Technische Universität München

TUM School of Computation, Information and Technology

Asymptotically Optimal Channel Estimation and Learning Suitable Compressive Sensing Matrices

Michael Koller

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

genehmigten Dissertation.

Vorsitz:		Prof. Dr. Reinhard Heckel
Prüfende der Dissertation:	1.	Prof. Dr.-Ing. Wolfgang Utschick
	2.	Prof. Dr.-Ing. Robert Schober

Die Dissertation wurde am 09.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 22.11.2022 angenommen.

Abstract

We study a sequence of conditional mean estimators (CMEs) and a method to learn an observation matrix for compressive sensing purposes. In both cases, we aim to recover a channel vector from noisy linear observations. The channel is assumed to be distributed according to an unknown probability density function (PDF) and we assume we have access to a training data set of channel samples.

First, we approximate the unknown channel PDF by means of a Gaussian mixture model (GMM). This is motivated by universal approximation properties of GMMs according to which for any given continuous PDF, there exists a sequence of GMMs which converges uniformly to it. For each GMM sequence element, a CME for channels distributed according to the GMM sequence element can be analytically calculated. Hence, the sequence of GMMs implies a sequence of CMEs. Assuming an invertible observation matrix, we show the pointwise convergence of the sequence of CMEs to the optimal CME which corresponds to the unknown channel PDF. However, since the channel PDF is unknown, this optimal CME cannot be computed, which makes one of the GMM-based CMEs an attractive approximation thereof. The pointwise convergence result holds more generally: Whenever a sequence of PDFs converges uniformly to the channel PDF, the corresponding sequence of CMEs converges pointwise to the optimal CME. We study the case of noninvertible observation matrices in numerical experiments and observe a convergent behavior.

Second, the goal is to learn an observation matrix with the restricted isometry property (RIP). To this end, we interpret a matrix with the RIP as a matrix which maps vectors of interest from a high-dimensional unit hypersphere to a low-dimensional unit hypersphere. In addition to this property, we argue that the vectors on the low-dimensional hypersphere should be uniformly distributed to combat noise. We formulate the goal to obtain such a matrix as the learning problem of matching the distribution of the channel in the range space of the matrix to a uniform distribution on the low-dimensional hypersphere. The distance between the distributions is measured by means of a maximum mean discrepancy (MMD) metric. Lastly, we observe that the GMM-based CME can be used in conjunction with the learned observation matrices.

Contents

Acronyms

AMP	approximate message passing
CAE	concrete autoencoder
CME	conditional mean estimator
CNN	convolutional neural network
DFT	discrete Fourier transform
EM	expectation-maximization
GAN	generative adversarial network
GMM	Gaussian mixture model
LBCS	learning-based compressive subsampling
LMMSE	linear minimum mean square error
LOS	line of sight
LS	least squares
MIMO	multiple-input multiple-output
MMD	maximum mean discrepancy
MSE	mean square error
NLOS	non-line of sight
O2I	outdoor-to-indoor
OMP	orthogonal matching pursuit
PDF	probability density function
PGDM	projected gradient descent method

RIP	restricted isometry property
SIMO	single-input multiple-output
SISO	single-input single-output
SNR	signal-to-noise ratio
ULA	uniform linear array

1 Introduction

In this dissertation, we are interested in the equation

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{n}$$

where $\boldsymbol{A}$ is a matrix and the remaining quantities are vectors. In the studied setting, the vector $\boldsymbol{y}$ is given and $\boldsymbol{n}$ represents an unknown noise realization. The noise is modeled by means of a known Gaussian probability density function (PDF). The goal is to estimate the vector $\boldsymbol{h}$ and we measure the estimation error in terms of the mean square error (MSE). The vector $\boldsymbol{h}$ is a realization of a random variable with a PDF $f_{\boldsymbol{h}}$ which is continuous. The PDF $f_{\boldsymbol{h}}$ is not assumed to be known but we assume a (training) data set $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$ of realizations $\boldsymbol{h}_t$ to be given.

In this context, we discuss the following two main topics:

1. We propose an algorithm to estimate $\boldsymbol{h}$ and study its optimality (Chapter 3).
2. We propose an algorithm to obtain a matrix $\boldsymbol{A}$ suitable for the problem of estimating $\boldsymbol{h}$ (Chapter 4).

Since we work with a given data set, both topics make use of machine learning methods. The main tools used throughout the dissertation are introduced in Chapter 2. Chapter 5 presents a numerical simulation where the two proposed algorithms are combined. Lastly, Chapter 6 concludes the dissertation with an outlook.

1.1 Asymptotically Optimal Estimation

The first main topic (Chapter 3) is an algorithm to estimate $\boldsymbol{h}$. We use Gaussian mixture models (GMMs) to estimate the PDF $f_{\boldsymbol{h}}$ corresponding to $\boldsymbol{h}$. Every GMM is a PDF $f^{(K)}$ which consists of $K \in \mathbb{N}$ components. GMMs are universal approximators and can therefore approximate any continuous PDF arbitrarily well [1]. More precisely, in theory, there always exists a sequence $(f^{(K)})_{K=1}^{\infty}$ of GMMs which converges uniformly to the PDF $f_{\boldsymbol{h}}$. In practice, we take this as a motivation to use an expectation-maximization (EM) algorithm together with the given data set to obtain a K-component

GMM $f^{(K)}$ that approximates $f_{\boldsymbol{h}}$. If the approximation is perfect, i.e., if $f^{(K)} = f_{\boldsymbol{h}}$ holds, then the conditional mean estimator (CME), which achieves the minimum MSE, for $\boldsymbol{h}$ can be computed in closed form. If the approximation is not perfect, we still compute a closed-form estimator based on $f^{(K)}$ and we can observe that this GMM-based estimator converges to the CME for $K \to \infty$. We prove this result for the case where the matrix $\boldsymbol{A}$ is invertible. The noninvertible case is evaluated in numerical experiments and also shows a convergent behavior.

1.2 Learning a Compressive Sensing Matrix

The second main topic (Chapter 4) is an algorithm to design a matrix $\boldsymbol{A}$ for the problem of estimating $\boldsymbol{h}$. Here, we are mainly interested in underdetermined systems where $\boldsymbol{A}$ is a wide matrix that maps from an N-dimensional space to an m-dimensional space with $m < N$. Compressive sensing theory studies the problem of estimating $\boldsymbol{h}$ in this setting when $\boldsymbol{h}$ lies in a suitable form of a low-dimensional subspace. In particular, compressive sensing theory provides properties which, if a matrix $\boldsymbol{A}$ fulfills them, guarantee good estimates. One of these properties is the restricted isometry property (RIP). The RIP comes along with a restricted isometry constant which is a nonnegative number where a smaller number indicates a better matrix. We interpret a matrix $\boldsymbol{A}$ with the RIP as a function which maps normalized vectors $\boldsymbol{h}/\|\boldsymbol{h}\|$ into a thin layer around the m-dimensional unit hypersphere. The layer's thickness determines the restricted isometry constant so that in the best case, all $\boldsymbol{A}\boldsymbol{h}/\|\boldsymbol{h}\|$ lie exactly on the m-dimensional unit hypersphere (restricted isometry constant equal to zero). In addition to requiring this hypersphere mapping property, we argue that it is desireable to distribute the points $\boldsymbol{A}\boldsymbol{h}/\|\boldsymbol{h}\|$ isotropically around the origin to combat the influence of the noise. Assuming that all $\boldsymbol{h}$ are realizations of a random variable $\boldsymbol{h}$, in the best case, the random variable $\boldsymbol{A}\boldsymbol{h}/\|\boldsymbol{h}\|$ is thus uniformly distributed on the m-dimensional unit hypersphere. This leads us to propose an optimization problem which aims to determine the matrix $\boldsymbol{A}$ such that the distance between the distribution of $\boldsymbol{A}\boldsymbol{h}/\|\boldsymbol{h}\|$ and the uniform distribution on the m-dimensional unit hypersphere is minimized. We use a kernel-based maximum mean discrepancy (MMD) [2] to measure the distance between the two distributions and to formulate the distribution matching optimization problem as a learning problem. Finally, we incorporate the constraint that all entries of the matrix $\boldsymbol{A}$ should have the same modulus into the MMD minimization problem. This constant modulus constraint on $\boldsymbol{A}$ is motivated by one of the applications considered in the dissertation.

1.3 Motivation and Application

The main motivation for studying the described problems and the application presented in the numerical experiments is channel estimation [3]. In this context, the vector $\boldsymbol{y}$ is called observation and the goal is to estimate the channel $\boldsymbol{h}$. The observation corresponds, e.g., to the receive signal at a base station with a certain number of antennas. The matrix $\boldsymbol{A}$ is called observation matrix and it can, for example, represent (the positions of) pilot symbols. It can also represent analog phase shifters which can, for example, be used in order to connect a large number of antennas to a smaller number of receiver chains. In this case, the matrix $\boldsymbol{A}$ is a wide matrix where the number of rows corresponds to the number of receiver chains and the number of columns corresponds to the number of antennas. All elements of a phase shifters matrix have the same constant modulus. This case is the main motivation for designing such a matrix in Chapter 4.

2 Preliminaries

The notation and concepts introduced here are used in the main chapters.

