4.4}$$

In the second stage, $p(\mathcal{H}_i)$ measures our subjective prior over different hypotheses before the data is observed. In many cases it is reasonable to assign equal probabilities to different hypotheses, thus reducing the hypothesis selection to selecting the model with the highest evidence[2] $p(\mathcal{Z}|\mathcal{H}_i)$. The evidence can be expressed as the following integral:

$$p(\mathcal{Z}|\mathcal{H}_i) = \int p(\mathcal{Z}|\boldsymbol{w}_i, \mathcal{H}_i) p(\boldsymbol{w}_i|\mathcal{H}_i) d\boldsymbol{w}_i. \tag{4.5}$$

Maximizing this integral with respect to the unknown model $\mathcal{H}_i$ is known as the evidence maximization procedure, or Evidence Procedure (EP) [Mac92, Mac94]. The evidence integral (4.5) plays a crucial role in the development of Schwarz's

[2] In the Bayesian literature, the evidence is also known as the likelihood for the hypothesis $\mathcal{H}_i$

approach to model order estimation [Sch78] (Bayesian Information Criterion), as well as in a Bayesian interpretation of Rissanen's MDL principle, as well as its variations [Ris96, Ris78, Lan01].

Relevance Vector Machines (RVM), originally proposed by M. Tipping [Tip01], are an example of the marginal estimation method that, for a set of hypotheses $\mathcal{H}_i$, iteratively approximates (4.1) by alternating between the model selection, i.e., maximizing (4.5) with respect to $\mathcal{H}_i$, and inferring the corresponding model parameters from maximization of (4.3). RVMs have been initially proposed to find sparse solutions to general linear problems. However, they can be quite effectively adapted to the estimation of the impulse response of wireless channels, thus resulting in an effective channel parameter estimation and model selection scheme within the Bayesian framework.

The material presented in the paper is organized as follows: Section 4.1 introduces general assumptions we take to apply the EP algorithm, as well as the used notations, Section 4.2 explains the framework of the EP in the context of wireless channels. In Section 4.3 we explain how model selection is implemented within the presented framework and discuss the relationship between the EP and the MDL criterion for model selection. Finally, Section 4.4 presents some application results illustrating the performance of the RVM-based estimator in synthetic as well as in actual wireless environments.

4.1 Signal model

The method we develop is primarily based on the signal model of the sampled multipath channel corresponding to the planar wave assumption, discussed in Sections 3.1.3 and 3.1.4.

Initially we will, however, restrict ourselves to estimating the model order L along with the vector $\boldsymbol{w}_p$ in (3.12), rather than the constituting parameters τ_l, ϕ_l, ν_l, and a_l. We will also quantize, although arbitrarily fine[3], the search space for the multipath delays τ_l. Thus, we do not try to estimate the channel parameters with infinite resolution, but rather fix the search grid at a certain accuracy, which is dictated by a particular application.

The size of the delay search space L_0 and the resulting quantized delays $\mathcal{T} = \{T_1, \ldots, T_{L_0}\}$, form the initial model hypothesis $\mathcal{H}_0$, which would manifest itself in the columns of the matrix $\boldsymbol{K}$ in (3.12).

As it can be seen, our idea lies in finding the closest approximation of the continuous-time model (3.10) with the discrete-time equivalent (3.12). By incorporating the model selection in the analysis, we also strive to find the most compact representation (in terms of the number of components), while preserving good approximation quality. Thus, our goal is to estimate the channel parameters $\boldsymbol{w}_p$ as

[3]There is actually a limit beyond which it makes no sense to make the search grid finer, since it will not decrease the variance of the estimates, which is lower-bounded by the Cramer-Rao lower bound [FTH+99].

well as to determine how many multipath components $L \leq L_0$ are present in the measured impulse response. The application of the RVM framework to this problem follows in the next section.

4.2 Evidence maximization, Relevance Vector Machines and wireless channels

We begin our analysis following the steps outlined in the beginning of this chapter. In order to ease the algorithm description we first assume that $P = 1$, i.e., only a single sensor is used. Extensions to the case $P > 1$ is carried out later in Section 4.2.2. To simplify the notations, we also drop the subscript index p in our further notations.

From (3.12) it follows that the observation vector $\boldsymbol{z}$ is a linear combination of the vectors from the column-space of $\boldsymbol{K}$, weighted according to the parameters $\boldsymbol{w}$ and embedded in the correlated noise $\boldsymbol{\xi}$. In order to correctly assess the order of the model, it is imperative to take the noise process into account. It follows from (3.15) that the covariance matrix of the noise is proportional to the unknown spectral height N_0, which should therefore be estimated from the data. Thus, the model hypotheses $\mathcal{H}_i$ should include the term N_0. In the following analysis we assume that $\beta = N_0^{-1}$ is Gamma-distributed [Tip01], with the corresponding probability density function (pdf) given as

$$p(\beta|c,d) = \frac{c^d}{\Gamma(d)}\beta^{d-1}\exp(-c\beta), \tag{4.6}$$

with parameters c and d predefined so that (4.6) accurately reflects our *a priori* information about N_0. In the absence of any *a priori* knowledge one can make use of a non-informative (i.e., flat in the logarithmic domain) prior by fixing the parameters to small values $d = c = 10^{-4}$ [Tip01]. Furthermore, to steer the model selection mechanism, we introduce an extra parameter (hyperparameter) α_l, $l = 1, \ldots, L_0$, for each column in $\boldsymbol{K}$. This parameter measures the contribution or relevance of the corresponding weight w_l in explaining the data $\boldsymbol{z}$ from the likelihood $p(\boldsymbol{z}|\boldsymbol{w}_i, \mathcal{H}_i)$. This is achieved by specifying the prior $p(\boldsymbol{w}|\boldsymbol{\alpha})$ for the model weights:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{l=1}^{L_0} \frac{\alpha_l}{\pi}\exp(-|w_l|^2\alpha_l), \tag{4.7}$$

which is in our case a zero-mean complex multivariate Gaussian. High values of α_l will render the contribution of the corresponding column in the matrix $\boldsymbol{K}$ 'irrelevant', since the weight w_l is likely to have a very small value (hence they are termed *relevance hyperparameters*). This will enable us to prune the model by setting the corresponding weight w_l to zero, thus effectively removing the corresponding column from the matrix and the corresponding delay T_l from the delay search space

$\mathcal{T}$. We also see that α_l^{-1} is nothing else as the prior variance of the model weight w_l. Also note that the prior (4.7) implicitly assumes statistical independence of the multipath contributions.

To complete the Bayesian framework, we also specify the prior over the hyperparameters. Similarly to the noise contribution, we assume the hyperparameters α_l to be Gamma-distributed with the corresponding pdf

$$p(\boldsymbol{\alpha}|a,b) = \prod_{l=1}^{L} \frac{b^a}{\Gamma(a)} \alpha_l^{a-1} \exp(-b\alpha_l),$$

where a and b are fixed at some values that ensure an appropriate form of the prior. Again, we can make this prior non-informative by fixing a and b to small values, e.g., $a = b = 10^{-4}$.

Now, let us define the hypothesis $\mathcal{H}_i$ more formally. Let $\mathcal{P}(\mathcal{S})$ be a power set consisting of all possible subsets of basis vector indices $\mathcal{S} = \{1, \ldots, L_0\}$, and $i \mapsto \mathcal{P}(i)$ be the indexing of $\mathcal{P}(\mathcal{S})$ such that $\mathcal{P}(0) = \mathcal{P}(\mathcal{S})$. Then for each index value i the hypothesis $\mathcal{H}_i$ is the set $\mathcal{H}_i = \{\beta; \alpha_j, j \in \mathcal{P}(i)\}$. Clearly, the initial hypothesis $\mathcal{H}_0 = \{\beta; \alpha_j, j \in \mathcal{S}\}$ includes all possible potential basis functions.

Now we are ready to outline the learning algorithm that estimates the model parameters $\boldsymbol{w}$, β, and hyperparameters $\boldsymbol{\alpha}$ from the measurement data $\boldsymbol{z}$.

4.2.1 Learning algorithm

Basically, learning consists of inferring the values of $\boldsymbol{w}_i$ and the hypothesis $\mathcal{H}_i$ that maximize the posterior (4.2): $p(\boldsymbol{w}_i, \mathcal{H}_i|\mathcal{Z}) \equiv p(\boldsymbol{w}_i, \boldsymbol{\alpha}_i, \beta|\boldsymbol{z})$. Here $\boldsymbol{\alpha}_i$ denotes the vector of all evidence hyperparameters associated with the ith hypothesis. The latter expression can also be rewritten as

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta|\boldsymbol{z}) = p(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\alpha}, \beta) p(\boldsymbol{\alpha}, \beta|\boldsymbol{z}). \tag{4.8}$$

The explicit dependence on the hypothesis index i has been dropped to simplify the notation. We recognize that the first term $p(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\alpha}, \beta)$ in (4.8) is the weight posterior and the other one $p(\boldsymbol{\alpha}, \beta|\boldsymbol{z})$ is the hypothesis posterior. From this point we can start with the Bayesian two-step analysis as has been indicated before.

Assuming the parameters $\boldsymbol{\alpha}$ and β are known, estimation of model parameters consists in finding values $\boldsymbol{w}$ that maximize $p(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\alpha}, \beta)$. Using Bayes' rule we can rewrite this posterior as

$$p(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\alpha}, \beta) \propto p(\boldsymbol{z}|\boldsymbol{w}, \boldsymbol{\alpha}, \beta) p(\boldsymbol{w}|\boldsymbol{\alpha}, \beta). \tag{4.9}$$

Consider the Bayesian graphical model [Hec95] in Fig. 4.1. This graph captures the relationship between different variables involved in (4.8). It is a useful tool to represent the dependencies between the variables involved in the analysis in order to factor the joint density function into contributing marginals.

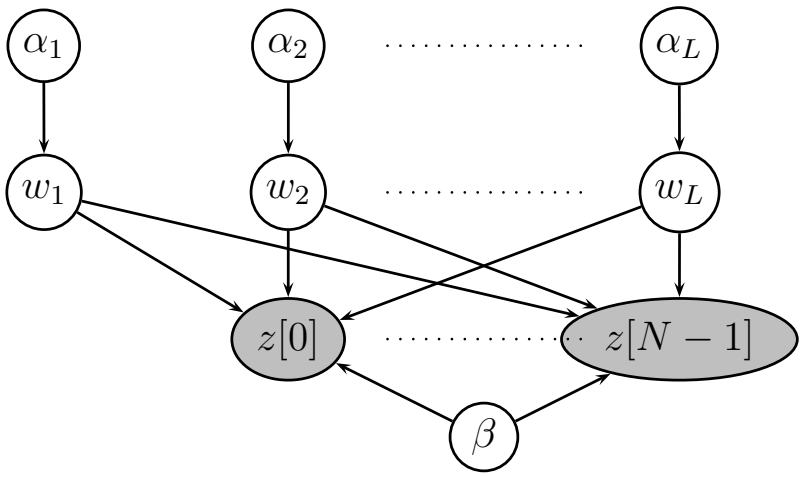


Figure 4.1: Graphical model representing the dependence structure of the discrete-time model of the wireless channel.

It immediately follows from the structure of the graph in Fig. 4.1 that $p(\boldsymbol{z}|\boldsymbol{w},\boldsymbol{\alpha},\beta) = p(\boldsymbol{z}|\boldsymbol{w},\beta)$ and $p(\boldsymbol{w}|\boldsymbol{\alpha},\beta) = p(\boldsymbol{w}|\boldsymbol{\alpha})$, i.e., $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ are conditionally independent given $\boldsymbol{w}$ and β, and $\boldsymbol{w}$ and β are conditionally independent given $\boldsymbol{\alpha}$. Thus, (4.9) is equivalent to

$$p(\boldsymbol{w}|\boldsymbol{z},\boldsymbol{\alpha},\beta) \propto p(\boldsymbol{z}|\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha}), \tag{4.10}$$

where the second factor on the right-hand side is given in (4.7). The first term is the likelihood of $\boldsymbol{w}$ and β given the data. From (3.12) it follows that

$$p(\boldsymbol{z}|\boldsymbol{w},\beta) = \frac{\exp\{-(\boldsymbol{z}-\boldsymbol{K}\boldsymbol{w})^H\beta\boldsymbol{\Lambda}^{-1}(\boldsymbol{z}-\boldsymbol{K}\boldsymbol{w})\}}{\pi^N|\beta^{-1}\boldsymbol{\Lambda}|}.$$

Since both right-hand factors in (4.10) are Gaussian densities, $p(\boldsymbol{w}|\boldsymbol{z},\boldsymbol{\alpha},\beta)$ is also a Gaussian density with the covariance matrix $\boldsymbol{\Phi}$ and mean $\boldsymbol{\mu}$ given as

$$\boldsymbol{\Phi} = (\boldsymbol{A} + \beta\boldsymbol{K}^H\boldsymbol{\Lambda}^{-1}\boldsymbol{K})^{-1}. \tag{4.11}$$

$$\boldsymbol{\mu} = \beta\boldsymbol{\Phi}\boldsymbol{K}^H\boldsymbol{\Lambda}^{-1}\boldsymbol{z}, \tag{4.12}$$

The matrix $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{\alpha})$ is a diagonal matrix that contains the evidence parameters α_l on its main diagonal. Clearly, $\boldsymbol{\mu}$ is a maximum *a-posteriori* (MAP) estimate of the parameter vector $\boldsymbol{w}$ under the hypothesis $\mathcal{H}_i$, with $\boldsymbol{\Phi}$ being the covariance matrix of the resulting estimates. This completes the model fitting step.

Our next step is to find parameters $\boldsymbol{\alpha}$ and β that maximize the hypothesis posterior $p(\boldsymbol{\alpha},\beta|\boldsymbol{z})$ in (4.8). This density function can be represented as $p(\boldsymbol{\alpha},\beta|\boldsymbol{z}) \propto p(\boldsymbol{z}|\boldsymbol{\alpha},\beta)p(\boldsymbol{\alpha})p(\beta)$, where $p(\boldsymbol{z}|\boldsymbol{\alpha},\beta)$ is the evidence term and $p(\boldsymbol{\alpha})p(\beta)$ is the hypothesis prior. As it was mentioned earlier, it is quite reasonable to choose non-informative hypothesis priors since we would like to give all possible hypotheses $\mathcal{H}_i$

an equal chance of being valid. This can be achieved by setting a, b, c, and d to very small values. In fact, as it will follow from the learning algorithm, it is possible to set them to zero, effectively forming uniform (over the logarithmic scale) hyperpriors for $\boldsymbol{\alpha}$ and β. This formulation of prior distributions is related to automatic relevance determination (ARD) [Nea96, Mac94]. As a consequence of this assumption, the maximization of the model posterior is equivalent to the maximization of the evidence, which is known as the Evidence Procedure [Mac92].

The evidence term $p(\boldsymbol{z}|\boldsymbol{\alpha}, \beta)$ can be expressed as

$$\begin{aligned} p(\boldsymbol{z}|\boldsymbol{\alpha}, \beta) &= \int p(\boldsymbol{z}|\boldsymbol{w}, \beta) p(\boldsymbol{w}|\boldsymbol{\alpha}) d\boldsymbol{w} \\ &= \frac{\exp\Big(- \boldsymbol{z}^H (\beta^{-1}\boldsymbol{\Lambda} + \boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H)^{-1}\boldsymbol{z}\Big)}{\pi^N |\beta^{-1}\boldsymbol{\Lambda} + \boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H|}, \end{aligned} \tag{4.13}$$

which is equivalent to (4.5), where conditional independencies between variables have been used to simplify the integrands. In the Bayesian literature this quantity is known as *marginal likelihood* and its maximization with respect to the unknown hyperparameters $\boldsymbol{\alpha}$ and β is a *type-II maximum likelihood* method [Ber85]. To ease the optimization, several terms in (4.13) can be expressed as a function of the weight posterior parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Phi}$ as given by (4.11) and (4.12). Then, by taking the derivatives of the logarithm of (4.13) with respect to $\boldsymbol{\alpha}$ and β and by setting them to zero, we obtain its maximizing values as

$$\alpha_l = \frac{1}{\Phi_{ll} + |\mu_l|^2}, \tag{4.14}$$

$$N_0 = \beta^{-1} = \frac{\mathrm{tr}[\boldsymbol{\Phi}\boldsymbol{K}^H\boldsymbol{\Lambda}^{-1}\boldsymbol{K}] + (\boldsymbol{z} - \boldsymbol{K}\boldsymbol{\mu})^H\boldsymbol{\Lambda}^{-1}(\boldsymbol{z} - \boldsymbol{K}\boldsymbol{\mu})}{N}. \tag{4.15}$$

In (4.14) μ_l and Φ_{ll} denote the lth element of, respectively, the vector $\boldsymbol{\mu}$, and the main diagonal of the matrix $\boldsymbol{\Phi}$. Unlike the maximizing values obtained in the original RVM paper [Tip01, (18)], (4.15) is derived for the extended, more general case of colored additive noise $\boldsymbol{\xi}$ with the corresponding covariance matrix $\beta^{-1}\boldsymbol{\Lambda}$ arising due to the MF processing at the receiver. Clearly, if the noise is assumed to be white, expressions (4.14) and (4.15) coincide with those derived in [Tip01].

Thus, for a particular hypothesis $\mathcal{H}_i$ the learning algorithm proceeds by repeated application of (4.11) and (4.12), alternated with the update of the corresponding evidence parameters $\boldsymbol{\alpha}_i$ and β from (4.14) and (4.15), as depicted in Fig. 4.2, until some suitable convergence criterion has been satisfied. Provided a good initialization $\boldsymbol{\alpha}_i^{[0]}$ and $\beta^{[0]}$ is chosen, the scheme in Fig. 4.2 converges after j iterations to the stationary point of the system of coupled equations (4.11), (4.12), (4.14), and (4.15). Then, the maximization (4.1) is performed by selecting the hypothesis that results in the highest posterior (4.2).

In practice, however, we will observe that during the re-estimation process many of the hyperparameters α_l tend to very large values, or, in fact, become numerically

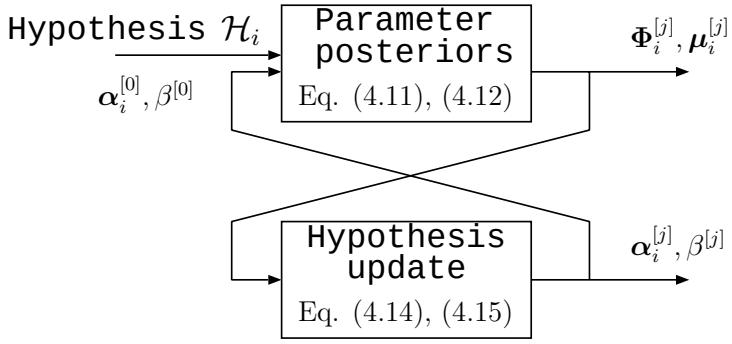


Figure 4.2: Iterative learning of the parameters; The superscript $[j]$ denotes the iteration index.

indistinguishable from infinity given the computer accuracy[4]. This enables us to approximate (4.1) by performing an on-line model selection: starting from the initial hypothesis $\mathcal{H}_0$, we prune the hyperparameters that become larger than a certain threshold as the iterations proceed by setting them to infinity. In turn, this sets the corresponding coefficient w_l to zero, thus "switching off" the lth column in the kernel matrix $\boldsymbol{K}$ and removing the delay T_l from the search space $\mathcal{T}$. This effectively implements the model selection by creating smaller hypotheses $\mathcal{H}_i < \mathcal{H}_0$ (with fewer basis functions), without performing an exhaustive search over all the possibilities. The choice of the threshold will be discussed in Section 4.3.

4.2.2 Extensions to multiple channel observations

In this subsection we extend the above analysis to multiple channel observations or multiple-antenna systems. When detecting multipath components any additional channel measurement (either in time, by observing several periods of the burst waveform $u(t)$, or in space, by using multiple-sensor antennas) can be used to increase detection quality. Of course, it is important to make sure that the multipath components are time-invariant within the observation interval and space invariant over the length of the array. The basic idea how to incorporate several channel observations is quite simple: in the original formulation each hyperparameter α_l was used to control a single weight w_l and thus the corresponding column in the design matrix $\boldsymbol{K}$. Having several channel observations we can tie them together by utilizing only a single hyperparameter for a physical multipath component present in channel observations. Usage of a single parameter in this case expresses the channel coherence property in the Bayesian framework. The corresponding graphical model that illustrates this idea for a single hyperparameter α_l is depicted in Fig. 4.3. It is interesting to note that similar ideas, though in a totally different context, were adopted to train neural networks by allowing a single hyperparameter to control a group of weights [Nea96].

[4]In the finite sample size case, however, this will only happen in the high SNR regime. Otherwise, α_l will take large but still finite values.

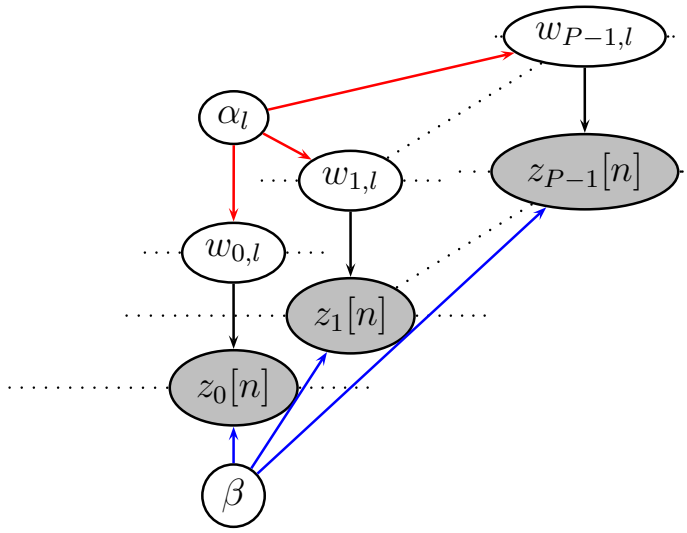


Figure 4.3: Usage of α_l in a multiple-observation discrete-time wireless channel model to represent P coherent channel measurements.

Now, let us return to (3.12). It can be seen that the weights $\boldsymbol{w}_p$ capture the structure induced by multiple antennas. However, for the moment we ignore this structure and treat the components of $\boldsymbol{w}_p$ as a wide-sense stationary (WSS) process over the individual channels, $p = 0, \ldots, P-1$. We will also allow each sensor to have a different MF. This might not necessarily be the case for wireless channel sounding, but thus a more general situation can be considered. Different matched filters result in different design matrices $\boldsymbol{K}_p$, and thus different noise covariance matrices $\boldsymbol{\Sigma_p}$, $p = 0, \ldots, P-1$. The only requirement is that the variance of the input noise remains the same and equals $N_0 = \beta^{-1}$ for all channels, so that $\boldsymbol{\Sigma_p} = N_0 \boldsymbol{\Lambda}_p$, and the noise components are statistically independent among the channels. Then, by defining

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}} &= \beta^{-1} \begin{bmatrix} \boldsymbol{\Lambda}_0 & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{\Lambda}_{P-1} \end{bmatrix}, \quad \tilde{\boldsymbol{A}} = \underbrace{\begin{bmatrix} \boldsymbol{A} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{A} \end{bmatrix}}_{P \times P \text{ block matrix}}, \\ \tilde{\boldsymbol{K}} &= \begin{bmatrix} \boldsymbol{K}_0 & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{K}_{P-1} \end{bmatrix}, \quad \tilde{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{z}_0 \\ \vdots \\ \boldsymbol{z}_{P-1} \end{bmatrix}, \quad \tilde{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{w}_0 \\ \vdots \\ \boldsymbol{w}_{P-1} \end{bmatrix}, \end{aligned} \tag{4.16}$$

we rewrite equation (3.12) as

$$\tilde{\boldsymbol{z}} = \tilde{\boldsymbol{K}} \tilde{\boldsymbol{w}} + \tilde{\boldsymbol{\xi}}. \tag{4.17}$$

A crucial point of this system representation is that the hyperparameters α_l are shared by P channels as it can be seen in the structure of the matrix $\tilde{\boldsymbol{A}}$. This will have a corresponding effect on the hyperparameter re-estimation algorithm.

From the structural equivalence of (3.12) and (4.17) we can easily infer that equations (4.11) and (4.12) are modified as follows:

$$\boldsymbol{\Phi}_p = (\boldsymbol{A} + \beta \boldsymbol{K}_p^H \boldsymbol{\Lambda}_p^{-1} \boldsymbol{K}_p)^{-1}, \tag{4.18}$$

$$\boldsymbol{\mu}_p = \beta \boldsymbol{\Phi}_p \boldsymbol{K}_p^H \boldsymbol{\Lambda}_p^{-1} \boldsymbol{z}_p, \quad p = 0, \ldots, P-1. \tag{4.19}$$

The expressions for the hyperparameter updates become a bit more complicated but are still straight-forward to compute. It is shown in the Appendix E that:

$$\alpha_l = \frac{P}{\sum_{p=0}^{P-1} \left(\Phi_{p,ll} + |\mu_{p,l}|^2 \right)}, \tag{4.20}$$

$$\begin{aligned} N_0 = \beta^{-1} = \frac{1}{NP} \Bigg(& \sum_{p=0}^{P-1} \mathrm{tr}[\boldsymbol{\Phi}_p \boldsymbol{K}_p^H \boldsymbol{\Lambda}_p^{-1} \boldsymbol{K}_p] + \\ & + \sum_{p=0}^{P-1} (\boldsymbol{z}_p - \boldsymbol{K}_p \boldsymbol{\mu}_p)^H \boldsymbol{\Lambda}_p^{-1} (\boldsymbol{z}_p - \boldsymbol{K}_p \boldsymbol{\mu}_p) \Bigg) \end{aligned} \tag{4.21}$$

where $\mu_{p,l}$ is the lth element of the MAP estimate of the parameter vector $\boldsymbol{w}_p$ given by (4.19), and $\Phi_{p,ll}$ is the lth element on the main diagonal of $\boldsymbol{\Phi}_p$ from (4.18). Comparing the latter expressions with those developed for the single channel case, we observe that (4.20) and (4.21) use multiple channels to improve the estimates of the noise spectral height and channel weight hyperparameters. They also offer more insight into the physical meaning of the hyperparameters $\boldsymbol{\alpha}$. On the one hand, the hyperparameters are used to regularize the matrix inversion (4.18), needed to obtain the MAP estimates of the parameters $w_{p,l}$ and their corresponding variances. On the other hand, they are acting as the inverse of the second noncentral moments of the coefficients $w_{p,l}$, as can be seen from (4.20).

4.3 Model selection and basis pruning

The ability to select the best model to represent the measured data is an important feature of the proposed scheme, and thus it is paramount to consider in more detail how the model selection is effectively achieved. In Section 4.2.1 we have shortly mentioned that during the learning phase many of the hyperparameters α_l's tend to large values, meaning that the corresponding weights w_l's will cluster around zero according to the prior (4.7). This will allow us to set these coefficients to zero, thus effectively pruning the corresponding basis function from the design matrix. However the question how large a hyperparameter has to grow in order to prune its corresponding basis function has not yet been discussed. In the original RVM paper [Tip01], the author suggests using a threshold α_{th} to prune the model. The empirical evidence collected by the author suggests setting the threshold to "a sufficiently

large number" (e.g., $\alpha_{\text{th}} = 10^{12}$). However, our theoretical analysis presented in the following section will show that such high thresholds are only meaningful in very high SNR regimes, or if the number of channel observations P is sufficiently large. In more general, and often more realistic, scenarios such high thresholds are absolutely impractical. Thus, there is a need to study the model selection in the context of the presented approach more rigorously.

Below, we present two methods for implementing model selection within the proposed algorithm. The first method relies on the statistical properties of the hyperparameters α_l, when the update equations (4.18), (4.19), (4.20), and (4.21) converge to a stationary point. The second method exploits the relationship that we will establish between the proposed scheme and the Minimum Description Length principle [WK85, Mac03, Grü05, BRY98], thus linking the EP to this classical model selection approach.

4.3.1 Statistical analysis of the hyperparameters in the stationary point

The decision to keep or to prune a basis function from the design matrix is based purely on the value of the corresponding hyperparameter α_l. In the following we analyze the convergence properties of the iterative learning scheme depicted in Fig. 4.2 using expressions (4.18), (4.19), (4.20), and (4.21), and the resulting distribution of the hyperparameters once convergence is achieved.

We start our analysis of the evidence parameters α_l by making some simplifications to make the derivations tractable:

- P channels are assumed.
- The same MF is used to process each of the P sensor output signals, i.e., $\boldsymbol{K}_1 = \ldots = \boldsymbol{K}_P = \boldsymbol{K}$ and $\boldsymbol{\Sigma}_1 = \ldots = \boldsymbol{\Sigma}_P = \boldsymbol{\Sigma} = \beta^{-1}\boldsymbol{\Lambda}$.
- The noise covariance matrix $\boldsymbol{\Sigma} = \beta^{-1}\boldsymbol{\Lambda}$ is known, and $\boldsymbol{B} = \boldsymbol{\Sigma}^{-1}$.
- We assume the presence of a single multipath component, i.e., $L = 1$, with known delay τ. Thus, the design matrix is given as $\boldsymbol{K} = [\boldsymbol{r}(\tau)]$, where $\boldsymbol{r}(\tau) = [R_{uu}(-\tau), R_{uu}(T_s-\tau), \ldots, R_{uu}((N-1)T_s-\tau)]^T$ is the associated basis function.
- The hyperparameter associated with this component is denoted as α.

Our goal is to consider the steady-state solution α_∞ for hyperparameter α in this simplified scenario. In this case (4.18) and (4.19) simplify to

$$\phi = (\alpha + \boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau))^{-1},$$
$$\mu_p = \phi \boldsymbol{K}^H \boldsymbol{B}\boldsymbol{z}_p = \frac{\boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{z}_p}{\alpha + \boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau)}, \quad p = 0, \ldots, P-1.$$

Inserting these two expressions into (4.20) yields

$$\alpha^{-1} = \phi + \frac{\sum_p |\mu_p|^2}{P} = \frac{1}{\alpha + \boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau)} + \frac{\sum_p \left| \frac{\boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{z}_p}{\alpha + \boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau)} \right|^2}{P}. \tag{4.22}$$

From (4.22) the solution α_∞ is easily found to be

$$\alpha_\infty = \frac{(\boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau))^2}{\frac{1}{P}\sum_p |\boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{z}_p|^2 - \boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau)}. \tag{4.23}$$

A closer look at (4.23) reveals that the right-hand side expression might not always be positive since the denominator can be negative for some values of $\boldsymbol{z}_p$. This contradicts the assumption that the hyperparameter α is positive[5]. Moreover, all terms in (4.22) are positive, and thus the resulting solution must be positive as well. A further analysis of (4.22) reveals, that (4.20) converges to (4.23) if, and only if, the denominator of (4.23) is positive:

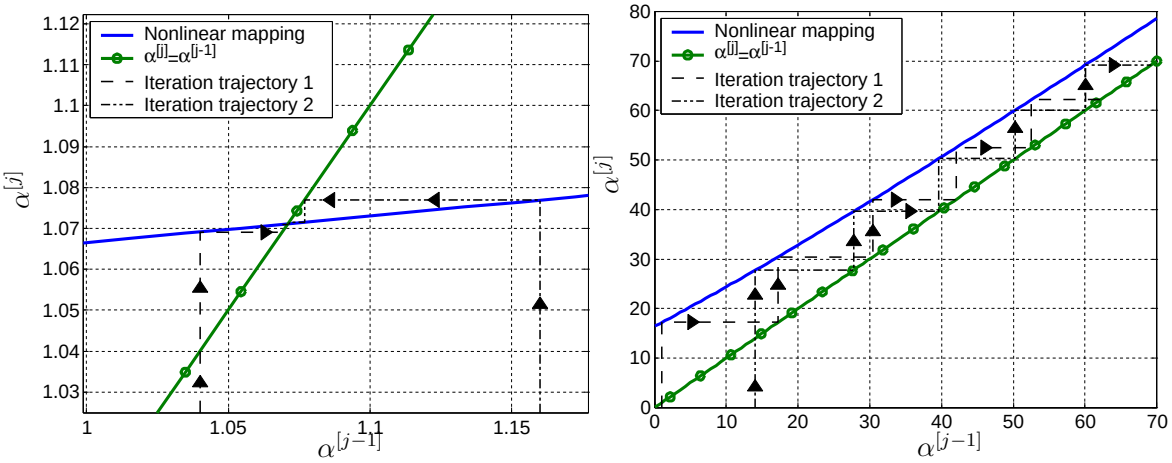
$$\frac{1}{P}\sum_p |\boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{z}_p|^2 > \boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau). \tag{4.24}$$

Otherwise, the iterative learning scheme depicted in Fig. 4.2 diverges, i.e., $\alpha_\infty = \infty$. This can be inferred by interpreting (4.20) as a nonlinear dynamical system that, at iteration j, maps $\alpha^{[j-1]}$ into the updated value $\alpha^{[j]}$. The nonlinear mapping is given by the right-hand side of (4.20), where the quantities $\boldsymbol{\Phi}_p$ and $\boldsymbol{\mu}_p$ depend on the values of the hyperparameters at iteration $j-1$. In Fig. 4.4 we show several iterations of this mapping that illustrate how the solution trajectories evolve. If condition (4.24) is satisfied, the sequence of solutions $\{\alpha^{[j]}\}$ converges to a stationary point (Fig. 4.4(a)) given by (4.23). Otherwise, $\{\alpha^{[j]}\}$ diverges (Fig. 4.4(b)). Thus, (4.22) is a stationary point only provided the condition (4.24) is satisfied:

$$\alpha_\infty = \begin{cases} \frac{(\boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau))^2}{\frac{\sum_p |\boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{z}_p|^2}{P} - \boldsymbol{r}(\tau)^H \boldsymbol{B}\boldsymbol{r}(\tau)}; & \text{condition (4.24) is satisfied} \\ \infty; & \text{otherwise.} \end{cases} \tag{4.25}$$

Practically, this means that for a given measurement $\boldsymbol{z}_p$, and known noise covariance matrix $\boldsymbol{B}$, we can immediately decide whether a given basis function $\boldsymbol{r}(\tau)$ should be included in the basis by simply checking if (4.24) is satisfied or not. A similar analysis is performed in [FT02], where the behavior of the likelihood function with respect to a single parameter is studied. The obtained convergence results coincide with ours when $P = 1$. Expression (4.24) is, however, more general and accounts for multiple channel observations and colored noise. In [FT02] the authors also suggest that testing (4.24) for a given basis function $\boldsymbol{r}(\tau)$ is sufficient to find a sparse representation and no further pruning is necessary. In other words, each basis

[5] Recall that α^{-1} is the prior variance of the corresponding parameter w. This constrains α to be nonnegative.



(a) (b)

Figure 4.4: Evolution of the two representative solution trajectories for two cases: (a) $\{\alpha^{[j]}\}$ converges, (b) $\{\alpha^{[j]}\}$ diverges.

function in the design matrix $\boldsymbol{K}$ is subject to the test (4.24) and, if the test fails, i.e., (4.24) does not hold for the basis function under test, the basis function is pruned. In case of wireless channels, however, we have observed experimentally that even in simulated high-SNR scenarios such pruning results in a significantly overestimated number of multipath components. Moreover, it can be inferred from (4.24) that, as the SNR increases, the number of functions pruned with this approach decreases, resulting in less and less sparse representations. This motivates us to perform a more detailed analysis of (4.25).

Let us slightly modify the assumptions we made earlier. We now assume that the multipath delay τ is unknown. The design matrix is constructed similarly but this time $\boldsymbol{K} = [\boldsymbol{r}_l]$, where

$$\boldsymbol{r}_l = [R_{uu}(-T_l), \ldots, R_{uu}((N-1)T_s - T_l)]^T$$

is the basis function associated with the delay $T_l \in \mathcal{T}$ used in our discrete-time model.

Under these assumptions the input signal $\boldsymbol{z}_p$ is nothing else but the basis function $\boldsymbol{r}(\tau)$ scaled and embedded in the additive complex zero-mean Gaussian noise with covariance matrix $\boldsymbol{\Sigma}$, i.e.,

$$\boldsymbol{z}_p = w_p \boldsymbol{r}(\tau) + \boldsymbol{\xi}_p. \tag{4.26}$$

Let us further assume that $w_p \in \mathbb{C}$, $p = 0, \ldots, P-1$ are unknown but fixed complex scaling factors. In further derivations we assume, unless explicitly stated otherwise, that the condition (4.24) is satisfied for the basis $\boldsymbol{r}_l$. By plugging (4.26)

into (4.23) and rearranging the result with respect to α_∞^{-1} we arrive at:

$$\begin{aligned}\alpha_\infty^{-1} =& \frac{|\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}(\tau)|^2 \sum_p |w_p|^2}{P|\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l|^2} + \frac{2\sum_p \mathrm{Re}\{w_p \boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}(\tau)\boldsymbol{\xi}_p^H \boldsymbol{B}\boldsymbol{r}_l\}}{P|\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l|^2} + \\ & \frac{\boldsymbol{r}_l^H \boldsymbol{B}\left(\sum_p \boldsymbol{\xi}_p \boldsymbol{\xi}_p^H\right)\boldsymbol{B}\boldsymbol{r}_l}{P|\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l|^2} - \frac{1}{\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l}.\end{aligned} \tag{4.27}$$

Now, we consider two scenarios. In the first scenario $\tau = T_l \in \mathcal{T}$, i.e., the discrete-time model matches the observed signal. Although unrealistic, this allows to study the properties of α_∞^{-1} more closely. In the second scenario, we study what happens if the discrete-time model does not match perfectly the measured signal. This case helps us to define how the model selection rules have to be adjusted to consider possible misalignment of the path component delays in the model.

Model match: $\tau = T_l$

In this situation, $\boldsymbol{r}_l = \boldsymbol{r}(\tau)$, and thus (4.27) can be further simplified according to

$$\alpha_\infty^{-1} = \frac{\sum_p |w_p|^2}{P} + \frac{2\sum_p \mathrm{Re}\{w_p \boldsymbol{\xi}_p \boldsymbol{B}\boldsymbol{r}_l\}}{P(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)} + \frac{\boldsymbol{r}_l^H \boldsymbol{B}\left(\sum_p \boldsymbol{\xi}_p \boldsymbol{\xi}_p^H\right)\boldsymbol{B}\boldsymbol{r}_l}{P(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)^2} - \frac{1}{\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l}, \tag{4.28}$$

where the only random quantity is the additive noise term $\boldsymbol{\xi}_p$. This allows us to study the statistical properties of the finite stationary point in (4.25).

Equation (4.28) shows how the noise and multipath component contribute to α_∞^{-1}. If all w_p are set to be zero, i.e., there is no multipath component, then $\alpha_\infty^{-1} = \alpha_\mathrm{n}^{-1}$ reflects only the noise contribution:

$$\alpha_\mathrm{n}^{-1} = \frac{\boldsymbol{r}_l^H \boldsymbol{B}\left(\sum_p \boldsymbol{\xi}_p \boldsymbol{\xi}_p^H\right)\boldsymbol{B}\boldsymbol{r}_l}{P(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)^2} - \frac{1}{\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l}. \tag{4.29}$$

On the other hand, in the absence of noise, i.e., in the infinite SNR case, the corresponding hyperparameter α_∞^{-1} includes the contribution of the multipath component[6] α_s^{-1}:

$$\alpha_\mathrm{s}^{-1} = \frac{\sum_p |w_p|^2}{P} + \frac{2\sum_p \mathrm{Re}\{w_p \boldsymbol{\xi}_p^H \boldsymbol{B}\boldsymbol{r}_l\}}{P(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)}. \tag{4.30}$$

In a realistic case, both noise and multipath component are present, and α_∞^{-1} consists of the sum of two contributions $\alpha_\infty^{-1} = \alpha_\mathrm{s}^{-1} + \alpha_\mathrm{n}^{-1}$. Both quantities α_s^{-1} and α_n^{-1} are random variables with pdf's depending on the number of channel observations P, the basis function $\boldsymbol{r}_l$, and the noise covariance matrix $\boldsymbol{\Sigma}$. In the sequel we analyze their statistical properties.

[6] Actually, the second term in the resulting expression vanishes in a perfectly noise-free case, and then $\alpha_\mathrm{s}^{-1} = \sum_p |w_p|^2 / P$.

We first consider α_{s}^{-1}. The first term on the right-hand side of (4.30) is a deterministic quantity that equals the average power of the multipath component. The second one, on the other hand, is random. The product $\mathrm{Re}\{w_p \boldsymbol{\xi}_p^H \boldsymbol{B}\boldsymbol{r}_l\}$ in (4.30) is recognized as the cross-correlation between the additive noise term and the basis function $\boldsymbol{r}_l$. It is Gaussian distributed with expectation and variance given as

$$\begin{aligned} E\left\{ \frac{2\sum_p \mathrm{Re}\{w_p \boldsymbol{\xi}_p^H \boldsymbol{B}\boldsymbol{r}_l\}}{P(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)} \right\} &= 0, \quad \text{and} \\ E\left\{ \left(\frac{2\sum_p \mathrm{Re}\{w_p \boldsymbol{\xi}_p^H \boldsymbol{B}\boldsymbol{r}_l\}}{P(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)} \right)^2 \right\} &= \frac{2\sum_p |w_p|^2}{P^2(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)}, \end{aligned} \tag{4.31}$$

respectively, where $E\{\cdot\}$ denotes the expectation operator. Thus, α_{s}^{-1} is distributed as

$$\alpha_{\mathrm{s}}^{-1} \sim \mathcal{N}\Big(\frac{\sum_p |w_p|^2}{P}, \frac{2\sum_p |w_p|^2}{P^2(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)} \Big), \tag{4.32}$$

which is a normal distribution with the mean given by the average power of the multipath component and variance proportional to this power.

Now, let us consider the term α_{n}^{-1}. In (4.29) the only random element is $\sum_{p=1}^{P} \boldsymbol{\xi}_p \boldsymbol{\xi}_p^H$. This random matrix is known to have a complex Wishart distribution [CNSS03, Goo63] with the scale matrix $\boldsymbol{\Sigma}$ and P degrees of freedom. Let us denote

$$\boldsymbol{c} = \frac{\boldsymbol{B}\boldsymbol{r}_l}{\sqrt{P}\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l} \quad \text{and} \quad x = \boldsymbol{c}^H \sum_{p=1}^{P} \boldsymbol{\xi}_p \boldsymbol{\xi}_p^H \boldsymbol{c}. \tag{4.33}$$

It can be shown that x is Gamma-distributed, i.e., $x \sim \mathcal{G}(P, \sigma_c^2)$, with the shape parameter P and the scale parameter σ_c^2 given as

$$\sigma_c^2 = \boldsymbol{c}^H \boldsymbol{\Sigma} \boldsymbol{c} = \frac{1}{P(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)}.$$

The pdf of x reads

$$p(x|P, \sigma_c^2) = \frac{x^{P-1}}{\Gamma(P)(\sigma_c^2)^P} \mathrm{e}^{-x/\sigma_c^2}. \tag{4.34}$$

The mean and the variance of x are easily computed to be

$$\begin{aligned} \mathrm{E}\{x\} &= P\sigma_c^2 = \frac{1}{\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l}, \\ \mathrm{Var}\{x\} &= P(\sigma_c^2)^2 = \frac{1}{P(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)^2}. \end{aligned} \tag{4.35}$$

Taking the term $-1/(\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l)$ in (4.29) into account, we see that the resulting variable $\tilde{\alpha}_{\mathrm{n}}^{-1}$ is a zero mean random variable with the pdf

$$p_{\tilde{\alpha}_{\mathrm{n}}^{-1}}(x|P, \sigma_c^2) = \frac{(x - \mathrm{E}\{x\})^{P-1}}{\Gamma(P)(\sigma_c^2)^P} \mathrm{e}^{-(x-\mathrm{E}\{x\})/\sigma_c^2}, \tag{4.36}$$

which is equivalent to (4.34), but shifted so as to correspond to a zero-mean random variable. However, it is known that only positive values of α_{n}^{-1} occur in practice. The probability mass of the negative part of (4.36) equals the probability that the condition (4.24) is not satisfied and the resulting α_{∞}^{-1} eventually diverges to infinity and is pruned. Taking this into account the pdf of α_{n}^{-1} reads

$$p_{\alpha_{\mathrm{n}}^{-1}}(x) = P_{\mathrm{n}}\delta(x) + (1 - P_{\mathrm{n}})\mathcal{I}^{+}(x)\tilde{p}_{\alpha_{\mathrm{n}}^{-1}}(x|P, \sigma_c^2), \tag{4.37}$$

where $\delta(\cdot)$ denotes a Dirac delta function, P_{n} is defined as

$$P_{\mathrm{n}} = \int_{-1/(\boldsymbol{r}_l^H \boldsymbol{B} \boldsymbol{r}_l)}^{0} \tilde{p}_{\alpha_{\mathrm{n}}^{-1}}(x|P, \sigma_c^2)dx,$$

and $\mathcal{I}^{+}(\cdot)$ is the indicator function of the set of positive real numbers:

$$\mathcal{I}^{+}(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0. \end{cases}$$

A closer look at (4.37) shows that as P increases the variance of the Gamma distribution decreases, with α_{n}^{-1} concentrating at zero. In the limiting case as $P \to \infty$, (4.37) converges to a Dirac delta function localized at zero, i.e., $\alpha_{\mathrm{n}} = \infty$. This allows natural pruning of the corresponding basis function. This situation is equivalent to averaging out the noise, as the number of channel observations grows. Practically, however, P stays always finite, which means that (4.32) and (4.37) have a certain finite variance.