2.1 Notation

We write $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C})$ for the probability density function (PDF) of a real-valued Gaussian random variable with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{C}$. We write $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{C})$ to evaluate this density at a suitable vector $\boldsymbol{x}$:

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{C}) = \frac{1}{\sqrt{\det(2\pi \boldsymbol{C})}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right). \tag{2.1}$$

Analogously, we use $\mathcal{N}_{\mathbb{C}}$ to denote the PDF of a complex-valued circularly-symmetric Gaussian random variable. Generally, we write $\boldsymbol{z} \sim p$ to indicate that $\boldsymbol{z}$ is a random variable with PDF p.

The supremum norm of a continuous function $f : \mathbb{K}^N \to \mathbb{K}$ is $\|f\|_\infty = \sup_{\boldsymbol{x} \in \mathbb{K}^N} |f(\boldsymbol{x})|$ with $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. A sequence $(f^{(K)})_{K=1}^{\infty}$ of continuous functions $f^{(K)} : \mathbb{K}^N \to \mathbb{K}$ is said to converge uniformly to $f : \mathbb{K}^N \to \mathbb{K}$ if

$$\lim_{K \to \infty} \|f - f^{(K)}\|_\infty = 0 \tag{2.2}$$

holds. For a vector $\boldsymbol{d} \in \mathbb{K}^N$, $\mathrm{diag}(\boldsymbol{d}) \in \mathbb{K}^{N \times N}$ is the $N \times N$ matrix with $\boldsymbol{d}$ on its main diagonal and zeros elsewhere. For a matrix $\boldsymbol{D} \in \mathbb{K}^{N \times N}$, $\mathrm{diag}(\boldsymbol{D}) \in \mathbb{K}^N$ is the vector on its main diagonal. Further, $\|\cdot\|$ denotes the Euclidean norm, $\boldsymbol{0} \in \mathbb{K}^N$ or $\boldsymbol{0}_N$ denotes the zero vector in $\mathbb{K}^N$, $\boldsymbol{I} \in \mathbb{K}^{N \times N}$ or $\boldsymbol{I}_N$ is the identity matrix in $\mathbb{K}^{N \times N}$, $\boldsymbol{A}^{\mathrm{H}}$ denotes the adjoint (conjugate transpose) of a matrix $\boldsymbol{A}$, and $\boldsymbol{A} \otimes \boldsymbol{B}$ denotes the Kronecker product of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Lastly, $\mathrm{vec}(\boldsymbol{A}) \in \mathbb{K}^{mN}$ denotes stacking the columns of a matrix $\boldsymbol{A} \in \mathbb{K}^{m \times N}$ into a long vector, and j is the imaginary unit.

2.2 Signal Model

The signal model considered throughout the dissertation is

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{n} \tag{2.3}$$

where $\boldsymbol{y} \in \mathbb{C}^m$ is called the *observation*, $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ is called the *observation matrix*, $\boldsymbol{h} \in \mathbb{C}^N$ is called the *channel*, and $\boldsymbol{n} \in \mathbb{C}^m$ is called the *noise*. The noise is assumed to be a realization of a Gaussian random variable $\boldsymbol{n} \sim \mathcal{N}_{\mathbb{C}}(\boldsymbol{0}, \boldsymbol{\Sigma})$ with mean vector $\boldsymbol{0} \in \mathbb{C}^m$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{C}^{m \times m}$.

One topic of interest in the context of the signal model (2.3) is to design a channel estimation algorithm. There, we assume that the observation $\boldsymbol{y}$, the observation matrix $\boldsymbol{A}$, and the noise PDF are given, and the goal is to obtain an estimate of the channel $\boldsymbol{h}$. We propose such an algorithm in Chapter 3. Another topic of interest is to design the observation matrix $\boldsymbol{A}$. There, one goal could be to design $\boldsymbol{A}$ such that it harmonizes well with a certain channel estimation algorithm. A particular challenge when designing $\boldsymbol{A}$ can be to incorporate structural matrix constraints. In Chapter 4, we propose an algorithm to find an observation matrix which can harmonize with compressive sensing channel estimation algorithms and which is constrained to have constant modulus entries. In contrast to this, the standard procedure in compressive sensing theory is to draw $\boldsymbol{A}$ randomly, see Section 2.3 for details. Advantages of designed matrices can be a reduced computational complexity of the corresponding channel estimation algorithms, an improved estimation performance as compared to random matrices, or better compatibility for analog implementations.

2.2.1 Special Cases

The receive signal

$$\boldsymbol{y} = \boldsymbol{h} + \boldsymbol{n} \tag{2.4}$$

corresponding to a single-antenna mobile terminal which transmits a pilot to an N-antenna base station is a special case of the signal model in (2.3), see, e.g., [3, 4]. This case is interesting for us because potential channel estimation algorithms can be studied without having to consider the difficulty of finding a suitable observation matrix. Another reason is that the observation matrix (the identity matrix) is invertible which addresses a special case considered in Chapter 3.

The multiple-input multiple-output (MIMO) signal model

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{P} + \boldsymbol{N} \tag{2.5}$$

with the channel matrix $\boldsymbol{H} \in \mathbb{C}^{N_{\text{rx}} \times N_{\text{tx}}}$, the pilot matrix $\boldsymbol{P} \in \mathbb{C}^{N_{\text{tx}} \times N_p}$, and the noise matrix $\boldsymbol{N} \in \mathbb{C}^{N_{\text{rx}} \times N_p}$, where an N_{tx}-antenna mobile terminal transmits N_p pilots to an N_{rx}-antenna base station (see, e.g., [4]), is another special case of (2.3). This can be seen using the definitions $\boldsymbol{h} = \operatorname{vec}(\boldsymbol{H})$, $\boldsymbol{y} = \operatorname{vec}(\boldsymbol{Y})$, $\boldsymbol{n} = \operatorname{vec}(\boldsymbol{N})$, and $\boldsymbol{A} = \boldsymbol{P}^{\mathrm{T}} \otimes \boldsymbol{I}_{N_{\text{rx}}}$ such that we have $m = N_{\text{rx}} N_p$ and $N = N_{\text{rx}} N_{\text{tx}}$.

Another special case is the single-input single-output (SISO) transmission over a frequency-selective fading channel $\boldsymbol{H} \in \mathbb{C}^{N_c \times N_t}$ where N_c is the number of subcarriers and N_t is the number of time slots. If only N_p of the $N_c N_t$ resource elements are occupied with pilots, then these can be selected via a matrix $\boldsymbol{A} \in \{0,1\}^{N_p \times N_c N_t}$ which represents the pilot positions. Defining $\boldsymbol{h} = \mathrm{vec}(\boldsymbol{H}) \in \mathbb{C}^{N_c N_t}$, the observed signal is given by (2.3) with $m = N_p$ and $N = N_c N_t$.

The cases mentioned so far are studied in Chapter 3. The next case is what we are concerned with in Chapter 4. There, we consider the setting where a single-antenna mobile terminal transmits a pilot to an N-antenna base station which does not have a dedicated receiver chain for every antenna. Instead, the N antennas are connected to $m < N$ receiver chains via an analog mixing network which is represented by a matrix $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ such that $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{n}$ is the base station's receive signal, see, e.g., [5–9]. The analog mixing network is assumed to consist of phase shifters so that every entry of $\boldsymbol{A}$ has the form

$$[\boldsymbol{A}]_{k,l} = \frac{1}{\sqrt{m}} \exp(\mathrm{j}\, \phi_{k,l}) \tag{2.6}$$

with a phase $\phi_{k,l} \in \mathbb{R}$. We refer to such a matrix as a *constant modulus matrix* because all entries have a constant modulus constraint: $|[\boldsymbol{A}]_{k,l}| = \frac{1}{\sqrt{m}}$. Such a setting is often approached from a compressive sensing perspective.

2.3 Compressive Sensing

One part of compressive sensing theory studies the problem of recovering the channel $\boldsymbol{h}$ from an observation of the form (2.3). There exist recovery guarantees if the matrix has the restricted isometry property (RIP) [10]. Assuming the channel lies in an (infinite) union of subspaces or some abstract subset $\mathcal{H} \subset \mathbb{C}^N$ (see, e.g., [11–13]), the observation matrix $\boldsymbol{A}$ is said to have the RIP if the restricted isometry constant $\delta \geq 0$ is small in

$$(1-\delta)\|\boldsymbol{h}\|^2 \leq \|\boldsymbol{A}\boldsymbol{h}\|^2 \leq (1+\delta)\|\boldsymbol{h}\|^2 \tag{2.7}$$

for all $\boldsymbol{h} \in \mathcal{H}$.