The pruning problem can be approached from the perspective of classical detection theory. To prune a basis function, we have to decide if the corresponding value of α^{-1} has been generated by the noise distribution (4.37), i.e., the *null hypothesis*, or by the pdf of $\alpha_{\mathrm{s}}^{-1} + \alpha_{\mathrm{n}}^{-1}$, i.e., the *alternative hypothesis*. Computing the latter is difficult. The problem might be somewhat relaxed by taking the assumption that α_{s}^{-1} and α_{n}^{-1} are statistically independent. However proving the plausibility of this assumption is difficult. Even if we were successful in finding the analytical expression for the pdf of the alternative hypothesis, such model selection approach is hampered by our inability to evaluate (4.32) since the gains w_p's are not known *a priori*. However, we can still use (4.37) to select a threshold.

Recall that the presented algorithm allows to learn (estimate) the noise spectral height $N_0 = \beta^{-1}$ from the measurements. Assuming that we know β, and, as a consequence, the whole matrix $\boldsymbol{B}$ then, for any basis function $\boldsymbol{r}_l$ in the design matrix $\boldsymbol{K}$ and the corresponding hyperparameter α_l, we can decide with an *a priori* specified probability ρ that α_l is generated by the distribution (4.37). Indeed, let $\alpha_{\mathrm{th}}^{-1}$ be a ρ-quantile of (4.37) such that the probability $P(\alpha^{-1} \leq \alpha_{\mathrm{th}}^{-1}) = \rho$. Since (4.37) is known exactly, we can easily compute $\alpha_{\mathrm{th}}^{-1}$ and prune all the basis functions for which $\alpha_l^{-1} \leq \alpha_{\mathrm{th}}^{-1}$.

Model mismatch: $\tau \neq T_l$

The analysis performed above relies on the knowledge that the true multipath delay τ belongs to $\mathcal{T}$. Unfortunately, this is often unrealistic and the model mismatch $\tau \notin \mathcal{T}$ must be considered. To be able to study how the model mismatch influences the value of the hyperparameters we have to make a few more assumptions.

Let us for simplicity select the model delay T_l to be a multiple of the chip period T_p. We will also need to assume a certain shape of the correlation function $R_{uu}(t)$ to make the whole analysis tractable.

Schematically, the model mismatch can be visualized as shown in Fig. 4.5. The fixed basis function $\boldsymbol{r}$ is positioned at the delay $T_l \in \mathcal{T}$. The true multipath component $\boldsymbol{r}(\tau)$ is not necessarily aligned with the basis $\boldsymbol{r}$.

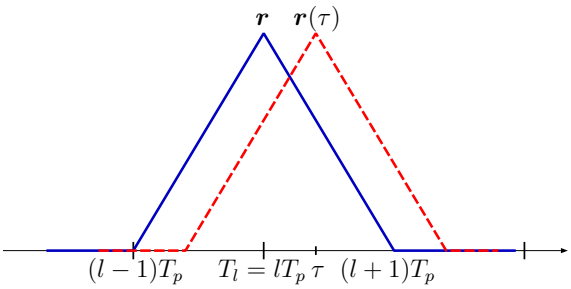


Figure 4.5: Model mismatch.

Just as in the previous case, we can split the expression (4.27) into the the multipath component contribution α_s^{-1}

$$\alpha_s^{-1} = \frac{|\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}(\tau)|^2 \sum_p |w_p|^2}{P|\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l|^2} + \frac{2\sum_p \mathrm{Re}\{w_p \boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}(\tau)\boldsymbol{\xi}_p^H \boldsymbol{B}\boldsymbol{r}_l\}}{P|\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l|^2}, \tag{4.38}$$

and the same noise contribution α_n^{-1} defined in (4.29).

It can be seen that the *weighted* and normalized correlation

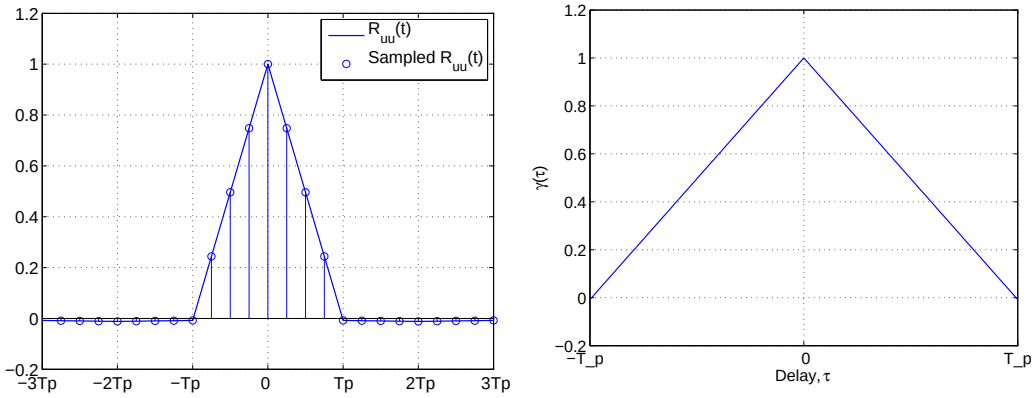
$$\gamma(\tau) = \frac{\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}(\tau)}{\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}_l} \tag{4.39}$$

makes (4.38) differ from (4.30), and as such it is the key to the analysis of the model mismatch. Note that this function is bounded as

$$|\gamma(\tau)| \leq 1,$$

with equality following only if $\tau = T_l$. Note also that in our case for $|\tau - T_l| < T_p$ the correlation $\gamma(\tau)$ is strictly positive.

Due to the properties of the sounding sequence $u(t)$, the magnitude of $R_{uu}(t)$ for $|t| > T_p$ is sufficiently small and in our analysis of model mismatch can be safely assumed to be zero. Furthermore, if $\boldsymbol{r}_l$ is chosen to coincide with the multiple of the sampling period $T_l = lT_s$, then it follows from (3.15) that the product $\boldsymbol{r}_l^H \boldsymbol{B} =$



(a) (b)

Figure 4.6: Evaluated correlation functions a)$R_{uu}(t)$ and b) $\gamma(\tau)$ for the linear approximation case.

$\boldsymbol{r}_l^H \boldsymbol{\Sigma}^{-1} = \beta \boldsymbol{e}_l^H$ is a vector with all elements being zero except the lth element, which is equal to β. Thus, the product $\boldsymbol{r}_l^H \boldsymbol{B}\boldsymbol{r}(\tau)$ for $|\tau - T_l| < T_p$ must have a form identical to that of the correlation function $R_{uu}(t)$ for $|t| < T_p$. It follows that when $|\tau - T_l| \geq T_p$ the correlation $\gamma(\tau)$ can be assumed to be zero, and it makes sense to analyze (4.38) only when $|\tau - T_l| < T_p$.

To proceed further, we need to assume the shape of the correlation function $R_{uu}(t)$. We will consider consequently two cases: the linear approximation, when the main lobe of the $R_{uu}(t)$ is approximated by a triangular pulse with the width equal to $2T_p$, and the cosine approximation when the main lobe of the $R_{uu}(t)$ is approximated by the raised cosine function with the width $2T_p$.

Let us start with the linear approximation. This approximation is exact when the shaping pulse used to form the sounding signal $u(t)$ is a simple rectangular pulse with the pulse width T_p. In Fig. 4.6 we show the evaluated correlation function $R_{uu}(t)$, as well as the corresponding correlation function $\gamma(\tau)$ corresponding to this case.

Since the true value of τ is unknown, we define a probability density over it. It is reasonable to assume τ to be uniformly distributed in the interval $[-(l-1)T_p, (l+1)T_p]$. This in turn induces the corresponding distribution over $\gamma(\tau)$, and, correspondingly, over $|\gamma(\tau)|^2$, which enters the first term on the right-hand side of (4.38).

In the linear approximation case it can be easily found that

$$\gamma(\tau) \sim \mathcal{U}(0,1), \tag{4.40}$$

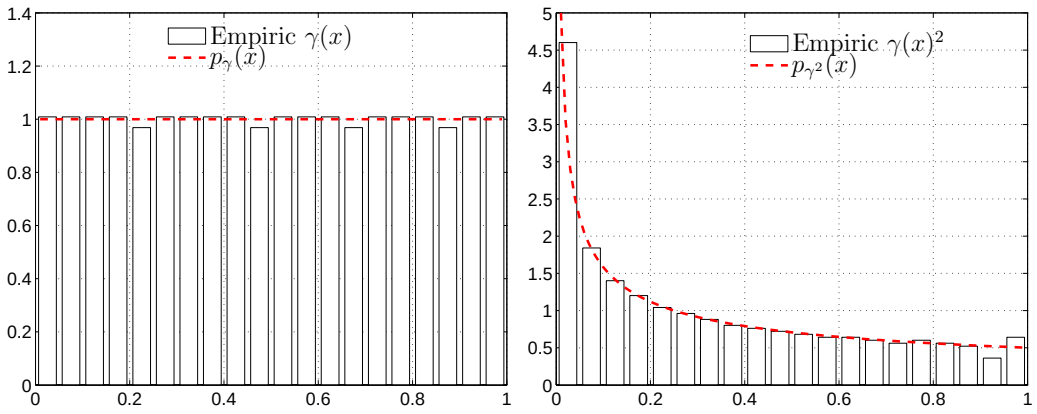
where $\mathcal{U}(0,1)$ is a uniform distribution over the interval $[0,1]$ with the corresponding pdf $p_\gamma(x) = 1$.

The distribution of the $|\gamma(\tau)|^2$ can also be easily found. The corresponding pdf

$p_{\gamma^2}(x)$ is given as

$$p_{\gamma^2}(x) = \frac{1}{2\sqrt{x}}. \tag{4.41}$$

In Fig. 4.7 we plot the empirical (based on Monte Carlo simulations) and theoretical pdf's of the $\gamma(\tau)$ and $\gamma(\tau)^2$ terms, respectively, for the linear approximation case.



(a) (b)

Figure 4.7: Comparison between the empirical and theoretical pdf's of the a)$\gamma(\tau)$ and b)$|\gamma(\tau)|^2$ for the linear approximation case.

Let us now consider the cosine approximation case. This approximation makes sense if the sounding pulse $p(t)$ defined in Sec. 3.1.1 is a square root raised cosine pulse (Fig. 4.8).

Again, assuming that τ is uniformly distributed in the interval $[-(l-1)T_p, (l+1)T_p]$, it can be shown that

$$\gamma(\tau) \sim \mathcal{B}(0.5, 0.5), \tag{4.42}$$

where $\mathcal{B}(0.5, 0.5)$ is a Beta distribution [EHBP00] with both distribution parameters being equal to 1/2. The corresponding pdf $p_\gamma(x)$ in this case is given as

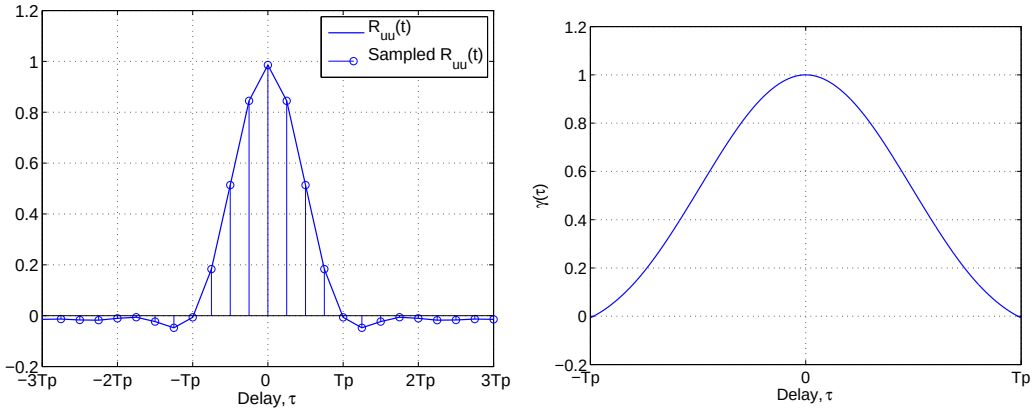
$$p_\gamma(x) = \frac{1}{B(0.5, 0.5)} x^{-\frac{1}{2}}(1-x)^{-\frac{1}{2}}, \tag{4.43}$$

where $B(\cdot,\cdot)$ is a Beta-function [AS72] with $B(0.5, 0.5) = \pi$.

It is also straight-forward to compute the pdf of the $\gamma(\tau)^2$ term:

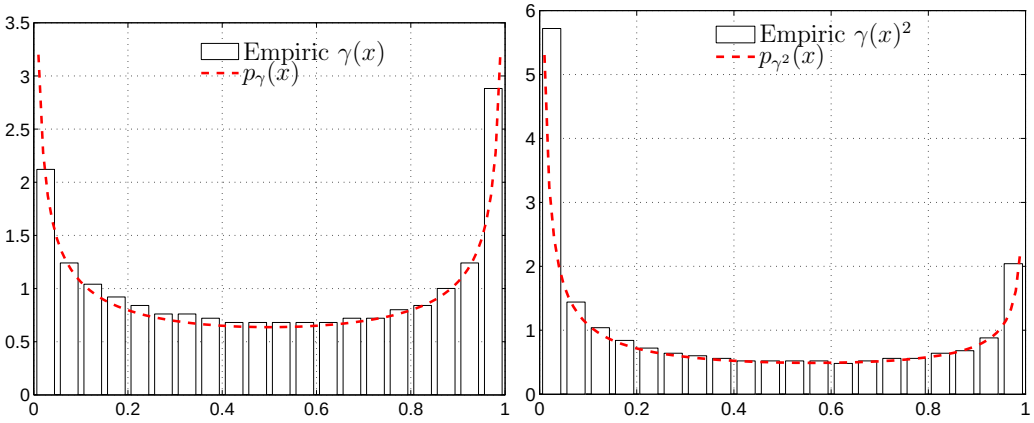
$$p_{\gamma^2}(x) = \frac{1}{\pi} x^{-\frac{3}{4}}(1-\sqrt{x})^{-\frac{1}{2}}. \tag{4.44}$$

The corresponding empirical as well as theoretical pdf's that correspond to this case are shown in Fig. 4.9.



(a) (b)

Figure 4.8: Evaluated correlation functions a)$R_{uu}(t)$ and b) $\gamma(\tau)$ for the cosine approximation case.



(a) (b)

Figure 4.9: Comparison between the empirical and theoretical pdf's of the a)$\gamma(\tau)$ and b)$|\gamma(\tau)|^2$ for the cosine approximation case.

It is interesting to note that similar model mismatch analysis was done in [KK99], though in totally different context of speech coding.

Now we have to find out how this information can be utilized to design an appropriate threshold. In the case of perfectly matched model the threshold is selected based on the noise distribution (4.37). In the case of model mismatch, the term (4.38) measures the amount of the interference resulting from the model imperfection.

Indeed, if $|\tau - T_l| \geq T_p$, then the resulting $\gamma(\tau) = 0$, and thus $\alpha_s^{-1} = 0$. The

corresponding evidence parameter α_∞^{-1} is then equal to the noise contribution α_n^{-1} only and will be pruned using the method we described for the matched model case. If however, $|\tau - T_l| < T_p$, then a certain fraction of α_s^{-1} will be added to the noise contribution α_n^{-1}, thus causing the interference. In order to be able to take this interference into account and adjust the threshold accordingly, we propose the following approach.

The amount of interference added is measured by the magnitude of α_s^{-1} in (4.38). It consists of two terms: the first one is the multipath power, scaled by the factor $\gamma(\tau)^2$:

$$\gamma(\tau)^2 \frac{\sum_p |w_p|^2}{P}. \tag{4.45}$$

The second term is a cross product between the multipath component and the additive noise, scaled by $\gamma(\tau)$:

$$\gamma(\tau) \frac{2 \sum_p \mathrm{Re}\{w_p \boldsymbol{\xi}_p^H \boldsymbol{B} \boldsymbol{r}_l\}}{P(\boldsymbol{r}_l^H \boldsymbol{B} \boldsymbol{r}_l)}. \tag{4.46}$$

Both terms have the same physical interpretation as in (4.30), but with scaling factors $\gamma(\tau)$ depending on the true value of τ.

We see that in (4.38) there are quite a few unknowns: we do not know the true multipath delay τ, the multipath gains w_p, as well as the instantaneous noise value $\boldsymbol{\xi}$. To be able to circumvent this uncertainties, we consider the large sample size case, i.e, $P \to \infty$ and invoke the law of large numbers to approximate (4.45) and (4.46) by their expectations.

First of all, using (4.31) it is easy to see that

$$E\left\{ \gamma(\tau) \frac{2 \sum_p \mathrm{Re}\{w_p \boldsymbol{\xi}_p^H \boldsymbol{B} \boldsymbol{r}_l\}}{P(\boldsymbol{r}_l^H \boldsymbol{B} \boldsymbol{r}_l)} \right\} = 0.$$

The other term (4.45) converges to $\gamma(\tau)^2 \alpha_p^{-1}$ as P grows, where α_p^{-1} is the power of the multipath component. So, even in the high SNR regime and infinite number of channel observations P the term (4.45) does not go to zero. In order to assess how large it is, we approximate the gains of the multipath component w_p by the corresponding MAP estimate μ_p obtained with (4.19).

The correlation function $\gamma(\tau)$ can also be taken into account. Since we know the distributions of both $\gamma(\tau)$ and $\gamma(\tau)^2$, we can summarize these by the corresponding mean values. In fact, we will need the mean only for $\gamma(\tau)^2$ since it enters the irreducible part of α_s^{-1}.

For the cosine approximation case it is easily computed as:

$$E\{\gamma(\tau)^2\} = \int_0^1 \frac{x}{\pi} x^{-\frac{3}{4}} (1 - \sqrt{x})^{-\frac{1}{2}} dx = \frac{B(2.5, 0.5)}{\pi} = \frac{3}{8}. \tag{4.47}$$

Having obtained the mean, we can approximate the interference $\hat{\alpha}_s^{-1}$ due to the model mismatch as

$$\hat{\alpha}_s^{-1} = 3/8 \times \frac{\sum_{p=0}^{P-1} |\mu_p|^2}{P}, \tag{4.48}$$

The final threshold that accounts for the model mismatch is then obtained as

$$\hat{\alpha}_{\text{th}}^{-1} = \hat{\alpha}_s^{-1} + \alpha_{\text{th}}^{-1}, \tag{4.49}$$

where α_{th}^{-1} is the threshold developed earlier for the matched model case.

4.3.2 Improving the learning algorithm to cope with the model selection

Under the light of the model selection strategy considered here we anticipate two major problems arising with the learning algorithm discussed in Section 4.2. The first one is the estimation of the channel parameters that requires computation of the posterior (4.18). Even for the modest sizes of the hypothesis $\mathcal{H}_i$ (from 100 to 200 basis functions), the matrix inversion is computationally very intensive. This issue becomes even more critical if we consider a hardware implementation of the estimation algorithm. The second problem arises due to the non-vanishing correlation between the basis vectors $\boldsymbol{r}_l$ constituting the design matrix $\boldsymbol{K}$. A very undesirable consequence of this correlation is that the evidence parameters α_l associated with these vectors become also correlated, and thus no longer represent the contribution of a single basis function. As the consequence the developed model selection rules are no longer applicable.

It is, however, possible to circumvent these two difficulties by modifying the learning algorithm as discussed below. The basic idea consists in estimating the channel parameters for each basis independently. In other words, instead of solving (4.18), (4.19), (4.20), and (4.21) jointly for all L basis functions, we find a solution for each basis vector separately. First, the new data vector $\boldsymbol{x}_{p,l}$ for the lth basis is computed as

$$\boldsymbol{x}_{p,l} = \boldsymbol{z}_p - \sum_{k=1,k\neq l}^{L} \boldsymbol{r}_k \mu_{p,l}. \tag{4.50}$$

This new data vector $\boldsymbol{x}_{p,l}$ now contains the information relevant to the basis $\boldsymbol{r}_l$ only. It is then used to update the corresponding posterior statistics as well as evidence parameters exclusively for the lth basis as follows:

$$\Phi_l = (\alpha_l + \beta \boldsymbol{r}_l^H \boldsymbol{\Lambda}^{-1} \boldsymbol{r}_l)^{-1}, \tag{4.51}$$

$$\mu_{p,l} = \beta \Phi_l \boldsymbol{r}_l^H \boldsymbol{\Lambda}^{-1} \boldsymbol{x}_{p,l}, \quad p = 0, \ldots, P-1. \tag{4.52}$$

Note that expressions (4.51) and (4.52) are now scalars, unlike their matrix counterparts (4.18) and (4.19). Similarly, we update the evidence parameters as

$$\alpha_l = \frac{P}{\sum_{p=1}^{P} \left(\Phi_l + |\mu_{p,l}|^2 \right)}. \tag{4.53}$$

Updates (4.51), (4.52), and (4.53) are performed for all L components sequentially. Once all components are updated, we update the noise hyperparameter N_0:

$$N_0 = (\beta^{-1}) = \frac{1}{NP}\left(\sum_{p=1}^{P} \mathrm{tr}[\boldsymbol{\Phi}(\boldsymbol{K})^H \boldsymbol{\Lambda}^{-1} \boldsymbol{K}] + \right. \\ \left. + \sum_{p=1}^{P} (\boldsymbol{z}_p - \boldsymbol{K}\boldsymbol{\mu}_p)^H \boldsymbol{\Lambda}^{-1} (\boldsymbol{z}_p - \boldsymbol{K}\boldsymbol{\mu}_p) \right). \tag{4.54}$$

The above updating procedures constitute one single iteration of the modified learning algorithm. This iteration is repeated until some suitable convergence criterion is satisfied. Note that the procedure described here is an instance of the SAGE algorithm. This opens a potential to unite both SAGE and Evidence Procedure, allowing to implement simultaneous parameter and model order estimation within the SAGE framework.

The above iterative method is an instance of a general technique called successive interference cancellation. It allows solving both anticipated problems. First of all, there is no need to compute matrix inversion at each iteration. Second, the obtained values of α now reflect the contribution of a single basis function only, since they were estimated while the contribution of other bases was canceled in (4.50).

Now, at the end of each iteration, once the new value of the noise is obtained using (4.54), we can decide to prune some of the components, as described in Section 4.3.1.

4.3.3 MDL principle and Evidence Procedure

The goal of this section is to establish a relationship between the classical information-theoretic criteria for model selection, such as Minimum Description Length (MDL) [WK85, Mac03, Grü05], and the Evidence Procedure discussed here.

The MDL criterion was originally formulated from the perspective of coding theory as a solution to the problem of balancing the code complexity and the resulting length of the encoded data. Formally, if the length of the data $\mathcal{Z}$ encoded with the code $\mathcal{H}_i$ is given as $Len(\mathcal{Z}|\mathcal{H}_i)$, and the corresponding code complexity is given as $C(\mathcal{H}_i)$, then the optimal code in the MDL-sense is achieved as the solution to the following optimization problem:

$$\hat{\mathcal{H}} = \underset{\mathcal{H}_i}{\mathrm{argmin}} \Big\{ Len(\mathcal{Z}|\mathcal{H}_i) + C(\mathcal{H}_i) \Big\}.$$

This concept can naturally be transferred to general model selection schemes by treating $Len(\mathcal{Z}|\mathcal{H}_i)$ as a measure of model performance and $C(\mathcal{H}_i)$ as measure of the model complexity. In the context of Maximum Likelihood (ML) estimation, model performance arises naturally as the likelihood function evaluated at its maximum, i.e., at the ML estimates of the parameters. Formalizing the model complexity term, on the other hand, is not a trivial task. This term has been approximated under

some very specific assumptions in the seminal paper [Ris78] by Rissannen:

$$\mathrm{DL}(\mathcal{H}_i) = \underbrace{-\log(p(\boldsymbol{z}|\boldsymbol{w}_{ML}^{\mathcal{H}_i}))}_{\text{model performance}} + \underbrace{0.5L\log(N)}_{\text{model complexity}}, \tag{4.55}$$

where $\mathrm{DL}(\mathcal{H}_i)$ is the so-called description length of the model $\mathcal{H}_i$, L is the assumed model order (number of parameters), N is the number of observed data samples, $\boldsymbol{w}_{ML}^{\mathcal{H}_i}$ is an L-dimensional ML estimate of the model parameters under hypothesis $\mathcal{H}_i$, and $\boldsymbol{z}$ is the observed data. Thus, joint model and parameter estimation schemes should aim at minimizing the DL so as to find the compromise between the model fit (likelihood) and the number of the parameters involved (complexity). Equation (4.55) has been used ever since in many signal processing applications involving model selection. However, for general problems the complexity term $0.5L\log(N)$ might not always be adequate. In order to account explicitly for the complexity of a particular model structure, a quantity called *stochastic complexity* has been introduced (see [Grü05, Ris96, BRY98]).

The Bayesian interpretation of the stochastic complexity term obtained for likelihood functions from an exponential family (see [Grü05] for more details) is of particular interest for our problem at hand.

$$\begin{aligned}\mathrm{DL}(\mathcal{H}_i) = &\underbrace{-\log(p(\boldsymbol{z}|\boldsymbol{w}_{MAP},\mathcal{H}_i)}_{\text{model performance}} + \\ &\underbrace{\frac{L}{2}\log\frac{N}{2\pi} - \log(p(\boldsymbol{w}_{MAP}|\mathcal{H}_i)) + \log(\sqrt{|\boldsymbol{I}_1(\boldsymbol{w}_{MAP})|})}_{\text{stochastic complexity}}.\end{aligned} \tag{4.56}$$

Here $\boldsymbol{I}_1(\boldsymbol{w}_{MAP})$ is the Fisher information matrix of a single sample evaluated at the MAP estimate of the model parameter vector, and $p(\boldsymbol{w}_{MAP}|\mathcal{H}_i)$ is the corresponding prior for this vector. We will now show, that the Evidence Procedure employed in our model selection scheme results in a very similar expression.

Let us once again come back to the evidence term (4.13). To exemplify the main message that we want to convey here, we will compute the integral in (4.13) differently. For each model hypothesis defined as in Section 4.2, let us define $\Delta(\boldsymbol{w}_i) = -\log(p(\boldsymbol{z}|\boldsymbol{w}_i,\beta_i)) - \log(p(\boldsymbol{w}_i|\boldsymbol{\alpha}_i))$. Then equation (4.13) can be expressed as

$$p(\boldsymbol{z}|\boldsymbol{\alpha}_i,\beta_i) = \int \exp(-\Delta(\boldsymbol{w}_i))d\boldsymbol{w}_i. \tag{4.57}$$

In our case $\Delta(\boldsymbol{w}_i)$ is known to be quadratic, since both $p(\boldsymbol{z}|\boldsymbol{w}_i,\beta_i)$ and $p(\boldsymbol{w}_i|\boldsymbol{\alpha}_i)$ are Gaussian. Now, we expand $\Delta(\boldsymbol{w}_i)$ in a Taylor series around the argument that maximizes the integrand in (4.57), which is the MAP estimate of the model parameters $\boldsymbol{\mu}_i$ given in (4.12). This technique is also known as Laplace's method

[Mac03, ch. 27]. Proceeding in this way we obtain

$$\begin{aligned}\Delta(\boldsymbol{w}_i) = \Delta(\boldsymbol{\mu}_i) + (\boldsymbol{w}_i - \boldsymbol{\mu}_i)\frac{\partial \Delta(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_i}\bigg|_{\boldsymbol{w}_i=\boldsymbol{\mu}_i} + \\ \frac{1}{2}(\boldsymbol{w}_i - \boldsymbol{\mu}_i)^H \frac{\partial^2 \Delta(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_i \partial \boldsymbol{w}_i^H}\bigg|_{\boldsymbol{w}_i=\boldsymbol{\mu}_i} (\boldsymbol{w}_i - \boldsymbol{\mu}_i)\end{aligned} \tag{4.58}$$

It is easily verified that

$$\frac{\partial^2 \Delta(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_i \partial \boldsymbol{w}_i^H}\bigg|_{\boldsymbol{w}_i=\boldsymbol{\mu}_i} = 2\boldsymbol{\Phi}_i^{-1} \text{ and } \frac{\partial \Delta(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_i}\bigg|_{\boldsymbol{w}_i=\boldsymbol{\mu}_i} = \mathbf{0}. \tag{4.59}$$

By inserting the right-hand side of (4.58) in (4.57) and making use of (4.59) we arrive to

$$p(\boldsymbol{z}|\boldsymbol{\alpha}_i, \beta_i) = \exp(-\Delta(\boldsymbol{\mu}_i)) \int \exp(-(\boldsymbol{w}_i - \boldsymbol{\mu}_i)^H \boldsymbol{\Phi}_i^{-1}(\boldsymbol{w}_i - \boldsymbol{\mu}_i)) d\boldsymbol{w}_i \tag{4.60}$$

which can be easily integrated. For a hypothesis $\mathcal{H}_i$ with $L = |\mathcal{P}(i)|$ parameters it equals

$$p(\boldsymbol{z}|\boldsymbol{\alpha}_i, \beta_i) = \exp(-\Delta(\boldsymbol{\mu}_i))\pi^L |\boldsymbol{\Phi}_i|. \tag{4.61}$$

By taking the logarithm of (4.61) and changing the sign of the resulting expression we arrive at the final expression for the negative log −evidence

$$\begin{aligned}-\log(p(\boldsymbol{z}|\boldsymbol{\alpha}_i, \beta_i)) = -\log(p(\boldsymbol{z}|\boldsymbol{\mu}_i, \beta_i)) - \\ \log(p(\boldsymbol{\mu}_i|\boldsymbol{\alpha}_i)) - L\log(\pi) - \log(|\boldsymbol{\Phi}_i|).\end{aligned} \tag{4.62}$$

By noting that $\boldsymbol{\Phi}_i$ has been computed using N data samples, and that $\log(|\boldsymbol{\Phi}_i/N|) = \log(|\boldsymbol{I}_1^{-1}(\boldsymbol{\mu}_i)|)$, we rewrite (4.62) as

$$\begin{aligned}\mathrm{DL}(\mathcal{H}_i) = \underbrace{-\log(p(\boldsymbol{z}|\boldsymbol{\mu}_i, \beta_i))}_{\text{model performance}} + \\ \underbrace{L\log(\frac{N}{\pi}) - \log(p(\boldsymbol{\mu}_i|\boldsymbol{\alpha}_i)) + \log(|\boldsymbol{I}_1(\boldsymbol{\mu}_i)|)}_{\text{model complexity}},\end{aligned} \tag{4.63}$$

We note that (4.56) and (4.63) are essentially similar, with the distinction that the latter accounts for complex data. Thus we conclude that maximizing evidence (or minimizing the negative log −evidence) is equivalent to minimizing the DL (see Fig. 4.10).

On the one hand, the model performance gets better as we use higher model orders, with smallest negative likelihood achieved for the full model $\mathcal{H}_0$. On the

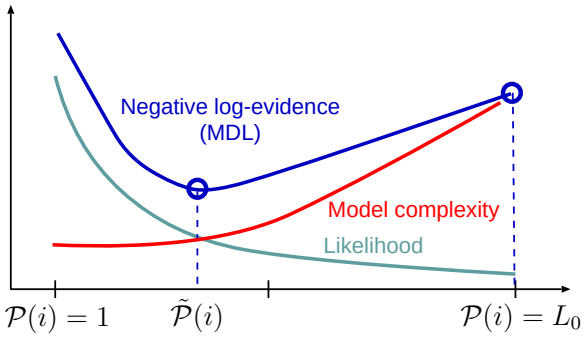


Figure 4.10: Model selection by evidence evaluation.

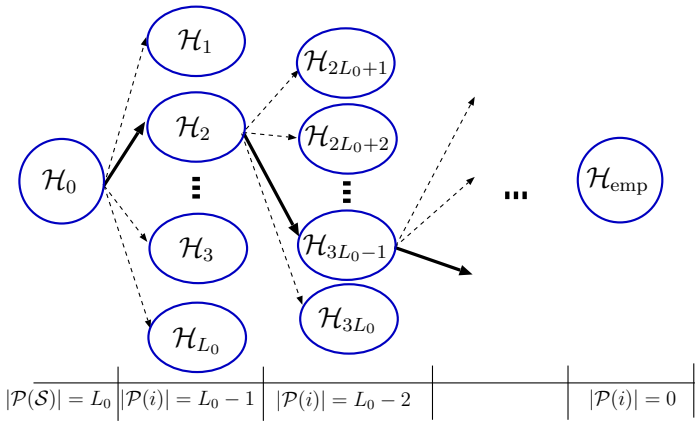


Figure 4.11: Model selection by evidence evaluation.

other hand, the model complexity grows as we consider larger hypotheses, achieving a maximum at $\mathcal{H}_0$. The balance between the two terms corresponds to the model $\tilde{\mathcal{H}}$ with the highest evidence.

Let us now consider how this can be exploited in our case. Within the Evidence Procedure framework we always start with the full hypothesis $\mathcal{H}_0$, which includes all basis functions. We then prune some of the basis functions from the initial hypothesis according to the specific rules. In Section 4.3.1 we developed a threshold that allows optimal pruning of the basis functions. However, the MDL principle can also be used for this purpose. In general, the MDL concept assumes presence of multiple *estimated* models. The model that minimizes the DL functional is then picked as the optimal one. In our case, evaluating the DL functional for all possible hypotheses $\mathcal{H}_i$ is way to complex. In order to make this procedure more efficient, we can exploit the estimated evidence information.

Consider the graph shown in Fig. 4.11. Each node on the graph corresponds to a certain hypothesis $\mathcal{H}_i$ consisting of $|\mathcal{P}_i|$ basis functions. An edge emanating from a node is associated with a certain basis function from the hypothesis $\mathcal{H}_i$. Should the path through the graph include this edge, the corresponding basis function would be pruned, leading to a new hypothesis with fewer basis functions. Clearly, the optimal

path through the graph should be the one that minimizes the DL criterion. Now, let us propose a strategy to find the optimal model without evaluating all possible paths through the graph.

At the initial stage, we start in the leftmost node, which corresponds to the full hypothesis $\mathcal{H}_0$. We then proceed with the learning algorithm using the iterative scheme depicted in Fig. 4.2 to obtain the estimates of the evidence parameters $\boldsymbol{\alpha}_0$ for each basis function in $\mathcal{H}_0$. Once convergence is achieved, we evaluate the corresponding description length DL_0 for this hypothesis using (4.63). Since the optimal path should decrease the DL, the hypothesis at the next stage $\mathcal{H}_i$ is selected by moving along the edge that corresponds to the basis function with the largest value of α (i.e., the basis function with the smallest evidence). For the newly selected hypothesis $\mathcal{H}_i$ we again estimate the evidence parameters $\boldsymbol{\alpha}_i$ and the corresponding description length DL_i. If $\mathrm{DL}_0 < \mathrm{DL}_i$, then the hypothesis $\mathcal{H}_0$ achieves the minimum of the description length and it is then selected as the solution. Otherwise, i.e., if $\mathrm{DL}_0 > \mathrm{DL}_i$, we continue along the graph, each time pruning a basis function with the smallest evidence and comparing the description length at each stage. We proceed so until the DL does not decrease any more, or until we stop at the last node that has no basis functions at all. Such an empty hypothesis corresponds to the case when there is no structure in the observed data at all. In other words it corresponds to the case when the algorithm failed to find any multipath components.

4.4 Application of the RVM to wireless channels

The application of the proposed channel estimation scheme coupled with the considered model selection approach requires two major components: 1) it needs a proper construction of the kernel design matrix that is dense enough to ensure good delay resolution, and 2) the iterative nature of the algorithm requires a good initialization.

The construction of the design matrix $\boldsymbol{K}$ can be done with various approaches, depending on how much *a priori* information we have about the possible positions of the multipath components. The columns of the matrix $\boldsymbol{K}$ contain the shifted versions of the kernel $R_{uu}(nT_s - T_l)$, $l = 1, \ldots, L_0$, where T_l are the possible positions of the multipath components that form the search space $\mathcal{T}$. The delays T_l can be selected uniformly to cover the whole delay span or might be chosen so as to sample some areas of the impulse response more densely, where multipath components are likely to appear. Note that the delays T_l are not constrained to fall on a regular grid. The power-delay profile (PDP) may be a good indicator of how to place the multipath components.

Initialization of the model hyperparameters can also be done quite effectively. In the sequel we propose two different initialization techniques.

The simplest one consists in evaluating the condition (4.24) for all the basis functions in the already created design matrix $\boldsymbol{K}$. For those basis functions that satisfy condition (4.24), the corresponding evidence parameter is initialized using (4.23). Other basis functions are removed from the design matrix $\boldsymbol{K}$. Such initialization

assumes that there is no interference between the neighboring basis functions. It makes sense to employ it when the minimal spacing between the elements in $\mathcal{T}$ is at most half the duration of the sounding pulse T_p.

Alternatively it is better to use independent evidence initialization. This type of initialization is in fact coupled with the construction of the design matrix $\boldsymbol{K}$ and relies on the successive interference cancellation scheme discussed in the Section 4.3.2. To make the procedure work, we need to set the initial channel coefficients to zero, i.e., $\boldsymbol{\mu}_p \equiv 0$. The basis vectors $\boldsymbol{r}_l$ are computed as usual according to the delay search space $\mathcal{T}$. The initialization iterations start by computing (4.50). The basis $\boldsymbol{r}_l$ that is best aligned with the residual $\boldsymbol{x}_{p,l}$ is selected. If the selected $\boldsymbol{r}_l$ satisfies condition (4.24), it is included in the design matrix $\boldsymbol{K}$, and the corresponding parameters Φ_l, $\mu_{p,l}$, and α_l are computed according to (4.51), (4.52), and (4.53), respectively. These steps are continued until all bases with delays from the search space $\mathcal{T}$ are initialized, or until the basis vector that does not satisfy the condition (4.24) is encountered.

Of course, in order to be able to use this initialization scheme, it is crucial to get a good initial noise estimate. The initial noise parameter $N_0^{[0]}$ can in most cases be estimated from the tails of the channel impulse response, where multipath components are unlikely to be present or too weak to be detected. Generally, we have observed that the algorithm is less sensitive to the initial values of the hyperparameters $\boldsymbol{\alpha}$, but proper initialization of the noise spectral height is crucial.

Now we can describe the simulation setup used to assess the performance of the proposed algorithm.

4.4.1 Simulation setup

The generation of the synthetic channel is done following the block-diagram shown in Fig. 3.1: a single period $u(t)$ of the sounding sequence $s(t)$ is filtered by the channel with the impulse response $\boldsymbol{h}(\tau)$, and complex white Gaussian noise is added to the channel outputs to produce the received signal $\boldsymbol{y}(t)$. The received signal is then run through the MF. The continuous-time signals at the output of the MF are represented with cubic splines. The resulting spline representation is then used to obtain the sampled output $z_p[n]$, $p = 0, \ldots, P-1$, with $n = 0 \ldots N-1$. Output signals $z_p[n]$ are then used as the input to the estimation algorithm.

For all P channel observations we use the same MF, and thus $\boldsymbol{\Phi}_p = \boldsymbol{\Phi}$, $\boldsymbol{K}_p = \boldsymbol{K}$, and $\boldsymbol{\Sigma}_p = \boldsymbol{\Sigma}$, $p = 0, \ldots, P-1$. Without loss of generality, we assume a shaping pulse of the duration $T_p = 10$nsec. The sampling period is assumed to be $T_s = T_p/N_s$, where N_s is the number of samples per chip used in the simulations. The sounding waveform $u(t)$ consists of $M = 255$ chips. We also assume the maximum delay spread in all simulations to be $\tau_{spread} = 1.27\mu$sec. With these parameters, a one-sample/chip resolution results in $N = 128$ samples. The autocorrelation function $R_{uu}(t)$ is also represented with cubic splines, allowing a proper construction of the design matrix $\boldsymbol{K}$ according to the predefined delays in $\mathcal{T}$.

Realizations of the channel parameters $w_{l,p}$ are randomly generated according to (4.7).

The performance of the algorithm is also evaluated under different SNR's at the output of the MF, defined as

$$\mathrm{SNR} = 10\log_{10}\Big(\frac{1/\alpha}{N_0}\Big). \tag{4.64}$$

We assumed that in the case $L > 1$ all simulated multipath components have the same expected power α^{-1}.

4.4.2 Numerical simulations

Let us now demonstrate the performance of the model selection schemes discussed in Section 4.3 on synthetic, as well as on measured channels.

Multipath detection with perfectly matching model

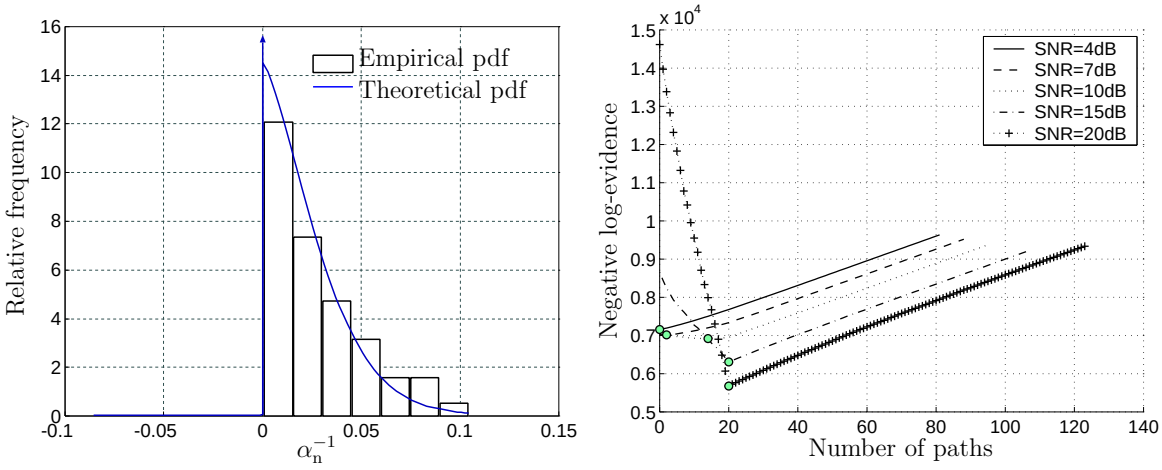
First we consider the distribution of the hyperparameters once the stationary point has been reached. In order to do that, we apply the learning algorithm to the full hypothesis $\mathcal{H}_0$. The delays in $\mathcal{H}_0$ are evenly positioned over the length of the impulse response: $\mathcal{T} = \{lT_s; l = 0, \ldots, N-1\}$, i.e., $L_0 = N$. Here, we simulate the channel with a single multipath component, i.e., $L = 1$, having the delay τ' equal to a multiple of the sampling period T_s. Thus, in the design matrix $\boldsymbol{K}$ corresponding to the full hypothesis $\mathcal{H}_0$ there will be a basis function that coincides with the contribution of the true multipath component. Once the parameters have been learned, we partition all the hyperparameters α into those attributed to the noise, i.e., α_{n}, and one parameter that corresponds to the multipath component α_{s}, i.e., the one associated with the delay $T_l = \tau'$.

In a next step, we compare the obtained histogram of α_{n}^{-1} with the theoretical pdf $p_{\alpha_{\mathrm{n}}^{-1}}(x)$ given in (4.37). The corresponding results are shown in Fig. 4.12(a). A very good match between the empirical and theoretical pdf's can be observed.

Similarly, we investigate the behavior of the negative log-evidence versus the size of the hypothesis. We consider a similar simulation setup as above, however with more than just one multipath component to make the results more realistic. Figure 4.12(b) depicts the evaluated negative log-evidence (4.62) as a function of the model order, evaluated for a single realization, when the true number of components is $L = 20$, and the number of channel observations is $P = 5$.

Note that, as the SNR increases, there are fewer components subject to the initial pruning, i.e., those that do not satisfy condition (4.24). We also observe that the minimum of the negative log-evidence (i.e., maximum of the evidence) becomes more pronounced as the SNR increases, which has an effect of decreasing the variance of the model order estimates.

In order to find the best possible performance of the algorithm, we first perform some simulations assuming that the discrete-time model (3.12) perfectly matches



(a) (b)

Figure 4.12: Evidence-based model selection criteria. a) Empirical (bar plot) and theoretical (solid line) pdf's of hyperparameters α_n^{-1} (SNR = 10dB, and $P = 10$), b) Negative log-evidence as a function of the model order (number of paths) for different SNR values ($P = 5$, and $L = 20$).

the continuous-time model (3.10), i.e., $\tau_l \in \mathcal{T},\ l = 1, \ldots, L$. This is realized by drawing uniformly L out of N possible delay values in the interval $[0, T_s(N-1)]$. Again, $\mathcal{T} = \{lT_s; l = 0, \ldots, N-1\}$. The number of multipath components in the simulated channels is set to $L = 5$ and the channel is sampled with $N_s = 2$ samples per chip.