In the most prominent case, $\mathcal{H}$ is the set of all p-sparse vectors $\boldsymbol{s} \in \mathbb{C}^N$ (p nonzero entries). If $\boldsymbol{h}$ is not sparse itself, it may be sparsely represented in a basis $\boldsymbol{\Psi} \in \mathbb{C}^{N \times N}$ such that $\boldsymbol{h} = \boldsymbol{\Psi}\boldsymbol{s}$ holds. Writing

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{\Psi}\boldsymbol{s} + \boldsymbol{n} \tag{2.8}$$

Algorithm 1 Orthogonal Matching Pursuit (OMP) [10]

Require: matrix $\boldsymbol{C} \in \mathbb{C}^{m \times L}$, observation $\boldsymbol{y}$, sparsity p

$S^{(0)} \leftarrow \emptyset$, $\boldsymbol{s}^{(0)} \leftarrow \boldsymbol{0}$

for $i = 1$ to p **do**

  $j^* \leftarrow \arg\max_{j \in \{1,\dots,L\}} \{[\boldsymbol{C}^{\mathrm{H}}(\boldsymbol{y} - \boldsymbol{C}\boldsymbol{s}^{(i-1)})]_j\}$

  $S^{(i)} \leftarrow S^{(i-1)} \cup \{j^*\}$

  $\boldsymbol{s}^{(i)} \leftarrow \arg\min_{\tilde{\boldsymbol{s}}} \{\|\boldsymbol{y} - \boldsymbol{C}\tilde{\boldsymbol{s}}\|, \operatorname{support}(\tilde{\boldsymbol{s}}) \subset S^{(i)}\}$

end for

return $\boldsymbol{s}^{(p)}$ *// p-sparse vector*

the greedy recovery algorithm orthogonal matching pursuit (OMP) [14–16] can be used to obtain an estimate $\hat{\boldsymbol{s}}$ of $\boldsymbol{s}$ and the channel is then estimated as $\hat{\boldsymbol{h}} = \boldsymbol{\Psi}\hat{\boldsymbol{s}}$. OMP is summarized in Algorithm 1 and would be used with $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{\Psi}$ here.

Random matrices provide a well-known source for matrices with the RIP. For example, a Gaussian random matrix where the entries are drawn independently with $[\boldsymbol{A}]_{k,l} \sim \mathcal{N}_{\mathbb{C}}(0, \frac{1}{\sqrt{m}})$ is known to have the RIP with high probability [10]. Furthermore, bounded random variables are sub-Gaussian random variables [10] and because sub-Gaussian random matrices have the RIP with high probability [10], there also exist constant modulus matrices with the RIP. To this end, the entries of $\boldsymbol{A}$ are drawn independently and have the form

$$\frac{1}{\sqrt{m}} \exp(\mathrm{j}\,\phi) \quad \text{with} \quad \phi \sim \mathcal{U}([0, 2\pi]) \tag{2.9}$$

where ϕ is uniformly distributed in the interval $[0, 2\pi]$.

2.4 Gaussian Mixture Models

A Gaussian mixture model (GMM) is a PDF of the form [17]

$$f^{(K)} : \mathbb{R}^N \to \mathbb{R},\ \boldsymbol{x} \mapsto f^{(K)}(\boldsymbol{x}) = \sum_{k=1}^{K} p(k)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{C}_k). \tag{2.10}$$

The summands represent the K GMM *components*. Each component consists of a *mixing coefficient* $p(k)$ with $\sum_{k=1}^{K} p(k) = 1$ and of a Gaussian PDF with mean vector $\boldsymbol{\mu}_k \in \mathbb{R}^N$ and covariance matrix $\boldsymbol{C}_k \in \mathbb{R}^{N \times N}$. Given a realization $\boldsymbol{x} \in \mathbb{R}^N$, the probability $p(k \mid \boldsymbol{x})$ of it stemming from component k can be computed [17]:

$$p(k \mid \boldsymbol{x}) = \frac{p(k)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{C}_k)}{\sum_{i=1}^{K} p(i)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i)}. \tag{2.11}$$

Being able to calculate these *responsibilities* (2.11) plays an important role in Chapter 3.

In Chapter 3, we are also interested in approximating an unknown continuous PDF f via a K-component GMM. To this end, all mixing coefficients $\{p(k)\}_{k=1}^{K}$, all mean vectors $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$, and all covariance matrices $\{\boldsymbol{C}_k\}_{k=1}^{K}$ of the K-component GMM need to be determined. Maximum likelihood estimates of all theses parameters can be obtained using an expectation-maximization (EM) algorithm and a data set $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$ of realizations $\boldsymbol{h}_t$ of a random variable $\boldsymbol{h} \sim f$. Such a data set is assumed to be given. Details about the GMM fitting process can be found, e.g., in [17]. A list of convergence properties of the EM algorithm can be found, e.g., in [18].

The motivation for approximating the unknown PDF f via a GMM comes from [1, Theorem 5] according to which GMMs are able to approximate any continuous PDF arbitrarily well. We write

$$\mathcal{C} = \left\{ f : \mathbb{R}^N \to \mathbb{R} : f \geq 0, \int f(\boldsymbol{x}) d\boldsymbol{x} = 1, f \text{ is continuous} \right\} \tag{2.12}$$

for the set of all continuous PDFs. Then, one part of [1, Theorem 5] says that for any $f \in \mathcal{C}$, there exists a sequence of GMMs which converges uniformly to f:

Theorem 1 (Universal Approximation Property)**.** *Denoting the standard Gaussian density by $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, let*

$$\mathcal{M}_K = \left\{ h : h(\boldsymbol{x}) = \sum_{k=1}^{K} c_k \frac{1}{\sigma_k^N} \mathcal{N}\left(\frac{\boldsymbol{x} - \boldsymbol{\mu}_k}{\sigma_k}; \mathbf{0}, \boldsymbol{I} \right) \right\}$$

be the class of K-component location-scale finite Gaussian mixtures with

$$\boldsymbol{\mu}_k \in \mathbb{R}^N, \sigma_k > 0, c_k \geq 0 \quad \textit{for all } k \in \{1, \ldots, K\}$$

and $\sum_{k=1}^{K} c_k = 1$. For any $f \in \mathcal{C}$, there exists a sequence $(f^{(K)})_{K=1}^{\infty}$ with $f^{(K)} \in \mathcal{M}_K$ which converges uniformly to f.

2.5 Maximum Mean Discrepancy

Maximum mean discrepancy (MMD) can be used to compute a distance between probability distributions and is used in Chapter 4. If $\mathcal{X}$ is a metric space and $\boldsymbol{x}$ and $\boldsymbol{y}$ are two random variables with respective probability distributions p and q defined on $\mathcal{X}$, MMD quite literally computes a maximization of the discrepancy between two means [2, Definition 2]:

$$\mathrm{MMD}_{\mathcal{F}}(p, q) = \sup_{f \in \mathcal{F}} \left(\mathrm{E}_{\boldsymbol{x} \sim p}[f(\boldsymbol{x})] - \mathrm{E}_{\boldsymbol{y} \sim q}[f(\boldsymbol{y})] \right). \tag{2.13}$$

Here, $\mathcal{F}$ is a suitably chosen set of functions $f : \mathcal{X} \to \mathbb{R}$. Depending on $\mathcal{F}$, (2.13) can be a metric and then we have $\mathrm{MMD}_{\mathcal{F}}(p, q) = 0$ if and only if $p = q$, and $\mathrm{MMD}_{\mathcal{F}}(p, q) > 0$ otherwise. An overview of suitable function sets $\mathcal{F}$ can be found, e.g., in [19].

Solving the optimization problem (2.13) can be challenging, depending on $\mathcal{F}$. One particularly interesting set $\mathcal{F}$ in this regard is the unit ball in a reproducing kernel Hilbert space. A brief introduction to reproducing kernel Hilbert spaces can be found in [2]. Every such Hilbert space is associated with a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Conversely, every kernel defines an associated Hilbert space. Intuitively speaking, a kernel k can represent all functions which are an element of the reproducing kernel Hilbert space. For this reason, the optimization with respect to the whole set $\mathcal{F}$ in (2.13) can be expressed using the kernel only and the computation of the supremum is avoided.

Using a suitable kernel k, (2.13) can be expressed as [2, Lemma 6]

$$\mathrm{MMD}_k^2(p, q) = \mathrm{E}_{\boldsymbol{x},\boldsymbol{x}'}[k(\boldsymbol{x}, \boldsymbol{x}')] - 2\,\mathrm{E}_{\boldsymbol{x},\boldsymbol{y}}[k(\boldsymbol{x}, \boldsymbol{y})] + \mathrm{E}_{\boldsymbol{y},\boldsymbol{y}'}[k(\boldsymbol{y}, \boldsymbol{y}')]. \tag{2.14}$$

Here, $\boldsymbol{x}$ and $\boldsymbol{x}'$ are independent random variables with distribution p, and $\boldsymbol{y}$ and $\boldsymbol{y}'$ are independent random variables with distribution q. Among the best known kernels is the Gaussian kernel:

$$k_\sigma : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R},\ (\boldsymbol{x}, \boldsymbol{y}) \mapsto k_\sigma(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{y}\|^2} \tag{2.15}$$

with a parameter $\sigma > 0$. The Gaussian kernel is a typical choice when MMD_k is used, as, e.g., [20–22] demonstrate. One reason is that the associated reproducing kernel Hilbert space enjoys the property of being universal which makes (2.14) a metric [2, Theorem 5].