In this simulation we evaluate the detection performance by counting the errors made by the algorithms. Two types of errors can occur: (a) an *insertion error*– an erroneous detection of a non-existing component; (b) a *deletion error*– a loss of an existing component. The case when an estimated delay $\hat{T}_l$ matches one of the true simulated delays is called a *hit*. We further define the *multipath detection rate* as the ratio between the number of hits to the true number of components L plus the number of insertion errors. It follows that the detection rate is equal to 1 only if the number of hits equals the true number of components. If, however, the algorithm makes any deletion or insertion errors, the detection rate is then strongly smaller than 1. We study the detection rates for both model selection schemes versus different SNR's. The presented results are averaged over 300 independent channel realizations.

We start with the model selection approach based on the threshold selection using the ρ-quantile of the noise distribution - quantile-based model selection. The results shown in Fig. 4.13(a) are obtained for $\rho = 1 - 10^{-6}$ and different numbers of channel observations P. It can be seen that, as P increases, the detection rate significantly improves. To obtain the results shown in Fig. 4.13(b) we fix the

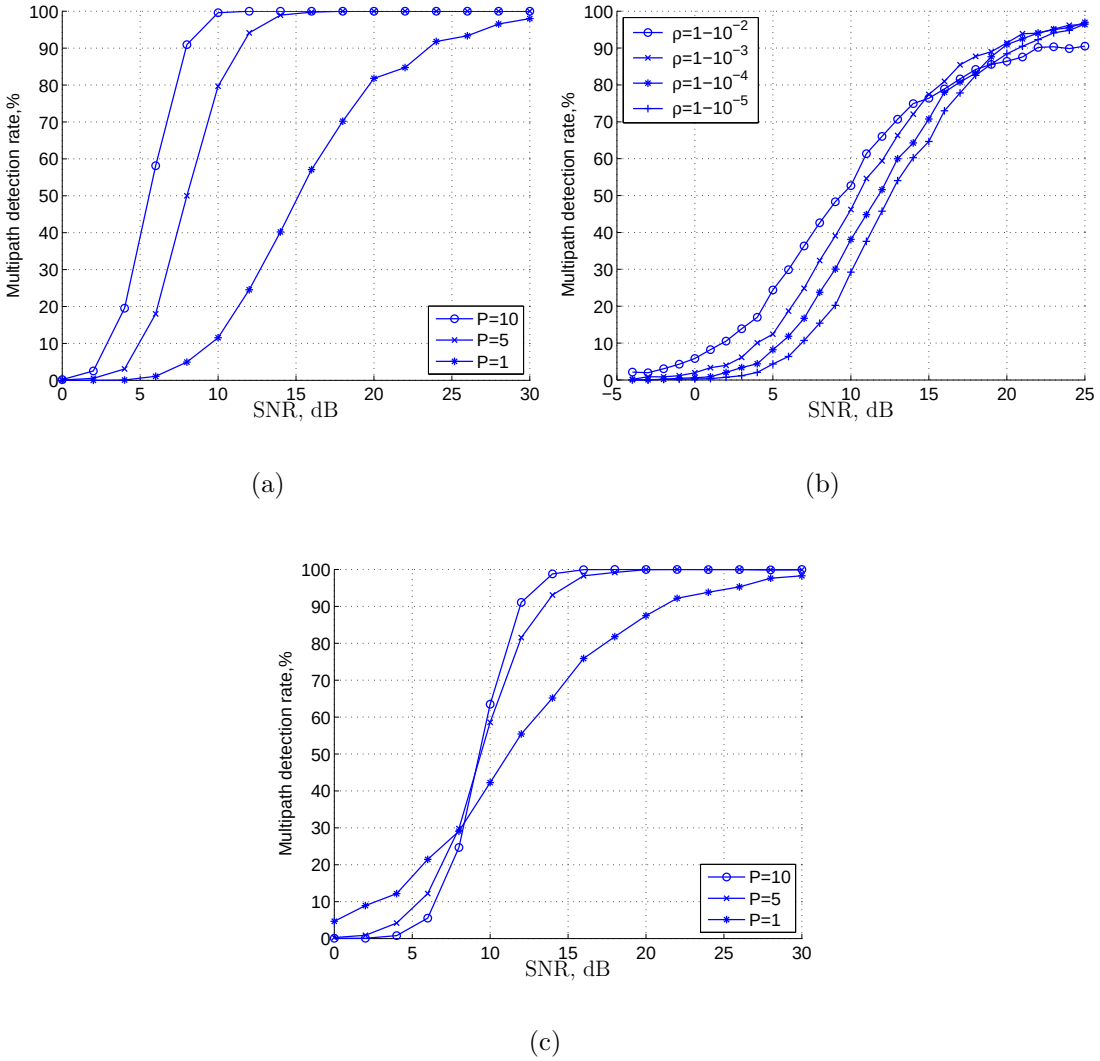


Figure 4.13: Multipath detection rates based on the EP. (a) Quantile-based model selection versus P: $\rho = 1 - 10^{-6}$, $L = 5$; (b) Quantile-based model selection versus ρ: $P = 5$, $L = 5$; (c) Negative log-evidence-based detection versus P.

number of channel observations at $P = 5$ and vary the value of the quantile ρ. It can be seen that as ρ approaches unity, the threshold is placed higher, meaning that fewer noise components can be mistakenly detected as multipath components, thus slightly improving the detection rate. However higher thresholds require a higher SNR to achieve the same detection rate, as compared for the thresholds obtained with lower ρ.

The next plot in Fig.4.13(c) shows the multipath detection rate when the model is selected based on the evaluation of the negative log-evidence under different model hypotheses (negative log-evidence model selection). It is interesting to note that

in this case the reported curves behave quite differently from those shown in Fig. 4.13(a). First, we see that for the case $P = 1$ the behavior of this method is slightly better, compared to the threshold-based method in Fig. 4.13(a). But as P grows, the performance of the multipath detection does not increase proportionally, but rather exhibits a threshold-like behavior. In other words, multipath detection based on the negative log-evidence and alike MDL-based model selection requires the SNR above a certain threshold in order to operate reliably. Furthermore, this threshold is independent of the number of channel observations P.

Thus from Fig. 4.13(a) and Fig. 4.13(c) we can conclude that the quantile-based method performs better in a sense that it can always be improved by increasing the number of channel observations. Further, model selection using the thresholding approach can be performed on-line, concurrent with parameters estimation, while in the other case multiple models have to be learned.

Now, let us consider how the EP performs when the multipath component delays are on the real line, rather than on a discrete grid. Clearly, this case corresponds more to the real-life situation.

Multipath detection with model mismatch

In the real world the delays of the multipath components do not necessarily coincide with the elements in $\mathcal{T}$ used to approximate the continuous-time model (3.10). By using the discrete-time models to approximate the continuous-time counterparts, we would necessarily expect some performance degradation in terms of an increased number of components.

Since there is an inevitable mismatch between the continuous-time and discrete-time models, it is worth asking how densely we should quantize the delay line to form the design matrix in order to achieve the best performance. It is convenient to select the delays in $\mathcal{T}$ of the discrete-time model as a multiple of the sampling period T_s. As the sampling rate increases the true delay values get closer to some elements in $\mathcal{T}$, thus approaching the continuous-time model (3.10).

We simulate a channel with a single multipath component that has a random delay, uniformly distributed in the interval $[0, \tau_{spread}]$.

The criterion used here to assess the performance of the algorithm is the probability of correct path extraction. This probability is defined to be the conditional probability that, given any path is detected at all, the absolute difference between the delay estimate and the true delay is less than the chip pulse duration T_p. Notice that the probability of correct path extraction is conditioned on the path detection, i.e., it is evaluated for the cases when the estimation algorithm is able to find at least one component.

It is also interesting to compare the performance of the EP with other parameter estimation techniques. Here we consider the SAGE algorithm [FTH$^+$99] that has become a popular multipath parameter estimation technique. The SAGE algorithm, however, does not provide any information about the number of multipath components. To make the comparison fair, we augment it with the standard MDL

criterion [Ris78, WK85] to perform model selection.

Thus, we are going to compare three different model selection algorithms: the quantile-based (or threshold-based) scheme with a pre-selected quantile $\rho = 1-10^{-6}$, the SAGE+MDL method, and negative log-evidence method. We are also going to use the threshold-based method to demonstrate the difference between two EP initialization schemes: the joint initialization, and the independent initialization, discussed in Section 4.4. In all simulations the negative log-evidence method was initialized using independent initialization.

We start with channels sampled with $N_s = 1$ sample/chip resolution and $P = 5$ channel observations. We see that the studied methods have different probabilities of path detection (Fig.4.14(a)), i.e., they require different SNR to achieve the same path detection probability. The threshold-based methods can be, however, adjusted by selecting the quantile ρ appropriately. As we see, with $\rho = 1-10^{-6}$, the threshold-based and SAGE+MDL methods achieve the same probabilities of path detection. The resulting probabilities of correct path extraction are shown in Fig. 4.14(b). Note that for low SNR comparisons of the methods is meaningless, since too few paths are detected. However, above SNR $\approx$ 15dB, with all methods we can achieve similar high path detection probabilities, which allows direct comparison of the correct path extraction probabilities. We can hence infer that, in this regime, model selection with negative log-evidence is superior to other methods, since it has higher probabilities of path extraction. In other words this means that at higher SNR this method will introduce fewer artifacts.

Now, let us increase the sampling rate and study the case $N_s = 2$ (Fig. 4.14(c), and Fig. 4.14(d)). We see that the probabilities of path extraction are now higher for all methods. A slight difference between the two EP initialization schemes can also be observed. Note however that the performance increase is higher for the SAGE+MDL and negative log-evidence algorithms, which both rely on the same model selection concept.

Finally, the last case with $N_s = 4$ is shown in Fig. 4.14(e) and Fig. 4.14(f). Again SAGE+MDL and negative log-evidence schemes achieve higher correct path extraction probabilities as compared to the threshold-based method. The performance of the latter also increases with the sampling rate, but unfortunately not as fast as that of the Description-Length based model selection. Theses plots also demonstrate the difference between the two proposed initializations of the EP. In Fig. 4.14(f) we see that in this case the independent initialization outperforms the joint one. As already mentioned, this distinction becomes noticeable, once the basis functions in $\boldsymbol{K}$ exhibit significant correlation, what is the case for $N_s \gtrsim 2$.

4.4.3 Results for measured channels

We also apply the proposed algorithm to the measured data collected in in-door environments. Channel measurements were done with the MIMO channel sounder PropSound manufactured by Elektrobit Oy (see Appendix D). The basic setup for channel sounding is equivalent to the block-diagram shown in Fig. 3.1. In the

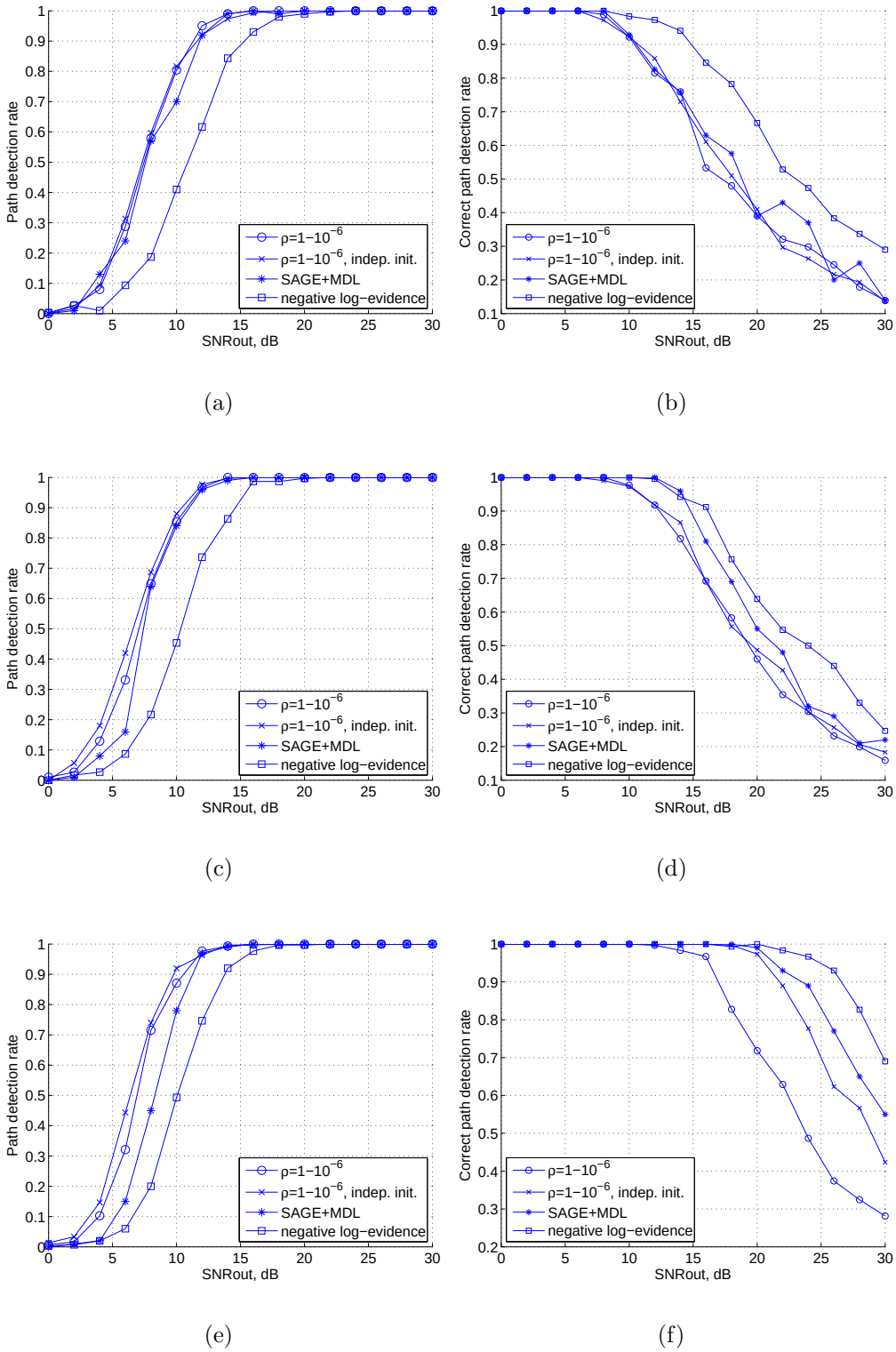


Figure 4.14: Comparison of the model selection schemes in a single path scenario. (a,c,e) path detection probability, and (b,d,f) probability of correct path extraction for $P = 5$, and (a,b) $N_s = 1$; (c,d) $N_s = 2$; and (e,f) $N_s = 4$.

conducted experiment the sounder operated at the carrier frequency 5.2GHz with a chip period of $T_p = 10$nsec. The output of the matched filter was sampled with the period $T_s = T_p/2$, thus resulting in a resolution of 2 samples per chip. The sounding sequence consisted of $M = 255$ chips, resulting in the burst waveform duration of $T_u = MT_p = 0.255\mu$sec.

Based on visual inspection of the PDP of the measured channels, the delays T_l in the search space $\mathcal{T}$ are positioned uniformly in the interval between 250nsec and 1000nsec, with spacing between adjacent delays equal to T_s. This corresponds to the delay search space $\mathcal{T}$ consisting of 151 elements. The initial estimate of the noise floor is obtained from the tail of the measured PDP. The algorithm stops once the relative change of the evidence parameters between two successive iterations is smaller than 0.0001%. The corresponding detection results for different number of channel observations are shown in Fig. 4.15. When $P = 1$ (see Fig. 4.15(a)), the independent initialization results in only 9 basis functions constituting the initial hypothesis $\mathcal{H}_0$. The final estimated number of components is found to be $L = 8$. As expected, increasing the number of channel observations P makes it possible to detect and estimate components with smaller SNR. For the case of $P = 5$ we detect already $L = 12$ components (Fig. 4.15(b)), and for $P = 32$, $L = 15$ components (Fig. 4.15(c)). This shows that increasing the number of observations not necessarily brings a proportional increase of the detected components, thus suggesting that there might be a limit given by the true number of multipath components.

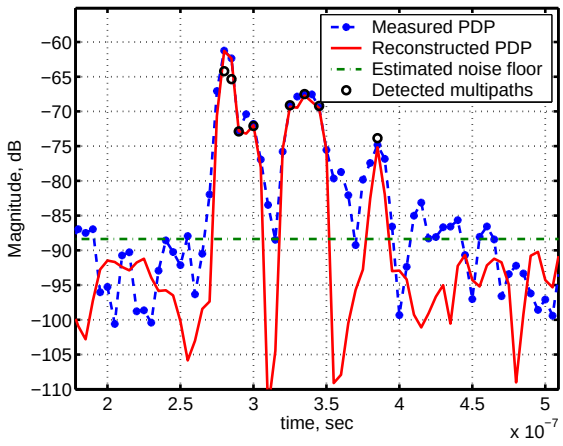
4.5 SAGE iterations and SAGE-RVM algorithm

The Evidence Procedure developed so far can be applied to estimating the model order L, as well as the corresponding multipath delays τ_l with fixed resolution. The SAGE algorithm discussed in Section 3.2 allows estimating other multipath parameters as well, but does not have model selection capabilities. Joining the two approaches will allow to take the best from both SAGE and Evidence Procedure approaches, thus giving rise to the new SAGE-RVM algorithm, discussed in the following section.

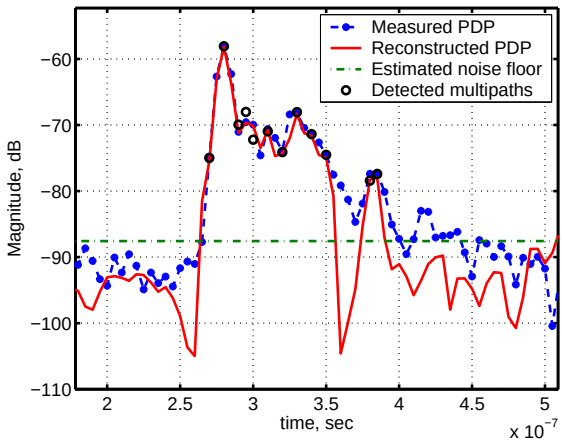
4.5.1 Basic steps of the SAGE-RVM algorithm

The reader familiar with the SAGE algorithm has already noticed that the modifications to the EP learning introduced in Section 4.3.2 are inspired by the SAGE Expectation-Maximization steps. The key to the algorithm improvement lies in estimating data relevant to a single wavefront only by canceling the influence of the other components, i.e., in terms of the SAGE terminology, estimating hidden data. The hidden data allows to estimate both the evidence parameters, which are then used in model selection, and the other multipath parameters, like Doppler frequency, DoA, multipath gain, etc.

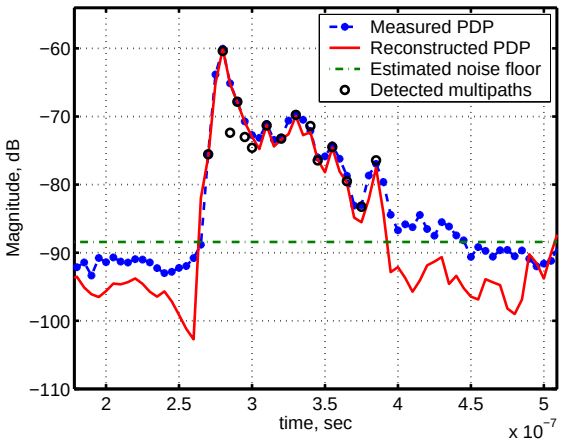
Let us now go through the major steps of the SAGE-RVM algorithm. We generally



(a) $P = 1$; Estimated number of multipath components $L = 8$.



(b) $P = 5$; Estimated number of multipath components $L = 12$.



(c) $P = 32$; Estimated number of multipath components $L = 15$.

Figure 4.15: Multipath detection results for quantile-based method with $\rho = 1 - 10^{-6}$.

assume a receive antenna array with P elements and I consecutive SIMO channel observations, as explained in Section 3.1.4. Thus, in total we have $J = I \times P$ channel observations.

SAGE-RVM initialization

The initialization of the SAGE-RVM algorithm begins with the independent EP initialization as explained in Section 4.4. This results in the initial model order L, design matrix $\boldsymbol{K}$, coefficient vector $\boldsymbol{\mu}_j$, $j = 0..J-1$, corresponding evidence

parameters $\boldsymbol{\alpha}$, and initial additive noise spectral height $N_0 = \beta^{-1}$.

Now, using (3.13) we can extract the initial Doppler frequency, DoA, and multipath gain from the estimated coefficients $\boldsymbol{\mu}_j$. This is done by transforming $\boldsymbol{\mu}_j = [\mu_{1j} \dots \mu_{Lj}]^T$ into the matrix $\boldsymbol{W}_l$ as it was done for the SAGE algorithm initialization, explained in Section 3.2.1. The initial values of the DoA ϕ_l, Doppler frequency ν_l, and multipath gain a_l are then found as solutions to (3.30), (3.31), and (3.33).

This would finalize the initialization step of the SAGE-RVM algorithm. Note that this initialization is basically equivalent to the SAGE initialization, with the distinction that the EP expressions are used instead of those of the Matching Pursuit.

SAGE-RVM iterations

Basically, the iterations of the SAGE-RVM algorithm reproduce (with some modifications) the initialization step. At the each iteration the hidden data $\boldsymbol{x}_{j,l}$ for the lth multipath component is computed as

$$\boldsymbol{x}_{j,l} = \boldsymbol{z}_j - \sum_{k=1,k\neq l}^{L} \boldsymbol{r}_k \mu_{j,k},$$

which is then used to update the delay τ_l of the lth multipath component: the value of the new delay τ_l is found as the maximizer of

$$\tau_l' = \underset{\tau}{\operatorname{argmax}} \sum_j \left| \boldsymbol{x}_{j,l}^H \boldsymbol{r}(\tau) \mu_{j,l} \right|, \tag{4.65}$$

where $\boldsymbol{r}(\tau) = [R_{uu}(-\tau), \dots, R_{uu}((N-1)T_s - \tau)]^T$. Note that in (4.65) the search space for the multipath delay is the whole real line, rather than a discrete set. Once the optimum delay is found, we adjust the corresponding basis function associated with this component as

$$\boldsymbol{r}_l' = [R_{uu}(-\tau_l'), R_{uu}(T_s - \tau_l'), \dots, R_{uu}((N-1)T_s - \tau_l')]^T. \tag{4.66}$$

With the new basis it is possible to update the corresponding posterior statistics as well as evidence parameters exclusively for this multipath component using now the hidden data $\boldsymbol{x}_{j,l}$ only:

$$\Phi_l' = (\alpha_l + \beta (\boldsymbol{r}_l')^H \boldsymbol{\Lambda}^{-1} \boldsymbol{r}_l')^{-1}, \tag{4.67}$$

$$\mu_{j,l}' = \beta \Phi_l' (\boldsymbol{r}_l')^H \boldsymbol{\Lambda}^{-1} \boldsymbol{x}_{j,l}, \quad j = 0, \dots, J-1. \tag{4.68}$$

Having updated the parameter posterior statistics we update the corresponding evidence parameter:

$$\alpha_l' = \frac{J}{\sum_{j=0}^{J-1} \left(\Phi_l + |\mu_{j,l}|^2 \right)}. \tag{4.69}$$

At this stage, we can perform model selection using the threshold-based rules developed earlier to test if the basis $\boldsymbol{r}'_l$ stays in the model. If we decide not to prune the basis, we proceed to the estimation of the DoA, Doppler frequency and the multipath gain. Otherwise, the corresponding components are removed from the analysis.

To estimate the DoA, Doppler frequency, and multipath gain, we construct the matrix $\boldsymbol{W}'_l$ using the updated weight coefficients $\mu'_{j,l}$ as explained in Section 3.2.1. The new update value of the DoA is found as the solution to the following maximization problem:

$$\phi'_l = \underset{\phi}{\operatorname{argmax}} |a_l^* \boldsymbol{c}^H(\phi) \boldsymbol{W}_l \boldsymbol{d}^H(\nu_l)|, \tag{4.70}$$

where a_l is the multipath gain, $\boldsymbol{c}(\phi)$ is the steering vector of the array (3.3), and $\boldsymbol{d}(\nu)$ is a Doppler vector, defined in (3.32).

The update for the Doppler frequency ν_l is found similarly as a solution to

$$\nu'_l = \underset{\nu}{\operatorname{argmax}} |a_l^* \boldsymbol{c}^H(\phi'_l) \boldsymbol{W}_l \boldsymbol{d}^H(\nu)|. \tag{4.71}$$

Finally, the updated value of the multipath gain a_l is found as

$$a'_l = \frac{\boldsymbol{c}^H(\phi'_l) \boldsymbol{W}_l \boldsymbol{d}^H(\nu'_l)}{||\boldsymbol{c}(\phi'_l)||^2 ||\boldsymbol{d}(\nu'_l)||^2}. \tag{4.72}$$

The update steps (4.65)-(4.72) are subsequently performed for all L components. Once all the components are updated, we can update the noise parameter N_0 as

$$N'_0 = \frac{1}{NJ} \left(\sum_{j=0}^{J-1} \operatorname{tr}[\boldsymbol{\Phi}' \boldsymbol{K}'^H \boldsymbol{\Lambda}^{-1} \boldsymbol{K}'] + \sum_{j=0}^{J-1} (\boldsymbol{z}_j - \boldsymbol{K}' \boldsymbol{\mu}'_j)^H \boldsymbol{\Lambda}^{-1} (\boldsymbol{z}_j - \boldsymbol{K}' \boldsymbol{\mu}'_j) \right), \tag{4.73}$$

where $\boldsymbol{K}'$ is the updated design matrix with columns defined by (4.66), and $\boldsymbol{\Phi}'$ and $\boldsymbol{\mu}'_j$ are posterior statistics, updated according to (4.67) and (4.68), respectively.

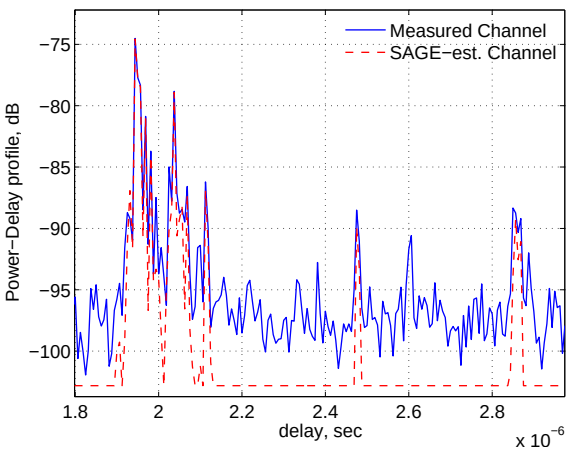
This completes a single iteration of the SAGE-RVM algorithm. We see from the preceding discussion that SAGE-RVM is in fact a modified version of the SAGE algorithm that allows online model selection.

4.5.2 Some application examples

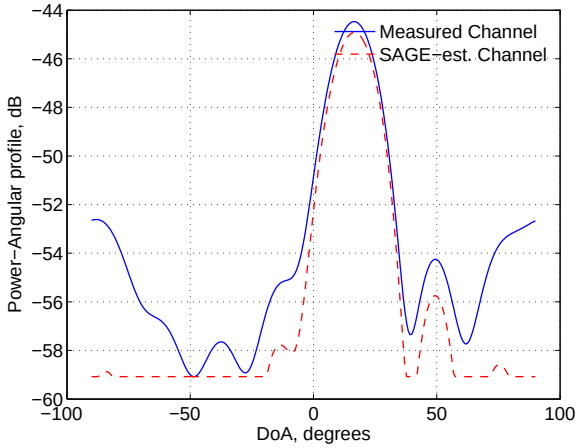
Let us now consider some application examples. We apply the SAGE-RVM algorithm to the FTW data (Appendix C) since it is used later for the channel prediction, too. To implement the model selection, we use the quantile-based method with $\rho = 1 - 10^{-6}$.

Similarly to the SAGE algorithm, we demonstrate the resulting goodness-of-fit (Fig. 4.16) to the measured data achieved with the SAGE-RVM algorithm for a single measurement.

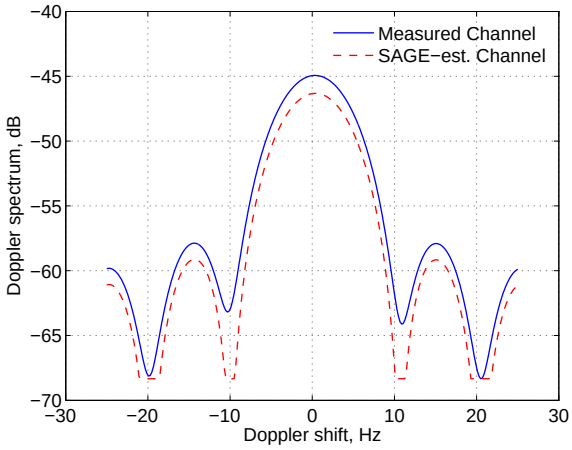
Since the channel is in general time-varying, the multipath parameters as well as the model order are a function of time. In Fig. 4.17 we plot the evolution of the



(a) Estimated Power-Delay Profile.



(b) Power-Angular Profile.



(c) Estimated Doppler spectrum.

Figure 4.16: (a,b,c) Goodness-of-fit for the SAGE-RVM algorithm. The number of estimated components is $L = 14$.

estimated multipath parameters as a function of the walked distance (the speed of the mobile transmitter in this case was $\approx$ 1m/s). The size of the markers on the plot is proportional to the inverse of the estimated evidence parameter α_l^{-1}, i.e., proportional to the estimated power of a multipath component.

Note that the approximation results are very similar to that of the SAGE algorithm, but the number of the wavefronts is estimated optimally in accordance with the Ocham's razor principle.

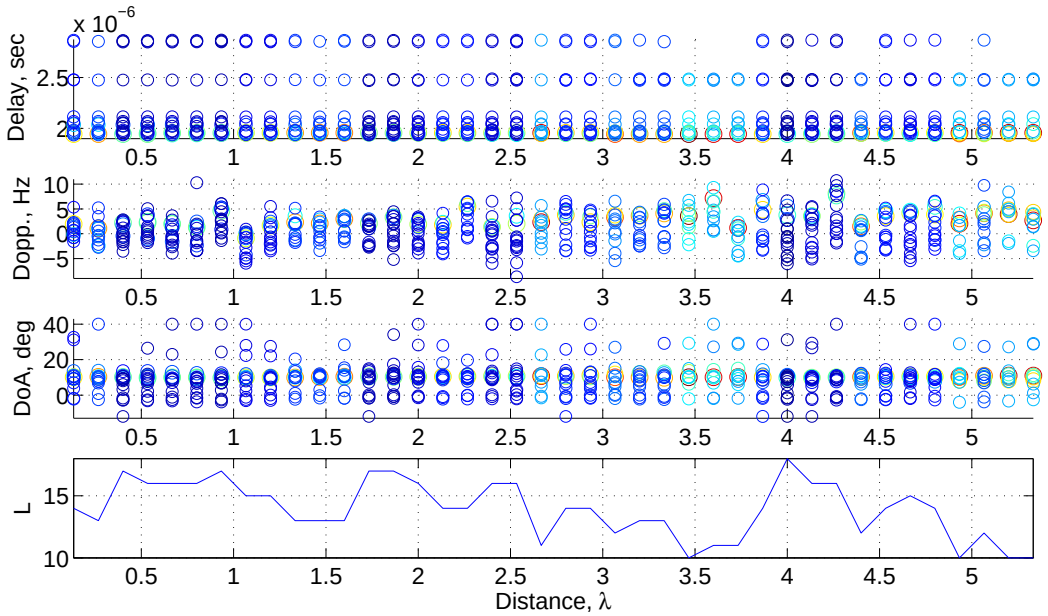


Figure 4.17: Evolution of the estimated multipath parameters (Delays, Doppler frequency, DoA, and number of wavefronts L).

4.6 Discussion and conclusions

In this chapter we have considered an extension and application of the Evidence Procedure to the estimation of wireless channels. Let us now summarize and discuss the performance and properties of the EP and SAGE-RVM algorithms.

4.6.1 Evidence Procedure

The Evidence Procedure is in many respects similar to the SAGE algorithm. It is also a model-based parameter estimation algorithm, but unlike SAGE, the EP optimizes the penalized model performance error. The penalty introduced in the EP framework allows to find a compromise between the model performance (size of the approximation error) and the number of the components in the approximation.

The proper penalty is introduced naturally within the Bayesian framework. The Bayesian approach results in the Maximum *a posteriori* (MAP) estimate. We know, that MAP is basically equivalent to the ML approach with the distinction that the former requires specification of the *a priori* information. This prior information is then used in the model selection criteria.

The application of the Evidence Procedure to the wireless channels was developed based on the methods known in the literature as Relevance Vector Machines. The RVM algorithm was developed as a Machine Learing technique and had to be significantly extended to allow its application to the wireless channels. First, we extended the RVM to the complex domain and colored additive noise. From the methodological point of view, an important innovation is the Bayesian graphi-

cal channel model that represents a probabilistic structure of the multipath MIMO channel. This graphical model is not only the basis for the probabilistic inference of the model parameters, but it is also a completely new way of representing the multipath structure of the channel. The evidence parameters, introduced in the model, also called hyperparameters in the original RVM paper, can be interpreted as a simple form of the hypermodel. We can say so because each evidence parameter α_l controls the contribution of the corresponding multipath component.

These evidence parameters are the key to the model selection. Note that here we also exploit this "hyper"-model concept: based on α we control the sparsity of the total model. Assuming a single path scenario we are able to find the statistical laws that govern the values of the evidence parameters once the estimation algorithm has converged to the stationary point. It is shown that in low SNR scenarios the evidence parameters do not attain infinite values, as has been assumed in Tipping's original RVM formulation, but stay finite with values depending on the particular SNR level. This knowledge enabled us to develop model selection rules based on the discovered statistical laws behind the evidence parameters.

In order to be able to apply these rules in practice, we also proposed a modified learning algorithm that exploits the principle of successive interference cancellation. This modification not only allows to avoid computationally intensive matrix inversions, but also removes the interference between the neighboring basis functions in the design matrix.

The model mismatch case is also considered in our analysis. We are able to assess the possible influence of the finite algorithm resolution and, to some extent, take it into account by adjusting the corresponding model selection rules. This step eventually minimizes the number of the obtained estimation artifacts

We also showed the relationship between the EP and the classical model selection based on the MDL criterion. It was found that the maximum of the evidence corresponds to the minimum of the corresponding description length criterion. Thus, EP can be used as the classical MDL-like model selection scheme, but also allows faster and more efficient threshold-based implementation.

The EP framework was also compared with the multipath estimation using the SAGE algorithm augmented with the MDL criterion.

According to the simulation results, the Description-Length based methods, i.e., negative log-evidence and SAGE+MDL method, give better results in terms of the achieved probabilities of correct path extraction. They also improve faster as the sampling rate grows. However, these model selection strategies require learning multiple models in parallel, which, of course, imposes additional computational load. The threshold-based method, on the other hand, allows to perform model selection on-line, thus being more efficient, but its performance increase with the growing sampling rate is more modest. The performance of the threshold-based method also depends on the value of the quantile ρ. In our simulations we set $\rho = 1-10^{-6}$, which results in the same probability of path detection as in the SAGE+MDL algorithm. However, other values of ρ can be used, thus giving a way to further optimize the performance of the threshold-based method.

The comparison between the SAGE and EP schemes clearly shows that estimating evidence parameters really pays off. Introducing them in the computation of the model complexity, as it is done in the negative log-evidence approach, results in the best performance, compared to the other two methods. Although the negative log-evidence method needs a slightly higher SNR to reliably detect channels, it however results in the highest probability of path extraction.

The threshold-based method also opens perspectives for on-line remodeling, i.e., removing, or even adding new, components during the estimation of the model parameters which might result in much better and sparser models. Since the evidence parameters reflect the contribution of the multipath components, they might also be useful in applications, where it is necessary to define some measure of confidence for a multipath component.

4.6.2 SAGE-RVM algorithm

By borrowing some of the ideas implemented in the SAGE algorithm, we also made the EP algorithm more efficient. This union of both SAGE and Evidence Procedure gives birth to the SAGE-RVM algorithm. The theoretical foundation that makes this possible lies in the SAGE algorithm itself.

In general, SAGE is a general-purpose parameter estimation technique, which is not necessarily bundled with estimating delay, DoA, and Doppler frequency. It can similarly be used to estimate the evidence parameters α. The latter allows us to invoke the model selection criteria we developed within the EP framework. It is this step that gives birth to the SAGE-RVM algorithm. In other words, it is possible to treat SAGE-RVM as a modification of the SAGE algorithm that, in addition to the standard list of multipath parameters, estimates the associated evidence parameters as well.

The key step in SAGE-RVM is the estimation of the hidden data (E-step of the SAGE algorithm). The hidden data allows us to estimate and update parameters for a single component only, including the evidence parameter. In connection to the EP, this not only means that we avoid computing matrix inversions used in obtaining the posterior statistics (4.18), and (4.12), but also that we are able to estimate other multipath parameters.

It also important to stress the importance of the proper noise estimation. The EP framework allows estimating the noise value directly. As we have seen, the whole model selection mechanism assumes the noise variance to be known. Practically, we need to estimate it and use the estimate as the true value, which might not be the best choice. Further, since the noise estimate and model selection are coupled, errors in the model selection might propagate in the estimation of the noise statistics, and the other way around. To decouple this dependency, it might be advantageous not to update the noise estimate at all, or at lest freeze it after a couple of iterations in order to avoid error propagation.

Chapter 5

Channel tracking

Estimation algorithms discussed in Chapters 3 and 4 allow to estimate parameters of the multipath components constituting the impulse response of a multipath channel. Now, for each estimation window we can represent a wireless channel by a set of parameters describing the detected wavefronts.

However, wireless channels are usually time-varying. In order to properly reconstruct the dynamics of the underlying wavefronts it is important to keep proper parameter associations between the consecutive channel observations, i.e., we need to track the multipath components over time. This brings us to the problem of parameter association and tracking. A similar problem is sometimes referred to as *parameter warping* [MS00b].

In general, parameter tracking/association is not a trivial problem since there is no *a priori* model that can be used to ease this task. However, this model can be constructed or learned iteratively, as the algorithm proceeds. In fact, the hypermodel of the multipath dynamics is an appropriate model that can assist multipath tracking.

Thus, multipath tracking needs a hypermodel for optimal performance, while a hypermodel learning algorithm relies on the output of the tracking algorithm that supplies it with learning data. We suggest to resolve this interdependency in the spirit of the classical sequential Bayesian estimation (see, for example, [MS00b]).

Let us consider the block diagram of the proposed sequential tracking and prediction scheme, depicted in Fig. 5.1.

We assume that we want to reconstruct K tracks from the multipath estimates $\{\boldsymbol{\theta}_l[q]\}_{l=1}^{L}$, so that $K \leq L$, where q refers to the estimation window sample, as defined in Section 3.1.4.

The dynamics of each track is captured by the corresponding deterministic *hypermodel*, i.e., predictor $\mathcal{H}_k(\cdot)$, in a sense that

$$\hat{\boldsymbol{\theta}}_k[q] = \mathcal{H}_k(\boldsymbol{\theta}_k[q-1], \boldsymbol{\theta}_k[q-2], \ldots). \tag{5.1}$$

Expression (5.1) is equivalent to the *prediction step* of Bayesian sequential estimation.

Once the prediction is obtained, we can define a distance measure $f(\cdot,\cdot)$ between the predictions $\hat{\boldsymbol{\theta}}_k[q]$ and newly obtained estimates $\{\boldsymbol{\theta}_l[q]\}_{l=1}^{L}$. The associations are then made so as to minimize the resulting distance between the predictions and the estimates. The details on the association algorithm are presented in Section 5.1.

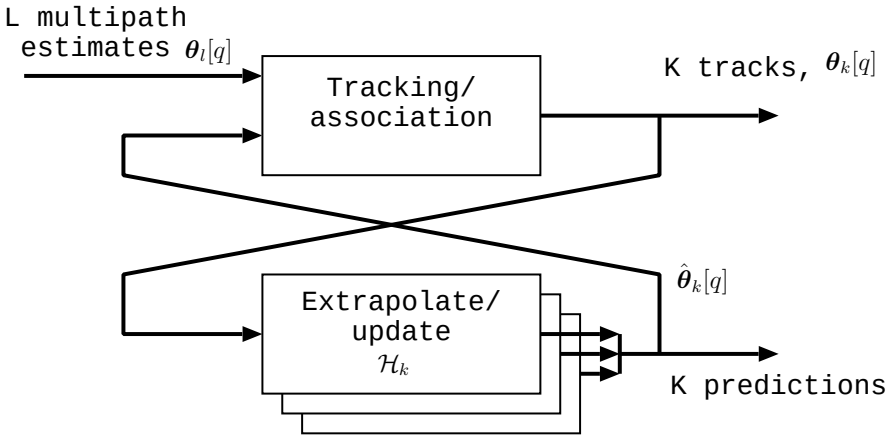


Figure 5.1: Iterative multipath tracking and adaptation of the track hypermodels $\mathcal{H}_k$.

The obtained associations are then used to recursively update the hypermodels. In Sections 5.2 and 5.3 we consider the corresponding hypermodel realizations and the corresponding learning algorithms. This constitutes the *update step* of sequential estimation. Note that the proposed scheme is similar in the reasoning to the Dual Estimation [Hay01, ch. 5], used within the Kalman Filter framework to jointly estimate the states as well as the system's observation or transition models.

Let us now consider these steps in more details.

5.1 Multipath tracking

Let us start by assuming that the estimation algorithm finds and estimates $L[q]$ multipath components for the qth channel estimation window. Depending on a particular estimation algorithm the number of estimated components might vary with time. Let us also assume that we are interested in reconstructing the dynamics of $K[q]$ components, which we also call tracks, so that $K[q] \leq L[q]$. In the sequel we drop the explicit dependency on the estimation window index q to simplify the notations, however we assume that both L and K are in general a function of q.

We already know that a multipath component is described by a parameter vector $\boldsymbol{\theta}_k[q]$. The parameters constituting $\boldsymbol{\theta}_k[q]$ can be split into two subsets. This is done since not all of the multipath parameters are used in the tracking algorithm. For instance, in case of SIMO channels, only the multipath delay, Doppler shift, and DoA uniquely identify the multipath component. The complex multipath gain, on the other hand, does not help much, since given two multipath components with identical delays, Doppler frequencies, and DoA's, the estimation algorithm will not be able to separate them.

The first subset $\boldsymbol{s}_k[q] \subset \boldsymbol{\theta}_k[q]$ consists of the parameters related to the structure of the channel, namely multipath delay $\tau_k[q]$, Doppler frequency $\nu_k[q]$, and DoA $\phi_k[q]$. These multipath parameters are then going to be used in the tracking and

association algorithm.

The second subset $\boldsymbol{a}_k[q] \subset \boldsymbol{\theta}_k[q]$ includes multipath parameters that are not used in tracking. As an example, this subset might consist of a complex multipath gain $a_k[q]$, but it also might include other multipath parameters for which long-term predictors are to be designed. As we will see later, it makes sense to use different hypermodels for $\boldsymbol{s}_k[q]$ and $\boldsymbol{a}_k[q]$.

Let us for the moment assume that the dynamics of each track is captured by a certain known deterministic hypermodel $\mathcal{H}_k(\cdot)$ consisting of two separate predictors $\mathcal{S}_k(\cdot)$ and $\mathcal{A}_k(\cdot)$ in a sense that

$$\begin{aligned} \hat{\boldsymbol{s}}_k[q] &= \mathcal{S}_k(\boldsymbol{s}_k[q-1], \boldsymbol{s}_k[q-2], \ldots), \\ \hat{\boldsymbol{a}}_k[q] &= \mathcal{A}_k(\boldsymbol{a}_k[q-1], \boldsymbol{a}_k[q-2], \ldots), \end{aligned} \tag{5.2}$$

where $\hat{\boldsymbol{\theta}}_k[q] = \hat{\boldsymbol{s}}_k[q] \cup \hat{\boldsymbol{a}}_k[q]$ is the predicted set of parameters for the kth multipath track.

Now, we can formulate the tracking problem as follows : having found L estimated parameters $\boldsymbol{s}_l[q]$, $l = 1, \ldots, L$, it is required to assign them optimally to the K existing tracks in order to reconstruct the proper temporal sequence of the multipath parameters $\boldsymbol{\theta}_k[q]$, $k = 1 \ldots K$.