Compared to (2.13), computing (2.14) no longer involves solving an optimization problem. It is however still necessary to know both p and q. In some applications (in particular in Chapter 4), only realizations $\{\boldsymbol{x}_i\}_{i=1}^M$ and $\{\boldsymbol{y}_j\}_{j=1}^N$ of $\boldsymbol{x} \sim p$ and $\boldsymbol{y} \sim q$, respectively, are given and we are still interested in determining whether $p = q$ or whether p and q are at least close to each other. In this situation, a biased estimate of (2.14) can be computed [2, Equation (6)]:

$$\begin{aligned}\mathrm{MMD}_k^2\left(\{\boldsymbol{x}_i\}_{i=1}^M, \{\boldsymbol{y}_j\}_{j=1}^N\right) = \\ \sum_{i=1}^M \sum_{j=1}^M \frac{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{M^2} - 2\sum_{i=1}^M \sum_{j=1}^N \frac{k(\boldsymbol{x}_i, \boldsymbol{y}_j)}{MN} + \sum_{i=1}^N \sum_{j=1}^N \frac{k(\boldsymbol{y}_i, \boldsymbol{y}_j)}{N^2}.\end{aligned} \tag{2.16}$$

The smaller the value of $\mathrm{MMD}_k^2(\{\boldsymbol{x}_i\}_{i=1}^M, \{\boldsymbol{y}_j\}_{j=1}^N)$, the closer p and q are to each other. One main result of [2] is that (2.16) converges in probability to (2.14) at a rate of $\mathcal{O}((M+N)^{-\frac{1}{2}})$.

2.6 Distribution Matching

The biased MMD estimate (2.16) is differentiable because the Gaussian kernel (2.15) is differentiable. In the literature, this fact is utilized in the context of machine learning applications, e.g., [20–23]. One possible application, which is related to the considerations in Chapter 4, can be described as follows. A set $\{\boldsymbol{x}_i\}_{i=1}^M$ of natural images $\boldsymbol{x}_i$ is given and it is assumed that they are all samples of a random variable $\boldsymbol{x}$ with distribution p. The goal is to draw new samples of $\boldsymbol{x}$, or, in other words, to generate new natural images. To this end, a set $\{g_\phi\}_{\phi\in\Phi}$ of functions g_ϕ is defined where every g_ϕ is differentiable with respect to the parameter ϕ and where Φ is a constraint set for the parameter. For instance, every g_ϕ could be a neural network where ϕ collects all learnable parameters (all weights and biases). Then, an auxiliary latent random variable $\boldsymbol{y}'$ with the distribution q' is chosen (e.g., a standard Gaussian random variable) and corresponding random variables $g_\phi(\boldsymbol{y}')$ with respective distributions q_ϕ are defined. Typically, $\boldsymbol{y}'$ is chosen such that sampling this random variable is simple. The goal is to find $\phi^\star \in \Phi$ such that the distribution $q_{\phi^\star}$ is close (ideally equal) to p. This amounts to solving an optimization problem:

$$\mathrm{MMD}_k^2(p, q_{\phi^\star}) = \min_{\phi\in\Phi} \mathrm{MMD}_k^2(p, q_\phi). \tag{2.17}$$

New images are now generated by first drawing samples $\boldsymbol{y}'_1, \boldsymbol{y}'_2, \ldots$ of $\boldsymbol{y}'$, which is simple, and computing the images $g_{\phi^\star}(\boldsymbol{y}'_1), g_{\phi^\star}(\boldsymbol{y}'_2), \ldots$, which are then samples of the distribution $q_{\phi^\star}$.

Since p is not known and q_ϕ is generally difficult to compute, (2.17) cannot be solved directly. However, if samples (training data) $\{\boldsymbol{x}_i\}_{i=1}^M$ and $\{\boldsymbol{y}'_j\}_{j=1}^N$ are given, (2.16) can be used to define an alternative optimization problem:

$$\min_{\phi\in\Phi} \mathrm{MMD}_k^2\left(\{\boldsymbol{x}_i\}_{i=1}^M, \{g_\phi(\boldsymbol{y}'_j)\}_{j=1}^N\right). \tag{2.18}$$

A problem of this form is considered in Chapter 4. Such optimization problems are solved via stochastic gradient descent, e.g., [20–23], because the objective function is differentiable, and are considered easy to optimize [23]. Convergence guarantees are part of ongoing research. Solving an optimization problem via stochastic gradient descent often involves some form of a grid or a random search. This is discussed next.

2.7 Random Search

For the following explanation of the grid and the random search procedures [24], we assume we want to train a classical feed-forward neural network using a stochastic gradient descent algorithm. We distinguish between the optimization parameters of the neural network and the hyperparameters of the neural network and of the stochastic gradient descent algorithm. The values of the optimization parameters are derived during the training (or learning) process whereas the hyperparameters are set before the training process begins. Weights and biases constitute the neural network's optimization parameters. Examples of hyperparameters are the number of hidden layers of the neural network, the stochastic gradient descent algorithm's learning rate (i.e., the gradient step size), or the batch size (i.e., the number of samples used to compute the stochastic gradient). The optimization parameters are typically randomly initialized (e.g., from a uniform or a normal distribution) and the hyperparameters are typically determined via a grid or a random search procedure [24].

To perform a grid search, we determine sets of possible hyperparameter values, for example, $P_{\text{hl}} = \{5, 10, 25, 50\}$ for the number of hidden layers, $P_{\text{lr}} = \{10^{-5}, 10^{-3}, 10^{-1}\}$ for the learning rate, and $P_{\text{bs}} = \{10, 50, 100, 250, 500\}$ for the batch size. Then, we train the neural network once for every combination of the hyperparameter values, i.e., for every element of the Cartesian product $P_{\text{hl}} \times P_{\text{lr}} \times P_{\text{bs}}$. In this example, we would therefore train the neural network $4 \cdot 3 \cdot 5 = 60$ times. In the end, a held out validation data set is used to choose the best of the 60 trained neural networks.

The authors of [24] argue that the hyperparameters space can be explored more efficiently if a random search procedure is employed. To this end, we would also train the neural network, e.g., 60 times but this time the hyperparameter values would be randomly drawn for every neural network. If $\text{uniform}([a, b])$ denotes drawing a sample from a continuous uniform distribution on the interval $[a, b]$, a hyperparameter value can be determined via

$$e^{\text{uniform}([\ln(a),\ln(b)])} \tag{2.19}$$

followed by rounding to the nearest integer if necessary [24]. Using the numerical example for the learning rate from before, we would, e.g., set $a = 10^{-5}$ and $b = 10^{-1}$ to determine the learning rate as $e^{\text{uniform}([\ln(10^{-5}),\ln(10^{-1})])}$ for a given neural network. Here, too, a held out validation data set is used to choose the best of the 60 trained neural networks.

3 Asymptotically Optimal Channel Estimation

This chapter is mainly based on our work in [25, 26]. In this chapter, we distinguish between a random variable $\boldsymbol{x}$ and a realization $\boldsymbol{x}$ thereof.

3.1 A Sequence of Conditional Mean Estimators

We write $f_{\boldsymbol{h}}$ for the probability density function (PDF) of the channel random variable $\boldsymbol{h}$. Further, we write $f_{\boldsymbol{n}} = \mathcal{N}_{\mathbb{C}}(\mathbf{0}, \boldsymbol{\Sigma})$ for the noise PDF and $f_{\boldsymbol{y}}$ for the observation PDF in the model

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{n} \tag{3.1}$$

with $\boldsymbol{A} \in \mathbb{C}^{m \times N}$.

A good channel estimator is a function $\tilde{\boldsymbol{h}} : \mathbb{C}^m \to \mathbb{C}^N$ which leads to a small mean square error (MSE)

$$\mathrm{E}\left[\left\|\boldsymbol{h} - \tilde{\boldsymbol{h}}(\boldsymbol{y})\right\|^2\right]. \tag{3.2}$$

It is known that a minimizer is given by the conditional mean estimator (CME) [27]

$$\hat{\boldsymbol{h}} : \mathbb{C}^m \to \mathbb{C}^N,\ \boldsymbol{y} \mapsto \hat{\boldsymbol{h}}(\boldsymbol{y}) = \mathrm{E}\left[\boldsymbol{h} \mid \boldsymbol{y} = \boldsymbol{y}\right] = \int \boldsymbol{h} f_{\boldsymbol{h}|\boldsymbol{y}}(\boldsymbol{h}|\boldsymbol{y}) d\boldsymbol{h} \tag{3.3}$$

where $f_{\boldsymbol{h}|\boldsymbol{y}}(\boldsymbol{h} \mid \boldsymbol{y})$ denotes the conditional PDF of $\boldsymbol{h}$ given $\boldsymbol{y} = \boldsymbol{y}$ evaluated at $\boldsymbol{h}$. For our purposes, we first rewrite the conditional PDF via

$$f_{\boldsymbol{h}|\boldsymbol{y}}(\boldsymbol{h} \mid \boldsymbol{y}) = \frac{f_{\boldsymbol{y}|\boldsymbol{h}}(\boldsymbol{y} \mid \boldsymbol{h}) f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} = \frac{f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} \tag{3.4}$$

and the CME is then given by

$$\hat{\boldsymbol{h}} : \boldsymbol{y} \mapsto \hat{\boldsymbol{h}}(\boldsymbol{y}) = \mathrm{E}[\boldsymbol{h} \mid \boldsymbol{y} = \boldsymbol{y}] = \int \boldsymbol{h} \frac{f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} d\boldsymbol{h}. \tag{3.5}$$

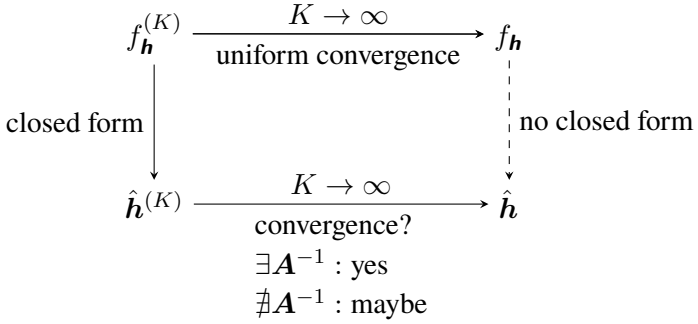

Figure 3.1: In Chapter 3, we discuss the following question: If a sequence $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ of probability density functions converges uniformly to the channel probability density function $f_{\boldsymbol{h}}$, does the corresponding sequence $(\hat{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ of conditional mean estimators converge to the channel conditional mean estimator $\hat{\boldsymbol{h}}$?