The hypermodels $\mathcal{S}_k(\cdot)$ are the key elements in solving this problem, since they provide predictions $\hat{\boldsymbol{s}}_k[q] = \mathcal{S}_k(\boldsymbol{s}_k[q-1], \boldsymbol{s}_k[q-2], \ldots)$ for the K tracks of interest. The optimum associations should then minimize some distance functional between predictions $\hat{\boldsymbol{s}}_k[q]$, $k = 1 \ldots K$, and newly estimated parameters $\boldsymbol{s}_l[q]$, $l = 1, \ldots, L$.

Now, let us formulate the association problem more formally.

5.1.1 Dynamic programming and assignment problem

Consider three possible track continuation scenarios at time q, shown as directed graphs in Fig. 5.2. As an example, we consider cases when $K = L$ (Fig. 5.2(a)), when $K < L$ (Fig. 5.2(b)), and when $K > L$ (Fig. 5.2(c)).

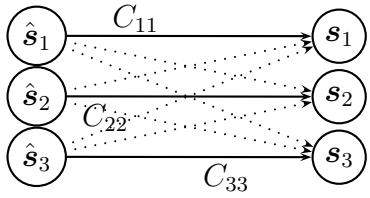
The graph edges indicate possible track continuations as connections between the predicted $\hat{\boldsymbol{s}}_k[q]$ and the newly estimated $\boldsymbol{s}_l[q]$ parameters. Each connection induces a dynamic cost $C_{kl}[q]$ computed as

$$C_{kl}[q] = f(\hat{\boldsymbol{s}}_k[q], \boldsymbol{s}_l[q]) + \mu C_k[q-1]. \tag{5.3}$$

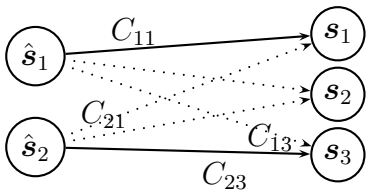
Here $C_k[q-1]$ is the cost accumulated by the kth track up to the time $q-1$, and $0 \leq \mu \leq 1$ is a forgetting factor. The function $f(\cdot, \cdot)$ measures the closeness between the predicted and estimated structure parameters.

Now, let us define a binary variable x_{kl} such that:

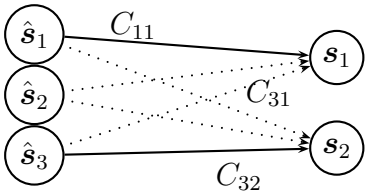
$$x_{kl} = \begin{cases} 1, & \text{if } \boldsymbol{s}_l[q] \text{ should be assigned to } \hat{\boldsymbol{s}}_k[q] \ ; \\ 0, & \text{otherwise.} \end{cases}$$



(a) $K = 3$, $L = 3$.



(b) $K = 2$, $L = 3$.



(c) $K = 3$, $L = 2$.

Figure 5.2: Possible track continuation scenarios.

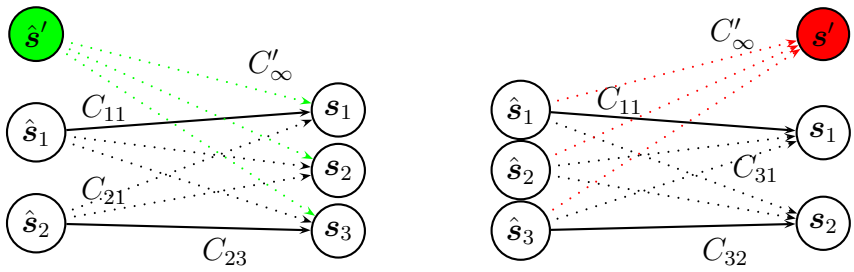
Then, optimal association should minimize the total induced cost Z:

$$\begin{aligned} &\underset{x_{kl}}{\operatorname{argmin}}\, Z = \sum_{k=1}^{K}\sum_{l=1}^{L} C_{kl}[q]x_{kl}, \quad \text{so that} \\ &\sum_{l=1}^{L} x_{kl} = 1,\ k = 1 \dots K, \quad \text{and} \quad x_{kl} \in \{0, 1\}. \end{aligned} \tag{5.4}$$

Formulation (5.4) known in the literature as the *assignment problem* and can be solved using linear programming methods [MS00b, Tah02]. A classical assignment problem occurs in situations when, for example, it is needed to assign several workers to different jobs. Each worker i requests a certain payment c_{ij} to perform a job j. The assignment problem appoints the workers to the corresponding job so that the total cost is minimized. This formulation is clearly equivalent to our case with the jobs being equivalent to the tracks of interest and workers to the estimated multipath components. Should we have just a single track, i.e., $K = 1$, then for $\mu = 1$ in (5.3) we obtain an instance of the classical Viterbi algorithm [Rab89]. However, in our case all of the K tracks have to be simultaneously associated with L candidates, which makes the problem more difficult.

The standard solution to (5.4) requires the assignment problem to be balanced: the number of workers and the number of jobs must be the same, which in our case translates into $K = L$. (Fig. 5.2(a)). This requirement can be easily satisfied by introducing dummy variables into the analysis.

Let us first consider the case when $K < L$. It follows that we need to augment the existing K tracks with $L - K$ dummy predictions $\hat{\boldsymbol{\theta}}'$, as shown in Fig. 5.3(a). The



(a) $K = 2$, $L = 3$. (b) $K = 3$, $L = 2$.

Figure 5.3: Augmented graphs for balancing the assignment problem.

weights C'_∞ along the edges between the dummy predictions $\hat{\boldsymbol{s}}'$ and the estimates $\boldsymbol{s}_l$, $l = 1, \ldots, L$ are set to a sufficiently large number, e.g., $C'_\infty = 10^{14}$, to make sure they will not affect the assignments for the real tracks[1]. Once the solution is found the dummy variables are removed and the remaining associations are used further in the algorithm for hypermodel updates. This is what is implemented in all our simulations. Alternatively, one can also consider this situation as a possibility to introduce new tracks in the analysis. This however stays outside the scope of this work.

The other case, when $K > L$, is a bit more difficult. It is basically equivalent to the situation when some of the tracks have ceased to exist (possibly only temporarily) due to the change in the propagation environment, or, simply, due to becoming to weak to be detected.

To balance the problem, we need to introduce $K - L$ dummy estimates $\boldsymbol{s}'$ (Fig. 5.3(b)), just as we did for the case ($K < L$), and solve the association problem as usual. However the dummy data $\boldsymbol{s}'$ cannot be used in making association and hypermodel updates since it is artificially added. As a consequence, the corresponding tracks have to be deleted from the analysis.

It might, however, be advantageous to refrain from deleting the tracks immediately. The estimation algorithm might have simply missed the component temporarily due to, for instance, unresolved multipath component superposition, and there is a chance it might be rediscovered several steps later. To account for this we leave the parameters of this track unchanged, hoping that this deletion happend only temporarily. This is done using the following strategy:

- First, we solve the augmented association problem with dummy variables.
- The tracks that are assigned the dummy parameters $\boldsymbol{s}'$ use the previous track parameters $\boldsymbol{\theta}_k[q-1]$ as their continuation, i.e., $\boldsymbol{\theta}_k[q] = \boldsymbol{\theta}_k[q-1]$.

We may also note how often we do not find any association for a certain track.

[1] How the assignments are going to be resolved between the dummy variables is absolutely unimportant.

This number can be used to guide our decision on whether the track must be really deleted or not.

5.1.2 Selecting the cost function

Measuring the similarity (closeness) between the predicted values $\hat{\boldsymbol{s}}_k[q]$ and the newly estimated ones $\boldsymbol{s}_l[q]$ with the distance function $f(\cdot,\cdot)$ is an important step in solving the association problem. The natural measure of such similarity would be absolute distance or Euclidian distance. However, straight-forward application of these concepts would result in inappropriate computations of the similarity measure, since the parameters in both $\boldsymbol{s}_l[q]$ and $\hat{\boldsymbol{s}}_k[q]$ have very different physical units.

In [SÖH+02] authors proposed a measure to evaluate the distance between two multipath components – multipath component distance (MCD). We can adopt the same measure in the computation of the cost function for track association.

The MCD between any two multipath components k and l is defined as

$$MCD_{kl}^2 = \sum_{i\in\{\tau,\nu,\phi\ldots\}} MCD_{i,kl}^2, \tag{5.5}$$

which is the radius of the hypersphere in the normalized multipath parameter distance space. The index i here spans several different physical dimensions describing the multipath component, and thus $MCD_{i,kl}$ is the distance between the multipath components along dimension i. In fact, an appropriately normalized Euclidian distance is the essence of the MCD.

Normalization is necessary. First of all, it is important to make sure that all multipath parameters contribute equally to the computation of the distance. Second, since the $MCD_{i,kl}$ are added together to obtain the final MCD_{kl}, we must make sure that we add quantities with the same physical units. In [SÖH+02] it was proposed to normalize $MCD_{i,kl}$ such that $0 \leq MCD_{i,kl} \leq 1$ for all multipath parameters.

A possible normalization for the delay is

$$MCD_{\tau,kl} = \frac{|\tau_k - \tau_l|}{\Delta\tau_{max}}, \tag{5.6}$$

where $\Delta\tau_{max} = \max_{k,l}|\tau_k - \tau_l|$ is the maximum delay spread. Practically $\Delta\tau_{max}$ is selected based on some *a priori* information of the channel delay spread.

Similarly, Doppler information can also be normalized as follows:

$$MCD_{\nu,kl} = \frac{|\nu_k - \nu_l|}{2\nu_{max}}, \tag{5.7}$$

where ν_{max} is the maximum Doppler frequency magnitude.

For spatial MIMO systems, component distance for the angular information, $MCD_{DoA,kl}$ and $MCD_{DoD,kl}$ can be computed as the normalized Euclidian distance between two points on a unit sphere. Figure 5.4 illustrates this concept for the case of the DoA. Here, ϕ_l and ϑ_l are azimuth and $\pi/2$-elevation angle of the lth multipath

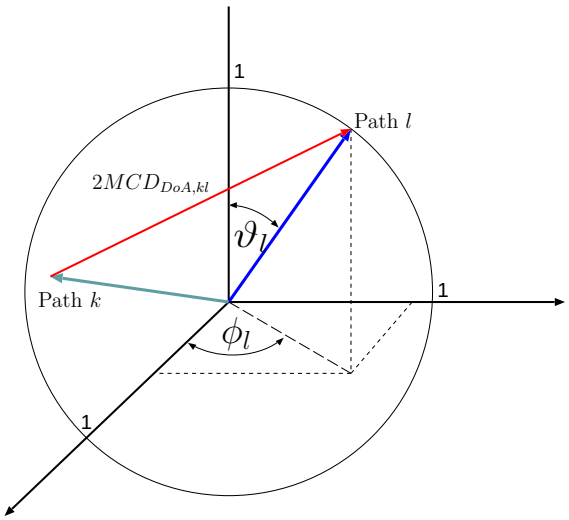


Figure 5.4: Geometrical definition of the spatial component $MCD_{DoA,kl}$.

component at the receiver, respectively. Thus the resulting distance $MCD_{DoA,kl}$ (or equivalently $MCD_{DoD,kl}$) can be computed as

$$MCD_{DoA,kl} = \frac{1}{2}\left|\begin{pmatrix}\sin(\vartheta_k)\sin(\phi_k)\\ \sin(\vartheta_k)\cos(\phi_k)\\ \cos(\vartheta_k)\end{pmatrix} - \begin{pmatrix}\sin(\vartheta_l)\sin(\phi_l)\\ \sin(\vartheta_l)\cos(\phi_l)\\ \cos(\vartheta_l)\end{pmatrix}\right|. \tag{5.8}$$

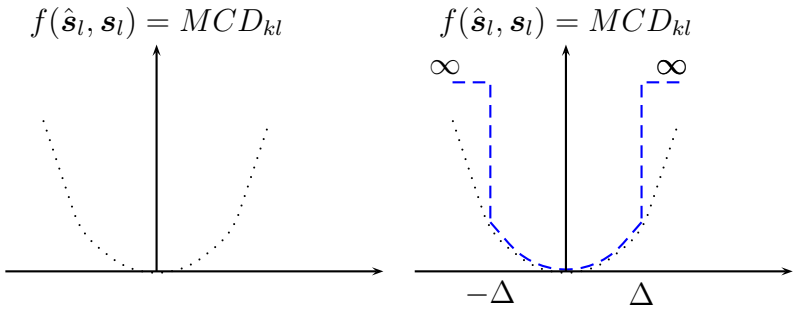
Although each of the contributing distances $MCD_{i,kl}$ is normalized, the resulting measure MCD_{kl} is not. Deciding to normalize MCD_{kl} might lead to inconsistent results should the same data be processed with different number of physical dimensions. Thus, as suggested in [SÖH+02], it more appropriate to leave the resulting distance unnormalized.

Further, we will adopt several slight modifications to the resulting MCD to tailor it to our needs. The considered distance measure is a monotonically increasing function of the distance between the components (Fig. 5.5(a)). In case of tracking/association, it is reasonable to assume that the parameters of a single multipath component do not differ significantly between two consecutive channel blocks, i.e., there are no parameter jumps. Thus, we are more inclined to have a function that is monotonic only in a certain sensitivity region $\pm\Delta$ (Fig.5.5(b)). Outside the sensitivity region $\pm\Delta$ the cost function attains the maximum value irrespective of the value of the argument.

Let us consider the following example that illustrates the necessity to introduce this sensitivity region.

Example

Let us consider for simplicity a single track, i.e., $K = 1$, and $\boldsymbol{s}[q] = \{\tau[q]\}$.



(a) MCD introduced in [SÖH+02].

(b) Modified MCD for multipath tracking.

Figure 5.5: The form of the distance function $f(\cdot, \cdot)$ for a single parameter.

The predicted value of the multipath components (assuming the hyper model is known) is $\hat{\tau} = 3.5\mu$sec. The samples of the impulse response were obtained with the sampling period $T_s = 1\mu$sec.

The estimation algorithm finds two multipath components with the delays $\tau_1 = 3\mu$sec and $\tau_2 = 9\mu$sec. It is clear, that the optimal track continuation would be to choose τ_1 as the track continuation, since $f(\hat{\tau}, \tau_1) < f(\hat{\tau}, \tau_2)$.

Now, let us assume that $\tau_1 = 8.9\mu$sec and $\tau_2 = 9.0\mu$sec. Here again $f(\hat{\tau}, \tau_1) < f(\hat{\tau}, \tau_2)$, thus we are tempted to make the same assignment as before. But this would most likely correspond to the wrong physical multipath component since the estimated components arrive significantly later in time (in this case almost 6 sampling instances later). In this situation we must declare that there is no candidate to use as the track continuation.

As we can see, the sensitivity region allows us to exclude assignments of the multipath components that are too far away from the candidates. Taking this into account requires the appropriate re-normalization of the discussed $MCD_{i,kl}$ terms. For delay and Doppler frequency these modifications take the following form:

$$MCD_{\tau,kl} = \frac{|\tau_k - \tau_l|}{\Delta_\tau}, \quad MCD_{\nu,kl} = \frac{|\nu_k - \nu_l|}{\Delta_\nu}, \tag{5.9}$$

where Δ_τ and Δ_ν are the sensitivity regions for delays and Doppler spreads, respectively. Similarly, one can re-normalize the $MCD_{DoA,kl}$. The sensitivity regions should be chosen so as to reflect some *a priori* information about allowable parameter variations. This information might come from, for example, the known resolution ability of the measurement equipment, noise level, some specific features of the propagation environment, etc.

Note that sensitivity regions may transform a $K < L$ case into the $K \geq L$ case, when the number of allowable continuations is less than the number of tracks, even when number of estimated components L is large.

It is also convenient to include the component weighting in the computation of the total cost in (5.5) to amplify the influence of some parameters as compared to the other:

$$MCD_{kl}^2 = \sum_{i\in\{\tau,\nu,\phi\ldots\}} W_i \cdot MCD_{i,kl}^2, \qquad \sum_{i\in\{\tau,\nu,\phi\ldots\}} W_i = 1, \tag{5.10}$$

where W_i are some predefined weights. The weighting is useful since the different multipath parameters are estimated with different resolution. For example, it makes sense to give $MCD_{\tau,kl}$ more weight since the resolution in delay is usually much higher, as compared to the Doppler frequency of angular information.

5.2 Structure hypermodel $\mathcal{S}_k$ for channel tracking

Previously we defined very abstractly the hypermodels $\mathcal{S}_k$ and $\mathcal{A}_k$ associated with the tracked multipath components. In a sequel we explain how to construct the corresponding hypermodels and how they can be trained.

Since the whole tracking/prediction approach is Bayesian-inspired, we employ Bayesian sequential methods for learning track hypermodels $\mathcal{H}_k$ as well. First of all, the Bayesian methodology is quite general and, as we will see later, can be applied to constructing the hypermodels $\mathcal{S}_k$, as well as $\mathcal{A}_k$. Second, using a sequential method we can build the model from scratch as the data arrives.

As we previously mentioned, for parameter tracking/association we need a one-step-ahead predictor (5.1) for the parameter subset $\boldsymbol{s}_k[n]$ to compute the cost (5.3). This one-step prediction can be accomplished by a dedicated structure hypermodel $\mathcal{S}_k$. A small prediction horizon allows to approximate the trajectory of the track $\boldsymbol{s}_k[n]$ with relatively simple models. One such model is a so-called damped local linear trend (DLLT) [Har89] discussed below. Note that this is equivalent to assuming linear dynamics for the multipath parameters, as it was also suggested in [Sem03].

5.2.1 Damped local linear trend

Assuming that the multipath parameters evolve smoothly with time, we can try to locally approximate the parameter trajectories with straight lines (or more generally, polynomials). Here we will assume that the linear extrapolation is sufficient. The DLLT is a simple linear model that a) can be learned with the standard Kalman filter framework, and b) can be employed to implement the required one-step ahead extrapolation.

For a single kth track, the state-space representation of this filter is given as:

$$\begin{cases} \begin{bmatrix} \hat{\boldsymbol{s}}_k[q+1] \\ \boldsymbol{v}_k[q+1] \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{\Delta}_k \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{s}}_k[q] \\ \boldsymbol{v}_k[q] \end{bmatrix} + \boldsymbol{\xi}_k[q] \\ \boldsymbol{s}_k[q] = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{s}}_k[q] \\ \boldsymbol{v}_k[q] \end{bmatrix} + \boldsymbol{\epsilon}_k[q], \end{cases} \tag{5.11}$$

where $\boldsymbol{I}$ is an identity matrix of the appropriate size, $\boldsymbol{v}_k[n]$ is a vector of estimated DLLT slopes, and $\boldsymbol{\Delta}_k = \mathrm{diag}([\delta_\tau, \delta_\nu, \delta_\phi])$ are fixed damping factors for each of the multipath parameters. The damping factors are chosen such that $0 \leq \delta_\tau, \delta_\nu, \delta_\phi \leq 1$. Practically, we select $\boldsymbol{\Delta}_k = 0.1\boldsymbol{I}$. Also note that when $\boldsymbol{\Delta}_k = \boldsymbol{0}$ the DLLT converges to the classical random walk model.

Although for tracking we need only $\mathcal{L} = 1$ step prediction, higher prediction horizons can be realized by recursive application of the transition equation (5.11) exactly $\mathcal{L}$ times. It has been shown [Har89] that for an $\mathcal{L}$-step-ahead predictor based on the information up to the moment of time q, eq. (5.11) converges to the value $\hat{\boldsymbol{s}}_k = \hat{\boldsymbol{s}}_k[q] + \boldsymbol{v}_k^T[q](1 - \boldsymbol{\Delta}_k)^{-1}$ as $\mathcal{L} \to \infty$.

The disturbance terms $\boldsymbol{\epsilon}_k[q]$ and $\boldsymbol{\xi}_k[q]$ are assumed to be zero-mean Gaussian processes[2]. However their variances remain important design parameters. Since the multipath parameters cannot be estimated with zero variance, the observation noise $\boldsymbol{\epsilon}_k[n]$ can be related to the residual estimation uncertainty of the SAGE algorithm. Due to the unbiasedness and consistency of the SAGE-obtained estimates [FTH+99], the disturbance $\boldsymbol{\epsilon}_k[n]$ can be treated as a white Gaussian estimation noise. State noise $\boldsymbol{\xi}_k[n]$, on the other hand, is left as a free design parameter. Practically, we choose it so as to make sure that the ratio between the variance of the state noise and that of the observation noise is ≈ 0.01.

The Kalman filter allows to find the states of (5.11) iteratively, as the data arrives. Clearly, this requires a proper initializations. The initialization of the hypermodels $\mathcal{S}_k$ is chosen so as to repeat the last seen value. This can be achieved by selecting $\boldsymbol{v}_k[0] = 0$, and setting $\boldsymbol{s}_k[0]$ to the true estimated multipath parameters at $q = 0$. Assuming smooth parameter variations, the hypermodel predictions will not wander too far from the true future values. Such initialization is more likely to result in correct associations, and thus the proper values are going to be used to update the predictor coefficients during the later iterations.

5.3 Hypermodels $\mathcal{A}_k$

Once we solve the tracking/association problem, we can consider the evolution of the parameters $\boldsymbol{a}_k[q]$ and build predictors for them. As we mentioned, the set $\boldsymbol{a}_k[q]$ includes parameters that are not involved in tracking and for which long term-prediction is needed. The required hypermodels $\mathcal{A}_k$ might thus be more complicated, as compared to $\mathcal{S}_k$.

[2]Note that $\boldsymbol{\xi}_k[q]$ used in (5.11) should not be confused with the additive channel noise defined in Chapters 3 and 4.

Multipath power prediction is often a desired output of channel forecasting. In power prediction we are mostly interested in accurately modeling the evolution of the multipath gains and extrapolating it beyond the observation interval. Thus, $\boldsymbol{a}_k[q] = \{a_k[q]\}$.

In the sequel we present several possible implementations of the hypermodel structures for gain prediction and the corresponding learning strategies.

5.3.1 Adaptive Linear Predictor (ALP)

The first predictor we propose is based on a simple linear model. The structure of such a predictor is given as :

$$\hat{a}_k[q+\mathcal{L}] = \sum_{m=0}^{Q-1} c_k[m]a_k[q-m] = \boldsymbol{c}_k[q]^T\boldsymbol{\alpha}_k[q], \tag{5.12}$$

where $\mathcal{L} \geq 1$ is the prediction interval, and $Q > 0$ is the order of the predictor. In (5.12) $\boldsymbol{\alpha}_k[q] = [a_k[q], \ldots, a_k[q-Q+1]]^T$ is a vector of delayed gain observations, and $\boldsymbol{c}_k[q] = [c_0[q], \ldots, c_{Q-1}[q]]^T$ are the time-varying predictor coefficients.

Due to the linearity of (5.12) the coefficients $\boldsymbol{c}_k[q]$ can be estimated and updated with the classical Recursive Least Squares (RLS) algorithm [MS00b].

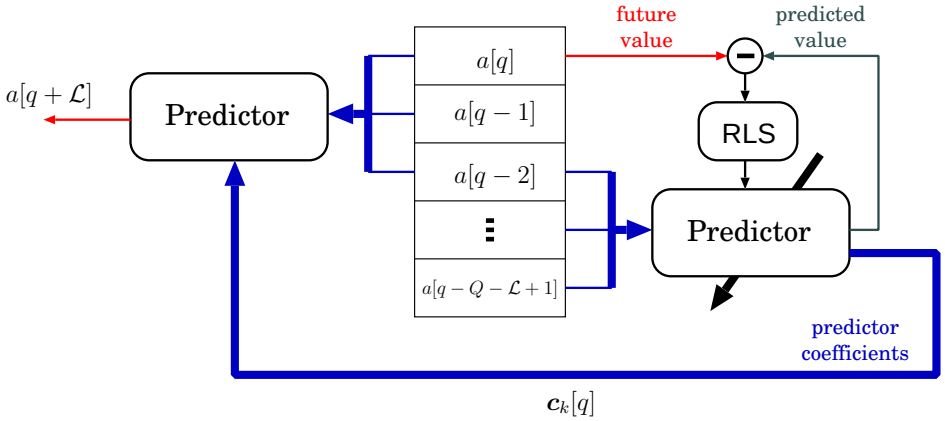


Figure 5.6: Structure of the ALP with RLS-based adaptation of predictor coefficients for $\mathcal{L} = 2$.

Model (5.12) and the RLS algorithm form the basis of the Adaptive Linear Predictor (ALP). The block diagram of the ALP learning is shown in Fig. 5.6. The samples $a_k[q]$ are stored in the buffer that is used to simultaneously update the predictor coefficients and make predictions. The size of the buffer needed to store all the necessary data is $Q + \mathcal{L}$.

As it can be seen, we use the newest sample $a_k[q]$ to update the predictor coefficients. Once the new predictor coefficients are estimated, they are immediately used in obtaining forecasts.

Note that the ALP is trained for a fixed prediction horizon $\mathcal{L}$. If multiple prediction horizons are needed, we would be forced to re-learn the predictor, or train several of them in parallel for every value of $\mathcal{L}$, which of course increases the computational load.

5.3.2 Iterated Adaptive Linear Predictor (IALP)

Another type of the linear predictor we use in our work is an Iterated Adaptive Linear Predictor (IALP). Similarly to the ALP, this predictor utilizes structure (5.12) with $\mathcal{L} = 1$ to make predictions. However, it exploits the Kalman Filter framework to estimate the predictor coefficients. As we will see later, this predictor can be used in a manner that allows different prediction intervals $\mathcal{L}$ without the need to re-train the model.

For the case, $\mathcal{L} = 1$ we can cast this predictor in a state-space form as

$$\begin{cases} \begin{bmatrix} \hat{\boldsymbol{\alpha}}_k[q+1] \\ \boldsymbol{c}_k[q+1] \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \boldsymbol{c}_k[q]^T \\ \boldsymbol{I}_{Q-1\times Q} \end{bmatrix} \hat{\boldsymbol{\alpha}}_k[q] \\ \boldsymbol{I}\boldsymbol{c}_k[q] \end{bmatrix} + \begin{bmatrix} \boldsymbol{\eta}_{\alpha,k}[q] \\ \boldsymbol{\eta}_{c,k}[q] \end{bmatrix} \\ a_k[q] = \begin{bmatrix} 1 & \mathbf{0}_{1\times Q-1} \end{bmatrix} \hat{\boldsymbol{\alpha}}_k[q] + \varsigma_k[q], \end{cases} \tag{5.13}$$

where $\boldsymbol{c}_k[q] \in \mathbb{C}^Q$ is a vector of model coefficients, $\hat{\boldsymbol{\alpha}}_k[q] \in \mathbb{C}^Q$ a vector of delayed gain observations as in (5.12), and $\boldsymbol{I}_{Q-1\times Q}$ is a $Q-1 \times Q$ rectangular matrix with the 1's on the diagonal $I_{ii} = 1$, $i = 1, \ldots, Q-1$.

It can be seen that the KF is used not only to track filter states, but also to estimate predictor coefficients, which in the Kalman filter context is known as the Joint estimation problem [Hay01, ch. 5]. In this form the joint filter states are interdependent, forming the bilinear state-space representation. This nonlinearity prevents the application of the standard KF algorithm. However, it is still possible to apply the Joint Extended Kalman Filer (EKF) [Hay01, ch. 5] that circumvents the nonlinearity problem and enables joint estimation.

The role of the disturbance terms $\varsigma_k[q]$, $\boldsymbol{\eta}_{\alpha,k}[q]$ and $\boldsymbol{\eta}_{c,k}[q]$ in (5.13) is basically the same as of $\boldsymbol{\xi}_k[q]$ and $\boldsymbol{\epsilon}_k[q]$ in (5.11). They are assumed to be zero-mean Gaussian processes and their variances remain free design parameters. Here as well we kept the ratio between the variance of the state noise and that of the observation noise on the order of 0.01. We should however mention that there are methods to estimate the variance of the disturbance terms iteratively within the Bayesian framework [Hay01] so as to minimize the prediction error. Accommodation of this case presents a challenging predictor design problem that should be addressed in further research.

Note that although the hypermodel (5.13) is functionally equivalent to the ALP, it, however, differs significantly in the way the predictions for longer $\mathcal{L}$ are realized. The ALP is trained for a particular prediction horizon $\mathcal{L}$, as can be seen from Fig. 5.6, while the IALP makes forecasts for $\mathcal{L} > 1$ by the recursive application of the one-step-ahead state-transition equation in (5.13) exactly $\mathcal{L}$ times, hence its name "iterated" predictor. It is also worth saying that in the IALP case the predictions

are made based on the filtered states $\hat{\boldsymbol{\alpha}}_k[q]$ rather than directly on the samples of the time series $a_k[q]$, as it is the case with the ALP.

5.3.3 Nonlinear predictor based on Volterra models (AVNP)

The basis for the ALP and IALP hypermodels constitutes the so called *autoregressive* (AR) model [MS00a]. The idea of autoregression is simple – the future sample is represented by a linear combination of the past measurements. ALP and IALP differ in the way how the predictor coefficients are estimated and how the predictions are obtained, but they are both linear predictors.

The linear predictor is able to capture only the linear dependencies in the observed signal. If the signal that is to be predicted has some nonlinear structure then the linear predictor is only a suboptimal one.

A nonlinear extension of the AR model is known as a Nonlinear Autoregression (NAR). In this case the future sample is represented by a nonlinear combination of the past observations. The distinction between different NAR models lies in the way this nonlinearity is represented.

The type of predictor we consider here represents the nonlinearity using Volterra models [MS00a]. In general, the Rth discrete Volterra filter is given as

$$\begin{aligned} y[n] =& h_0 + \sum_m h_1[m]x[n-m] + \\ & \sum_m \sum_l h_2[n-l, n-m]x[n-l]x[n-m] + \dots \\ & + \sum_m \dots \sum_l h_R[n-l, \dots, n-m]x[n-l]\dots x[n-m], \end{aligned} \tag{5.14}$$

where $x[n]$ is an input signal and $y[n]$ is the output of the Volterra model, and $h_r[\cdots]$, $r = 0, \dots, R$ are the so-called Volterra kernels.

It can be seen that h_0 captures a possible bias in the data, $h_1[n]$ is a linear impulse response, $h_2[m, l]$ captures the quadratic nonlinearity, $h_3[m, l, k]$ captures the cubic one, and so on, until the Rth order[3].

Clearly, the number of coefficients needed to represent each order of nonlinearity grows exponentially. As the result more data is required to reliably estimate these coefficients, and thus the learning time increases. This might present extra difficulty should the channel exhibit fast temporal variations – there might be not enough samples to estimate the model coefficients. However, if the variations occurring in the signal are nonlinear, we might still capture them with the NAR and thus extend the range of model validity.

Now, let us discuss how the Volterra filter can be implemented to accomplish the task we need. The structure of the Volterra filter can be easily formed by combining

[3]Theoretically, infinite Volterra series, both of infinite order and infinite memory length, are possible, but they are of little practical interest for us. The infinite representation is usually truncated at a certain level that ensures sufficient approximation quality.

nonlinearly the elements from a simple delay line. The following examples illustrate this principle.

Example

Let us consider the following nonlinear system:

$$y[n] = x[n] - 4x^2[n-1] + 2x[n-1]x[n] + x[n-2]x[n] + x^3[n-3]$$

The signal flow diagram that implements this system is shown in Fig. 5.7.

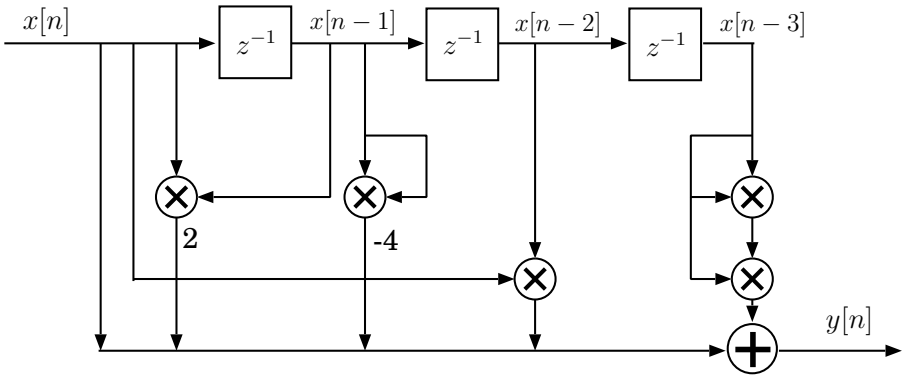


Figure 5.7: Signal flow diagram of the Volterra filter.

The number of elements in the delay line represents the filter memory, while the interconnections between the elements form the required nonlinearity.

It can be seen that the difference that distinguishes the Volterra filter from a simple linear FIR filter is the way the elements in the delay line are combined. In the simple FIR they are combined linearly, i.e., weighted and added together, while in the Volterra filter the elements are interconnected so as to represent the required nonlinearity and, only then, they are linearly combined.

It follows that transforming a linear ALP into the nonlinear Adaptive Volterra-based Nonlinear Predictor (AVNP) is actually not that difficult. Let us consider as an example a second order system with the linear and quadratic memory lengths equal to N_l and N_q, respectively. The model bias is assumed to be 0. Extension to the other possible configurations of Volterra models is straightforward.

Let us define a vector $\boldsymbol{h}_1 = [h_1[0], \ldots, h_1[N_l - 1]]$ that contains the coefficients of the linear kernel $h_1[m]$. Similarly we collect the coefficients of the quadratic part $h_2[m, l]$ into a vector $\boldsymbol{h}_2 = \mathrm{vec}(h_2[m, l])$ by stacking the columns of the matrix $h_2[m, l]$. Note that due to the symmetry of the kernel coefficients [MS00a], the number of unique elements in $h_2[m, l]$ is only $N_q(N_q + 1)/2$. Now, we form the joint coefficient vector $\boldsymbol{h}$ as

$$\boldsymbol{h} = \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \end{bmatrix}.$$

Further, let $M_N = \max(N_l, N_q)$ denote length of the required delay line, and let $a_k[q]$, $q = 0, \ldots, Q-1$ be the samples of the observed signal that is used to train the Volterra predictor. Let us further define a memory vector $\boldsymbol{\alpha}_k[q] = [a_k[q], a_k[q-1], \ldots, a_k[q-Q-M_N+1]]^T$ and the Volterra operator $\mathcal{V}(\boldsymbol{\alpha}_k[q], \boldsymbol{h})$ as

$$\mathcal{V}(\boldsymbol{\alpha}_k[q], \boldsymbol{h}) = \sum_m h_1[m]a_k[n-m] + \sum_m \sum_l h_2[n-l, n-m]a_k[n-l]a_k[n-m]. \tag{5.15}$$

Then, training the predictor consists in solving the system of simultaneous equations

$$\begin{aligned} &a_k[\mathcal{L}] = \mathcal{V}(\boldsymbol{\alpha}_k[0], \boldsymbol{h}), \\ &a_k[\mathcal{L}+1] = \mathcal{V}(\boldsymbol{\alpha}_k[1], \boldsymbol{h}), \\ &\ldots \\ &a_k[Q-1] = \mathcal{V}(\boldsymbol{\alpha}_k[Q-\mathcal{L}-1], \boldsymbol{h}), \end{aligned} \tag{5.16}$$

assuming $\mathcal{L} \geq 1$.

Expression (5.16) can also be transformed into the vector form as

$$\boldsymbol{\alpha}_k = \boldsymbol{A}_k \boldsymbol{h}, \tag{5.17}$$

where now $\boldsymbol{\alpha}_k = [a_k[0], a_k[1], \ldots, a_k[Q-1]]^T$ are the samples of the observed signal, and $\boldsymbol{A}_k = [\boldsymbol{A}_{k1}\, \boldsymbol{A}_{k2}]$ is the observation matrix (or design matrix), with elements composed from the samples $a_k[q]$ delayed and combined according to the particular nonlinear structure. For example,

$$\boldsymbol{A}_{k1} = \begin{bmatrix} a_k[0] & \ldots & a_k[1-N_l] \\ \vdots & \ldots & a_k[2-N_l] \\ a_k[Q-1] & \ldots & a_k[Q-N_l] \end{bmatrix} \tag{5.18}$$

$$\boldsymbol{A}_{k2} = \begin{bmatrix} a_k^2[0] & a_k[0]a_k[-1] & \ldots & a_k^2[-1] & \ldots \\ a_k^2[1] & a_k[1]a_k[0] & \ldots & a_k^2[0] & \ldots \\ \vdots & \vdots & & \vdots & \\ a_k^2[Q-1] & a_k[Q-1]a_k[Q-2] & \ldots & a_k^2[Q-2] & \ldots \end{bmatrix} \tag{5.19}$$

From (5.17) we see that the unknown coefficients $\boldsymbol{h}$ enter the equation linearly, and thus we can employ linear optimization methods to estimate them. In fact, it is straightforward to apply the RLS algorithm [MS00b] to estimate the coefficients $\boldsymbol{h}$ in a recursive way.

The block diagram of the AVNP learning algorithm is shown in Fig. 5.8. We see that it does not differ substantially in its overall structure from its counterpart in Fig. 5.6 for the ALP predictor.

The distinction arises in the way the estimated coefficient vector $\boldsymbol{h}$ is used to obtain the predictions: in the ALP case it is a simple scalar product with the delayed signal samples $a_k[q]$, while in the AVNP case the coefficients $\boldsymbol{h}$ are combined with the delayed signal samples according to the specific nonlinear structure.

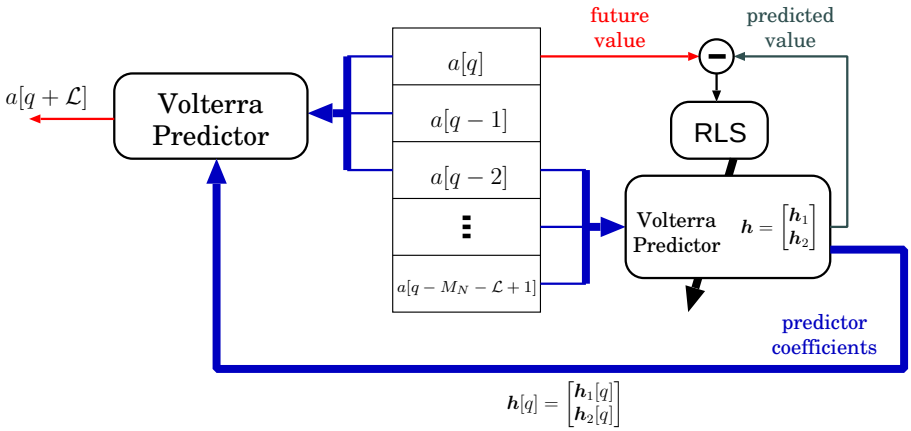


Figure 5.8: Structure of the Volterra model-based Nonlinear Predictor with RLS-based adaptation of predictor coefficients.

5.3.4 Nonlinear predictor based on Neural Networks (IANNP)

Similarly to the Volterra models, Neural Networks (NN) [Hay01] can also be used to capture the nonlinearity in NAR. By varying the coefficients of the NN we can create different nonlinear functions that result in different predictors. Just like in the case of other predictors, the hypermodel based on the NN-NAR should also be adjusted to the arriving data.

To adapt the network coefficients we will use here a structure similar to the IALP hypermodel. In other words we exploit the Kalman Filter framework to learn and adjust the filter coefficients recursively, giving rise to the Iterated Adaptive Neural Network-based Predictor (IANNP). The basics for applying the Kalman Filter methodology to NN learning are well described in [Hay01].

In general, the structure of a multilayer perceptron neural network is specified by

- The number of hidden layers,
- The number of neurons in each layer, including the input layer, and
- The form of the neuron activation functions.

The number of neurons (and thus the number of resulting coefficients) is directly proportional to the complexity of the resulting hypermodel. The more neurons are used, the more complex functions can this network approximate. It is known [HSW89] that a sufficiently big feedforward network with a single hidden layer can approximate any smooth bounded nonlinear function.

For us it is, however, important to make sure that the size of the network stays compact: the smaller the network is, the fewer coefficients need to be adapted. This minimizes the network learning time. We chose to implement the structure of the NN with a single hidden layer and a purely linear output layer, as shown in Fig. 5.9.

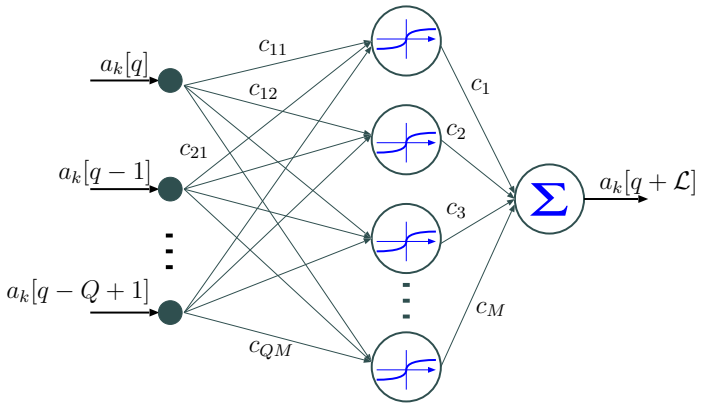


Figure 5.9: Structure of the Neural Network used for hypermodel approximation.

The neurons in the hidden layer all have sigmoidal activation functions. The number of neurons in the input layer, as well as in the hidden layer, are left as free network design parameters.

Now, let us consider the structure of the Kalman filter used to track and update the network coefficients.

Assuming that the NN has N_Q inputs and a total of M_Q weights, the state space of the corresponding NAR hypermodel is given as

$$\begin{cases} \begin{bmatrix} \hat{\boldsymbol{\alpha}}_k[q+1] \\ \boldsymbol{c}_k[q+1] \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} f(\boldsymbol{c}_k[q], \hat{\boldsymbol{\alpha}}_k[q]) \\ \boldsymbol{I}_{N_Q-1\times N_Q}\hat{\boldsymbol{\alpha}}_k[q] \end{bmatrix} \\ \boldsymbol{I}\boldsymbol{c}_k[q] \end{bmatrix} + \begin{bmatrix} \boldsymbol{\eta}_{\alpha,k}[q] \\ \boldsymbol{\eta}_{c,k}[q] \end{bmatrix} \\ a_k[q] = \begin{bmatrix} 1 & \mathbf{0}_{1\times(N_Q-1)} \end{bmatrix} \hat{\boldsymbol{\alpha}}_k[q] + \varsigma_k[q], \end{cases} \tag{5.20}$$

where $\boldsymbol{I}_{N_Q-1\times N_Q}$ denotes a $N_Q-1\times N_Q$ rectangular matrix with the 1's on the diagonal $I_{ii}=1$, where $i=1,\ldots,N_Q-1$, and $f(\cdot,\boldsymbol{c}_k[q])$ is a neural network parametrized by the time-varying coefficients $\boldsymbol{c}_k[q]$. In this formulation the vector $\boldsymbol{c}_k[q]$ consists of all the coefficients of the NN, shown in Fig. 5.9.

Similarly to the IALP, the standard Kalman filter cannot be used to solve this estimation problem due to the nonlinearity of the state-transition equation in (5.20). However, it is still possible to apply the Joint EKF framework to learn the coefficients of the neural network. For more information on Joint EKFs we refer the reader to [Hay01].

Also, just like in the IALP case the predictions for $\mathcal{L}>1$ are obtained by the recursive application of the state transition equation in (5.20).

5.4 Discussion and conclusions

Now let us discuss and conclude the ideas we introduced here for the tracking algorithm.

First note that we allow simultaneous tracking of K components. In theory, K might be as large as desired, upper bounded only by the true number of present

multipath component. In practice, we might however be limited in the available resources and reduce the number or tracked components to a possible minimum.

The main idea of the proposed tracking scheme is inspired by the sequential Bayesian estimation. The two major steps of the algorithm are the *prediction step* and the *update step*, just like prediction and propagation steps in the Bayesian sequential estimation [Hay01, MS00b].

Using the hypermodels we obtain single-step predictions of the multipath structure parameters – the prediction stage of the algorithm. The predicted structure defines the predicted "time-space position" of the tracked component. These predictions are then used to find which of the estimates obtained with the multipath estimation algorithm is associated with the current track.

An advantage of this scheme is its ability to construct the track hypermodel online from scratch. However, there is also a critical disadvantage, namely tracking errors propagation. If the association is wrong, consequently the wrong value is used as the track continuation and, as a result, the wrong value is used to update the hypermodel. Initially we try to minimize possible errors by assuming "singular" model structures. In other words, we initialize the models so as to repeat the last seen value at the output (i.e., to represent the random walk model). Assuming that the track dynamics does not change abruptly, such a "singular" model can be a good start to solve the associations in the beginning of the tracking, that in return triggers the adaptation of the model coefficients.

Tracks association

Track association is an essential part of the tracking approach we consider because it decides which estimates are going to be used for hypermodel updates.

Having K tracks results in a K dimensional constrained optimization procedure. This optimization task is formulated as a Dynamic Programming problem. The "dynamic" aspect arises simply because the current association cost (5.3) for the kth track depends on the previous costs evaluated along the optimal solution path. Since we minimize the total association costs we definitely prefer smooth track trajectories (with smaller costs). Smooth parameter change is crucial for the successful model building, since it prevents tracking error propagation.

Smooth parameter variations result from a high degree of correlation between the successive MIMO channel estimation windows. Thus, it is important to have spatially oversampled channel IR's. This will ensure smooth parameter trajectories that can be easily picked up and followed by the tracker.

Another very important aspect of the tracking algorithm is its sensitivity to the estimation artifacts. Such artifacts are forming mainly in the vicinity of strong components and the parameters of these artifacts are highly correlated with the parameters of the true component. Thus, the association algorithm is likely to assign similar costs to these estimates and quite possibly divert the component trajectory. The straightforward way to minimize the amount of estimation artifacts is to increase the intrinsic resolution of the measurement equipment. This is unfortunately not

always possible.

The tracker itself is not able to instantaneously distinguish between the true component and the artifact, but it might do it by observing the track component over time. The estimation artifacts eventually loose their influence once the physical component moves further away from the artifact position. Thus, it might be possible to detect these artifacts, return back in time, and adjust the tracker so that to avoid undesired tracking solutions. Of course, during this "time-reverse" the hypermodel will not be available for forecasting.

Hypermodels

Another important element in the tracking algorithm is the hypermodel.

The hypermodels we use are divided into two groups, depending on their function and application in the whole framework. We split the track hypermodel $\mathcal{H}_k$ into a set of two sub-models: structure hypermodel $\mathcal{S}_k$ and gain hypermodel $\mathcal{A}_k$. This dichotomy stems from the fact that the sub-models $\mathcal{S}_k$ and $\mathcal{A}_k$ are applied to different signals. The structure hypermodel $\mathcal{S}_k$ is the simplest one because it is needed to model relatively simple local dynamics of the track structure, which is then used in solving the association problem. We select it to be a simple damped Local Linear Trend model. Such model is linear and it can be easily updated with the standard Kalman filter.

On the other hand, the hypermodels $\mathcal{A}_k$ are more complex since the multipath gain variations are more difficult to model. In our work we consider hypermodels $\mathcal{A}_k$ for predicting complex multipath gain. We distinguish linear and nonlinear models (based on the hypermodel structure), as well as iterated one-step-ahead predictors, and $\mathcal{L}$-step predictors (depending on how the forecasts are realized).

The iterated and $\mathcal{L}$-step predictors differ in the way they come up with predictions for prediction intervals $\mathcal{L} > 1$. The iterated predictor, trained as a one-step-ahead predictor, obtains such forecasts by a closed loop prediction [Har89], i.e., by recursive application of the the one-step ahead prediction $\mathcal{L}$ times. The $\mathcal{L}$-step predictor, on the other hand, is trained for a specific prediction interval $\mathcal{L}$.

In learning the hypermodels we employ two different learning algorithms: the first one is based on the Recursive Least Squares, and the second one exploits Joint EKF methodology. RLS is used to train the $\mathcal{L}$-step predictors, while the joint EKF is used for iterated predictors. The types of the used hypermodels and the corresponding learning algorithms are summarized in Table 5.1.

	RLS	Joint EKF
Linear	ALP	IALP
Nonlinear	AVNP	IANNP

Table 5.1: Hypermodels used in multipath gain prediction.

The RLS algorithm is easier to implement and it has fewer free parameters, but it must be re-adapted for each new interval $\mathcal{L}$. EKF-based predictors exploit the

transition equation in the state-space formulation as a predicting function. They thus make predictions based on the filtered hypermodel states, but the state-space formulation has more free parameters that should be set up. Furthermore, with the closed-loop prediction we must be concerned with the stability of the resulting predictors. Although we propose to use quite a simple approach, this question deserves more rigorous mathematical investigation.

Although the linear models are usually easier to learn and interpret, the nonlinear can better represent the nonlinear dependencies in the signal. The disadvantage of the nonlinear models is that usually they require more data to reliably estimate the coefficients. This fact might render the usage of nonlinear models impractical, should we confront abrupt signal changes due to, for example, tracking or estimation errors.

Chapter 6

Multipath forecasting

The previous chapters discuss a sequence of channel processing steps, leading to the construction of multipath hypermodels. In this chapter we apply the proposed parameter estimation, tracking, and prediction algorithms to the measured channel data.

The multipath estimation, multipath parameter tracking, and prediction algorithms have several free parameters that have to be chosen to allow the application of the considered techniques. In Section 6.1 we discuss which and how these parameters can be selected. We also define the quality measure to assess the performance of the channel prediction algorithm.

In the Section 6.3 we consider tracking and prediction of the multipath component parameters estimated with the SAGE algorithm, which was discussed in Chapter 3. Using these data we also discuss hypermodel properties used for prediction.

In Section 6.4 we apply the prediction and tracking algorithms to the channel data estimated with the Evidence Procedure, namely, the SAGE-RVM algorithm proposed in Chapter 4.

Throughout this chapter we use the FTW data set (App. C) to demonstrate the results of channel prediction.

6.1 Choosing simulation parameters

Three major steps of the considered prediction framework, namely estimation, tracking and prediction require specification of several free parameters that control algorithm performance and properties. These parameters can be roughly grouped as:

- Channel estimation
  - Maximum number of components L to estimate.
  - Initialization of the SAGE-RVM and SAGE algorithms.
  - Initial noise variance used in the SAGE-RVM model selection.
- Channel tracking
  - Number of the tracks K to be reconstructed.

  - Forgetting constant μ_k used in computing the track cost (5.3).
  - Sensitivity regions $\mathbf{\Delta}$ used in computing the MCD in (5.10).
  - Weighting coefficients W_i in (5.10).

- Hypermodel design/ prediction
  - The damping factors $\mathbf{\Delta}$ used in the structure hypermodels $\mathcal{S}_k$ in (5.11).
  - Order of the hypermodels $\mathcal{A}_k$, as well as the corresponding structures of the nonlinearity in the case of the nonlinear hypermodels.
  - Selection of the disturbance parameters in the state-space formulations (5.13) and (5.20) of the predictor hypermodels.
  - Forgetting factor used in the RLS-based hypermodel learning algorithms.
  - Initialization of the $\mathcal{A}_k$ hypermodels.

Selecting or estimating these parameters is not a straightforward procedure. Below we provide some "rules of thumb" that have guided the selection of these parameters in our simulations.

Channel estimation

Channel estimation is a relatively autonomous procedure. The initialization of both the SAGE and SAGE-RVM algorithms has been discussed in the preceeding chapters. However, both algorithms require selection of the initial number of components L. SAGE cannot estimate the number of components, but SAGE-RVM can. From the analysis of FTW data with the SAGE-RVM algorithm, we established that the number of multipath components varies between $L = 7, \ldots, 18$. Thus, for the SAGE algorithm the initial number of components is chosen from this interval. Later we explicitly state how many components are used in the SAGE algorithm.

In order to implement model selection in the SAGE-RVM algorithm, we need to specify the initial variance of the noise. Practically, the noise variance is initialized by measuring the variance of the tails of the channel IR, where it is unlikely to observe any detectable multipath components.

Tracking

In comparison to channel estimation, the tracking algorithm has quite a few free parameters. First of all, it is the number of tracks K. This number is mainly dictated by the application constraints and available resources. One track is easier to reconstruct, but most likely it captures only a fraction of the total channel power. Many tracks are more difficult to track, but they reflect a larger portion of the total received power. As we will see from the experiments, the best strategy is to implement an intelligent track management algorithm that is able to decide how many tracks are needed and which components should be used in the tracker. Such

management algorithm is, however, outside the scope of this work. In the sequel we will demonstrate prediction results for different numbers of tracked components.

In the heart of the tracking/association algorithm lies the computation of the MCD. As we know, this requires the specification of the sensitivity regions Δ_τ, Δ_ν, and Δ_ϕ. In order to select them it might be helpful to exploit the resolution limitations induced by the measurement equipment. It is known that the acquisition bandwidth limits the resolution in the delay, the number of antennas influence the spacial resolution, and the length of the channel estimation window channel defines the resolution of the Doppler frequency.

Based on the corresponding parameters of the FTW data set, we used the following values: for the delay the sensitivity region Δ_τ was set to $2/(120 \cdot 10^6)$sec, which is double the inverse of the channel bandwidth. Concerning the sensitivity regions for Doppler and DoA parameters, we decided to set them to the maximum range. This was mainly done to account for the low data resolution in these domains, as compared to the resolution in the delay domain. In the computation of the MCD we also can specify the weighting W_i of individual parameter MCD_i's. As we previously said, this allows to control the contribution of some of the parameters to the final MCD. Taking this all into account we define the weighting coefficients as

$$W_\tau = 0.9, \quad W_\nu = 0.05, \quad W_\phi = 0.05.$$

Another important parameter is the forgetting factor μ_k for the dynamic track cost computation (5.3). The choice of this constant is quite arbitrary, but it should not be very high to make sure that the cost adapts to the time-varying environment, and it also should not be very low, to make sure that the reconstructed track remains smooth. In our simulations it was chosen to be $\mu_k = 0.9$ for all tracks.

Hypermodel design

The hypermodels also require specification of a set of parameters that control their behavior. Let us begin with the structure hypermodels $\mathcal{S}_k$.

To initialize the states $\hat{\boldsymbol{s}}_k[q]$ of the structure hypermodels we use the multipath parameters obtained for the first estimation window. States $\boldsymbol{v}_k[q]$ are set to zero to ensure that our untrained model will not produce absolutely irrelevant predictions.

For the DLLT hypermodel we also need to specify the slope constants $\boldsymbol{\Delta}_k$, as well as the parameters of the disturbance terms in the state-space model formulation. In our experiments we found that $\boldsymbol{\Delta}_k = 0.1\boldsymbol{I}$, $k = 1, \ldots, K$, provides sufficient model performance and good predictions. The disturbance terms $\boldsymbol{\xi}_k[q]$ and $\boldsymbol{\epsilon}_k[q]$ in (5.11) are assumed to be zero-mean Gaussian random variables. Their covariance matrices remain however an important design parameter and should be selected so as to optimize the agility of the resulting tracker and insensitivity to the errors. The values we used were mainly found by trial and error. The corresponding variances for the are specified in the Table 6.1.

	Delay τ		DoA ϕ		Doppler ν	
	$\sigma^2_\tau/\hat{\sigma}^2_\tau$	$\sigma^2_{\delta_\tau}/\hat{\sigma}^2_\tau$	$\sigma^2_\phi/\hat{\sigma}^2_\phi$	$\sigma^2_{\delta_\phi}/\hat{\sigma}^2_\phi$	$\sigma^2_\nu/\hat{\sigma}^2_\nu$	$\sigma^2_{\delta_\nu}/\hat{\sigma}^2_\nu$
$\frac{\text{var}\{\boldsymbol{\xi}[q]\}}{\text{var}\{\boldsymbol{\epsilon}[q]\}}$	0.02/0.9	0.02/0.9	0.025/1	0.025/1	0.025/1	0.025/1

Table 6.1: Variance of the disturbance terms in the state-space representation of the DLLT model for each of the structure parameters.

The parameters in Table 6.1 are related to the covariance matrices $\boldsymbol{\Sigma}_\xi$ of $\boldsymbol{\xi}_k[q]$, and $\boldsymbol{\Sigma}_\epsilon$ of $\boldsymbol{\epsilon}_k[q]$ as

$$\boldsymbol{\Sigma}_\xi = \text{diag}([\sigma^2_\tau, \sigma^2_\nu, \sigma^2_\phi, \sigma^2_{\delta_\tau}, \sigma^2_{\delta_\nu}, \sigma^2_{\delta_\phi}]^T)$$

$$\boldsymbol{\Sigma}_\epsilon = \text{diag}([\hat{\sigma}^2_\tau, \hat{\sigma}^2_\nu, \hat{\sigma}^2_\phi]^T)$$

Selecting parameters for the hypermodels $\mathcal{A}_k$ is a bit more tricky. First, for iterative learning we need to choose initial values for hypermodel parameters $\boldsymbol{c}_k[0]$. Generally, we select them as follows:

$$\boldsymbol{c}_k[0] = \begin{pmatrix} 1 \\ \boldsymbol{c}' \end{pmatrix},$$

where $\boldsymbol{c}' \sim \mathcal{N}(\boldsymbol{0}, \sigma\boldsymbol{I})$ is a vector of random hypermodel coefficients with zero mean and covariance matrix $\sigma\boldsymbol{I}$, with σ being some small number (e.g., $\sigma = 0.001$). Such initialization results in the "random walk" initialization of the hypermodel, i.e., we select the coefficients so as to be close to the random walk model. Random coefficient initialization allows then to average the prediction performance over different hypermodel solution trajectories.

It is also important to properly choose the order of hypermodels, as well the structure of nonlinearity, in case of AVNP or IANNP hypermodels. These are left as free design parameters and during the simulations we will consider several possible choices. As a general rule, we prefer compact models with few parameters, since training such predictors is simpler.

For the hypermodels learned with the RLS algorithm we also need to specify the forgetting constant. This, both for ALP and AVNP, was set to 0.9. This value was found to result in good prediction performance.

For the IALP and IANNP hypermodels, instead of the RLS forgetting factor, we need to specify the parameters of the disturbance terms present in the their state-space formulations. Similarly to the structure hypermodel $\mathcal{S}_k$, we find these values empirically though numerous experiments. We again assume the disturbance terms to be distributed as $\boldsymbol{\eta}_{\alpha,k}[q] \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_\alpha\boldsymbol{I})$, $\boldsymbol{\eta}_{c,k}[q] \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_c\boldsymbol{I})$, and $\varsigma_k[q] \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_\varsigma)$. The scaling factor for the covariance matrices are chosen so that $\sigma^2_\alpha/\sigma^2_\varsigma = 1/0.02$, and $\sigma^2_c/\sigma^2_\varsigma = 0.001/0.02$.

6.2 Measuring the prediction quality

It is also important to find a way to evaluate the performance of the prediction algorithm. We can anticipate that if the tracking algorithm makes errors, even temporarily, it leads to a burst-like degradation of the prediction performance. In addition, when the hypermodel is adapted, the corresponding transients also cause the prediction quality degradation.

A classical way to assess prediction quality is to compute the Prediction Gain (PG) that relates the power of the signal $a[q]$ to be predicted to the prediction error $e[q] = a[q] - a_{pred}[q]$ as

$$PG = 10 \log_{10} \left(\frac{P_{sig}}{P_{err}} \right), \tag{6.1}$$

where $P_{sig} = \sum_{q=1}^{N} |s[q]|^2/N$, and $P_{err} = \sum_{q=1}^{N} |e[q]|^2/N$ are the signal and prediction error powers, respectively, averaged over the segment of N samples.

We see that PG is equivalent to the Signal-to-Noise ratio. In our case, however, the straightforward application of (6.1) is not fully justified, since both the error signal and the signal we predict can exhibit short-term transient behavior, i.e., generally they are nonstationary. Thus, the computed average power might not be adequate.

A possible way to alleviate this problem is to consider a Segmental Prediction Gain – an equivalent of the Segmental SNR, often used in speech coding applications [O'S00]. The basic idea behind the segmental PG is quite simple: the data sequence is sectioned into relatively small chunks of size $\approx 2\lambda$, over which signal stationarity can be assumed. For each chunk i the individual PG_i is computed according to (6.1). The final Segmental PG is then found as an average over all the partial PG_i's over the whole data sequence.

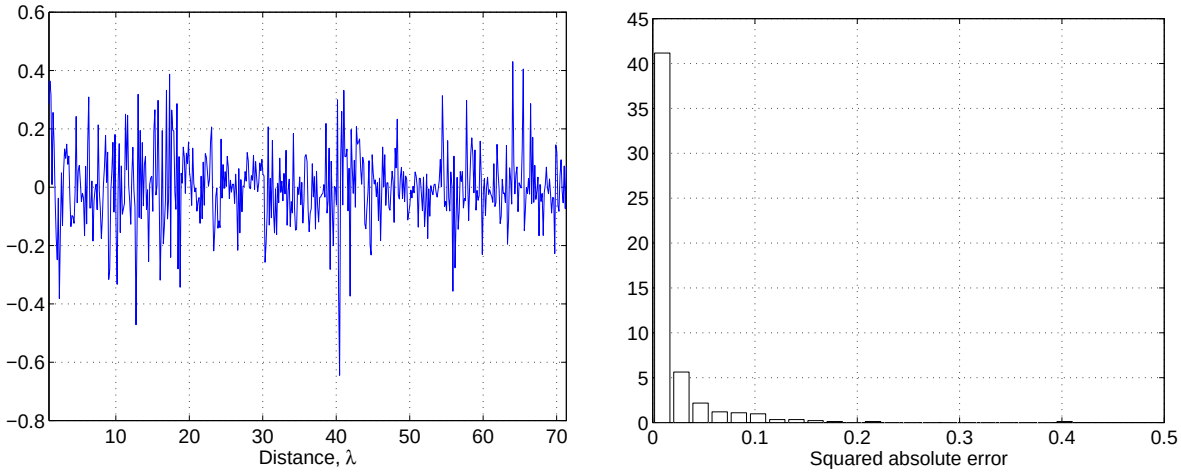
In all our further simulations we evaluate the Segmental PG only. The first value PG_0, corresponding to the initial hypermodel adaptation, is excluded from the computation of the Segmental PG.

We also develop this scheme a bit further by taking into the account the specifics of the resulting signals, as explained below.

Measuring the prediction error

The computation of the Prediction Gain requires estimation of the prediction error power. Although a relatively simple operation, it might result in the inadequate representation of the prediction quality due to the possible presence of outliers -- – instantaneous error bursts with high amplitudes. These outliers are mostly the result of transients and tracking errors. The outliers significantly affect the resulting value of the error signal power. In Fig. 6.1 we show a sample prediction error along with the corresponding histogram of prediction errors.

As we see the outliers constitute themselves as the long tails of the prediction error histogram (Fig. 6.1(b)). The probability of finding an outlier is quite low, but the effect on the computed variance is significant. In statistics it is common to amend this sensitivity to outliers by means of so-called robust statistics.



(a) Prediction error.

(b) Prediction error histogram.

Figure 6.1: A sample measured prediction error for a one-step-ahead ALP hypermodel.

It is known [Hub81] that the median is a robust estimator of the sample mean. The median is much less sensitive to outliers than the standard mean value. The following example illustrates the distinction between the median and the mean as a representative description of the random samples:

Example

Consider the following set of numbers: $\mathcal{A} = \{1, 2, 3\}$. It is required to compute the mean and the median of these samples. It is easily found that the mean equals $\mu_A = 2$, and the median is $m_A = 2$.

Now, let us assume that this set is extended with an outlier: $\mathcal{B} = \{1, 2, 3, 100\}$. The new values of the mean and median are $\mu_B = 26.5$, and $m_B = 2.5$, respectively.

We can see that the median is much less affected by the presence of the outlier in the data set, as compared to the mean.

In our experiments we thus compute the median of the squared absolute error instead of the mean. The corresponding Prediction Gain is then computed as in (6.1), but P_{sig} and P_{err} are evaluated as

$$P_{sig} = \text{median}\{|s[q]|^2\}, \text{ and } P_{err} = \text{median}\{|e[q]|^2\}. \tag{6.2}$$

This will result in the value of the noise power variance being less affected by the transients and temporary tracking errors. Thus, in computing the Segmental PG we will use the median power instead of the mean power.

Naive predictor

It also illustrative to compare the prediction properties of the trained hypermodels with the "simplest" predictor – the *Naive Predictor* . The Naive Predictor assumes that the future signal samples at the moment $q + \mathcal{L}$ are equal to the samples at the moment q. In other words, the Naive Predictor can be seen as a predictor that for any prediction interval $\mathcal{L} > 0$ returns the value equal to the current value at the moment q.

Such a predictor is termed "naive" because repeating the last seen value, especially for time-varying signals and long prediction horizons, and hoping that the signal does not change, is overly simplistic. However, in the absence of any other hypermodels, this might be the only strategy.

6.3 SAGE-based multipath prediction

Here we present the results of applying the prediction algorithm to the channel data estimated with the SAGE algorithm discussed in Chapter 3. In all experiments we use the FTW channel data set (Appendix C) to demonstrate the tracking and prediction results.

We consider different examples of multipath prediction with different simulation parameters. First, in Example 1 we consider a single track prediction over the distance of 28λ. Based on this example we also discuss the properties of the used prediction hypermodels. In Example 2 we extend the tracking distance to 71λ. Finally, in Example 3 we consider simultaneous tracking and prediction of several components.

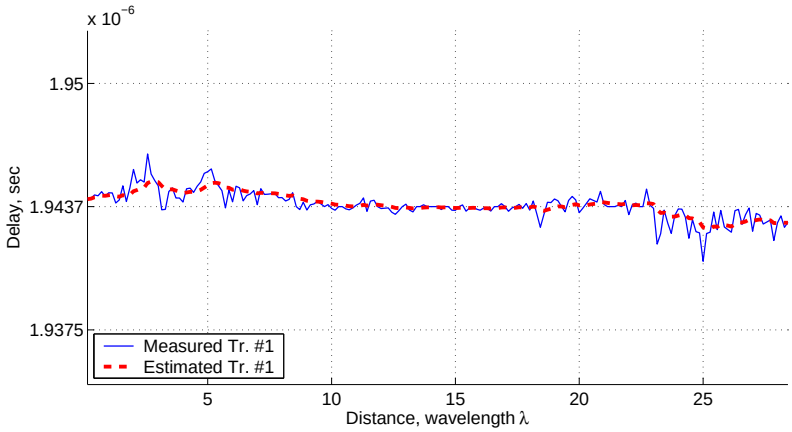
6.3.1 Tracking example 1: SIMO channel with a single track

In this example we consider a simple case of tracking a single multipath component, i.e., $K = 1$, over the distance of 28λ (or equivalently, ≈ 4.2m). The SAGE algorithm is set up to extract $L = 9$ components from the measured data. The strongest component from the estimated set is then used to initialize the corresponding structure hypermodel $\mathcal{S}_k$.

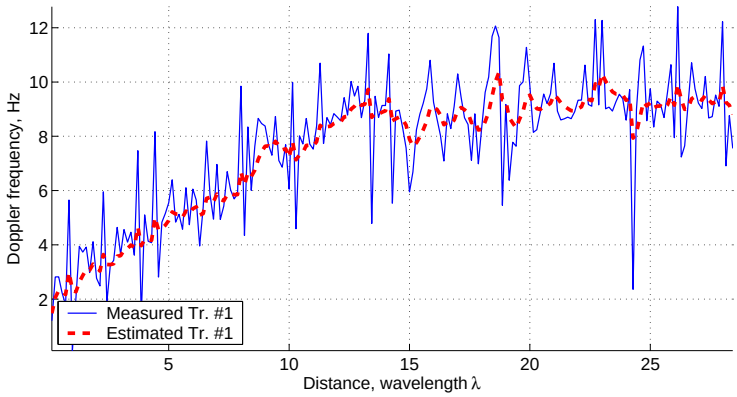
Let us first consider tracking results. In Fig. 6.2 we plot the evolution of the multipath parameters, i.e., delay, Doppler and DoA, as a function of the walked distance expressed in multiples of the carrier wavelength λ.

The first plot in Fig. 6.2(a) demonstrates the evolution of the corresponding multipath delay. The grid lines in Fig. 6.2(a) are drawn so as to coincide with the multiples of the sampling period. It can be seen that the delay trajectory resides mainly in the vicinity of a single sampling instance, however, it does deviate. These deviations are the results of the SAGE algorithm estimating parameters with a resolution better than the channel sampling period.

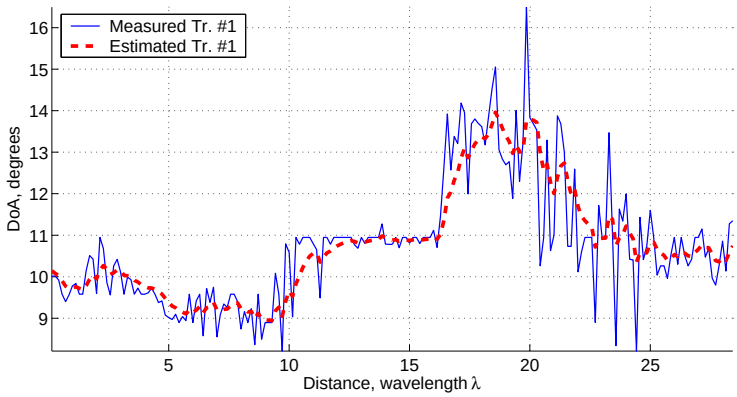
Next, in Fig. 6.2(b) we can see the evolution of the Doppler frequency trajectory. It is interesting to note how the Doppler frequency changes with time. The portion



(a) Multipath delay.



(b) Doppler frequency.



(c) Direction-of-Arrival.

Figure 6.2: (Example 1) Reconstructed trajectories of the track structure parameters.

of the data we use corresponds to the beginning of the measurement, i.e., when the transmitter starts moving. We see that initially we have very low Doppler frequency around 2Hz that increases to $\approx$ 10Hz, which corresponds to the movement with the velocity of $\approx$ 1m/s. This is the velocity with which the mobile transmitter was moved during the measurement campaign.

The evolution of the estimated DoA trajectory, shown in Fig. 6.2(c), is also quite interesting. It is relatively stable in the area up till 17λ. However, towards the end of the tracking interval the, DoA trajectory deviations grow. This is particularly visible after $\approx 20\lambda$.

A explanation for this behavior can be found once we consider the evolution of the corresponding complex gain and power, shown in Fig. 6.3. We see that in the

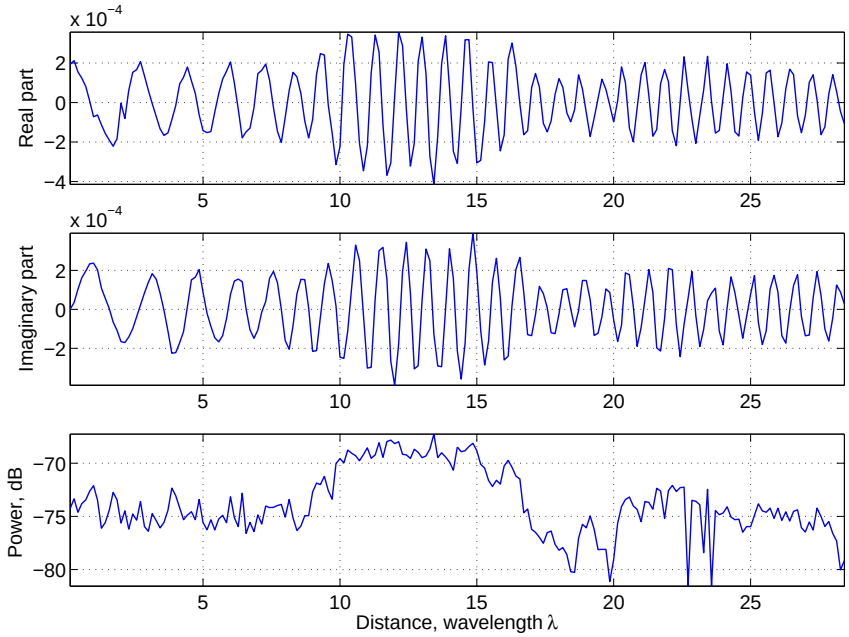


Figure 6.3: (Example 1) Evolution of the real and imaginary parts of the gain and of the power of the estimated track.

vicinity of the 20λ the power of the multipath component has dropped. This might happen naturally, due to the multipath component slowly getting out of sight of the antenna array. Alternatively, this might also result from errors in the tracking algorithm that picks up a wrong component continuation. In any case, the power of the track has fallen and thus more noise is affecting the parameter trajectories. Clearly, as the tracking algorithm continues the trajectory further, the hypermodels have to re-adapt to the new conditions. This in turn results in the temporary degradation of the prediction quality, since the hypermodels require time to adapt themselves to the new conditions and re-learn the new model coefficients. In case of tracking errors, the transients have a profound effect on both $\mathcal{S}_k$ and $\mathcal{A}_k$, and as a result on the prediction performance. But in case of gradual parameter change, the learning algorithm should be able to effectively cope with it.

This observation allows us to conclude that, in general, the agile hypermodels, i.e., those that are able to adapt faster, eventually minimize the influence of the transients and tracking errors on the prediction and tracking performance.

We see that in this experiment we have a strong and stable trajectory that can be used to train long-term gain predictors. In Fig. 6.4 we plot the spectrogram of the complex gain trajectory. We see that the complex gain is a time varying

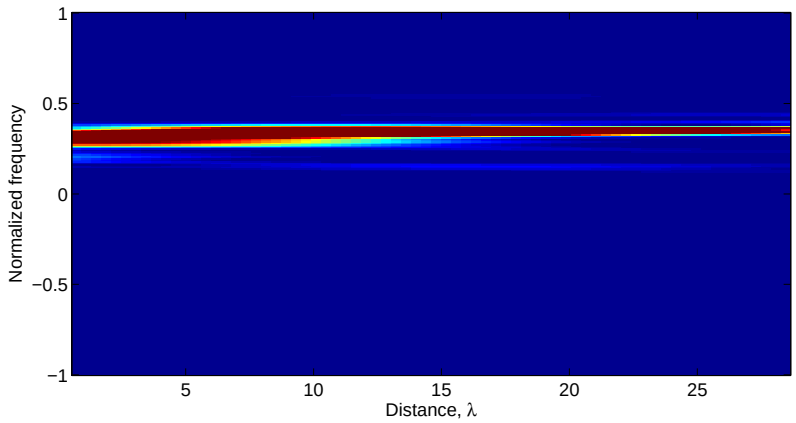


Figure 6.4: (Example 1) Spectrogram of the complex gain variation of the estimated track.

narrowband complex process, which exhibits a chirp-like behavior. Such signals can be viewed as complex exponential with a time-varying frequency. In theory, a complex exponential can be modeled with a simple first-order complex AR model. We, however, will need to continuously re-estimate the parameters of the AR models to cope with the signal nonstationarity, or use nonlinear models that try to capture the signal's chipred behavior.

In a sequel we will demonstrate the prediction performance of the gain hypermodels, considered in Section 5.3, using this trajectory data.

Adaptive Linear Predictor (ALP)

We begin with the application of the ALP hypermodel to the long-term forecast of the complex multipath gain. Keeping in mind that we want the predictor to adapt fast, we try to keep the predictor order as small as possible.

First, in Fig. 6.5 we illustrate a sample of a one-step-ahead prediction, i.e., $\mathcal{L} = 1$, with the predictor order $Q = 3$. For the FTW data set this corresponds to the spacial prediction horizon of $\lambda/7$, or, equivalently 20msec into the future. The initial portion of the predicted signal illustrates well the convergence properties of the hypermodel. It can be seen that after $\approx 4\lambda$ the hypermodel coefficients converge and the predictions start to follow closely the true gain variations. The Naive Predictor, on the other hand, is much less effective in this case. As expected this predictor simply compies the samples of the gain signal by $\mathcal{L} = 1$ samples into the

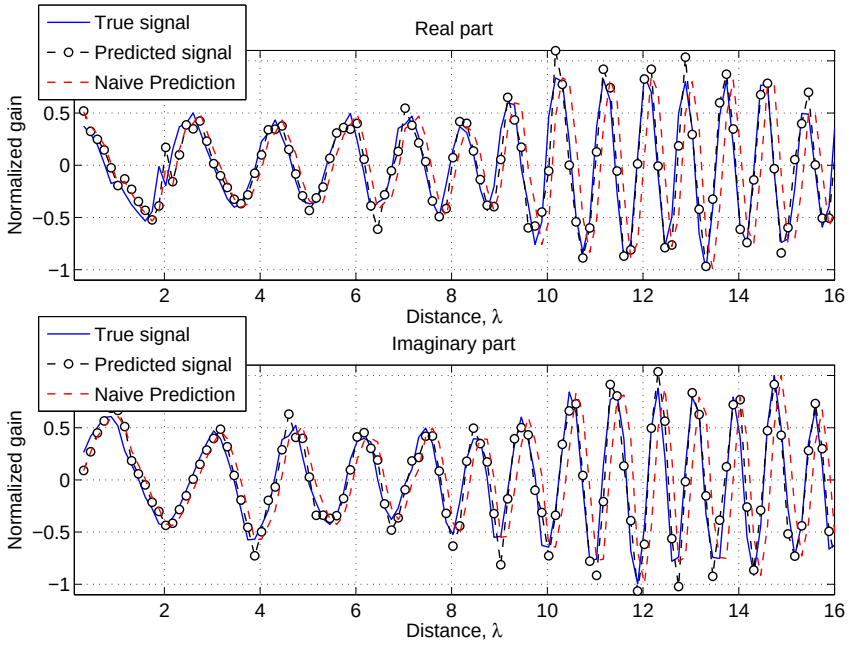


Figure 6.5: (Example 1) Complex gain prediction using the ALP hypermodel. $\mathcal{L} = 1$, $Q = 3$.

future. Such prediction is clearly inferior to the ALP-based prediction performance for this prediction horizon.

The next plot in Fig. 6.6 shows the evaluated PG as a function of the prediction horizon $\mathcal{L}$ for different model orders Q. The result in Fig. 6.6 are averaged over 100

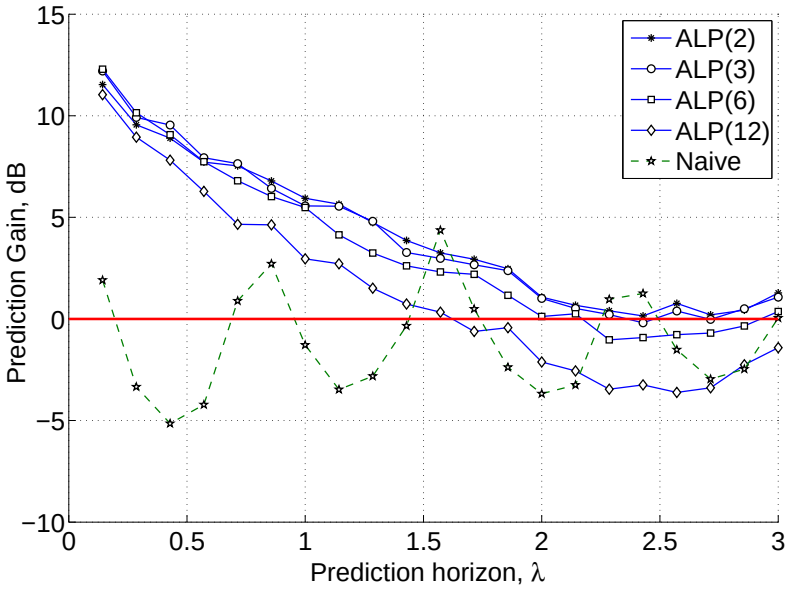


Figure 6.6: (Example 1) Prediction gain for the ALP hypermodel with different model orders.

random hypermodel coefficients initializations.

The first observation we make is that the prediction performance gets worse as the prediction horizon increases. This is quite logical since it is impossible to predict a non-deterministic process infinitely far into the future. What is also interesting is that by increasing the model order Q we do not gain any significant increase of PG. This might be the result of overfitting: the hypermodel with more parameters does not generalize well, especially for higher prediction horizons. However too few parameters lead, as we can see, to undermodeling, which also results in lower PG. In our simulations we empirically selected an optimum model order $Q = 3$ that achieves high PG with few coefficients.

We also observe that the Naive Predictor is not a monotonic function of the prediction horizon. Since the Naive Predictor simply repeats the last seen value, it thus exhibits oscillatory variations of the prediction quality like the gain signal itself. When the prediction interval $\mathcal{L}$ coincides with a multiple of the "signal period", we obtain higher PG values. Similarly, around half of this "signal period", the Naive Predictor results in a very poor prediction performance.

Iterated Adaptive Linear Predictor (IALP)

The next hypermodel we are going to discuss is the IALP. This is again a linear predictor that, unlike ALP, exploits the Kalman Filter framework to learn and adapt the hypermodel coefficients.

Similarly to the previous case we first show the initial portion of the predicted signal. In Fig. 6.7 we show the corresponding prediction results for a prediction interval $\mathcal{L} = 1$ and model order $Q = 3$. Interestingly, the IALP hypermodel adapts faster to the data. If we compare these prediction results to the similar ALP prediction experiment, shown in Fig. 6.5, we notice that the latter adapts after only $\approx 2\lambda$, i.e., the half of the ALP learning time. This is of course an advantage since we definitely prefer agile predictors. Thus, in general, we can expect a better PG performance with this predictor. The following plot in Fig. 6.8 proves this assertion. Again, the PG results where averaged over 100 independent model initializations. Although we cannot say that the PG increase is very high, but for short prediction horizons we win almost 1dB. Surprisingly, increasing the order Q of the predictor does not have a profound effect on the PG performance. Thus we can conclude that overfitting is less of a problem here.

It must, however, be stressed that the recursive nature of the IALP hypermodel brings along numerical stability issues. The "iterative" application of the state transition equation in (5.13) might lead to unstable predictor for long prediction intervals $\mathcal{L}$. In our implementation of the IALP, we use a simple-minded approach to avoid such instabilities by simply checking the range of the predicted gain samples. When they become larger than a certain predefined threshold, we re-adapt the hypermodel by generating initial hypermodel parameter values with slightly higher variance σ. This usually allows to find a solution that eventually leads to a stable trajectory (within the empirically defined range). The stability of the resulting

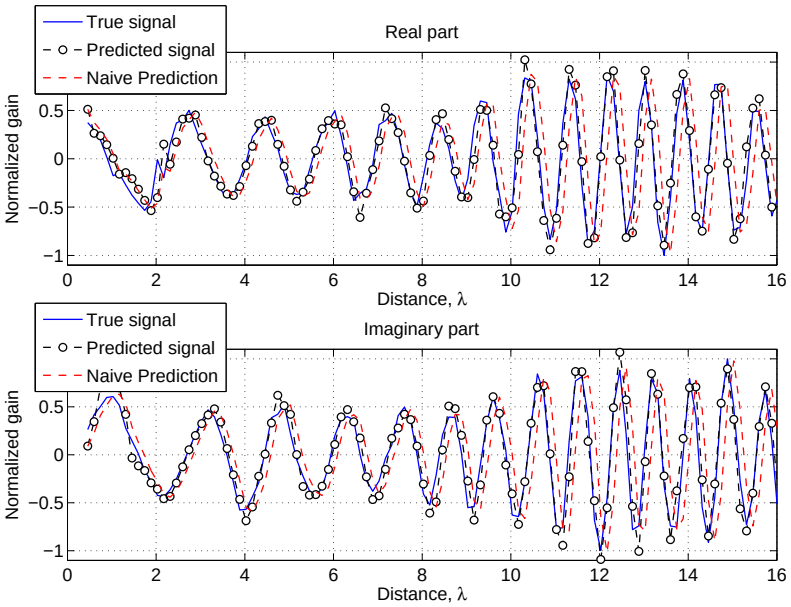


Figure 6.7: (Example 1) Complex gain prediction using the IALP hypermodel. $\mathcal{L} = 1$, $Q = 3$.

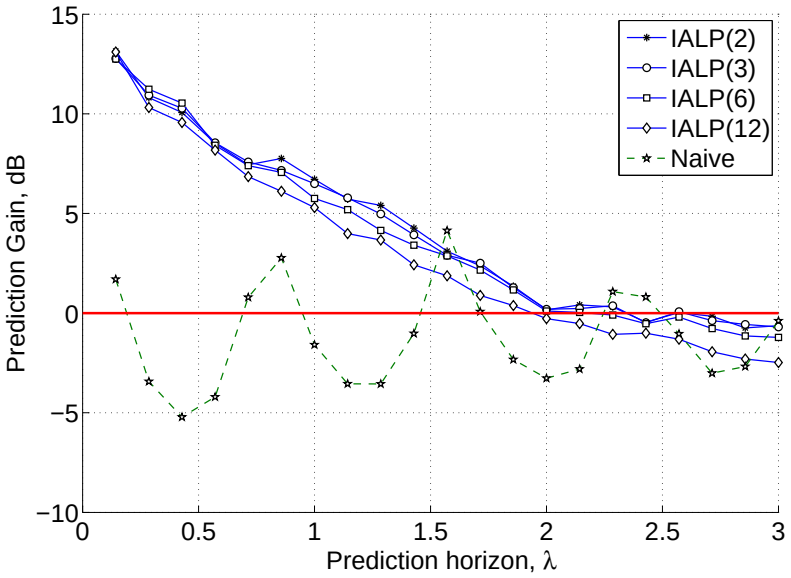


Figure 6.8: (Example 1) Prediction gain for the IALP hypermodel with different model orders.

predictor remains however a weak point of the used IALP algorithm.

Nonlinear Volterra-based predictor (AVNP)

Since the complex gain waveform is a chirp-like signal, a nonlinear model might be more appropriate. In theory, linear modeling of polynomial phase (chirped) signals

is suboptimal and a nonlinear structure is required. Here we apply several nonlinear models that can be used to approximate this nonlinearity. Again, we would like to have the simplest models to ensure that the resulting computational complexity is not high.

Volterra models, used in AVNP's, are good candidates for such approximation. By selecting the order of nonlinearity and memory length for each order we can approximate a large class of nonlinear systems. In the following we will try several different nonlinearity structures to find the best fitting one.

Keeping in mind that the ALP performed quite well with only three coefficients, we assume the length of the linear part of AVNP to be equal to 3. We also do not increase the order of nonlinearity beyond the cubic one. In our experiments we found that higher nonlinearity orders do not bring any considerable increase of the PG performance.

As in the previous cases, let us first consider a sample prediction result for $\mathcal{L} = 1$, shown in Fig. 6.9.

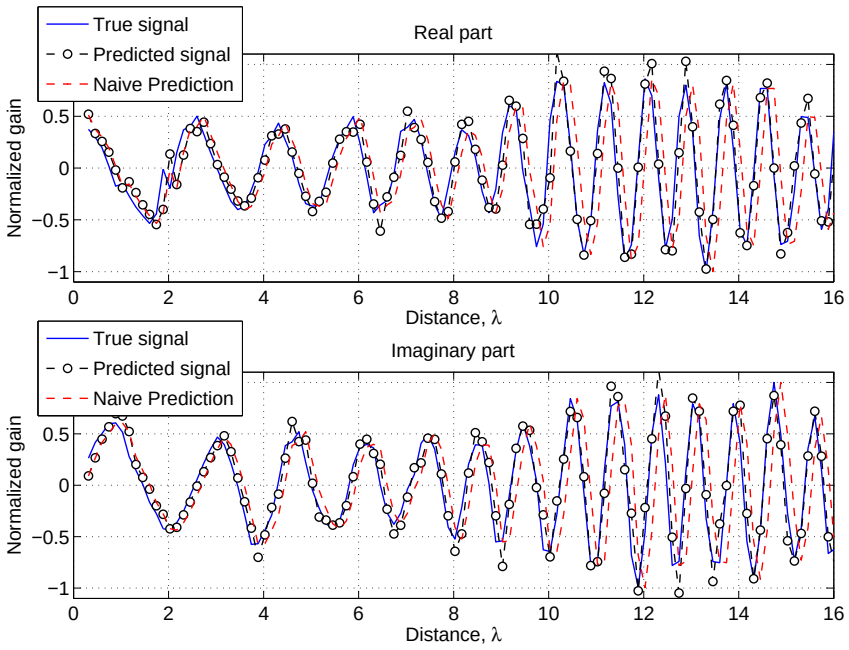


Figure 6.9: (Example 1) Complex gain prediction using the AVNP1 (Table 6.2) hypermodel. $\mathcal{L} = 1$.

In this experiment we use a simple quadratic Volterra model with a single coefficient for the nonlinear part. Note that similarly to the ALP hypermodel, this AVNP hypermodel requires roughly the same learning time to make useful predictions, i.e., approximately 3λ to 4λ. Similar learning times can be observed when we consider higher nonlinearity orders.

The configuration of the Volterra model used to generate these prediction results is not necessarily an optimal one. Although it is possible to objectively find the best structure, i.e., nonlinearity order and memory size, that would minimize the

prediction error, this is a quite challenging task. In the present work we choose the best model empirically by trial and error. Although suboptimal, this strategy, at least, allows to determine if Volterra models bring any advantage at all, as compared to the linear prediction methods.

A set of model structures we experiment with is summarized in the Table 6.2.

	AVNP1	AVNP2	AVNP3	AVNP4
Linear part	3	3	3	6
Quadratic part	1	3	3	3
Cubic part	0	1	3	3

Table 6.2: Memory lengths for the nonlinear terms of the AVNP hypermodel.

Using these configurations we can evaluate the PG experimentally. The corresponding prediction results, averaged over 100 random model initializations, are shown in Fig. 6.10.