Computing (3.5) analytically can be difficult. On top of this, even if all involved PDFs are given analytically, there may not exist an analytic expression of the integral and therefore of the CME $\hat{\boldsymbol{h}}$. Another problem might be that only the noise PDF $f_{\boldsymbol{n}}$—but not $f_{\boldsymbol{h}}$—is known, which is the assumption in this dissertation, and then, computing (3.5) is not possible. However, even if the channel PDF $f_{\boldsymbol{h}}$ is unknown, we can reasonably assume to have access to a sequence $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ of PDFs which converges uniformly to $f_{\boldsymbol{h}}$:

$$\lim_{K\to\infty} \|f_{\boldsymbol{h}} - f_{\boldsymbol{h}}^{(K)}\|_{\infty} = 0. \tag{3.6}$$

A prime example is the case where $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ is a sequence of Gaussian mixture models (GMMs). This case is studied in Section 3.3.

For every $K \in \mathbb{N}$, we now introduce an auxiliary channel random variable $\boldsymbol{h}^{(K)}$ with PDF $f_{\boldsymbol{h}}^{(K)}$, we introduce an auxiliary signal model

$$\boldsymbol{y}^{(K)} = \boldsymbol{A}\boldsymbol{h}^{(K)} + \boldsymbol{n} \tag{3.7}$$

and we write $f_{\boldsymbol{y}}^{(K)}$ for the PDF of $\boldsymbol{y}^{(K)}$. For every $K \in \mathbb{N}$, the CME

$$\hat{\boldsymbol{h}}^{(K)} : \boldsymbol{y} \mapsto \hat{\boldsymbol{h}}^{(K)}(\boldsymbol{y}) = \mathrm{E}[\boldsymbol{h}^{(K)} \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}] = \int \boldsymbol{h} \frac{f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} d\boldsymbol{h} \tag{3.8}$$

is then an optimal channel estimator for the model (3.7). In the case where $f_{\boldsymbol{h}}^{(K)}$ is a GMM, $\hat{\boldsymbol{h}}^{(K)}$ can be calculated analytically (see Section 3.3). Generally, if the sequence

$(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ is chosen such that every CME $\hat{\boldsymbol{h}}^{(K)}$ can be computed analytically, it is interesting to investigate whether $\hat{\boldsymbol{h}}^{(K)}$ can be used as an approximation of the optimal $\hat{\boldsymbol{h}}$, which is not available analytically. That is, we ask whether the uniform convergence of the PDFs in (3.6) implies that the sequence $(\hat{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ converges to $\hat{\boldsymbol{h}}$ in some suitable sense. A main result in this chapter is that we have at least pointwise convergence if the observation matrix $\boldsymbol{A}$ is invertible, see Theorem 2. Figure 3.1 schematically summarizes this result.

3.2 Pointwise Convergence

The PDF of a complex-valued random variable can be written in terms of its real and imaginary parts. For this reason, this section (and only this section) considers real-valued quantities only and the results apply to the setting discussed in Section 3.1 by stacking real and imaginary parts appropriately. The proof of the following theorem can be found in Appendix A. The theorem uses the notation from Section 3.1.

Theorem 2. *Let $\mathcal{C} = \big\{f : \mathbb{R}^N \to \mathbb{R} : f \geq 0, \int f(\boldsymbol{x})d\boldsymbol{x} = 1, f \text{ is continuous}\big\}$ be the set of all continuous PDFs. Let $f_{\boldsymbol{h}} \in \mathcal{C}$ and $f_{\boldsymbol{n}} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \in \mathcal{C}$. Further, let $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ be invertible and let $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ be a sequence of PDFs in $\mathcal{C}$ which converges uniformly to $f_{\boldsymbol{h}}$. For every $K \in \mathbb{N}$, define a function $\hat{\boldsymbol{h}}^{(K)} : \mathbb{R}^N \to \mathbb{R}^N$ via*

$$\hat{\boldsymbol{h}}^{(K)} : \boldsymbol{y} \mapsto \hat{\boldsymbol{h}}^{(K)}(\boldsymbol{y}) = \int \boldsymbol{h} \frac{f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} d\boldsymbol{h}. \tag{3.9}$$

Then, the sequence $(\hat{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ converges pointwise to $\hat{\boldsymbol{h}} : \mathbb{R}^N \to \mathbb{R}^N$ with

$$\hat{\boldsymbol{h}} : \boldsymbol{y} \mapsto \hat{\boldsymbol{h}}(\boldsymbol{y}) = \int \boldsymbol{h} \frac{f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} d\boldsymbol{h}. \tag{3.10}$$

That is, we have

$$\lim_{K\to\infty} \|\hat{\boldsymbol{h}}(\boldsymbol{y}) - \hat{\boldsymbol{h}}^{(K)}(\boldsymbol{y})\| = 0 \tag{3.11}$$

for every $\boldsymbol{y} \in \mathbb{R}^N$. Further, (3.11) holds uniformly for all $\boldsymbol{y} \in \mathcal{B}_r$ with $\mathcal{B}_r = \{\boldsymbol{y} \in \mathbb{R}^N : \|\boldsymbol{y}\| \leq r\}$ for any radius $r > 0$.

3.2.1 Discussion of Theorem 2

As a result of Theorem 2, the channel estimator $\hat{\boldsymbol{h}}^{(K)}$ can be viewed as an approximation of the optimal channel estimator $\hat{\boldsymbol{h}}$ for a $K \in \mathbb{N}$. Let $r > 0$ be given and let $\mathcal{B}_r = \{\boldsymbol{y} \in \mathbb{R}^N : \|\boldsymbol{y}\| \leq r\}$. According to Theorem 2, if we want the approximation

error to be bounded by a given threshold $\varepsilon_{\text{thr}} > 0$, there exists a $K_r \in \mathbb{N}$ which guarantees

$$\|\hat{\boldsymbol{h}}(\boldsymbol{y}) - \hat{\boldsymbol{h}}^{(K)}(\boldsymbol{y})\| \leq \varepsilon_{\text{thr}} \quad \text{for all } \boldsymbol{y} \in \mathcal{B}_r \tag{3.12}$$

if we choose $K \geq K_r$. This does not mean that the approximation error exceeds ε_{thr} for $\boldsymbol{y} \notin \mathcal{B}_r$. Due to (3.11), for any $\boldsymbol{y} \in \mathbb{R}^N$, there exists a $K_{\boldsymbol{y}} \in \mathbb{N}$ such that the approximation error falls below ε_{thr}. However, for some $\boldsymbol{y} \notin \mathcal{B}_r$, the sequence index $K_{\boldsymbol{y}}$ needs to be strictly larger than K_r to achieve the goal. Generally, we can expect the channel estimation performance of $\hat{\boldsymbol{h}}^{(K)}$ to be better for larger K.

The proof of Theorem 2 relies on $\boldsymbol{A}$ being invertible. If this is not the case, two problems immediately arise. These are discussed in detail in Appendix B. One problem is that the uniform convergence of $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ to $f_{\boldsymbol{h}}$ does generally not imply the uniform convergence of $(f_{\boldsymbol{y}}^{(K)})_{K=1}^{\infty}$ to $f_{\boldsymbol{y}}$ if $\boldsymbol{A}$ is not invertible (see Appendix B.2). As a consequence, a crucial step in the proof of Theorem 2 cannot be shown. Nonetheless, numerical studies in Section 3.6 suggest that even if the matrix $\boldsymbol{A}$ is not invertible, the performance of the estimator $\hat{\boldsymbol{h}}^{(K)}$ improves with increasing K.