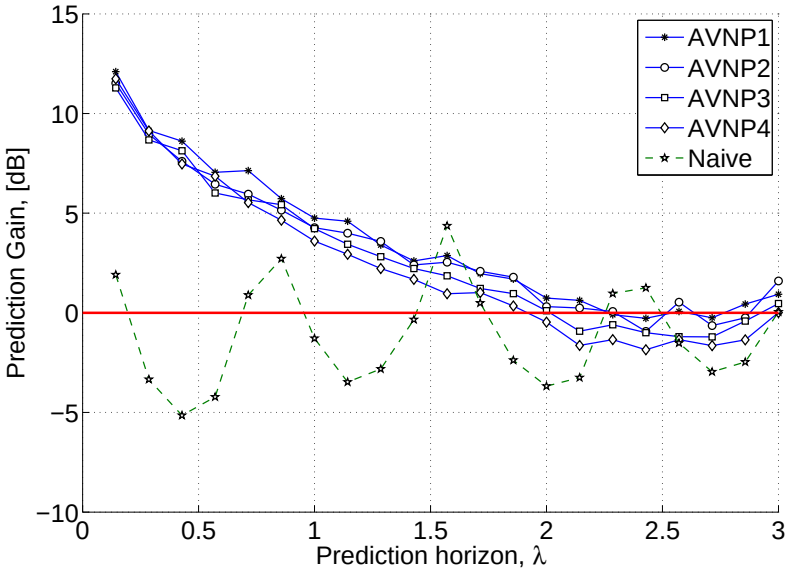


Figure 6.10: (Example 1) Prediction gain for the AVNP hypermodel with different model structures.

What we immediately see is that the resulting PG performance is inferior to that of the linear models. Especially for short prediction horizons, where the best performance is expected, the usage of nonlinearity does not bring any advantage and the achieved PG is even lower than that of the ALP predictor. For longer horizons the performance is not that different from the ALP predictor.

Based on that we can assume that Volterra models might not be an appropriate choice for modeling the dynamics of chirp-like signals we deal with. However, it is possible that other nonlinear models are more efficient in doing this.

Iterated nonlinear predictor based on Neural Networks (IANNP)

Another type of nonlinear predictor we consider is the Iterated Adaptive Neural Network Predictor (IANNP). Unlike AVNP, which is a nonlinear extension the ALP hypermodel, this predictor is the nonlinear extension of the IALP. Similarly to IALP this hypermodel exploits the Kalman filter framework for hypermodel parameter estimation and adaptation. It also makes predictions by recursive application of the state transition equation, which in the case of IANNP is a nonlinear function represented by a neural network (NN). We will consider several possible neural networks that we found to give interesting results. The used NN differ only in the number of neurons in the input and hidden layers.

We begin by illustrating a one-step-ahead prediction results for a sample network structure. For that we use a relatively small NN with 2 inputs and 3 neurons in the hidden layer, which gives a total of 9 network coefficients that are to be estimated. The corresponding results are shown in Fig. 6.11. It can be easily seen that this

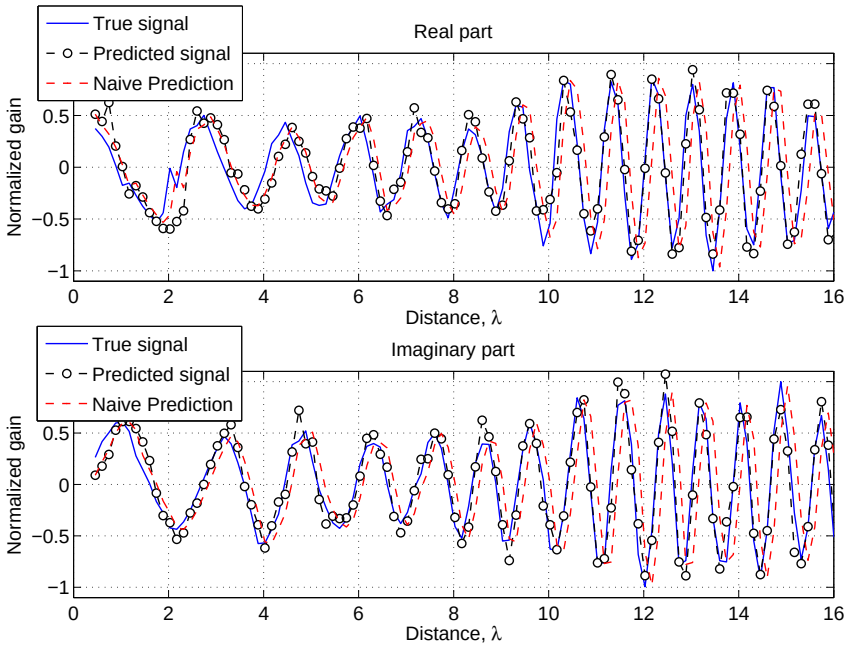


Figure 6.11: Complex gain prediction using the IANNP1 (Table 6.3) hypermodel. $\mathcal{L} = 1$.

predictor needs more than the others to adapt its coefficients. Unlike the IALP, the convergence time in this case is almost 4λ to 6λ, depending on the initialization. Such a long adaptation time eventually results in longer transients and lower PG. Keep in mind that the number of coefficients is relatively small, thus for larger networks we might expect even longer adaptation times.

As we said, the NN's we use in this experiment differ in the number of neurons in the input and hidden layers. The configurations we used to evaluate the PG are summarized in Table 6.3.

	IANNP1	IANNP2	IANNP3	IANNP4
Input Layer	2	2	7	7
Hidden Layer	3	7	2	7

Table 6.3: Number of neurons in the neural network used in the IANNP hypermodel.

The corresponding prediction gain computed using these hypermodels is shown in Fig. 6.12. We can see that although the PG performance for short prediction

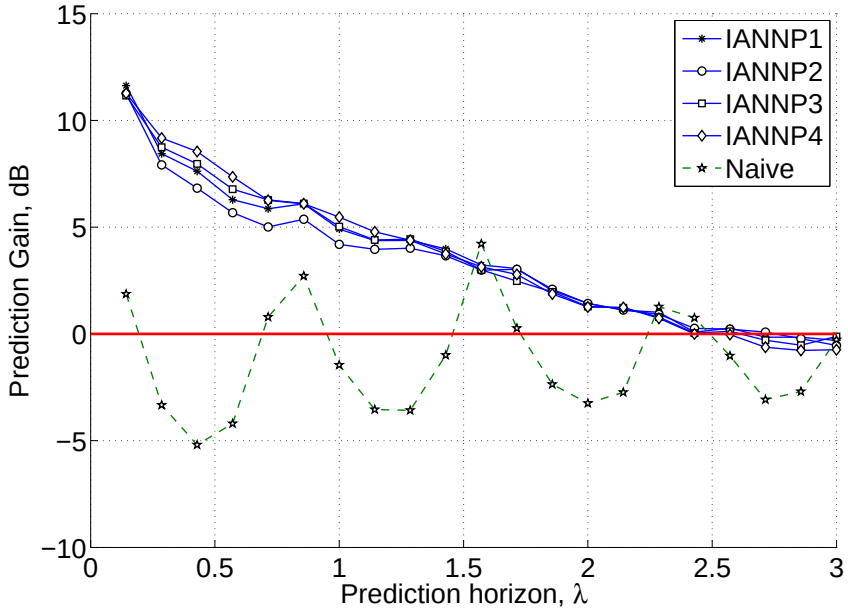


Figure 6.12: (Example 1) Prediction gain for the IANNP hypermodel with different network structures.

intervals is lower than for the corresponding linear hypermodels, it does not, however, degrade as fast. If we compare the IANNP with its linear counterpart, i.e., the IALP hypermodel, we find that, for shorter prediction intervals, the linear models still outperform IANNP. But when the prediction interval grows, the IANNP models deliver a better performance: it decays not as fast as the linear predictor, yielding a positive prediction gain as long as $\approx 2.5\lambda$.

We should also mention that unlike the IALP hypermodel, there are no stability issues with iterative prediction for long prediction intervals $\mathcal{L}$. Since the hidden layer uses sigmoidal activation functions, its output values are always bounded by ± 1 and thus the output of the hypermodel stays bounded even if it the transition equation is used recursively.

6.3.2 Tracking example 2: Extending tracking time

In this example we use the same simulation parameters as in the Example 1, but track the multipath component over a longer time interval. As we have seen,

the prediction results obtained in the previous examples were evaluated over the tracking distance of 28λ. Here we extend the observation and tracking time to 71λ, which corresponds to the tracking distance of $\approx$ 10m.

It is easier to analyze these plots together with the evolution of track power, shown in Fig. 6.14. We can see that up till $\approx 30\lambda$ the track evolution is quite stable, with slight variations of the signal power envelope. Again we observe a time dependent frequency variation of the track gain (see signal spectrogram in Fig. 6.15). The track trajectory is, however, interrupted at 30λ, which can be seen as a drop of the track power. A similar break occurs around $\approx 39\lambda$. This is most likely caused by errors in the tracking algorithm, which in turn might be caused by errors in parameter estimation. It is known that the SAGE algorithm occasionally introduces estimation artifacts, namely in the cases when several components are used to approximate a single true one. These artifacts might severely "confuse" the tracker. This "confusion" can be seen as noisy bursts in the parameter trajectories, since the artifacts have slightly different (but still close) parameter values. It is possible that, as the tracking proceeds, the artifacts become sufficiently different from the hypermodel predictions and the tracking algorithm might eventually pick up the correct component, as we see in our experiment around 30λ and 39λ. This however leads to parameter jumps, and as the result to possible transients.

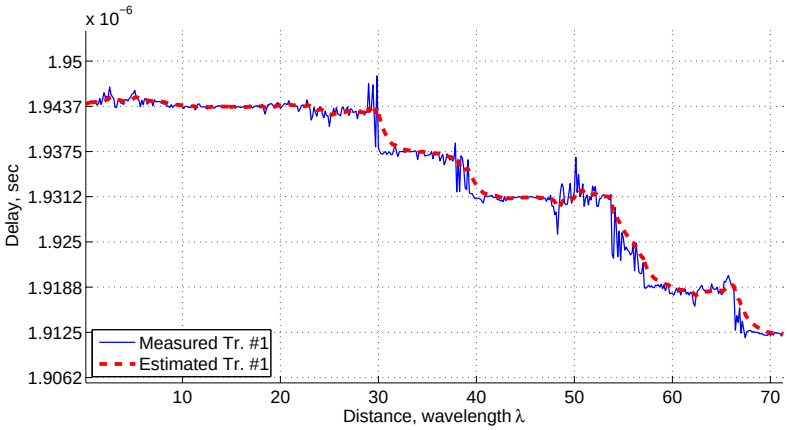
Further we notice that after $\approx 48\lambda$ the power of the track significantly decreases. This might be an absolutely natural process – the component simply becomes too weak due to the change of the propagation environment. We cannot, however, exclude the case of tracking errors. They might also lead to the track converging to improper continuations.

Still, we can identify the hypermodels that are able to adapt to these changes. Again, the signal we are going to predict is the time-varying complex gain. In this experiment we select several hypermodels to evaluate the PG performance. In particular, we use linear hypermodels of order 3, i.e., ALP(3), IALP(3), and nonlinear hypermodels AVNP3 and IANNP3. The corresponding results, averaged over 100 independent model initializations, are shown in Fig 6.16.

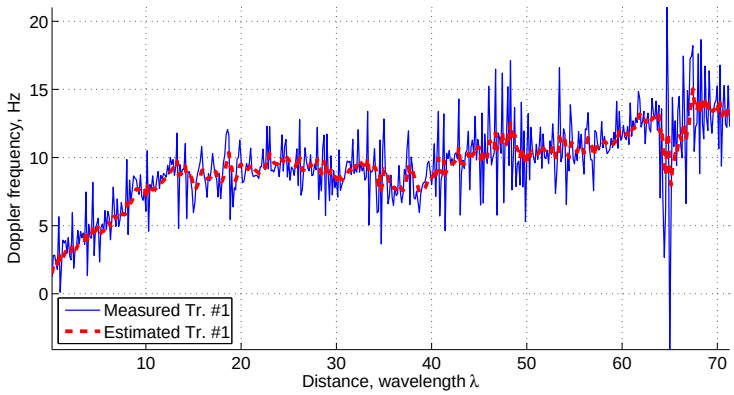
As we can see from the Fig. 6.16, there is a slight degradation of the prediction gain for the short prediction horizons, as compared to the previous example. This decrease comes from having more transients due to the hypermodel changes when compared to the shorter data record of Example 1. Due to the increased observation time, the obtained curves are now less noisy and the dependency of the PG on the prediction horizon can be seen more clearly.

It is also interesting to note that the hypermodel based on the neural network slightly outperforms other predictors for long prediction intervals (beyond $\approx 1.3\lambda$). We also see that IANNP hypermodels are better suited for prediction as compared to the AVNP. Thus, they capture the nonlinear dynamics of the observed signal more effectively. Nonetheless, for short prediction intervals the linear models are still better than the nonlinear models.

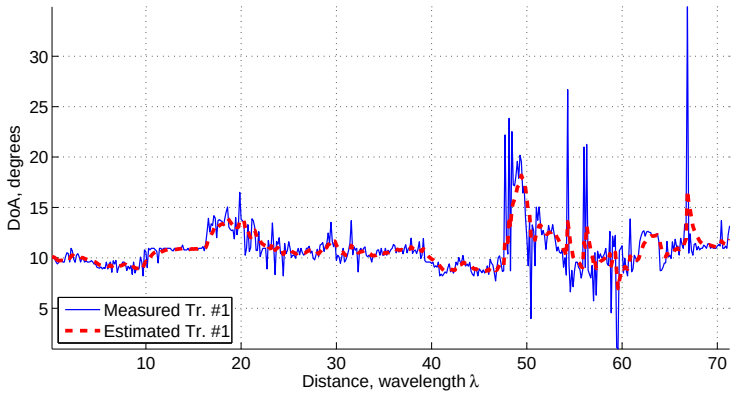
The PG performance obtained for this example demonstrates that we achieve positive PG up to a distance of $\approx 2.5\lambda$ (approx. 0.37m). Clearly, it is the actual



(a) Multipath delay.



(b) Doppler frequency.



(c) Direction-of-Arrival.

Figure 6.13: (Example 2) Reconstructed trajectories of the track structure parameters.

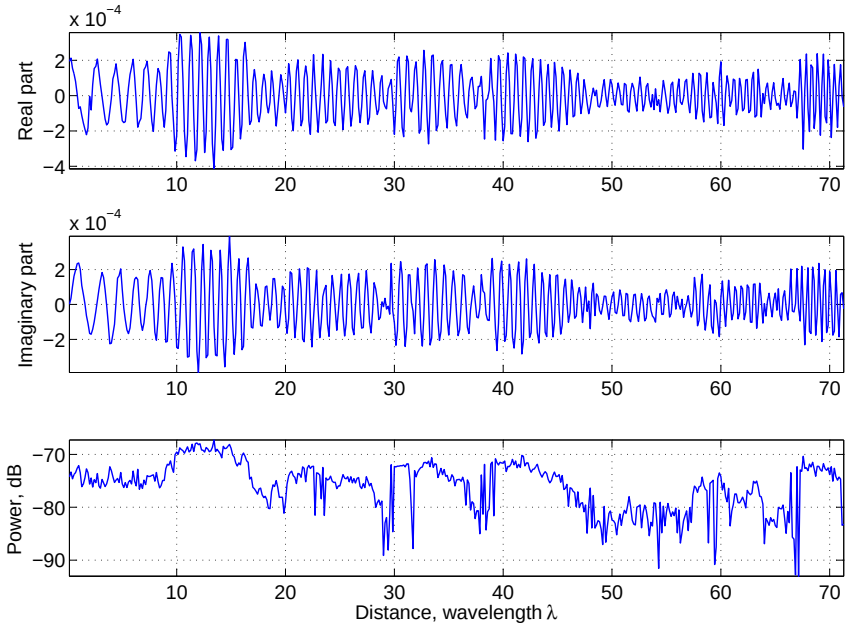


Figure 6.14: Example 2: Evolution of the real and imaginary parts of the gain and of the power of the estimated track.

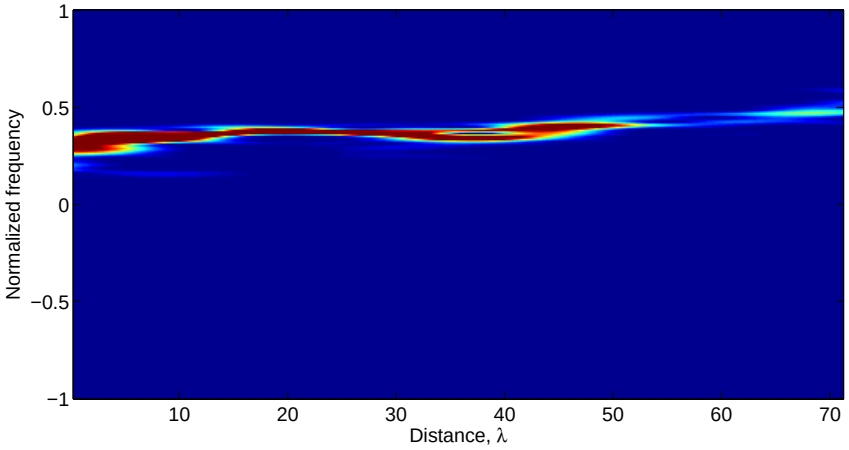


Figure 6.15: (Example 2) Spectrogram of the complex gain variation of the estimated track.

application that determines the minimum required PG, and thus the maximum possible prediction horizon.

6.3.3 Tracking example 3: Tracking multiple components

Here we will present some performance results for tracking several components simultaneously. We also consider tracking over a long distance, using the hyper-models we chose in Example 2. The SAGE algorithm is set up to estimate $L = 15$ components, the number of tracks is set to $K = 5$. Other experimental parameters

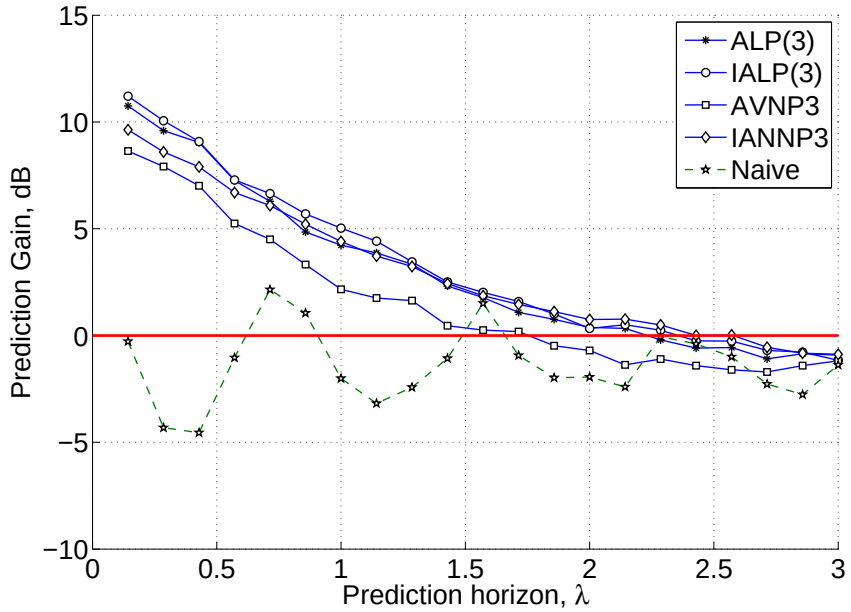


Figure 6.16: (Example 2) Prediction gain for a single track evaluated over the distance of 71λ (10m).

are left unchanged as in the previous examples.

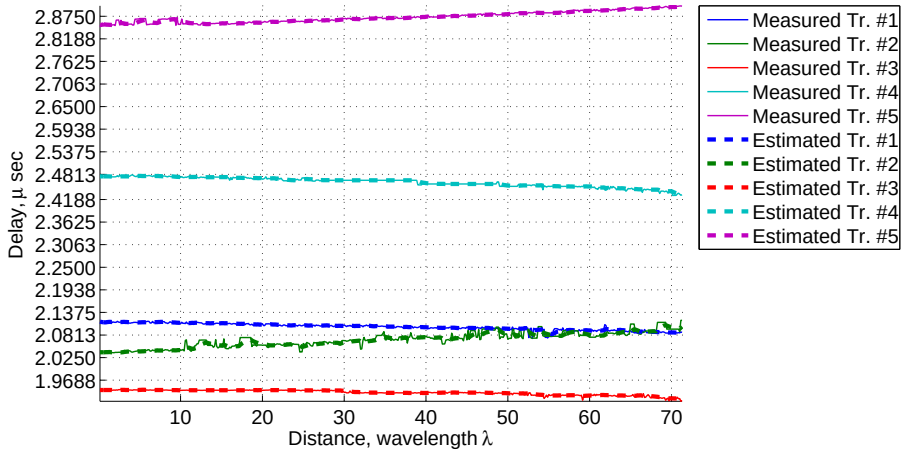
Unlike the previous examples, here we need to select $K > 1$ initial tracks, which is a "K out of L" selection problem. Despite its seeming simplicity, it is an important step of the algorithm. First of all, we want to track, and then predict, strong components. Second, the tracks should be initialized so as to minimize the possible influence of the estimation artifacts. For that we propose to use a simple multipath clustering.

Multipath clustering based on extracted multipath parameters has been addressed in a number of works [Shu04b, SG04, CBH+06, CCS+06]. These clusters are treated as geometrical objects that group components with close parameters into a single unit. Measuring the "closeness" between the multipath components can be effectively done using the MCD [SÖH+02] we discussed in our tracking algorithm. The initial components are then selected as the strongest component from each cluster.

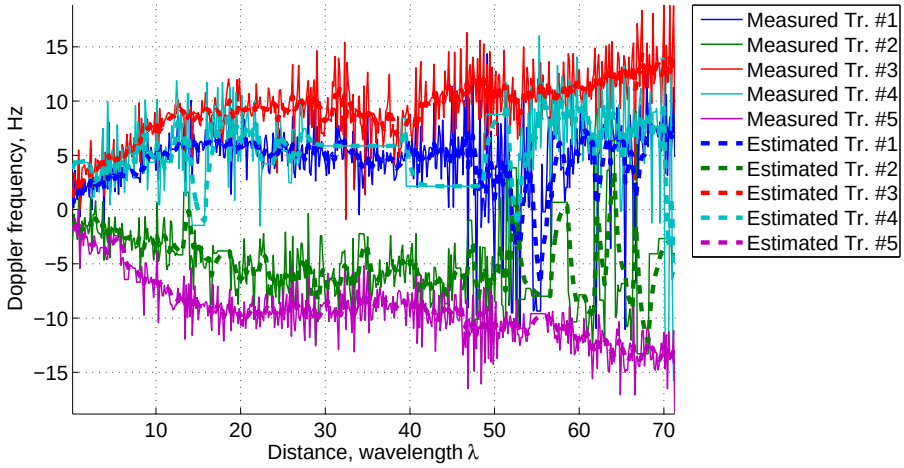
Let us here again start with plotting the multipath parameter trajectories. In Fig. 6.17 we plot the trajectories of the corresponding track delays (Fig. 6.17(a)), Doppler frequencies (Fig. 6.17(b)), and DoA's (Fig. 6.17(c)).

What we readily see is that cluster-based initialization performs quite well– we see that the tracked components are separated and do not interfere. Based on these trajectories we can also "guess" a geometrical distribution of the wavesources in the propagation environment.

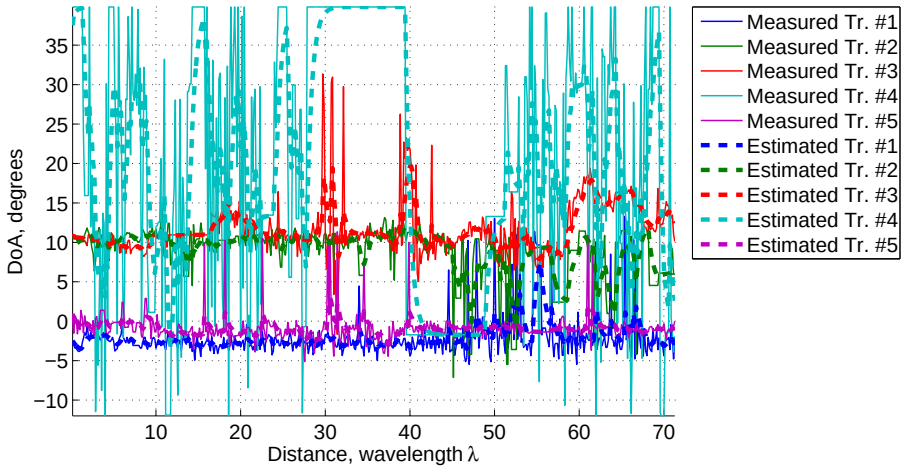
Track 3 is the closest one, i.e., it has the smallest propagation delay. This component, along with the Tracks 1 and 4, has positive Doppler frequency, thus we move towards it. Tracks 2 and 5, on the other hand, have negative Doppler frequencies



(a) Multipath delay.



(b) Doppler Frequency.



(c) Direction of Arrival.

Figure 6.17: (Example 3) Reconstructed multipath trajectories. $K = 5$.

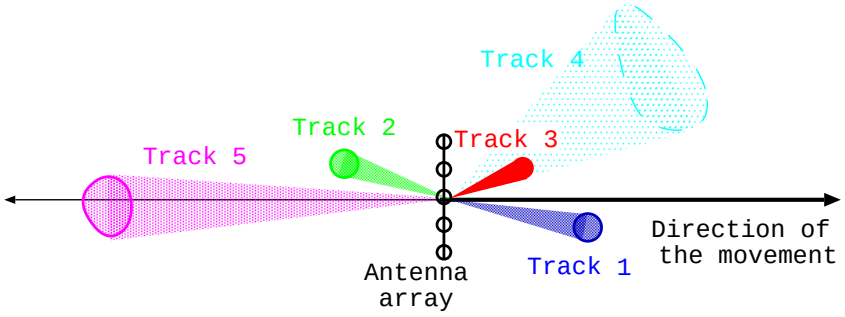


Figure 6.18: Virtual reconstructed geometry of wavesources distribution.

and thus we move away from them. Combining this information with the DoA and delay trajectories, we can construct an "approximate", or virtual, geometrical distribution of the wavesources, shown in Fig. 6.18. In Fig. 6.18 the distance from the antenna array is directly proportional to the corresponding track delay. Thus, the Track 3 is the closest to the array, while the Track 5 is the furthest.

Now, let us analyze the parameter dynamics of these tracks more closely. For that we again refer to the evolution of the track powers and complex gains, shown in Fig. 6.19, and 6.20, respectively.

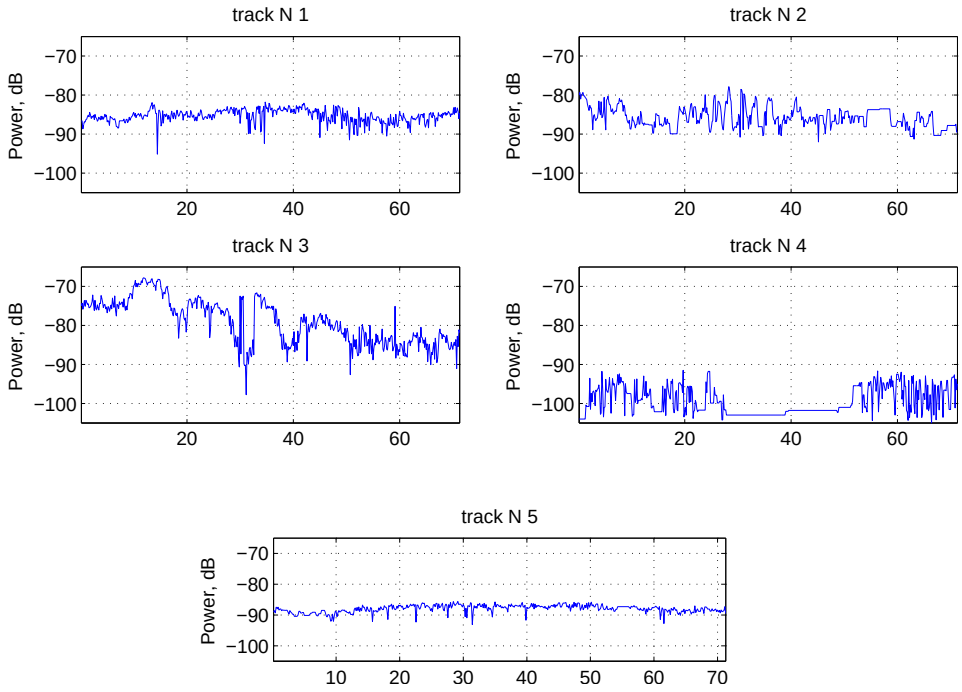


Figure 6.19: (Example 3) Evolution of the track powers.

First, we notice that the Track 4 is clearly very weak. In the interval 28λ to 52λ the tracker was not able to find any appropriate continuation, what can be

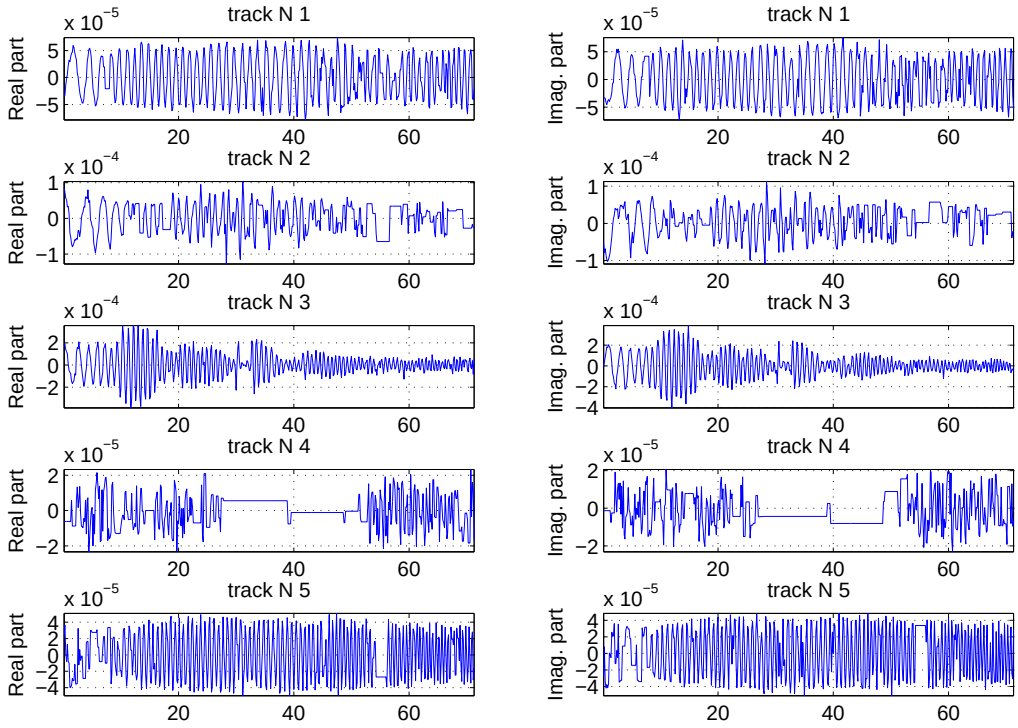


Figure 6.20: (Example 3) Evolution of the real and imaginary parts of the gain for the estimated tracks.

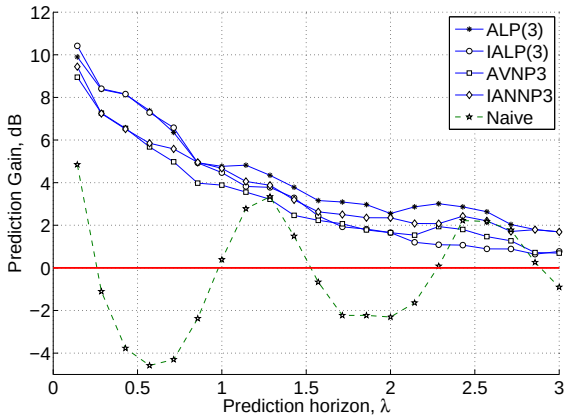
recognized from a horizontal line in the power evolution. We also see that the corresponding DoA and Doppler trajectories for this track are very noisy. Track 2 is also one that might not be useful for prediction. Although it is relatively strong, the corresponding gain signal is too incoherent, and thus the adapted hypermodel will have to re-adapt most of the time.

From Fig. 6.19 we can also identify that Track 3 is the strongest one. Its evolution is basically equivalent to the of the track from Example 2. Note, however, that these tracks are not *exactly* the same. Up to 30λ they behave identically, but after 30λ, in this example, we can observe a slightly different track continuation. This difference comes from the multiplicity of admissible solutions to the association problem– tracking several componets might result in a slightly different solution. Especially this might be the case when a tracking algorithm is "fooled" by estimation artifacts.

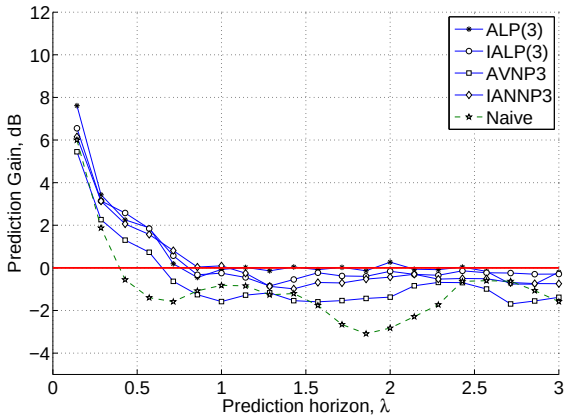
The remaining Tracks 1 and 5 are also potentially very promissing. They are weaker than Track 3, but their parameter evolution, despite occasional tracking errors, is quite stable. Track 5 preserves consistent structure over the length of almost 40λ.

Now, let us evaluate the prediction gain for these tracks. We will use the same hypermodels we used in Example 2, i.e., ALP(3), IALP(3), AVNP3, and IANNP3. The evaluated PG's for these tracks are summarized in Fig. 6.21.

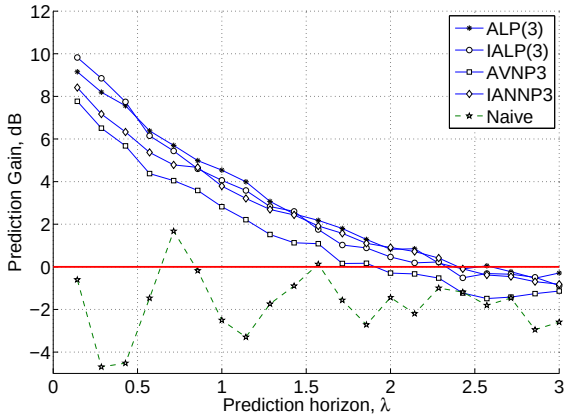
As expected, the best performance we obtain for Tracks 1, 3, and 5. Again the best



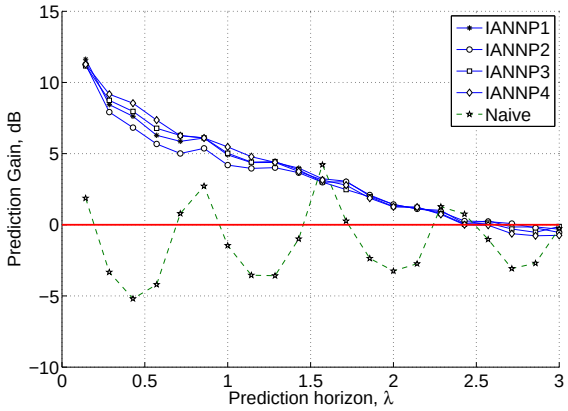
(a) Track N1



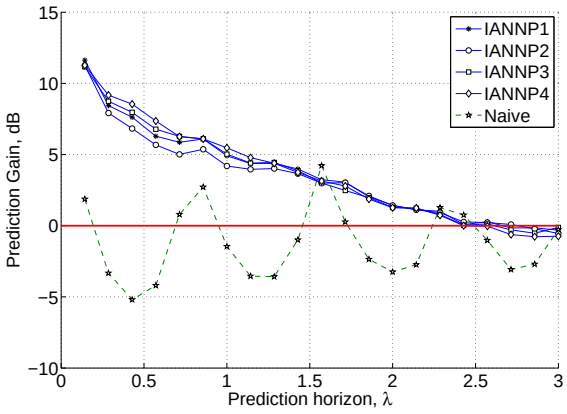
(b) Track N2



(c) Track N3



(d) Track N4



(e) Track N5

Figure 6.21: (Example 3)PG evaluated for $K = 5$ reconstructed tracks.

performance is achieved with linear predictors. They are simpler to adapt despite that they are not optimal in capturing the nonlinear behavior of the observed signal.

Volterra hypermodels are inferior to both linear as well as NN hypermodels. We might thus conclude that the polynomial structure of the Volterra model is inappropriate in track gain prediction. Neural networks, on the other hand, are much more promissing.

What is also interesting to observe, is that with Track 1 and 5 we achieve positive PG even beyond 3λ horizon. By comparing the achieved performance to that of Track 3, which is the strongest one, we can further conclude that it is not the power that is the major requirement for successful prediction, but rather the consistent signal structure with few to none tracking errors over a sufficiently long distance. This allows learning the coefficients of the hypermodels and still making reasonable predictions.

Examples of inconsistent signals are Tracks 2 and 4. The best predictor for these tracks is just the last seen value, what is actually implemented by the Naive Predictor. Clearly, it does not pay off to invest any resources for tracking these components.

We know that the estimation algorithm estimates more components than the number of tracks we reconstruct. This creates a potential for introducing track management schemes that can further improve prediction performance by excluding the useless tracks from the analysis and incorporating better components into the tracker.

6.4 Evidence Procedure-based multipath extraction and prediction

In this section we consider the application of the Evidence Procedure to the multipath extraction with the consecutive prediction of the resulting multipath components. The EP procedure considered here estimates channel parameters by using the SAGE-RVM approach discussed in the Section 4.5 of Chapter 4.

In order to be fair in the comparison, we apply this estimation algorithm to the same data that we used for the SAGE-based channel prediction. We also utilize the same tracker and predictor structures and parameters for the same reason.

In all the related SAGE-RVM related examples we consider the observation distance spanning 71λ and initialize the estimation algorithm with $L = 20$ components.

6.4.1 Tracking example 4: Single component tracking

Here we consider a single track example, whose parameters where estimated using the SAGE-RVM algorithm.

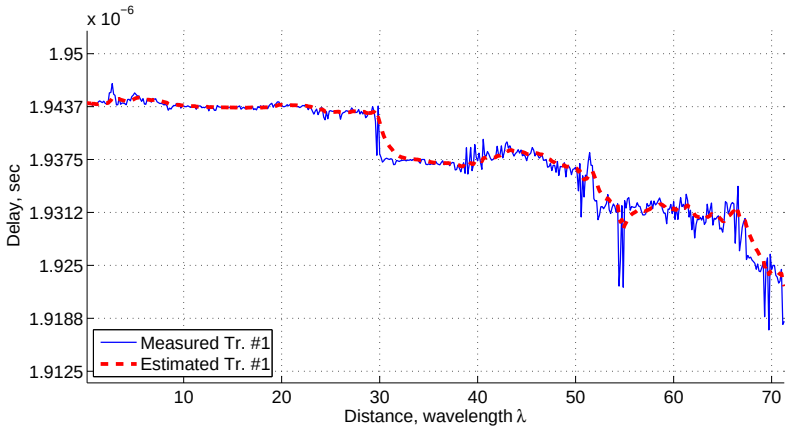
We begin by plotting the evolution of track structure parameters. The reconstructed trajectories are shown in Fig. 6.22(a) for delay, in Fig. 6.22(b) for Doppler frequency, and in Fig. 6.22(c) for DoA.

Clearly, the evolution of the track parameters is quite similar to that obtained in Example 2 with the SAGE estimates. But we also note some differences. We see that the delay trajectory is not the same after $\approx 40\lambda$, as compared to that in Fig. 6.13(a). When we consider the corresponding gain evolution, shown in Fig. 6.23, we notice that the track power has also fallen. Thus is is very likely that the tracker has picked up a different track. The evolution of the DoA trajectory is also different. Although we do still receive the component from roughly the same direction, the trajectory itself is much smoother. We see the outbreaks only where the tracking errors are likely to be, i.e., around 30λ, and 40λ, when the multipath delay "jumps" between the sampling instant.

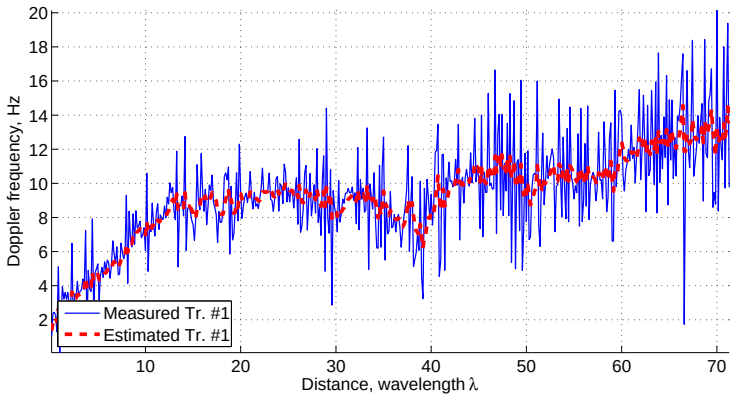
Note that in the case of the Evidence Procedure an estimate of the track power is readily available as the inverse of the corresponding evidence parameters α.

Similarly to Example 2, we plot the spectrogram of the corresponding gain signal in Fig. 6.24. Again, we see that the track gain is a slowly time-varying complex signal with narrow bandwidth. What is substantially different from Fig. 6.15 is that, after 40λ, the track is lost, as we have seen already from the other parameter evolutions. We will, however, not claim that the tracking based on SAGE estimates is better in general. According to the results in Fig. 6.14 and Fig. 6.15, we see that there we also have problems around 40λ, however, are able to track the component a bit longer up till 50λ. Note that this does not happen, as expected, for Track 3 in Example 3.

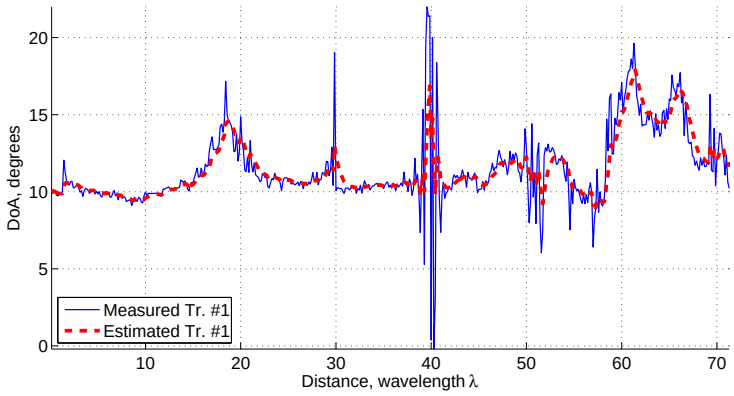
Now, let us consider the prediction results for this track. We apply four different prediction algorithms introduced in the previous chapter to the reconstructed



(a) Multipath delay.



(b) Doppler frequency.



(c) Direction-of-Arrival.

Figure 6.22: (Example 4) Reconstructed multipath trajectories. $K = 5$. (SAGE-RVM).

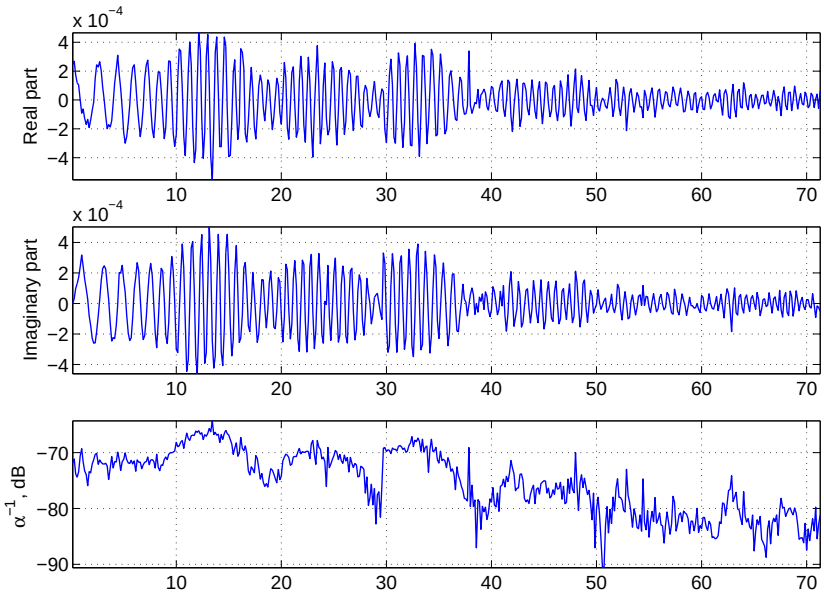


Figure 6.23: (Example 4) Evolution of the real and imaginary parts of the gain and of the power of the estimated track.

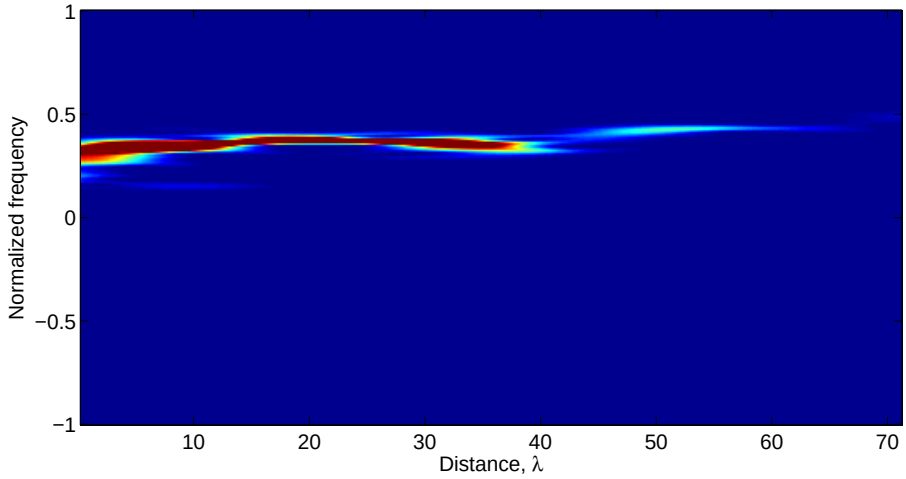


Figure 6.24: (Example 4) Spectrogram of the complex gain variation of the estimated track.

multipath gain. The evaluated PG characteristics are shown in Fig 6.25.

Interestingly, in this case we perform a bit better with nonlinear models, compared to Example 2. In general, despite the fact that we get slightly different parameter trajectories, the achieved averaged PG performance is pretty much the same as compared with the SAGE-based tracks and predictions.

6.4.2 Tracking example 5: Tracking several components

In this example we set the number of tracked components to $K = 5$ as in the simi-

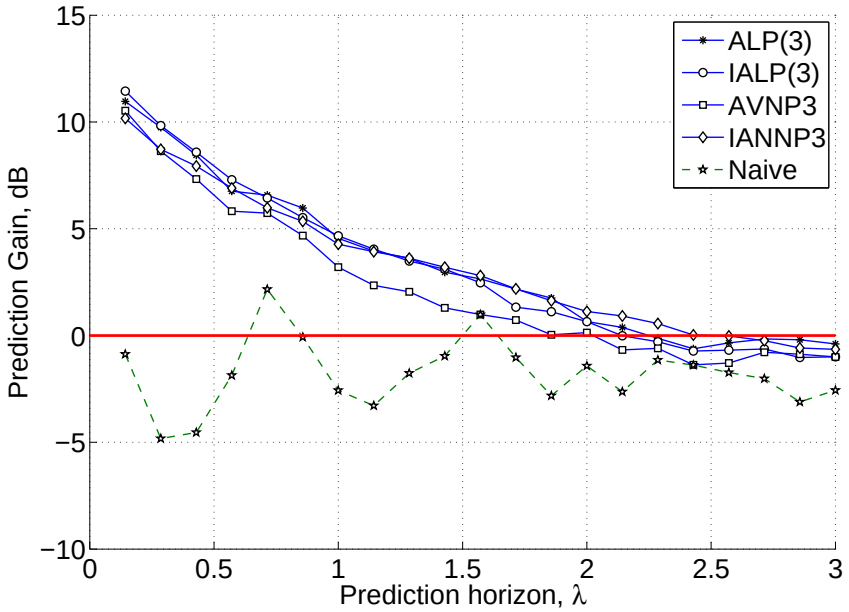


Figure 6.25: (Example 4) Prediction gain for a single track.

lar SAGE-based Example 3. Again, we leave the tracking and prediction parameters unchanged.

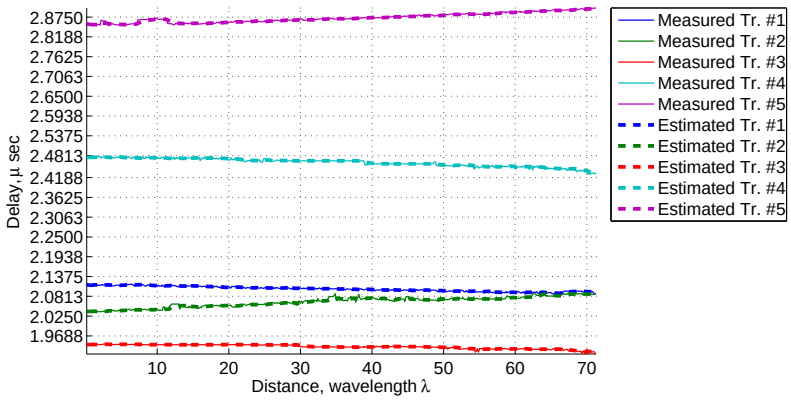
In Fig. 6.26 we show the evolution of the delay, Doppler and DoA trajectories for the selected multipath components. As we can see, there is not much difference between the structure parameter trajectories here and those in Example 3. This is actually the results we expect – the SAGE and SAGE-RVM algorithm possess a lot in common, and the parameter estimates obtained with these algorithms differ but not significantly. Since we use the same tracker setup we expect similar parameter evolutions.

The evolution of the complex gain and powers for the reconstructed tracks are shown in Fig. 6.28 and 6.27, respectively.

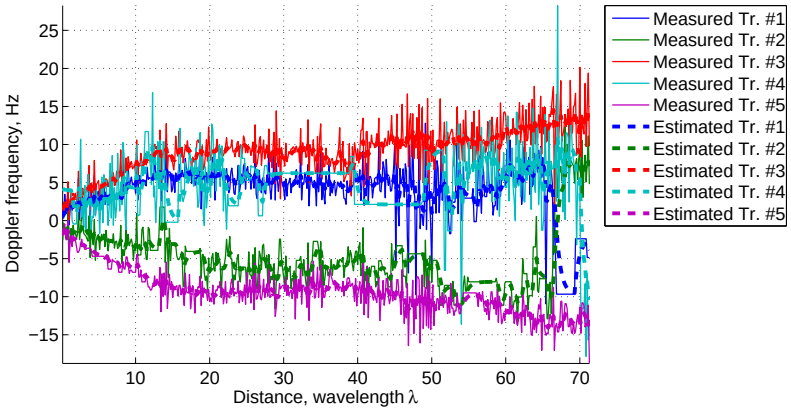
As we see, there is again not much difference. Again, we can identify Tracks 1, 3, and 5 as potentially useful for long-term prediction. Tracks 2 and 4 are less suited for prediction due to the inconsistent structure, just as in the Example 3.

Now, let us consider the prediction results for these tracks, shown in Fig. 6.29. The PG results where averaged over 100 independent model initializations.

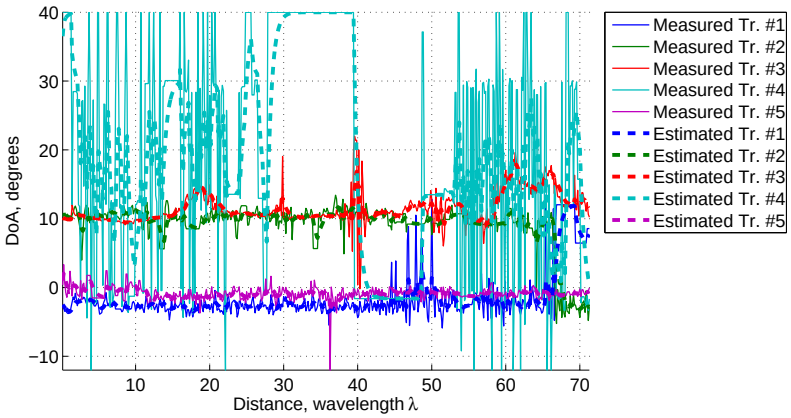
Comparing the results in Fig. 6.29 with those in Fig. 6.21 (i.e., with the SAGE-based prediction), we can notice a slight improvement (around 1dB) for all the observed tracks. This slight improvement might well be explained by the more smooth power envelopes of SAGE-RVM estimated tracks in Fig. 6.27 as compared to those estimated with the SAGE algorithm in Fig. 6.19. The parameter trajectories are smoother and, as a result, less noise is injected in the hypermodel update. Note that the PG increase is not that significant, but still it is better than in Example 3.



(a) Track delay.



(b) Doppler frequency.



(c) DoA.

Figure 6.26: (Example 5) Reconstructed multipath tracks. (SAGE-RVM).

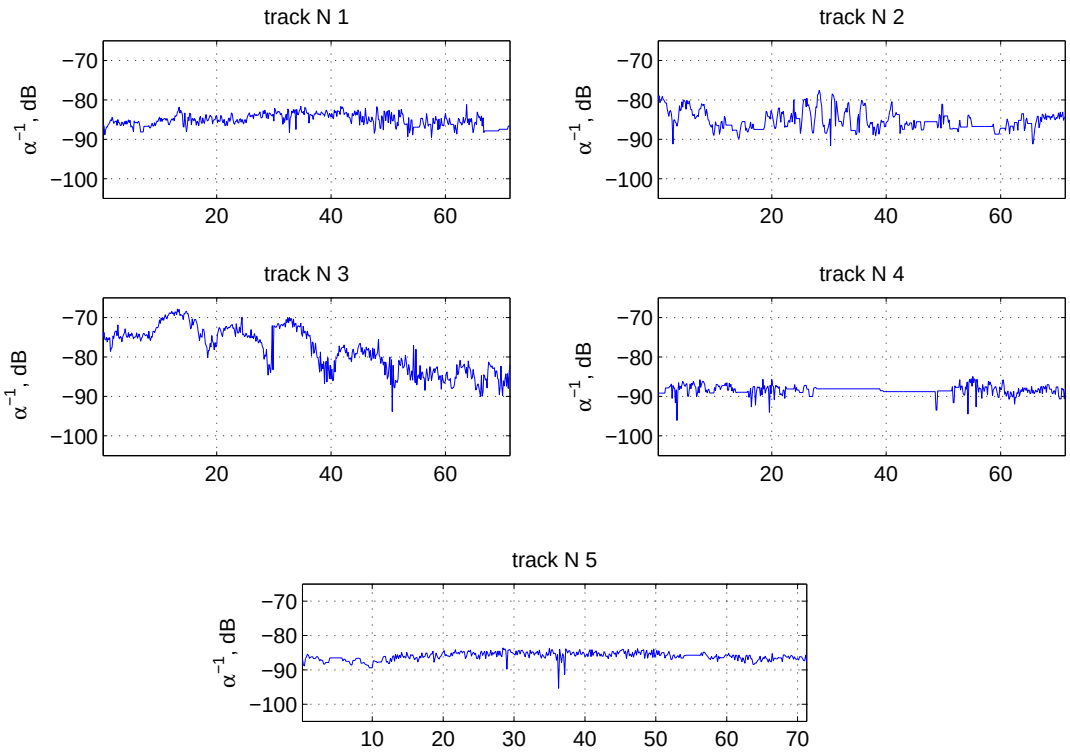


Figure 6.27: Evidence of the tracked components.

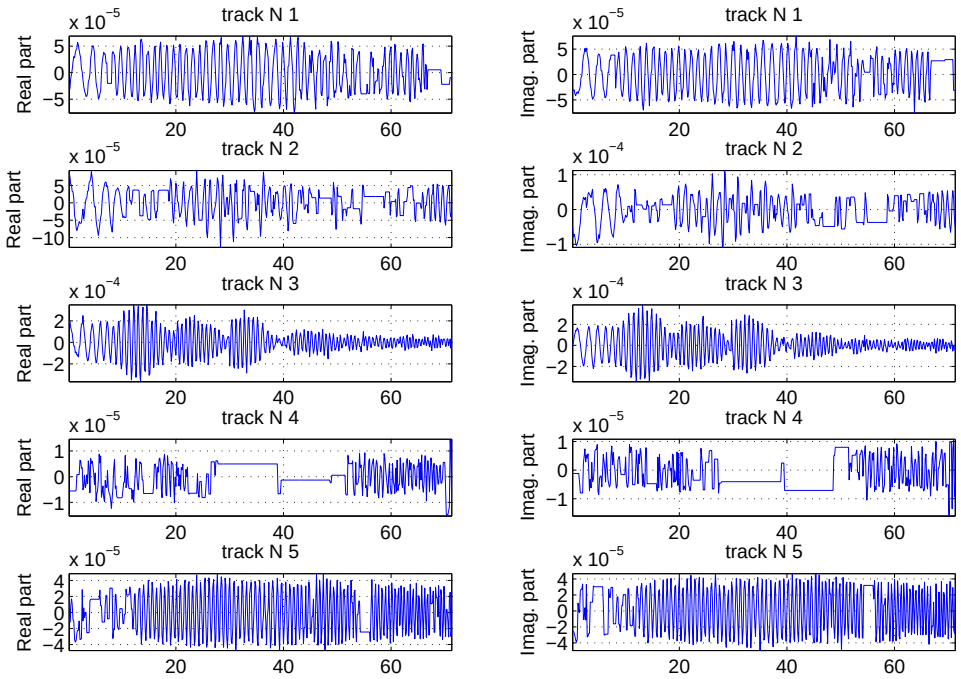
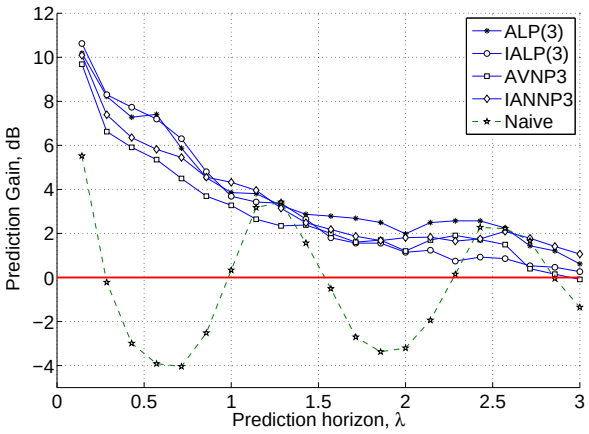
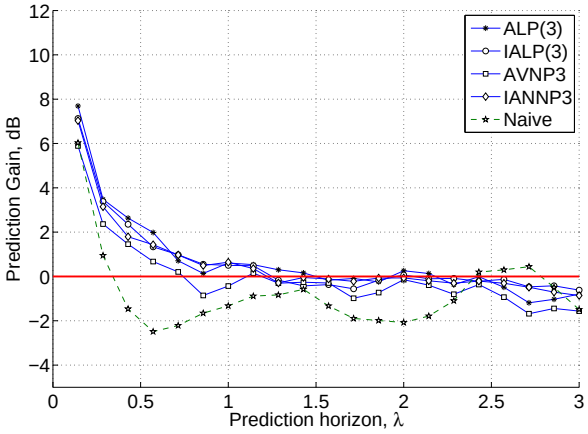


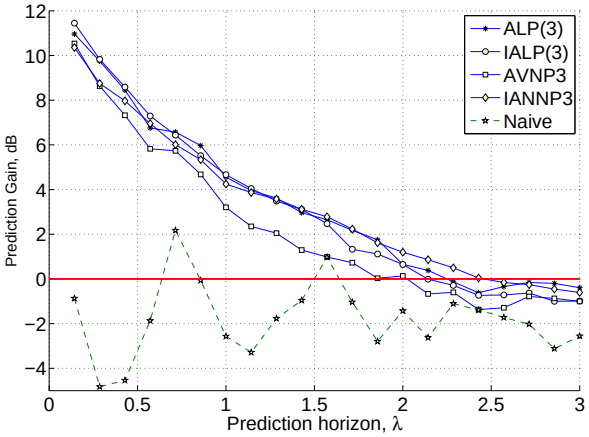
Figure 6.28: Evolution of the real and imaginary parts of the gain for the estimated tracks.



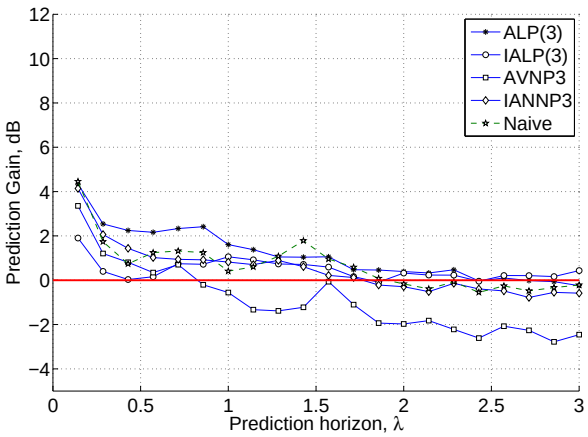
(a) Track N1



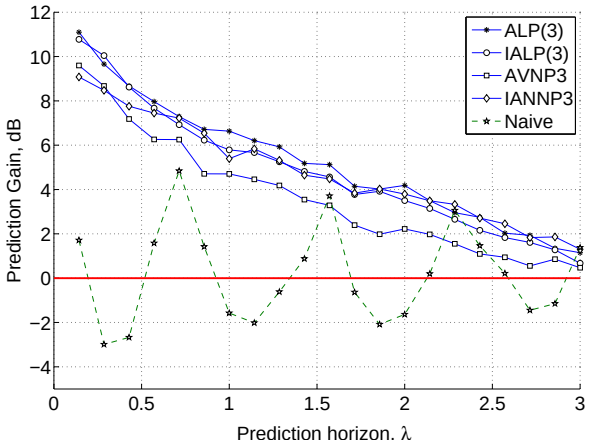
(b) Track N2



(c) Track N3



(d) Track N4



(e) Track N5

Figure 6.29: (Example 5) PG evaluated for $K = 5$ reconstructed tracks.

6.5 Discussion of the obtained tracking and prediction results

Based on the demonstrated experiments we can now discuss the obtained tracking and prediction results. Let us first begin with the tracking results.

6.5.1 Tracking

In this chapter we have considered several examples that demonstrate our multipath prediction approach. A short summary of the key experiment parameters is shown in Table 6.4.

	Estimation alg.	Tracking length	K	L
Experiment 1	SAGE	28λ	1	9
Experiment 2	SAGE	71λ	1	9
Experiment 3	SAGE	71λ	5	15
Experiment 4	SAGE-RVM	71λ	1	20
Experiment 5	SAGE-RVM	71λ	5	20

Table 6.4: Summary of the tracking and prediction experiments.

A first observation we can make from all the experiments, is that it is difficult, if not impossible, to track a component over an infinitely long time interval. There are areas, where tracking is quite stable, for example in Experiment 2 between 0λ and 30λ. During this tracking interval we expect a relatively good prediction performance. However, between 30λ and 40λ there are several instances where the tracker definitely picks up a wrong component, most likely an estimation artifact. It can well be seen that these power drops coinside in almost all cases with the jumps of the delay trajectory to the neighboring sampling instant. This is definitely the result of insufficient delay resolution: for the FTW data the estimation algorithm is simply unable to reliably estimate a component between the sampling instants. Indeed, the FTW data sampling period is 160MHz and the channel bandwidth is 120MHz, which corresponds to the oversampling factor of 1.33, which is too low. Such "jumps" lead to transients in the corresponding hypermodels and as the result to the further propagation of tracking errors.

The remedy to this situation is twofold: first of all, we should provide data with the highest possible resolution. And second, we should employ some track management algorithm that constantly monitors the "track health" and when the inconsistent behavior is detected, starts tracking anew, from scratch, possibly using different physical components from the set of estimated ones.

Of course, when the data has low resolution we can still make use of our prediction scheme. As we have shown in the examples, we obtain the best prediction results from agile hypermodels, which adapts fast to changing data, yet which are complex enough to adequately model the multipath dynamics.

In case of structure hypermodels, their agility can be controlled mainly through the specification of the disturbance parameters in the corresponding state-space formulation of the DLLT. We found these values empirically, but in the future it is imperative to find a functional relationship between the model disturbance parameters and the measured data resolution.

Let us now discuss the hypermodels we used for gain prediction.

6.5.2 Gain prediction

We will begin by highlighting some general observations about the hypermodels we used for gain prediction.

Previously we classified the hypermodels according to the way predictions are realized, i.e., iterated predictors (trained using the joint EKF algorithm) and $\mathcal{L}$-step predictors (trained using the RLS algorithm), as well as according to their structure, i.e., linear or nonlinear.

With respect to the way predictions are realized, we observed that iterative predictors perform a bit better than their $\mathcal{L}$-step counterparts. As a rule, they converge faster, and result in higher predictions gains. We should, however, be careful in the interpretation of these results. This difference might come from the fact that the iterated predictors are trained within the Kalman Filter framework and the corresponding predictions are based on the filtered hypermodel states. Also, the KF-based learning has more free parameters, than the RLS-based algorithm. This basically means that these parameters can be tuned to allow better performance. The set of these "tunable" parameters includes the specification of the observation and state noise variances, which in the RLS case is not needed. This extra degree of freedom allows fitting the models better to the data. Although we cannot say how to tune these parameters in a systematic way, we see that, since we found good parameters empirically, there must be an objective optimum that minimizes the prediction error.

We also consider linear and nonlinear structures for the hypermodel designs. Linear structures attract by their simplicity, which consequently results in easier hypermodel training. The nonlinear structures, however, also deserve attention.

Our empirical observation shows that the signals we are predicting are in fact chirped complex exponentials or, more generally, polynomial phase signals. For such kind of signals the linear predictors are not optimal, and a nonlinear predictor would be a more appropriate one. We do not claim that the nonlinear structures we use here are optimal, but they can still capture at least some of the nonlinear structure present in the signal. Theoretically, this might extend the model validity range and require less frequent adaptation. Unfortunately in practice we see that since more data is needed for training a nonlinear structure, the resulting models can not be fully trained, and, as a consequence, result in lower PG. The linear hypermodels, though not optimal in capturing the nonlinear dynamics of the observed signal, turn out to be more effective in multipath prediction since they adapt faster. Indeed, in some cases, usage of the nonlinear models might be an overcomplication. The

dynamics of a single component is much simpler than that of the whole channel, so an adaptive linear predictor with only a few coefficients might solve the task of predicting/tracking a time-varying narrowband signal sufficiently well. This is exactly what we observe in all experiments.

Let us now consider the performance of the used gain hypermodels in more details.

Adaptive Linear Predictor (ALP)

The ALP hypermodel is one of the simplest. It implements an AR model and, due to the linearity, it can be easily estimated and tracked using the classical RLS algorithm. The linearity of this model is a very attractive feature, since there are many algorithms that can be used to estimate the coefficients of the hypermodel. Especially, if one considers the hardware implementation of such filter. We use the RLS algorithm for fitting this hypermodel because it has a very fast convergence, but other adaptation algorithms, like LMS or its modifications, can also be used to recursively adapt filter coefficients.

In order to allow the algorithm to adapt to nonstationary data, we can specify a forgetting constant that allows only the recent data to influence the coefficient update. In our simulations the forgetting constant is fixed, but it might, however, be profitable to adjust it depending on the track performance. Weak tracks, with unstable structure, should rely more on the current information, while the stronger tracks should exploit their past dynamics more heavily.

The ALP hypermodel has quite good adaptation properties. We have observed that the predictions converge within a relatively short period of time spanning $\approx 4\lambda$. Taking into account that the channel acquisition period was 20msec, with roughly 7 samples per wavelength, we conclude that the predictor needs around ≈ 500msec of training time. Depending on how well the component can be tracked we can expect the prediction horizons for this hypermodel to extend[1] as far as $2.5\lambda - 3\lambda$, or equivalently $\approx 350 - 420$msec. Note that the time scales we specify here are obtained for the FTW channel data. If, for instance, we take a higher channel acquisition times to be able to estimate Doppler frequencies induced by, e.g., high-speed trains, we will need to rescale the 3λ-horizon appropriately. In this case we will certainly obtain shorter temporal (i.e., in seconds) prediction horizons.

Iterated Adaptive Linear Predictor (IALP)

This type of hypermodel again represents a linear predictor that relies on different learning strategy than the ALP. With the IALP, we employ the Kalman Filter framework to estimate filter coefficients. Although it is possible to formulate the KF to estimate the coefficients of the hypermodel for the fixed $\mathcal{L}$-step prediction, such iterative structure computationally is more efficient, since we can obtain forecasts for any desired prediction horizon without re-training the predictor.

[1] Measured with respect to the positive PG

The price we pay for this, however, is the stability of the predicted signal, especially for long prediction intervals. This requires extra measures for detecting instabilities.

The agility of the IALP predictor is regulated through the parameters of the disturbance terms in the state space model formulation. Indirectly these parameters influence the convergence properties of the algorithm in a similar way the RLS forgetting constant influences the performance of the ALP hypermodel. They definitely increase the list of parameters that are to be set up. For simplicity we chose the corresponding parameters empirically. However, we admit that a more rigorous study is needed to find objective rules for choosing the proper values so as to minimize the resulting prediction error.

These extra degrees of freedom allow to obtain more agile hypermodels. That is why the IALP predictor performs a bit better than the ALP hypermodel, especially for short prediction horizons. The IALP requires $\approx 2\lambda$ (or equivalently 280msec) to converge and achieves roughly similar prediction horizon of $2.5\lambda - 3\lambda$.

Adaptive Volterra-Based Nonlinear Predictors (AVNP)

This predictor is a nonlinear extension of the ALP hypermodel. In the AVNP, the relationship between the input and output are captured by a nonlinear polynomial filter – a Volterra filter. The major advantage of the Volterra models is the relatively simple learning procedure, which conceptually does not differ from learning the ALP.

The AVNP hypermodel has more degrees of freedom as compared to the ALP hypermodel, namely the order of the nonlinearity, and the memory size for each nonlinearity order. In our simulations we set these parameters empirically by trying several different model structures.

Finding an optimal structure that would minimize the prediction error is also possible, but it has not been addressed in our work. This can be solved by exploiting the Description Length criterion or the Evidence framework similar to the way we solved the model selection problem in the multipath estimation algorithm.

Although the best structure was found by trial and error, we did observe that the performance of the Volterra based predictor degrades with the growing nonlinearity order and memory length of nonlinear terms. This leads us to conclude that the used models simply do not capture the signal dynamics well. By increasing the number of parameters we simply overfit the data. As the result, the corresponding hypermodel fails to produce good predictions once it is applied to the prediction of unseen data. The higher number of coefficients involved in the model also increases the adaptation time of the predictor.

For example, in case of the AVNP3 hypermodel, the number of coefficients is 19. The corresponding learning time is similar to that of the ALP, i.e., $\approx 4\lambda$. In general we can say that with the AVNP predictors we have not achieved a desired performance and the higher model complexity does not bring any visible advantage.

Iterated Adaptive Neural Network-based Predictor (IANNP)

This is a second type of nonlinear structure we consider in our work. The IANNP is a nonlinear extension of the IALP predictor, with the distinction from the AVNP that the nonlinearity is represented with a neural network.

Similarly to the AVNP, selecting the optimal structure for this hypermodel is not trivial. In this case we need to specify the number of input neurons as well as the number of neurons in the hidden layer. Furthermore, the usage of the KF framework to learn the coefficients of the network also requires specification of the state and observation noise parameters, just like for the IALP hypermodel. This all creates extra degrees of freedom that have to be specified.

Again, it is possible to employ algorithms that optimize the network structure as it has been done in [Nea96], and estimate noise parameters so as to achieve the minimum of the prediction error. We stress, however, that the development of these algorithms falls outside the scope of this work. Here we also find suitable parameters by a simple trial and error approach.

The IANNP hypermodel utilizes more parameters than its linear counterpart IALP. The IANNP3 predictor, used in Experiments 2 to 5 has 19 coefficients. However, although the IANNP does not deliver outstanding performance for the short prediction horizons, it performs better than other hypermodels for long prediction intervals (approx. longer than 1.5 to 2λ). In favorable conditions (Track 5 in Examples 3 and 5) the positive PG extends as far as 3λ and even outperforms the linear models.

Chapter 7

Discussion and conclusions

Now, let us conclude the results obtained in this work and outline the future extensions of the multipath-based channel prediction.

The presented work addresses the prediction of multipath wireless MIMO channels. Common approaches attempt to model the dynamics of channel taps by building models using sampled channel data. Although such approaches are viable, they differ for wideband and narrowband channels, as well as for MIMO, SIMO/MISO and SISO channels. They often neglect the physical channel "background", i.e., the rich internal channel structure. Furthermore, they do not attempt to cancel the "fading" at its source by decoupling the interfering components.

In our work we account for the channel physics by viewing a channel as a sum of contributing wavefronts – multipath components. First of all, our approach is a general strategy. This framework can be applied not only to SIMO channels, as was demonstrated, but similarly to the MISO, MIMO, and SISO channels. It can also be easily adapted for wideband, as well as narrowband channels.

On a very abstract level, we simply decompose the channel into smaller subcomponents that eventually have simpler dynamics than the whole channel – *divide et impera* principle. This also removes the fading since the interfering components are resolved. The multipath-based decomposition we used in our work is just a possible way to go. Let us however explicitly state that other decompositions are also possible. In particular, we can model a channel as a collection of multipath clusters. Estimating clusters instead of the multipath components might result in easier tracking. We have observed that tracking individual components in a cluster, i.e., in areas with dense multipath components, might be difficult. Clusters might be superior in a sense that they "average" the trajectories of the individual multipath components to the trajectories of the cluster centers. The artifacts in this case will influence the cluster trajectory significantly less. In case of cluster tracking, it is quite possible there will be a need to redefine the learning algorithms, as well as to fit the hypermodels to the dynamics of the clusters.

Channel representation is basically a first step of our multistage channel prediction framework. Once we extract the appropriate structure, the next step is to keep this structure up-to-date. Thus, proper tracking is essential for the further prediction stage. In our work we implemented coupled tracking and hypermodel building. The concept we applied is not new in sequential data processing. Similar ideas are

implemented in Dual Kalman filter estimation [Hay01, ch. 5]. What is significantly different in our work is that usually we have estimated components than tracks. Thus, additionally we have to find proper associations between the estimates and the tracked components. We have solved the association quite effectively using Dynamical Programming techniques. However, this clearly increases the overall complexity of the tracker. Nonetheless, the obtained results prove the feasibility of the proposed scheme.

The increased tracker complexity inevitably leads to possible tracking errors and very undesirable hypermodel adaptation transients. The latter leads to prediction quality worsening. Further, we observed that proper tracking of individual components in clutter is difficult. Such cluttering occurs naturally in the vicinity of the LOS and of other strong components, as well as due to possible presence of estimation artifacts. Minimizing the tracking errors in such situations must be a prime goal of the future development of the tracking algorithm. The artifacts, for example, are best identified by the fast decay of their power envelope. This information is not available to the channel estimator but it is available to the tracker. By exploiting the knowledge of the component dynamics we can minimize the artifacts estimation by improving SAGE or SAGE-RVM initialization.

Furthermore, the tracking algorithm can be significantly improved, if we develop an intelligent way to control which components are to be tracked and which should be dropped from the tracker. In other words, we would like to maximize the number of tracked components, but at the same time save resources by tracking only those components that are potentially useful for applications where the tracking and prediction algorithm is to be used.

The final stage of our framework is the hypermodel construction. As we already mentioned, we have coupled this stage with the tracking algorithm to ease solving the association problem. Clearly, the goal of the hypermodels is to represent the dynamics of the multipath components, or other structures that are used to model a wireless channel, for instance clusters. In our work we have used hypermodels with a relatively small number of coefficients to ensure that the models a less sensitive to tracking errors. As can be seen, our main goal was long-term prediction of the multipath complex gain. For prediction of structure parameters, i.e., delay, Doppler frequency, and DoA we used a relatively simple structure, since for these parameters we mainly need one-step-ahead predictions that can be accomplished using simple linear models. For gain prediction we have used more elaborate, linear as well as nonlinear predictors, but have still tried to keep the structure as simple as possible.

From our empirical studies it follows that even with few coefficients we can ensure a positive prediction gain as far as 3λ into the future, which is significantly longer than any of the results known to us from the literature. We should also mention that the data we have used has very low spatial sampling.

For gain prediction, we tried both linear as well as nonlinear hypermodels, but generally linear models are preferred. They are simpler to interpret and to adapt. Moreover, once the components are extracted their dynamics is much simpler as compared to the dynamics of the whole channel, and thus simple linear predictors

might be sufficient. Nonlinear predictors might be usefull once we can improve the tracker and obtain longer segments that can be used for model building. Among the proposed linear models, we can say the ALP and IALP are, to a certain extent, equivalent. Altough for short prediction horizons we think that IALP should be used, while for longer horizons ALP is more appropriate since it avoids generating unstable predictors.

Keep in mind that when clusters are to be used instead of the multipath components, it is possible that linear models will no longer be adequate for prediction. Still, we believe that the main emphasis should stay more on developing accurate tracking methods, rather than on further refinements of hypermodels. The considered hypermodels are quite general and can be applied to different problems. Thus it will only be needed to try several off-the-shelf hypermodel structures to find the appropriate one. But without proper tracking, learning might go amiss for these models.

What is certainly left to be done in hypermodel design is the minimization of the number of free parameters, which were selected empirically in our work. Relating these parameters to the resolution of the channel data, as well as to the characteristics of the estimation algorithm is a possible way to proceed.

In conclusion, let us again point out that we have proved the viability of the multipath-based prediction approach. We have described and discussed the multistage wireless channel prediction procedure. We have also shown that it is possible to make use of hypermodels that allow long-term forecasts of channel parameters, which in turn can be used to mitigate fading in wireless communication.

Appendix A

Taylor approximation to the electrical distance (SIMO case)

Here we consider an approximation to the electrical distance $-\kappa\|\boldsymbol{r}_{l,p}(t)\|$ term for the simplified scenario. We assume a single wave source that contributes to the channel moving along the direction specified by the vector $\boldsymbol{x}$.

The corresponding scenario is depicted in Figure A.1.

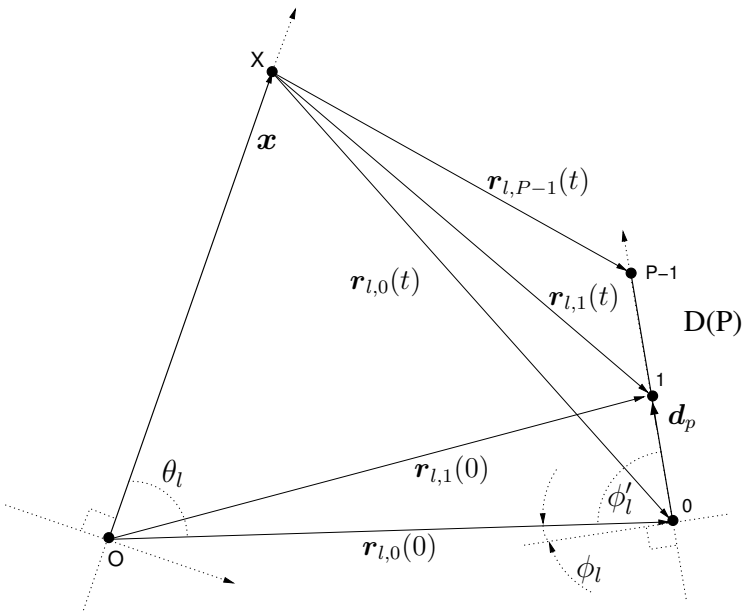


Figure A.1: Computing the electrical distance term for the SIMO case.

Here, the receiving antenna $D(P)$ is equipped with P sensors, $p = 0, \ldots, P-1$. The vectors $\boldsymbol{d}_p$ points from the arbitrary reference sensor (here indexed as $p = 0$) to another sensor in the array.

The distance $\boldsymbol{r}_{l,p}(t)$ can be computed as

$$\boldsymbol{r}_{l,p}(t) = \boldsymbol{r}_{l,p}(0) - \boldsymbol{x} = \boldsymbol{r}_{l,0}(0) + \boldsymbol{d}_p - \boldsymbol{x}. \tag{A.1}$$

Further, from eq. (A.1) the squared norm $\|\boldsymbol{r}_{l,p}(t)\|^2$ of this distance can be computed

as

$$\begin{aligned}
&\|\boldsymbol{r}_{l,p}(t)\|^2 = \langle \boldsymbol{r}_{l,0}(0) + \boldsymbol{d}_p - \boldsymbol{x}, \boldsymbol{r}_{l,0}(0) + \boldsymbol{d}_p - \boldsymbol{x}\rangle = \\
&= \|\boldsymbol{r}_{l,0}(0)\|^2 - 2\langle \boldsymbol{r}_{l,0}(0), \boldsymbol{d}_p\rangle - 2\langle \boldsymbol{r}_{l,0}(0), \boldsymbol{x}\rangle + \|\boldsymbol{d}_p\|^2 + \|\boldsymbol{x}\|^2 - 2\langle \boldsymbol{d}_p, \boldsymbol{x}\rangle = \\
&= \|\boldsymbol{r}_{l,0}(0)\|^2 \left[1 + \frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0}(0)\|^2} - \frac{2\|\boldsymbol{x}\|\cos(\theta_l)}{\|\boldsymbol{r}_{l,0}(0)\|} + \frac{2\|\boldsymbol{d}_p\|\sin(\phi_l)}{\|\boldsymbol{r}_{l,0}(0)\|}\right].
\end{aligned} \tag{A.2}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product operator. Since we are interested in $\|\boldsymbol{r}_{l,p}(t)\|$ rather than its squared value, we compute the square root of (A.2). In order to simplify the resulting expression we expand the square root of the right-hand side of (A.2) in a second-order Taylor series around zero. Using the fact that $\sqrt{1+y} \approx 1 + y/2 - y^2/8$, we continue as follows:

$$\begin{aligned}
&\|\boldsymbol{r}_{l,p}(t)\| = \|\boldsymbol{r}_{l,0}(0)\| \sqrt{1 + \frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0}(0)\|^2} + 2\frac{\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l)}{\|\boldsymbol{r}_{l,0}(0)\|}} \approx \\
&\approx \|\boldsymbol{r}_{l,0}(0)\| \left[1 + \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0}(0)\|^2} + \frac{\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l)}{\|\boldsymbol{r}_{l,0}(0)\|} - \right. \\
&- \frac{1}{8}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^4}{\|\boldsymbol{r}_{l,0}(0)\|^4} - \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2(\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))}{\|\boldsymbol{r}_{l,0}(0)\|^3} - \frac{1}{2}\frac{\|\boldsymbol{x}\|^2\cos(\theta_l)^2}{\|\boldsymbol{r}_{l,0}(0)\|^2} + \\
&\left. + \frac{\|\boldsymbol{x}\|\|\boldsymbol{d}_p\|\cos(\theta_l)\sin(\phi_l)}{\|\boldsymbol{r}_{l,0}(0)\|^2} - \frac{1}{2}\frac{\|\boldsymbol{d}_p\|^2\sin(\phi_l)^2}{\|\boldsymbol{r}_{l,0}(0)\|^2}\right] = \\
&= \|\boldsymbol{r}_{l,0}(0)\| + \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0}(0)\|} + \|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l) - \\
&- \frac{1}{8}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^4}{\|\boldsymbol{r}_{l,0}(0)\|^3} - \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2(\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))}{\|\boldsymbol{r}_{l,0}(0)\|^2} - \frac{1}{2}\frac{\|\boldsymbol{x}\|^2\cos(\theta_l)^2}{\|\boldsymbol{r}_{l,0}(0)\|} + \\
&+ \frac{\|\boldsymbol{x}\|\|\boldsymbol{d}_p\|\cos(\theta_l)\sin(\phi_l)}{\|\boldsymbol{r}_{l,0}(0)\|} - \frac{1}{2}\frac{\|\boldsymbol{d}_p\|^2\sin(\phi_l)^2}{\|\boldsymbol{r}_{l,0}(0)\|}.
\end{aligned}$$

By collecting together terms with the same order of the denominator, we finally arrive to the second-order approximation to the electrical distance term

$$\begin{aligned}
&\|\boldsymbol{r}_{l,p}(t)\| \approx \|\boldsymbol{r}_{l,0}(0)\| + \|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l) - \\
&- \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2 - \|\boldsymbol{x}\|^2\cos(\theta_l)^2 + 2\|\boldsymbol{x}\|\|\boldsymbol{d}_p\|\cos(\theta_l)\sin(\phi_l) - \|\boldsymbol{d}_p\|^2\sin(\phi_l)^2}{\|\boldsymbol{r}_{l,0}(0)\|} - \\
&- \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2(\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))}{\|\boldsymbol{r}_{l,0}(0)\|^2} - \frac{1}{8}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^4}{\|\boldsymbol{r}_{l,0}(0)\|^3} = \\
&= \|\boldsymbol{r}_{l,0}(0)\| + \|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l) - \\
&- \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2 - (\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))^2}{\|\boldsymbol{r}_{l,0}(0)\|} - \\
&- \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2(\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))}{\|\boldsymbol{r}_{l,0}(0)\|^2} - \frac{1}{8}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^4}{\|\boldsymbol{r}_{l,0}(0)\|^3}.
\end{aligned}$$

Thus, the second-order approximation of $\|\boldsymbol{r}_{l,p}(t)\|$ is given as

$$\begin{aligned}\|\boldsymbol{r}_{l,p}(t)\| \approx \|\boldsymbol{r}_{l,0}(0)\| + \|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l) - \\ - \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2 - (\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))^2}{\|\boldsymbol{r}_{l,0}(0)\|} - \\ - \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^2(\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{x}\|\cos(\theta_l))}{\|\boldsymbol{r}_{l,0}(0)\|^2} - \frac{1}{8}\frac{\|\boldsymbol{d}_p - \boldsymbol{x}\|^4}{\|\boldsymbol{r}_{l,0}(0)\|^3}.\end{aligned} \tag{A.3}$$

Appendix B

Taylor approximation to the electrical distance (MIMO case)

Let us now examine the change of the electrical distance $-\kappa\|\boldsymbol{r}_{l,m,p}(t)\|$ for the simplified MIMO case. We consider a MIMO channel with a transmitting sensor array $F(M)$ with M sensors, and a receiving array $D(P)$ with P sensors, respectively. The transmitting array $F(M)$ is assumed to be moving along the direction specified by the vector $\boldsymbol{x}$ without any rotation. We also assume linear sensor arrays at both channel ends to simplify the derivation. This is, however, not a critical assumption and extensions to more complex array geometries are quite straight forward.

The scenario corresponding to this case is depicted in Figure B.1

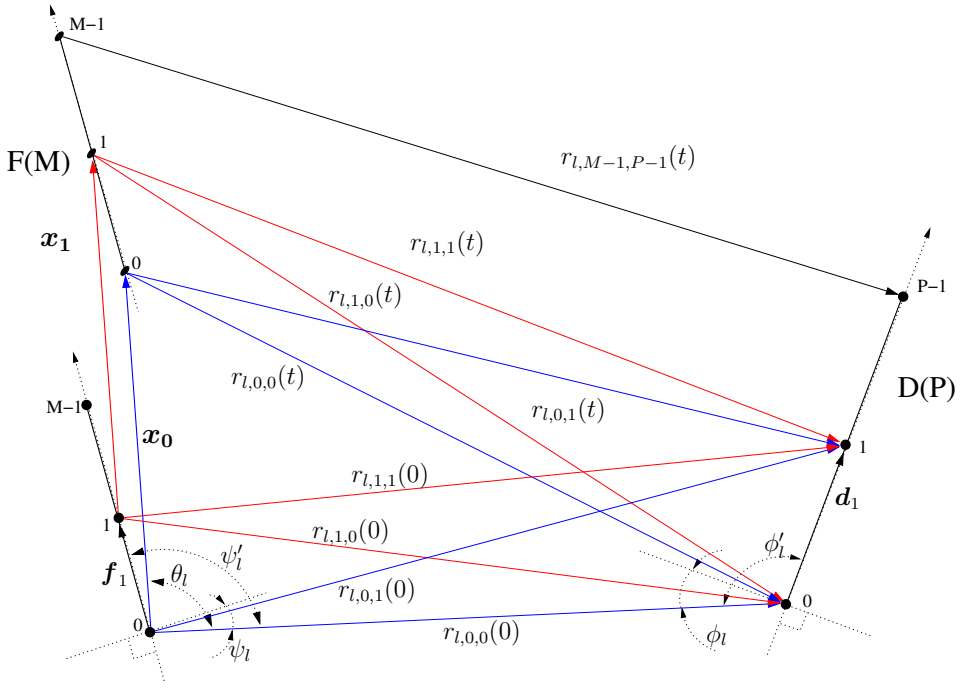


Figure B.1: Computing the electrical distance term for the MIMO case.