For a noninvertible $\boldsymbol{A}$, it seems difficult to infer a statement about the convergence of the PDFs $f_{\boldsymbol{y}}^{(K)}$, which correspond to $\boldsymbol{y}^{(K)}$, if only the uniform convergence of the PDFs $f_{\boldsymbol{h}}^{(K)}$, which correspond to $\boldsymbol{h}^{(K)}$, is given. However, we can say something about the convergence of the respective distribution functions. To this end, first observe that the uniform convergence of $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ to $f_{\boldsymbol{h}}$ implies the pointwise convergence as well. By Scheffe's lemma (e.g., [28]), the sequence of random variables $(\boldsymbol{h}^{(K)})_{K=1}^{\infty}$ then converges to the random variable $\boldsymbol{h}$ in distribution, i.e., we have the pointwise convergence of the sequence of distribution functions corresponding to $\boldsymbol{h}^{(K)}$ to the distribution function corresponding to $\boldsymbol{h}$. We denote this by $\boldsymbol{h}^{(K)} \xrightarrow{\text{d}} \boldsymbol{h}$. Since $\boldsymbol{n}$ is independent of $\boldsymbol{h}^{(K)}$ and of $\boldsymbol{h}$, also $(\boldsymbol{h}^{(K)}, \boldsymbol{n}) \xrightarrow{\text{d}} (\boldsymbol{h}, \boldsymbol{n})$ holds. Applying the continuous mapping theorem (e.g., [29]) using the continuous mapping

$$s : \mathbb{R}^{N+m} \to \mathbb{R}^{N+m},\ (\boldsymbol{h}, \boldsymbol{n}) \mapsto (\boldsymbol{h}, \boldsymbol{A}\boldsymbol{h} + \boldsymbol{n}) \tag{3.13}$$

implies $s(\boldsymbol{h}^{(K)}, \boldsymbol{n}) = (\boldsymbol{h}^{(K)}, \boldsymbol{y}^{(K)}) \xrightarrow{\text{d}} s(\boldsymbol{h}, \boldsymbol{n}) = (\boldsymbol{h}, \boldsymbol{y})$. In other words, with $(\boldsymbol{h}^{(K)}, \boldsymbol{y}^{(K)})$ we have an approximation of both the channel $\boldsymbol{h}$ and the observation $\boldsymbol{y}$. The author of [30] investigates whether in this case $\mathrm{E}[\boldsymbol{h}^{(K)} \mid \boldsymbol{y}^{(K)}]$ is an approximation of $\mathrm{E}[\boldsymbol{h} \mid \boldsymbol{y}]$. This is answered affirmatively for suitable distributions. Since we assume $f_{\boldsymbol{h}}$ to be unknown, a discussion of [30] is omitted. Note, $(\mathrm{E}[\boldsymbol{h}^{(K)} \mid \boldsymbol{y}^{(K)}])_{K=1}^{\infty}$ is a sequence of random variables and [30] provides conditions under which this sequence convergences in distribution to the random variable $\mathrm{E}[\boldsymbol{h} \mid \boldsymbol{y}]$. In contrast, we investigate the pointwise convergence of the functions $(\boldsymbol{y} \mapsto \mathrm{E}[\boldsymbol{h}^{(K)} \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}])_{K=1}^{\infty}$ to the

function $\boldsymbol{y} \mapsto \mathrm{E}[\boldsymbol{h} \mid \boldsymbol{y} = \boldsymbol{y}]$. Thus, Theorem 2 and [30] are related but study different mathematical objects and different types of convergence. In particular, Theorem 2 is not a consequence of [30].

We still need to address the question of how to obtain a sequence $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ of PDFs which converges uniformly to $f_{\boldsymbol{h}}$ and how to compute the estimator $\hat{\boldsymbol{h}}^{(K)}$ so that it can be used for channel estimation in practice. In the remainder of this chapter, we study the example of a sequence of GMMs which shows the desired convergence behavior and leads to a closed-form expression for $\hat{\boldsymbol{h}}^{(K)}$ (left column in Figure 3.1).

3.3 Example: Gaussian Mixture Models

We come back to the notation introduced in Section 3.1 where all quantities are complex-valued. Theorem 2 assumes a channel PDF $f_{\boldsymbol{h}}$ to be given and then requires a sequence $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ of PDFs which converges uniformly to $f_{\boldsymbol{h}}$. In light of Theorem 1 (in Section 2.4), there always exists a sequence of GMMs which fulfills that requirement. Henceforth, $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ denotes a suitable sequence of GMMs. As discussed next, GMMs have the benefit that $\hat{\boldsymbol{h}}^{(K)}$ from (3.8) can be computed in closed form for every $K \in \mathbb{N}$ which results in a practically useful channel estimator.

3.3.1 The GMM Estimator

For a given $K \in \mathbb{N}$, we are interested in obtaining an analytic expression for $\hat{\boldsymbol{h}}^{(K)}$ from (3.8). Since this estimator is based on GMMs for the remainder of Chapter 3, we call it the *GMM estimator* from now on. Let

$$f_{\boldsymbol{h}}^{(K)} : \mathbb{C}^N \to \mathbb{R},\ \boldsymbol{h} \mapsto f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h}) = \sum_{k=1}^{K} p(k) \mathcal{N}_{\mathbb{C}}(\boldsymbol{h}; \boldsymbol{\mu}_k, \boldsymbol{C}_k) \tag{3.14}$$

be the Kth GMM. For notational convenience, we omit the sequence index K when we write the GMM parameters so that we have $p(k), \boldsymbol{\mu}_k, \boldsymbol{C}_k$ instead of, e.g., $p^{(K)}(k), \boldsymbol{\mu}_k^{(K)}, \boldsymbol{C}_k^{(K)}$. Let $\boldsymbol{k}$ be the discrete mixing variable such that $(\boldsymbol{h}^{(K)} \mid \boldsymbol{k} = k) \sim \mathcal{N}_{\mathbb{C}}(\boldsymbol{\mu}_k, \boldsymbol{C}_k)$ is the kth Gaussian in the GMM (3.14). Since also the noise $\boldsymbol{n} \sim \mathcal{N}_{\mathbb{C}}(\boldsymbol{0}, \boldsymbol{\Sigma})$ is Gaussian, the observation $\boldsymbol{y}^{(K)} = \boldsymbol{A}\boldsymbol{h}^{(K)} + \boldsymbol{n}$ conditioned on $\boldsymbol{k} = k$ is Gaussian as well:

$$(\boldsymbol{y}^{(K)} \mid \boldsymbol{k} = k) \sim \mathcal{N}_{\mathbb{C}}(\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{C}}_k) \tag{3.15}$$

$$\text{with } \tilde{\boldsymbol{\mu}}_k = \boldsymbol{A}\boldsymbol{\mu}_k \text{ and } \tilde{\boldsymbol{C}}_k = \boldsymbol{A}\boldsymbol{C}_k\boldsymbol{A}^{\mathrm{H}} + \boldsymbol{\Sigma}. \tag{3.16}$$

Consequently, the PDF of $\boldsymbol{y}^{(K)}$ is the following GMM:

$$f_{\boldsymbol{y}}^{(K)} : \mathbb{C}^m \to \mathbb{R},\ \boldsymbol{y} \mapsto f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) = \sum_{k=1}^{K} p(k)\mathcal{N}_{\mathbb{C}}(\boldsymbol{y}; \tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{C}}_k). \tag{3.17}$$

The quantities obtained so far can now be used to compute the desired channel estimator $\hat{\boldsymbol{h}}^{(K)}$ from (3.8). To this end, let $\boldsymbol{y} \in \mathbb{C}^m$ and employ the law of total expectation:

$$\hat{\boldsymbol{h}}^{(K)}(\boldsymbol{y}) = \mathrm{E}[\boldsymbol{h}^{(K)} \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}] \tag{3.18}$$

$$= \sum_{k=1}^{K} p(k \mid \boldsymbol{y}^{(K)} = \boldsymbol{y})\, \mathrm{E}[\boldsymbol{h}^{(K)} \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}, \boldsymbol{k} = k]. \tag{3.19}$$

The first factor in (3.19) can be identified as a responsibility (cf. (2.11)) of the GMM $f_{\boldsymbol{y}}^{(K)}$ and can be calculated in closed form:

$$p(k \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}) = \frac{p(k)\mathcal{N}_{\mathbb{C}}(\boldsymbol{y}; \tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{C}}_k)}{\sum_{i=1}^{K} p(i)\mathcal{N}_{\mathbb{C}}(\boldsymbol{y}; \tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{C}}_i)}. \tag{3.20}$$

Since both $\boldsymbol{h}^{(K)}$ as well as $\boldsymbol{y}^{(K)}$ conditioned on $\boldsymbol{k} = k$ are Gaussian random variables, the second factor in (3.19) is the CME known as linear minimum mean square error (LMMSE) estimator and has a well-known closed-form expression. For later reference, we give this estimator the name $\hat{\boldsymbol{h}}_{\mathrm{LMMSE},k}$:

$$\hat{\boldsymbol{h}}_{\mathrm{LMMSE},k} : \mathbb{C}^m \to \mathbb{C}^N,\ \boldsymbol{y} \mapsto \hat{\boldsymbol{h}}_{\mathrm{LMMSE},k}(\boldsymbol{y}) = \mathrm{E}[\boldsymbol{h}^{(K)} \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}, \boldsymbol{k} = k] \tag{3.21a}$$

$$= \boldsymbol{C}_k \boldsymbol{A}^{\mathrm{H}} \tilde{\boldsymbol{C}}_k^{-1}(\boldsymbol{y} - \tilde{\boldsymbol{\mu}}_k) + \boldsymbol{\mu}_k. \tag{3.21b}$$

In summary, the closed-form expression of the GMM estimator $\hat{\boldsymbol{h}}^{(K)}$ reads:

$$\hat{\boldsymbol{h}}^{(K)} : \mathbb{C}^m \to \mathbb{C}^N \tag{3.22a}$$

$$\boldsymbol{y} \mapsto \hat{\boldsymbol{h}}^{(K)}(\boldsymbol{y}) = \sum_{k=1}^{K} \frac{p(k)\mathcal{N}_{\mathbb{C}}(\boldsymbol{y}; \tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{C}}_k)}{\sum_{i=1}^{K} p(i)\mathcal{N}_{\mathbb{C}}(\boldsymbol{y}; \tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{C}}_i)} \hat{\boldsymbol{h}}_{\mathrm{LMMSE},k}(\boldsymbol{y}) \tag{3.22b}$$

with $\hat{\boldsymbol{h}}_{\mathrm{LMMSE},k}$ defined in (3.21).

The formula in (3.22) is known and can also be found, e.g., in [31]. Therein, the authors assume that $f_{\boldsymbol{h}}^{(K)}$ in (3.14) is equal to the PDF $f_{\boldsymbol{h}}$ of the channel $\boldsymbol{h}$ (for some fixed K) so that (3.22) is the optimal channel estimator. They then study this estimator in the high signal-to-noise ratio (SNR) regime to derive suitable pilot symbols. In contrast to this, we view $f_{\boldsymbol{h}}^{(K)}$ as an approximation of the unknown channel PDF $f_{\boldsymbol{h}}$.

Algorithm 2 GMM Estimator

Offline GMM Training Phase

Require: training data $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$, number K of GMM components

1: $(\{p(k)\}_{k=1}^{K}, \{\boldsymbol{\mu}_k\}_{k=1}^{K}, \{\boldsymbol{C}_k\}_{k=1}^{K}) \leftarrow \mathrm{EM}(\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}, K)$ *// an EM-algorithm computes all parameters of* $f_{\boldsymbol{h}}^{(K)}$

Online Channel Estimation Phase

Require: observation $\boldsymbol{y}$, observation matrix $\boldsymbol{A}$, noise matrix $\boldsymbol{\Sigma}$

2: $\hat{\boldsymbol{h}}^{(K)} \leftarrow \mathbf{0}$

3: **for** $k = 1$ to K **do**

4: $\quad p(k \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}) \leftarrow \frac{p(k)\mathcal{N}_{\mathbb{C}}(\boldsymbol{y};\boldsymbol{A}\boldsymbol{\mu}_k,\boldsymbol{A}\boldsymbol{C}_k\boldsymbol{A}^{\mathrm{H}}+\boldsymbol{\Sigma})}{\sum_{i=1}^{K} p(i)\mathcal{N}_{\mathbb{C}}(\boldsymbol{y};\boldsymbol{A}\boldsymbol{\mu}_i,\boldsymbol{A}\boldsymbol{C}_i\boldsymbol{A}^{\mathrm{H}}+\boldsymbol{\Sigma})}$

5: $\quad \tilde{\boldsymbol{h}} \leftarrow \boldsymbol{C}_k\boldsymbol{A}^{\mathrm{H}}(\boldsymbol{A}\boldsymbol{C}_k\boldsymbol{A}^{\mathrm{H}} + \boldsymbol{\Sigma})^{-1}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}_k) + \boldsymbol{\mu}_k$

6: $\quad \hat{\boldsymbol{h}}^{(K)} \leftarrow \hat{\boldsymbol{h}}^{(K)} + p(k \mid \boldsymbol{y}^{(K)} = \boldsymbol{y})\tilde{\boldsymbol{h}}$

7: **end** *// many quantities in the loop can be precomputed*

8: **return** $\hat{\boldsymbol{h}}^{(K)}$ *// estimated channel, see* (3.22)

The approximation becomes better as K increases and due to Theorem 2, we can view $\hat{\boldsymbol{h}}^{(K)}$ as an approximation of the optimal channel estimator $\hat{\boldsymbol{h}}$. In that sense, we study $\hat{\boldsymbol{h}}^{(K)}$ in the "high number of components regime". The contribution of this chapter lies in providing a strong motivation to use $\hat{\boldsymbol{h}}^{(K)}$ for channel estimation even if the channel is not distributed according to $f_{\boldsymbol{h}}^{(K)}$.

3.3.2 Computing the GMM Estimator in Practice

At first glance, the GMM estimator $\hat{\boldsymbol{h}}^{(K)}$ now seems to be given analytically so that it can be used for channel estimation. However, we still need to discuss how to obtain the required parameters which appear in the formula in (3.22) in practice. To this end, we assume a set $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$ of realizations of the channel $\boldsymbol{h} \sim f_{\boldsymbol{h}}$ to be given. This training data set can then be used to fit a K-component GMM as explained in Section 2.4. The result of the fitting process are the mixing coefficients $\{p(k)\}_{k=1}^{K}$, the mean vectors $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$, and the covariance matrices $\{\boldsymbol{C}_k\}_{k=1}^{K}$ of the GMM $f_{\boldsymbol{h}}^{(K)}$. Thereafter, the mean vectors $\{\tilde{\boldsymbol{\mu}}_k\}_{k=1}^{K}$ and covariance matrices $\{\tilde{\boldsymbol{C}}_k\}_{k=1}^{K}$ of the GMM $f_{\boldsymbol{y}}^{(K)}$ can be determined via (3.16). After this offline initialization process, all quantities required to compute $\hat{\boldsymbol{h}}^{(K)}$ are given and $\hat{\boldsymbol{h}}^{(K)}$ can be used for online channel estimation. The whole process is summarized in Algorithm 2.

Note that since the covariance matrices $\{\tilde{\boldsymbol{C}}_k\}_{k=1}^{K}$ of $f_{\boldsymbol{y}}^{(K)}$ depend on the noise covariance matrix $\boldsymbol{\Sigma}$ (cf. (3.16)), the estimator $\hat{\boldsymbol{h}}^{(K)}$ also depends on it. In particular,

if the SNR changes, the estimator $\hat{\boldsymbol{h}}^{(K)}$ needs to be updated. However, this only requires updating $\{\tilde{\boldsymbol{C}}_k\}_{k=1}^K$—it is not necessary to fit a new GMM. The fitting process only determines $f_{\boldsymbol{h}}^{(K)}$, which does not depend on the SNR, and it only needs to be run once for a given $K \in \mathbb{N}$. Generally, it makes sense to precompute the quantities in (3.22) for a number of different SNRs so that computing $\hat{\boldsymbol{h}}^{(K)}$ online only amounts to matrix-vector multiplications. The complexity of computing the GMM estimator $\hat{\boldsymbol{h}}^{(K)}$ is discussed next.

3.3.3 Computational Complexity and Number of Parameters

As mentioned in Section 3.3.2, once the GMM fitting process, which yields $\{p(k)\}_{k=1}^K$, $\{\boldsymbol{\mu}_k\}_{k=1}^K$, and $\{\boldsymbol{C}_k\}_{k=1}^K$, is done, the remaining quantities in (3.22) can be precomputed for different SNRs. This is, in particular, also true for the costly inverses $\{\tilde{\boldsymbol{C}}_k^{-1}\}_{k=1}^K$ and determinants $\{\det(\tilde{\boldsymbol{C}}_k)\}_{k=1}^K$. Both of these quantities are required to evaluate the responsibilities in (3.22) because a Gaussian PDF with mean vector $\tilde{\boldsymbol{\mu}} \in \mathbb{C}^m$ and covariance matrix $\tilde{\boldsymbol{C}} \in \mathbb{C}^{m \times m}$ is given by

$$\boldsymbol{x} \mapsto \mathcal{N}_{\mathbb{C}}(\boldsymbol{x}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{C}}) = \frac{\exp(-(\boldsymbol{x}-\tilde{\boldsymbol{\mu}})^{\mathrm{H}} \tilde{\boldsymbol{C}}^{-1}(\boldsymbol{x}-\tilde{\boldsymbol{\mu}}))}{\pi^n \det(\tilde{\boldsymbol{C}})}. \tag{3.23}$$

It can now be seen that for $m \leq N$, an evaluation of $\mathcal{N}_{\mathbb{C}}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{C}})$ has a complexity of $\mathcal{O}(N^2)$. The last ingredient to computing $\hat{\boldsymbol{h}}^{(K)}$ is the LMMSE formula (3.21b) which costs $\mathcal{O}(N^2)$. Since this needs to be done K times (cf. (3.22)), the overall complexity of computing the GMM estimator $\hat{\boldsymbol{h}}^{(K)}$ is $\mathcal{O}(KN^2)$.