By following similar steps as in the SIMO case, we represent the $\boldsymbol{r}_{l,m,p}(t)$ term as

$$\boldsymbol{r}_{l,m,p}(t) = \boldsymbol{r}_{l,m,p}(0) - \boldsymbol{x} = \boldsymbol{r}_{l,m,0}(0) + \boldsymbol{d}_p - \boldsymbol{x} = \boldsymbol{r}_{l,0,0}(0) - \boldsymbol{f}_m + \boldsymbol{d}_p - \boldsymbol{x}. \quad \text{(B.1)}$$

Similarly to the SIMO case, the squared norm $\|\boldsymbol{r}_{l,m,p}(t)\|^2$ of the term of interest is then computed from (B.1) as

$$\begin{aligned}\|\boldsymbol{r}_{l,m,p}(t)\|^2 =&\langle \boldsymbol{r}_{l,0,0}(0) - \boldsymbol{f}_m + \boldsymbol{d}_p - \boldsymbol{x}, \boldsymbol{r}_{l,0,0}(0) - \boldsymbol{f}_m + \boldsymbol{d}_p - \boldsymbol{x}\rangle = \\ =&\|\boldsymbol{r}_{l,0,0}(0)\|^2 + \|\boldsymbol{x}\|^2 + \|\boldsymbol{f}_m\|^2 + \|\boldsymbol{d}_p\|^2 + \\ &+ 2\langle \boldsymbol{r}_{l,0,0}(0), \boldsymbol{d}_p\rangle - 2\langle \boldsymbol{r}_{l,0,0}(0), \boldsymbol{f}_m\rangle - 2\langle \boldsymbol{r}_{l,0,0}(0), \boldsymbol{x}\rangle + \\ &- 2\langle \boldsymbol{d}_p, \boldsymbol{f}_m\rangle - 2\langle \boldsymbol{d}_p, \boldsymbol{x}\rangle + 2\langle \boldsymbol{f}_m, \boldsymbol{x}\rangle,\end{aligned}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product operator. Thus, the length of the path is expressed as

$$\begin{aligned}\|\boldsymbol{r}_{l,m,p}(t)\| = \|\boldsymbol{r}_{l,0,0}(0)\| \Bigg[1 + \frac{\|\boldsymbol{x}\|^2 + \|\boldsymbol{f}_m\|^2 + \|\boldsymbol{d}_p\|^2}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \\ + \frac{-2\langle \boldsymbol{d}_p, \boldsymbol{f}_m\rangle - 2\langle \boldsymbol{d}_p, \boldsymbol{x}\rangle + 2\langle \boldsymbol{f}_m, \boldsymbol{x}\rangle}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \\ + \frac{2\|\boldsymbol{d}_p\|\sin(\phi_l) - 2\|\boldsymbol{f}_m\|\sin(\psi_l) - 2\|\boldsymbol{x}\|\cos(\theta_l)}{\|\boldsymbol{r}_{l,0,0}(0)\|} \Bigg]^{1/2} = \\ = \|\boldsymbol{r}_{l,0,0}(0)\| \left[1 + \frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \frac{2\|\boldsymbol{d}_p\|\sin(\phi_l) - 2\|\boldsymbol{f}_m\|\sin(\psi_l) - 2\|\boldsymbol{x}\|\cos(\theta_l)}{\|\boldsymbol{r}_{l,0,0}(0)\|} \right]^{1/2}.\end{aligned} \tag{B.2}$$

To simplify further analysis, we approximate the square root on the right-hand side of the previous expression around zero. By using the fact that $\sqrt{1+y} \approx 1 + y/2 - y^2/8$, we approximate (B.2) as

$$\begin{aligned}\|\boldsymbol{r}_{l,m,p}(t)\| \approx \|\boldsymbol{r}_{l,0,0}(0)\| \Bigg[1 + \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \\ + \frac{\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{f}_m\|\sin(\psi_l) - \|\boldsymbol{x}\|\cos(\theta_l)}{\|\boldsymbol{r}_{l,0,0}(0)\|} - \\ - \frac{1}{8}\left(\frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \frac{2\|\boldsymbol{d}_p\|\sin(\phi_l) - 2\|\boldsymbol{f}_m\|\sin(\psi_l) - 2\|\boldsymbol{x}\|\cos(\theta_l)}{\|\boldsymbol{r}_{l,0,0}(0)\|} \right)^2 \Bigg] = \\ \|\boldsymbol{r}_{l,0,0}(0)\| \Bigg[1 + \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \frac{\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{f}_m\|\sin(\psi_l) - \|\boldsymbol{x}\|\cos(\theta_l)}{\|\boldsymbol{r}_{l,0,0}(0)\|} - \\ - \frac{1}{8}\frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^4}{\|\boldsymbol{r}_{l,0,0}(0)\|^4} - \frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^2(\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{f}_m\|\sin(\psi_l) - \|\boldsymbol{x}\|\cos(\theta_l))}{\|\boldsymbol{r}_{l,0,0}(0)\|^3} - \\ - \frac{1}{2}\frac{(\|\boldsymbol{d}_p\|^2\sin(\phi_l)^2 - \|\boldsymbol{f}_m\|^2\sin(\psi_l)^2 - \|\boldsymbol{x}\|^2\cos(\theta_l)^2)}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} + \\ + \frac{\|\boldsymbol{d}_p\|\|\boldsymbol{f}_m\|\sin(\psi_l)\sin(\phi_l) + \|\boldsymbol{d}_p\|\|\boldsymbol{x}\|\sin(\phi_l)\cos(\theta_l) - \|\boldsymbol{f}_m\|\|\boldsymbol{x}\|\cos(\theta_l)\sin(\psi_l)}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} \Bigg].\end{aligned}$$

Thus we arrive to the final second order approximation of the path distance $\|\boldsymbol{r}_{l,m,p}(t)\|$:

$$\begin{aligned}
&\|\boldsymbol{r}_{l,m,p}(t)\| \approx \|\boldsymbol{r}_{l,0,0}(0)\| + \|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{f}_m\|\sin(\psi_l) - \|\boldsymbol{x}\|\cos(\theta_l) + \\
&+\frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^2}{\|\boldsymbol{r}_{l,0,0}(0)\|} - \frac{1}{2}\frac{\|\boldsymbol{d}_p\|^2\sin(\phi_l)^2 - \|\boldsymbol{f}_m\|^2\sin(\psi_l)^2 - \|\boldsymbol{x}\|^2\cos(\theta_l)^2}{\|\boldsymbol{r}_{l,0,0}(0)\|} + \\
&+\frac{\|\boldsymbol{d}_p\|\|\boldsymbol{f}_m\|\sin(\psi_l)\sin(\phi_l) + \|\boldsymbol{d}_p\|\|\boldsymbol{x}\|\sin(\phi_l)\cos(\theta_l) - \|\boldsymbol{f}_m\|\|\boldsymbol{x}\|\cos(\theta_l)\sin(\psi_l)}{\|\boldsymbol{r}_{l,0,0}(0)\|} \\
&-\frac{1}{2}\frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^2(\|\boldsymbol{d}_p\|\sin(\phi_l) - \|\boldsymbol{f}_m\|\sin(\psi_l) - \|\boldsymbol{x}\|\cos(\theta_l))}{\|\boldsymbol{r}_{l,0,0}(0)\|^2} - \\
&-\frac{1}{8}\frac{\|\boldsymbol{d}_p - \boldsymbol{f}_m - \boldsymbol{x}\|^4}{\|\boldsymbol{r}_{l,0,0}(0)\|^3}.
\end{aligned} \tag{B.3}$$

Appendix C

Description of the channel data (FTW)

Here we provide the details on the measured channel data, provided by the Forschungszentrum Telekommunkation Wien (FTW). This data has been mainly used to demonstrate different aspects and stages of the multipath-based channel prediction algorithm.

C.1 General data description

The MIMO channel sounding measurements were performed by Forschungszentrum Telekommunikation Wien, in Vienna, Austria, under the supervision of Helmut Hofstetter[BHMS01]. The measurements were done with the MIMO capable wideband vector channel sounder RUSK-ATM, manufactured by MEDAV [THR+00].

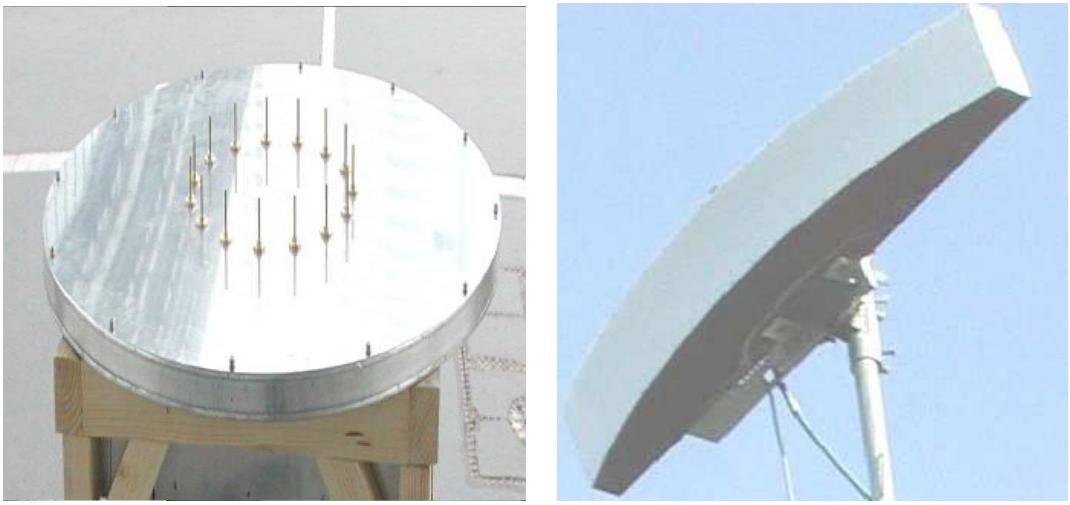
The sounder was specifically adapted to operate at the center frequency of 2GHz. The transmitted signal is generated in the frequency domain to ensure a pre-defined spectrum over 120MHz bandwidth, and an approximately constant envelope over time.

Two simultaneously multiplexed antenna arrays have been used at the transmitter and receiver side (Fig. C.1). The transmitter was a uniform circular array (Fig. C.1(a)) with 15 sensors spaced at approx. 6.45cm apart. The receiver was a fixed uniform linear array (Fig. C.1(b)), with 8 sensors spaced half a wavelength apart, which for the 2GHz carrier frequency corresponds to $\lambda/2 \approx 7.5$cm.

The measurements were performed outdoors, with the receiver array mounted on the roof of a building and the transmitter moving with a velocity of ≈ 1m/s. A MIMO channel snapshot was recorded every $T_r = 20$msec, thus resulting in a spatial resolution of $\approx \lambda/7$. Each MIMO snapshot thus consists in total of 15×8 individual SISO channel IR's that where obtained by temporal antenna multiplexing.

The individual SISO channels are sampled with the sampling rate $F_s = 1/T_s = 160$MHz, resulting in $N = 512$ samples. Some of the crucial sounding parameters are summarized in Table C.1.

In Fig. C.2 we show the relationship between some of the channel sounding parameters and the structure of the resulting MIMO channel impulse response.



(a) Transmitting antenna. (b) Receiving antenna.

Figure C.1: Transmitter and receiver array configurations.

Center frequency	$F_c = 2000$MHz
Measurement double-sided bandwidth	$B_m = 120$MHz
Channel sampling frequency	$F_s = 160$MHz
MIMO channel acquisition period	$T_r = 20$msec
Number of delay bins	$N = 512$
RX-array aperture	120°
TX-array aperture in azimuth	360°
TX-array aperture in elevation	60°

Table C.1: Parameters used in channel sounding.

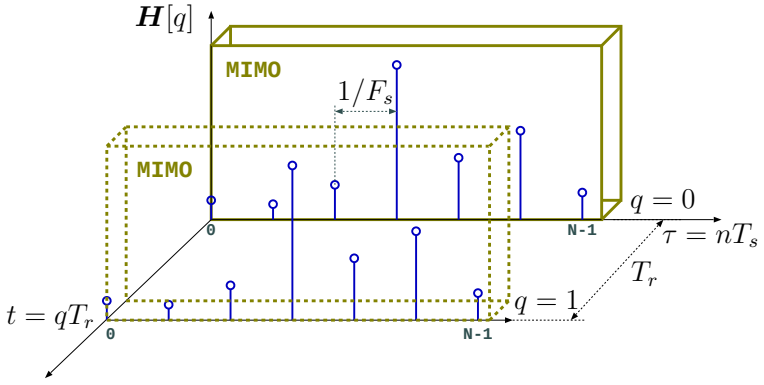


Figure C.2: Relationship between some of the sounding parameters and the structure of the impulse response.

C.1.1 Sample impulse response

The time domain representation of the channel impulse response often reveals a lot of useful details, in particular about the possible positions of the multipath components. A sample IR of the wireless SIMO channel from the FTW database is shown in Fig. C.3

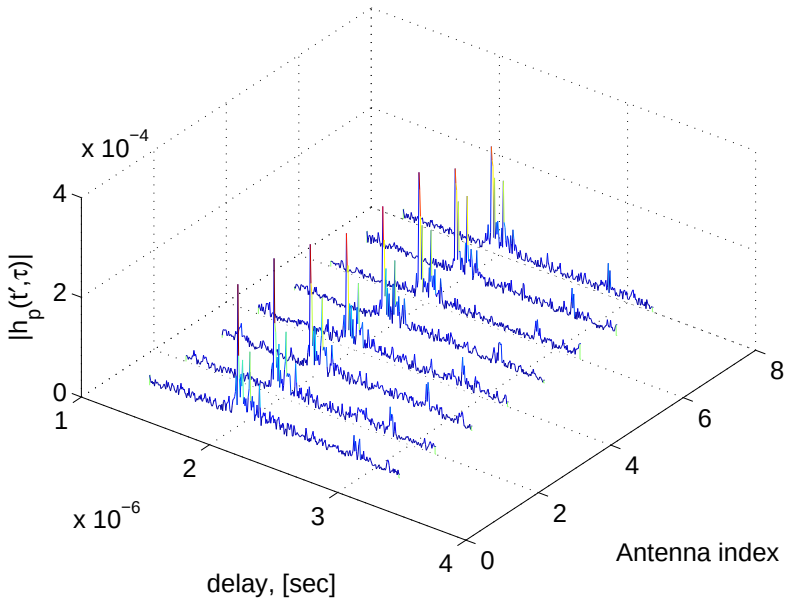


Figure C.3: A sample impulse response of the wireless SIMO channel.

From this example we can clearly identify some of the strongest components arriving around 2msec, it is also possible to see some distinct but relatively week multipaths around 2.8msec.

C.1.2 Doppler-Delay profile

Doppler spread is a key characteristic that defines the rate of channel variation with time. It is known that the channel coherence time T_C, i.e., the time span over which the channel remains roughly time-invariant, is inversely proportional to the Doppler bandwidth. Thus, the Doppler bandwidth defines the rate at which the channel is fading.

In Fig. C.4 we show the power spectral density (PSD) of the Doppler variation for the FTW measurement data. The estimate of the PSD was obtained by applying the Welch periodogram methods to the whole data set consisting of 4000 SIMO channel snapshots.

The PSD shown in Fig. C.4 reveals a very important message. First of all, the maximum Doppler shift $\nu_{\max}$ is bounded, i.e., $\nu_{\max} \leq 20$ Hz. Since the channel acquisition period was 20msec, or equivalently, 50Hz, we can say the channel data was not undersampled and no aliasing has been incurred. This allows re-sampling

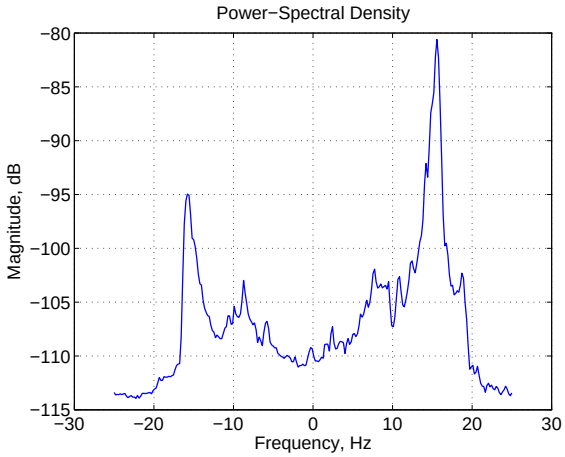


Figure C.4: Estimated Doppler bandwidth.

the channels to a higher spatial resolution, since this eases the task of multipath tracking. In our experiments with the FTW data, the channels were up-sampled by a factor of 5. Then in the estimation algorithm we used a length of the estimation window to be exactly $I = 5$, and thus the spatial sampling of the estimated multipath parameters remains 20msec.

Appendix D

Description of the channel data (Elektrobit)

Here we provide the details on the measured channel data provided by Elektrobit Oy, Finland. The measurement includes only a single MIMO channel snapshot, which prohibited the application of tracking and forecasting algorithms to it. However, we use this data to illustrate the performance of the Evidence Procedure algorithm.

Channel measurements were done with the MIMO-capable channel sounder PropSound manufactured by Elektrobit Oy. Parameters in the delay domain are estimated using the Spread Spectrum Direct Sequence technique (also known as the Pulse Compression technique), while other domains are measured by means of time-domain multiplexing. The DS sounding implementation is equivalent to the block diagram of the channel sounding shown in Fig. 3.1.

The PropSound channel sounder is designed to operate in the frequency range from 5.1 to 5.9GHz, with a chip rate of $1/T_p = 100$Mchips/sec.

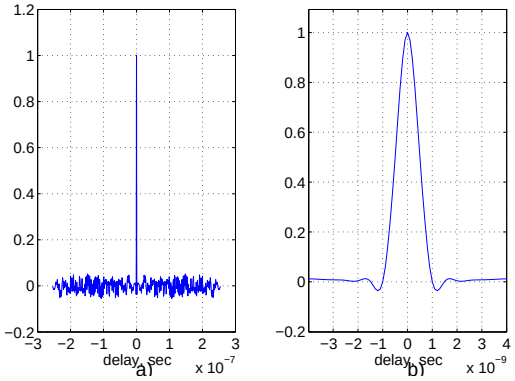


Figure D.1: Evaluated normalized autocorrelation sequence of the sounding signal $u(t)$. a) autocorrelation $R_{uu}(t)$, b) close-up on the main lobe of the $R_{uu}(t)$.

The output of the matched filter is sampled with the period $T_s = T_p/2$, thus resulting in 2 samples per chip resolution. The used sounding sequence consists of $M = 255$ chips resulting in the burst waveform duration of $T_u = 0.255\mu sec$.

In Fig. D.1 we show the resulting deterministic correlation function $R_{uu}(\tau)$ that is used in the estimation algorithm.

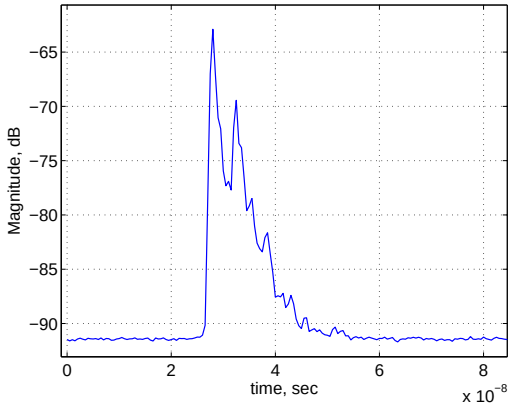


Figure D.2: Computed Power Delay Profile for the PropSound data

The measurement we use was performed indoor, in the Non-Line-Of-Sight setup. The measurement data includes a single MIMO channel, with $P_{TX} = 50$ transmitting and $P_{RX} = 33$ receiving sensors.

The delay profile shown in Fig. D.2 is the guiding tool we used to setup the EP algorithm. We see that the multipath components are likely to have delays from approx. 28nsec to 52nsec. This information is sufficient to select the initial delay search space for the setup of the estimation algorithm.

Appendix E

Evidence update expressions

To derive the update expressions for the evidence parameters in the multiple channels case, let us first rewrite (4.13) for the definitions (4.16). Since both terms under the integral are Gaussian densities, the result can be easily evaluated as

$$\begin{aligned} p(\tilde{\boldsymbol{z}}|\boldsymbol{\alpha},\beta) &= \int p(\tilde{\boldsymbol{z}}|\tilde{\boldsymbol{w}},\beta)p(\tilde{\boldsymbol{w}}|\boldsymbol{\alpha})d\tilde{\boldsymbol{w}} \\ &= \frac{\exp\Big(-\tilde{\boldsymbol{z}}^H(\beta^{-1}\tilde{\boldsymbol{\Lambda}}+\tilde{\boldsymbol{K}}\tilde{\boldsymbol{A}}^{-1}\tilde{\boldsymbol{K}}^H)^{-1}\tilde{\boldsymbol{z}}\Big)}{\pi^{PN}|\beta^{-1}\tilde{\boldsymbol{\Lambda}}+\tilde{\boldsymbol{K}}\tilde{\boldsymbol{A}}^{-1}\tilde{\boldsymbol{K}}^H|} \end{aligned} \tag{E.1}$$

Our goal is to find the values of $\boldsymbol{\alpha}$ and β that maximize (E.1). Now, let us define $\mathcal{L}(\boldsymbol{\alpha},\beta|\tilde{\boldsymbol{z}}) = \log(p(\tilde{\boldsymbol{z}}|\boldsymbol{\alpha},\beta))$. The desired values can be found by taking the derivative of $\mathcal{L}(\boldsymbol{\alpha},\beta|\tilde{\boldsymbol{z}})$ with respect to the parameters of interest and setting those to zero [Ber85]. Since it is often convenient to assume non-informative hyperpriors by setting a, b, c and d in (4.6) and (4.7) to very small values, the resulting prior in logarithmic domain will become uniform. As the result, it is more convenient to maximize with respect to $\log(\alpha_l)$ and $\log(\beta)$ since the derivatives of the prior terms will vanish. Before we begin, we prove the following matrix identity that we will exploit later

$$|\boldsymbol{B}^{-1}||\boldsymbol{A}^{-1}||\boldsymbol{A}+\boldsymbol{K}^H\boldsymbol{B}\boldsymbol{K}| = |\boldsymbol{B}^{-1}+\boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H| \tag{E.2}$$

$$\begin{gathered} |\boldsymbol{B}^{-1}||\boldsymbol{A}^{-1}||\boldsymbol{A}+\boldsymbol{K}^H\boldsymbol{B}\boldsymbol{K}| = \\ |\boldsymbol{B}^{-1}||\boldsymbol{A}^{-1}||\boldsymbol{K}^H[(\boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H)^{-1}+\boldsymbol{B}]\boldsymbol{K}| == \\ |\boldsymbol{B}^{-1}||\boldsymbol{A}^{-1}||\boldsymbol{K}||(\boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H)^{-1}+\boldsymbol{B}||\boldsymbol{K}^H| = \\ |\boldsymbol{K}||\boldsymbol{A}^{-1}||\boldsymbol{K}^H||[(\boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H)^{-1}+\boldsymbol{B}]\boldsymbol{B}^{-1}| = \\ |\boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H[(\boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H)^{-1}\boldsymbol{B}^{-1}+\boldsymbol{I}]| = \\ |\boldsymbol{B}^{-1}+\boldsymbol{K}\boldsymbol{A}^{-1}\boldsymbol{K}^H| \end{gathered}$$

Now, we can begin with the estimation of the hyperparameters α_l. Let us define $\tilde{\boldsymbol{B}}^{-1} = \beta^{-1}\tilde{\boldsymbol{\Lambda}}$. According to (E.2) we see that

$$\begin{gathered} |\tilde{\boldsymbol{B}}^{-1}+\tilde{\boldsymbol{K}}\tilde{\boldsymbol{A}}^{-1}\tilde{\boldsymbol{K}}^H| = \\ |\tilde{\boldsymbol{B}}^{-1}||\tilde{\boldsymbol{A}}^{-1}||\tilde{\boldsymbol{A}}+\tilde{\boldsymbol{K}}^H\tilde{\boldsymbol{B}}\tilde{\boldsymbol{K}}| = |\tilde{\boldsymbol{B}}^{-1}||\tilde{\boldsymbol{A}}^{-1}||\tilde{\boldsymbol{\Phi}}^{-1}|. \end{gathered}$$

Thus,

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}, \beta|\tilde{\boldsymbol{z}})}{\partial \log(\alpha_l)} = \frac{\partial}{\partial \log \alpha_l} \Bigg\{ -\log|\tilde{\boldsymbol{B}}^{-1}||\tilde{\boldsymbol{A}}^{-1}||\tilde{\boldsymbol{\Phi}}^{-1}| - \tilde{\boldsymbol{z}}^H(\tilde{\boldsymbol{B}}^{-1} + \tilde{\boldsymbol{K}}\tilde{\boldsymbol{A}}^{-1}\tilde{\boldsymbol{K}}^H)^{-1}\tilde{\boldsymbol{z}} + \sum_{l=1}^{L}(a\log\alpha_l - b\alpha_l) \Bigg\} =$$
$$\frac{\partial \log|\boldsymbol{A}|^P}{\partial \log \alpha_l} + \sum_{p=1}^{P} \frac{\partial \log|\boldsymbol{\Phi}_p|}{\partial \log \alpha_l} + (a - b\alpha_l)$$
$$-\tilde{\boldsymbol{z}}^H \frac{\partial(\tilde{\boldsymbol{B}} - \tilde{\boldsymbol{B}}\tilde{\boldsymbol{K}}(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{K}}^H\tilde{\boldsymbol{B}}\tilde{\boldsymbol{K}})^{-1}\tilde{\boldsymbol{K}}^H\tilde{\boldsymbol{B}})}{\partial \log \alpha_l}\tilde{\boldsymbol{z}}$$

where in the latter expression the Woodbury inversion identity [GL96] was used to expand the $(\tilde{\boldsymbol{B}}^{-1} + \tilde{\boldsymbol{K}}\tilde{\boldsymbol{A}}^{-1}\tilde{\boldsymbol{K}}^H)^{-1}$ term. Further,

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}, \beta|\tilde{\boldsymbol{z}})}{\partial \log(\alpha_l)} = P \operatorname{tr}\left[\boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial \log \alpha_l}\right] + \sum_{p=1}^{P} \operatorname{tr}\left[\boldsymbol{\Phi}_p^{-1}\frac{\partial \boldsymbol{\Phi}_p}{\partial \log \alpha_l}\right]$$
$$+(a - b\alpha_l) - \tilde{\boldsymbol{z}}^H\tilde{\boldsymbol{B}}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\Phi}}\frac{\partial(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{K}}^H\tilde{\boldsymbol{B}}\tilde{\boldsymbol{K}})}{\partial \log \alpha_l}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{K}}^H\tilde{\boldsymbol{B}}\tilde{\boldsymbol{z}} =$$
$$P - \sum_{p=1}^{P} \operatorname{tr}\left[\alpha_l \boldsymbol{E}_{ll}\boldsymbol{\Phi}_p\right] + (a - b\alpha_l) -$$
$$\tilde{\boldsymbol{z}}^H\tilde{\boldsymbol{B}}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\Phi}}\alpha_l\tilde{\boldsymbol{E}}_{ll}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{K}}^H\tilde{\boldsymbol{B}}\tilde{\boldsymbol{z}}$$

Here $\boldsymbol{E}_{ll}$ is a matrix with the lth element on the main diagonal equal to 1, and its other elements being zero. Similarly, $\tilde{\boldsymbol{E}}_{ll}$ is the P-times repetition of $\boldsymbol{E}_{ll}$ on its main diagonal. By noting that $\tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{K}}^H\tilde{\boldsymbol{B}}\tilde{\boldsymbol{z}}$, we arrive at

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}, \beta|\tilde{\boldsymbol{z}})}{\partial \log(\alpha_l)} = P - \sum_{p=1}^{P} \operatorname{tr}\left[\alpha_l \boldsymbol{E}_{ll}\boldsymbol{\Phi}_p\right] +$$
$$(a - b\alpha_l) - \tilde{\boldsymbol{\mu}}^H\alpha_l\tilde{\boldsymbol{E}}_{ll}\tilde{\boldsymbol{\mu}} = 0.$$

Solving for α_l, we arrive at the final expression for the hyperparameter update

$$\alpha_l = \frac{P + a}{\sum_{p=1}^{P}\left(\Phi_{p,ll} + |\mu_{p,l}|^2\right) + b}.$$

Similarly, for the noise estimate we proceed as

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}, \beta|\tilde{\boldsymbol{z}})}{\partial \log(\beta)} = \sum_{p=1}^{P} \frac{\partial \log|\boldsymbol{B}_p|}{\partial \log \beta} + \sum_{p=1}^{P} \frac{\partial \log|\boldsymbol{\Phi}_p|}{\partial \log \beta} + (c - d\beta)$$
$$-\tilde{\boldsymbol{z}}^H \frac{\partial(\tilde{\boldsymbol{B}} - \tilde{\boldsymbol{B}}\tilde{\boldsymbol{K}}(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{K}}^H \tilde{\boldsymbol{B}}\tilde{\boldsymbol{K}})^{-1}\tilde{\boldsymbol{K}}^H \tilde{\boldsymbol{B}})}{\partial \log \beta}\tilde{\boldsymbol{z}} =$$
$$\sum_{p=1}^{P} \frac{\partial \log \beta^N |\boldsymbol{\Lambda}_p^{-1}|}{\partial \log \beta} + \sum_{p=1}^{P} \mathrm{tr}\left[\boldsymbol{\Phi}_p^{-1} \frac{\partial \boldsymbol{\Phi}_p}{\partial \log \beta}\right] +$$
$$(c - d\beta) - \tilde{\boldsymbol{z}}^H \frac{\partial \beta \tilde{\boldsymbol{\Lambda}}^{-1}}{\partial \log \beta}\tilde{\boldsymbol{z}} +$$
$$\tilde{\boldsymbol{z}}^H \frac{\partial(\beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{K}}(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{K}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{K}})^{-1}\tilde{\boldsymbol{K}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1})}{\partial \log \beta}\tilde{\boldsymbol{z}} =$$
$$PN - \sum_{p=1}^{P} \mathrm{tr}\left[\boldsymbol{\Phi}_p^{-1}\boldsymbol{\Phi}_p \frac{\partial(\boldsymbol{A} + \boldsymbol{K}_p^H \beta \boldsymbol{\Lambda}_p^{-1}\boldsymbol{K}_p)}{\partial \log \beta}\boldsymbol{\Phi}_p\right] +$$
$$(c - d\beta) - \tilde{\boldsymbol{z}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{z}} + \tilde{\boldsymbol{z}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{K}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{z}} +$$
$$\tilde{\boldsymbol{z}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{K}} \frac{\partial(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{K}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{K}})}{\partial \log \beta}\tilde{\boldsymbol{K}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{z}} +$$
$$\tilde{\boldsymbol{z}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{K}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{z}} =$$
$$PN - \sum_{p=1}^{P} \mathrm{tr}\left[\boldsymbol{K}_p^H \beta \boldsymbol{\Lambda}_p^{-1}\boldsymbol{K}_p\boldsymbol{\Phi}_p\right] +$$
$$(c - d\beta) - \tilde{\boldsymbol{z}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{z}} + \tilde{\boldsymbol{z}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\mu}}$$
$$+\tilde{\boldsymbol{\mu}}^H \tilde{\boldsymbol{K}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\mu}}^H \tilde{\boldsymbol{K}}^H \beta\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{z}}.$$

Thus we arrive at the final expression:

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}, \beta|\tilde{\boldsymbol{z}})}{\partial \log(\beta)} = PN - \sum_{p=1}^{P} \mathrm{tr}\left[\boldsymbol{K}_p^H \beta \boldsymbol{\Lambda}_p^{-1}\boldsymbol{K}_p\boldsymbol{\Phi}_p\right] +$$
$$(c - d\beta) - \sum_{p=1}^{P} (\boldsymbol{z}_p - \boldsymbol{K}_p\boldsymbol{\mu}_p)^H \beta \boldsymbol{\Lambda}_p^{-1}(\boldsymbol{z}_p - \boldsymbol{K}_p\boldsymbol{\mu}_p) \stackrel{!}{=} 0.$$

By solving for β we obtain

$$\beta = (PN + c)\left(\sum_{p=1}^{P} \mathrm{tr}\left[\boldsymbol{K}_p^H \boldsymbol{\Lambda}_p^{-1} \boldsymbol{K}_p \boldsymbol{\Phi}_p \right] + \sum_{p=1}^{P} (\boldsymbol{z}_p - \boldsymbol{K}_p \boldsymbol{\mu}_p)^H \boldsymbol{\Lambda}_p^{-1} (\boldsymbol{z}_p - \boldsymbol{K}_p \boldsymbol{\mu}_p) + d \right)^{-1}.$$

Bibliography

[ADX02] A. Arredondo, K.R. Dandekar, and Guanghan Xu. Vector channel modeling and prediction for the improvement of downlink received power. *IEEE Trans. on Comm.*, 50(7):1121– 1129, Jul 2002.

[AJJF99] J.B. Andersen, J. Jensen, S.H. Jensen, and F. Frederiksen. Prediction of future fading based on past measurements. In *50th IEEE Conf. on Vehic. Tech., VTC'99*, volume 1, pages 151 – 155, 1999.

[AS72] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, 1972.

[Ber85] O. Berger. *Statistical decision theory and Bayesian analysis.* Springer, 2nd edition edition, 1985.

[BHMS01] E. Bonek, H. Hofstetter, C. Mecklenbruker, and M. Steinbauer. Double-directional superresolution radio channel measurements. *In Proc. Allerton Conf. Communication, Control, and Computing*, 3, Oct. 2001.

[BRY98] Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, October 1998.

[CBH+06] N. Czink, E. Bonek, L. Hentilä, J.-P. Nuutinen, and J. Ylitalo. Cluster-based MIMO channel model parameters extracted from indoor time-variant measurements. In *Proceeeding of the GlobeCom Conference*, 2006.

[CCS+06] N. Czink, P. Cera, J. Salo, E. Bonek, J.-P. Nuutinen, and J. Ylitalo. Improving clustering performance using multipath component distance. *Electronics Letters*, 42(1):33–5, 2006.

[CNSS03] K. Conradsen, A.A. Nielsen, J. Schou, and H. Skriver. A test statistic in the complex Wishart distribution and its application to change detection in polarimetric SAR data. *IEEE Transactions on Geoscience and Remote Sensing*, 41(1):4–19, 2003.

[DH00] H. Duel-Hallen, A. Shengquan Hu Hallen. Long-range prediction of fading signals. *IEEE Signal Processing Magazine*, 17(3):62 – 75, May 2000.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. John Wiley & Sons, Inc., second edition, 2000.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc.*, 39(1):1–38, 1977.

[DMA97] G.M. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Journal of Constructive Approximation*, 13:57–98, 1997.

[DXL01] Liang Dong, Guanghan Xu, and Hao Ling. Prediction of fast fading mobile radio channels in wideband communication systems. In *IEEE Global Telec. Conf., GLOBECOM '01.*, volume 6, pages 3287 – 3291, Nov 2001.

[EDHH98] T. Eyceoz, A. Duel-Hallen, and H. Hallen. Deterministic channel modeling and long range prediction of fast fading mobile radio channels. *IEEE Communications Letters*, 2(9):254–256, Sep. 1998.

[EHBP00] M. Evans, N. Hastings, B., and Peacock. *Statistical Distributions*. New York: Wiley, 3rd ed. edition, 2000.

[EK99] T. Ekman and G. Kubin. Nonlinear prediction of mobile radio channels: Measurements and MARS model designs. In *Proceedings of the IEEE Int. Conf. on Acoust., Speech, and Signal Proc.*, volume 5, pages 2667–2670, 1999.

[Ekm02] T. Ekman. *Prediction of Mobile Radio Channels: Modeling and Design*. PhD thesis, Uppsala University, Nov. 2002.

[FDHT96] B.H. Fleury, D. Dahlhaus, R. Heddergott, and M. Tschudin. Wideband angle of arrival estimation using the SAGE algorithm. In *IEEE 4th International Symposium on Spread Spectrum Techniques and Applications Proceedings*, pages 79 – 85, September 1996.

[FH94] J.A. Fessler and A.O. Hero. Space-alternating generalized expectation-maximization algorithm. *IEEE Transactions on Signal Processing*, 42:2664–2677, Oct. 1994.

[Fit98] W. J. Fitzgerald. The Bayesian approach to signal modelling. In *Proc. of IEE Colloquium on Non-Linear Signal and Image Processing (Ref. No. 1998/284)*, pages 9/1–9/5, May 1998.

[FT02] A. C. Faul and M. E. Tipping. Analysis of sparse Bayesian learning. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 383–389. MIT Press, 2002.

[FTH+99] B.H. Fleury, M. Tschudin, R. Heddergott, D. Dahlhaus, and K. Ingeman Pedersen. Channel parameter estimation in mobile radio environments using the SAGE algorithm. *IEEE Journal on Selected Areas in Communications*, 17(3):434–450, March 1999.

[FW88] M. Feder and E. Weinstein. Parameter estimation of superimposed signals using the EM algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(4):477–489, April 1988.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[Goo63] N.T. Goodman. Statistical analysis based on a certain multivariate complex Gaussian distribution (An introduction). *Ann. Math. Stat.*, 34:152–177, 1963.

[Grü05] P. Grünwald. A tutorial introduction to the minimum description length principle. In P. Grünwald, I.J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

[Har89] Andrew C. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press., 1989.

[Hay01] Simon Haykin, editor. *Kalman Filtering and Neural Networks*. John Wiley & Sons, Inc., 2001.

[Hec95] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Redmond, WA 98052, March 1995.

[HHDH99] S. Hu, H. Hallen, and A. Duel-Hallen. Physical channel modeling, adaptive prediction and transmitterdiversity for flat fading mobile channel. In *IEEE Workshop on Signal Proc. Advances in Wireless Comm. SPAWC*, pages 387–390, 1999.

[HN95] M. Haardt and J. Nossek. Unitary ESPRIT: How to obtained increased estimation accuracy with a reduced computational burden. *IEEE Trans. on Signal Processing*, SP-43:1232–1242, May 1995.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[Hub81] Peter. J. Huber. *Robust Statistics*. Wiley, 1981.

[HW98] J.K. Hwang and J.H. Winters. Sinusoidal modeling and prediction of fast fading processes. In *Global Telecom. Conf., GLOBECOM'98*, volume 2, pages 892–896. IEEE, 1998.

[KA00] Marvin K.Simon and Mohamed-Slim Alouini. *Digital Communication over the Fading Channels: A Unified Approach to Performance Analysis.* John Wiley & Sons, Inc., 2000.

[KK99] G. Kubin and W. B. Kleijn. Multiple-description coding (MDC) of speech with an invertible auditory model. In *Proc. IEEE Speech Coding Workshop*, pages 81–83, Porvoo, Finland, June 1999.

[KV96] H. Krim and M. Viberg. Two decades of array signal processing research: the parametric approach. *IEEE Signal Processing Mag.*, pages 67–94, July 1996.

[Lan01] A. Lanterman. Schwarz, Wallace, and Rissanen: Intertwining themes in theories of model order estimation. *International Statistical Review*, 69(2):182–215, January 2001.

[LVML96] T.I. Laakso, V. Välimäki, M.Karjalainen, and U.K. Laine. Splitting the unit delay [FIR/all pass filters design]. *IEEE Signal Processing Magazine*, 13(1):30–60, January 1996.

[Mac92] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[Mac94] D. J. C. MacKay. Bayesian methods for backpropagation networks. In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks III*, chapter 6, pages 211–254. Springer-Verlag, New York, 1994.

[Mac03] David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[Mol05] Andreas F. Molisch. *Wireless Communcations*. IEEE Press, 2005.

[Moo96] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.

[MS00a] V. John Mathews and Giovanni L. Sicuranza. *Polynomial Signal Processing*. Wiley, 2000.

[MS00b] Todd K. Moon and Wynn C. Stirling. *Mathematical Methods and Algorithms for Signal Processing*. Prentice-Hall, 2000.

[MZ93] Stephane Mallat and Zhifeng Zhang. Matching pursuit with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[NCP97] Boon Chong Ng, M. Cedervall, and A. Paulraj. A structured channel estimator for maximum-likelihood sequence. *IEEE Communications Letters*, 1(2):52–55, 1997.

[Nea96] R.M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. New York: Springer-Verlag, 1996.

[Oli99] M.W. Oliphant. The mobile phone meets the internet. *IEEE Spectrum*, 36(8):20–28, August 1999.

[O'S00] Douglas O'Shaughnessy. *Speech Communication, Human and Machine*. IEEE Press, 2000.

[PFM97] K. Pedersen, B. Fleury, and P. Mogensen. High resolution of the electtomagnetic waves in time-varying radio channels. In *In Proc. 8th IEEE International Symposium on Personal Indoor and Mobile communications. PIMRC'97*, Helsinki, Finland, September 1997.

[PNG03] Arogyaswami Paulraj, Rohit Nabar, and Dhananjay Gore. *Intoduction to Space-Time Wireless Communicartion*. Cambridge University Press, 2003.

[Poo96] V.H. Poor. *An Introduction to Signal Detection and Estimation*. Springer-Verlag, New York, USA, 1996.

[PRK93] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuits: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Asilomar Conference in Signals, Systems and Computers*, pages 40–44, 1993.

[Pro95] John G. Proakis. *Digital communication*. McGraw-Hill, 1995.

[Rab89] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, Vol. 77(2):257–286, February 1989.

[Rap02] Theodore S. Rappaport. *Wireless communications. Principles and practice.* Prentice Hall PTR, 2002.

[Ris78] J. Rissanen. Modeling by the shortest data description. *Automatica 14*, pages 465–471, 1978.

[Ris96] Jorma J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January 1996.

[RK89] R. Roy and T. Kailath. ESPRIT- estimation of signal parameters via rotation-invariance technique. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 37:984–995, July 1989.

[Sch78] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461464, 1978.

[Sch86] R.O. Schimdt. Multiple emitter location and signal parameter estimation. *IEEE Trans. on Antennas and Propagation*, AP-34:276–280, March 1986.

[Sem03] Sven Semmelrodt. *Methoden zur prädiktiven Kanalschätzung für adaptive Übertragungstechniken im Mobilfunk*. PhD thesis, Kassel Univeristy, 2003. Written in German.

[SF05] D. Shutin and B. Fleury. Application of the evidence procedure to the estimation of the number of paths in wireless channels. In *Proceedings of International Conference on Acoustics Speech and Signal Processing, ICASSP'2005*, volume III, pages 749–752, Philadelphia, USA, April 2005.

[SG04] D. Shutin and G.Kubin. Cluster analysis of wireless channel impulese responses with Hidden Markov Models. In *Proceedings of International Conference on Acoustics Speech and Signal Processing, ICASSP'2004*, pages Vol.4, 949–952, Montreal, Canada, May 2004. IEEE.

[SG05] D. Shutin and G.Kubin. Power prediction of multipath components in wireless MIMO channels. In *Proceeding of the 5th International Conference on Information, Communications and Signal Processing, ICICS'05*, pages 1546–1550, Bangkok, Thailand, 2005.

[Shu04a] D. Shutin. Cluster analysis of wireless channel impulse responses. In *Proceedings of International Zurich Seminar on Communications, IZS'2004*, pages 124–127, Zurich, Switzerland, February 2004.

[Shu04b] D. Shutin. Clustering wireless channel impulse responses in angular-delay domains. In *Proceedings of VI International Workshop on Signal Processing Advances in Wireless Communications, SPAWC2004*, pages 253 – 257, Lisbon, Portugal, July 2004.

[SK04] D. Shutin and H. Koeppl. Application of the evidence procedure to linear problems in signal processing. In *Proceedings of the 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, pages 124–127, Munich, Germany, July 2004.

[SKF] D. Shutin, G. Kubin, and B.H. Fleury. Application of the Evidence Procedure to the estimation of wireless channels. Submitted for publication in EURASIP journal on Applied Signal Processing.

[SÖH+02] M. Steinbauer, H. Özcelik, H. Hofstetter, C.F. Mecklenbräuker, and E. Bonek. How to quantify multipath seperation. *IEICE Transactions on Electronics, Special Issue on Signals,Systems and Electronics Technology*, March 2002.

[Tah02] H.A. Taha. *Operations Research*. Prentice Hall International, 2002.

[TGS05] J.A. Tropp, A.C. Gilbert, and M.J. Strauss. Simultaneous sparse approximation via greedy pursuit. In *Proceedings of IEEE International Conference on the Acoustics, Speech, and Signal Processing*, volume 5, pages 721–724, March 2005.

[THR+00] R. Thomä, D. Hampicke, A. Richter, G. Sommerkorn, A. Schneider, U. Trautwein, and W. Wirnitzer. Identification of time-variant directional mobile radio channels. *IEEE Trans. on Instrumen. and Meas.*, 49:2:357–364, Apr. 2000.

[Tip01] Michael Tipping. Sparse Bayesian learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, June 2001.

[TSA98] V. Tarokh, N. Seshadri, and A.R.Calderbank. Space-time codes for high data rate wireless communication: Performance criterion and code construction. *IEEE Transactions on Information Theory*, 44(2):744 – 765, March 1998.

[VTR00] R. Vaughan, P. Teal, and R. Raich. Short-term mobile channel prediction using discrete scatterer propagation model and subspace signal processing algorithms. In *52nd IEEE Conf. on Vehic. Tech.*, volume 2, pages 751 – 758, 2000.

[WK85] M. Wax and T. Kailath. Detection of signals by information theoretic criteria. *IEEE Transactions on Acoustics, Speech, and Signal Processing* ,, Volume: 33(2):387– 392, April 1985.

[ZAB99] M. Zeng, A. Annamalai, and V.K. Bhargava. Recent advances in cellular wireless communications. *IEEE Comm. Magazine*, 37(9):128 – 138, 1999.