In some cases, we are interested in reducing the number of parameters of the GMM $f_{\boldsymbol{h}}^{(K)}$. The parameters are the mixing coefficients $\{p(k)\}_{k=1}^K$, the mean vectors $\{\boldsymbol{\mu}_k\}_{k=1}^K$, and the covariance matrices $\{\boldsymbol{C}_k\}_{k=1}^K$ so that we have $K + KN + K\frac{N(N+1)}{2}$ scalar parameters if we take symmetries in $\{\boldsymbol{C}_k\}_{k=1}^K$ into account. One reason to reduce the number of parameters is to reduce the computational complexity. As demonstrated in Section 3.3.3.1, cases exist where the complexity of computing $\hat{\boldsymbol{h}}^{(K)}$ can be reduced to $\mathcal{O}(KN\log(N))$. Another reason to reduce the number of parameters is to improve the GMM fitting process. We can expect to need a larger number T_{tr} of training data $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$ in order to successfully fit a GMM with a larger number of parameters. For a given amount of training data, we can either choose a smaller K to control the number of GMM parameters, or, as discussed next, we can impose structure on the covariance matrices. The following two examples are motivated by commonly encountered matrix structures in mobile communications.

3.3.3.1 Circulant covariance matrices

In the first example, we force every covariance matrix of the GMM $f_{\boldsymbol{h}}^{(K)}$ to have the circulant structure

$$\boldsymbol{C}_k = \boldsymbol{F}^{\mathrm{H}} \operatorname{diag}(\boldsymbol{c}_k) \boldsymbol{F} \tag{3.24}$$

where $\boldsymbol{F} \in \mathbb{C}^{N \times N}$ is the discrete Fourier transform (DFT) matrix and $\boldsymbol{c}_k \in \mathbb{C}^N$. This significantly reduces the number of GMM parameters to $K + KN + KN$. The motivation comes from the fact that channel covariance matrices are Toeplitz structured if the base station employs a uniform linear array (ULA) and that Toeplitz matrices are well approximated by circulant matrices [32]. The charm of circulant matrices is that thanks to fast Fourier transforms, matrix-vector multiplications can be performed in $\mathcal{O}(N \log(N))$ time. This is particularly interesting if we have $\boldsymbol{A} = \boldsymbol{I}$ and $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I} = \sigma^2 \boldsymbol{F}^{\mathrm{H}} \boldsymbol{F}$ because the LMMSE formula (3.21b) simplifies to

$$\hat{\boldsymbol{h}}_{\mathrm{LMMSE},k}(\boldsymbol{y}) = \boldsymbol{F}^{\mathrm{H}} \operatorname{diag}(\boldsymbol{d}_k) \boldsymbol{F} (\boldsymbol{y} - \boldsymbol{\mu}_k) + \boldsymbol{\mu}_k \tag{3.25}$$

where the ith entry of the vector $\boldsymbol{d}_k$ is given by $[\boldsymbol{d}_k]_i = \frac{[\boldsymbol{c}_k]_i}{[\boldsymbol{c}_k]_i + \sigma^2}$, such that (3.25) can be calculated in $\mathcal{O}(N \log(N))$ time. On top of this, with a circulant covariance matrix, the Gaussian density can be written as

$$\boldsymbol{x} \mapsto \mathcal{N}_{\mathbb{C}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{F}^{\mathrm{H}} \operatorname{diag}(\boldsymbol{c}) \boldsymbol{F}) = \frac{\exp(-(\boldsymbol{F}(\boldsymbol{x} - \boldsymbol{\mu}))^{\mathrm{H}} \operatorname{diag}(\boldsymbol{c})^{-1} \boldsymbol{F}(\boldsymbol{x} - \boldsymbol{\mu}))}{\pi^n \prod_{i=1}^N [\boldsymbol{c}]_i}. \tag{3.26}$$

In contrast to (3.23), this can be computed in $\mathcal{O}(N \log(N))$ time. Consequently, computing $\hat{\boldsymbol{h}}^{(K)}$ has a complexity of $\mathcal{O}(KN \log(N))$. Even if $\boldsymbol{A} \neq \boldsymbol{I}$, such that we may not arrive at a complexity of $\mathcal{O}(KN \log(N))$, circulant covariance matrices can still be interesting to control the number of GMM parameters and thereby also the GMM fitting process. In Section 3.6, we investigate how the amount of training data and the number of GMM parameters influence the performance of $\hat{\boldsymbol{h}}^{(K)}$.

To implement the circulant structure (3.24), e.g., in Python, we can make use of the fact that diagonal covariance matrices are a commonly used constraint for GMM components so that there exist specialized versions of the expectation-maximization (EM) fitting algorithm for this case. We can fit a GMM with diagonal covariance matrices on Fourier transformed training data $\{\boldsymbol{F}\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$, which yields the mean vectors $\{\boldsymbol{F}\boldsymbol{\mu}_k\}_{k=1}^K$ and covariance matrices $\{\operatorname{diag}(\boldsymbol{c}_k)\}_{k=1}^K$, so that we obtain the corresponding quantities for the original data $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$ as $\{\boldsymbol{\mu}_k\}_{k=1}^K$ and $\{\boldsymbol{F}^{\mathrm{H}} \operatorname{diag}(\boldsymbol{c}_k) \boldsymbol{F}\}_{k=1}^K$ by making use of the DFT matrix. A numerical evaluation of the circulant GMM estimator is presented in Section 3.6.2.

3.3.3.2 Kronecker covariance matrices

Another common channel covariance matrix structure results from the assumption that the scattering in the vicinity of the transmitter and of the receiver are independent of each other, cf. [33]. In that case, we can force the covariance matrices to have the form

$$\boldsymbol{C}_k = \boldsymbol{C}_{\mathrm{tx},k} \otimes \boldsymbol{C}_{\mathrm{rx},k} \tag{3.27}$$

where $\boldsymbol{C}_{\mathrm{tx},k}$ and $\boldsymbol{C}_{\mathrm{rx},k}$ represent a transmit and a receive side spatial covariance matrix, respectively. Using the signal model (2.5), if mobile terminals have N_{tx} antennas and the base station has N_{rx} antennas, the training data $\{\boldsymbol{H}_t\}_{t=1}^{T_{\mathrm{tr}}}$ consist of channel matrices $\boldsymbol{H}_t \in \mathbb{C}^{N_{\mathrm{rx}} \times N_{\mathrm{tx}}}$. If no constraints on the covariance matrices of the GMM are to be taken into account, we fit a K-component GMM on the vectorized training data $\{\mathrm{vec}(\boldsymbol{H}_t)\}_{t=1}^{T_{\mathrm{tr}}}$, see also Section 2.2.1.

In contrast, in order to obtain a GMM with matrices of the form (3.27), we first fit two GMMs: We fit a K_{tx}-component GMM on all rows of $\{\boldsymbol{H}_t\}_{t=1}^{T_{\mathrm{tr}}}$, and we fit a K_{rx}-component GMM on all columns of $\{\boldsymbol{H}_t\}_{t=1}^{T_{\mathrm{tr}}}$. This results in two sets of GMM parameters: $\{(\boldsymbol{\mu}_{\mathrm{tx},i}, \boldsymbol{C}_{\mathrm{tx},i})\}_{i=1}^{K_{\mathrm{tx}}}$ and $\{(\boldsymbol{\mu}_{\mathrm{rx},j}, \boldsymbol{C}_{\mathrm{rx},j})\}_{j=1}^{K_{\mathrm{rx}}}$. These are now combined to obtain the GMM $f_{\boldsymbol{h}}^{(K)}$ with $K = K_{\mathrm{tx}} K_{\mathrm{rx}}$ components: $\{(\boldsymbol{\mu}_{\mathrm{tx},i} \otimes \boldsymbol{\mu}_{\mathrm{rx},j}, \boldsymbol{C}_{\mathrm{tx},i} \otimes \boldsymbol{C}_{\mathrm{rx},j}) : 1 \leq i \leq K_{\mathrm{tx}}, 1 \leq j \leq K_{\mathrm{rx}}\}$. Lastly, given these K mean vectors and covariance matrices, we compute the mixing coefficients $\{p(k)\}_{k=1}^{K}$ corresponding to $f_{\boldsymbol{h}}^{(K)}$ by performing one E-step (cf. [17]) of the EM algorithm. Overall, the GMM obtained as described has a significantly reduced number of parameters: $K + K_{\mathrm{tx}} N_{\mathrm{tx}} + K_{\mathrm{rx}} N_{\mathrm{rx}} + K_{\mathrm{tx}} \frac{N_{\mathrm{tx}}(N_{\mathrm{tx}}+1)}{2} + K_{\mathrm{rx}} \frac{N_{\mathrm{rx}}(N_{\mathrm{rx}}+1)}{2}$ in contrast to $K + K N_{\mathrm{tx}} N_{\mathrm{rx}} + K \frac{N_{\mathrm{tx}} N_{\mathrm{rx}}(N_{\mathrm{tx}} N_{\mathrm{rx}}+1)}{2}$. Plugging the Kronecker product matrices into the LMMSE formula (3.21b) does not necessarily lead to a computational advantage due to the inverse being a matrix which is generally not a Kronecker product of two smaller matrices. This inverse, however, can be approximated by means of a Kronecker product, cf., e.g., [34], if the computational complexity is an issue. A numerical evaluation of the Kronecker GMM estimator is presented in Section 3.6.2.

3.4 Channel Models for Numerical Evaluation

Section 3.6 presents a numerical evaluation of the GMM estimator. We consider the single-input multiple-output (SIMO), the multiple-input multiple-output (MIMO), and the wideband single-input single-output (SISO) signal models described in Section 2.2.1. The channels are generated according to one of the following two models.

3.4.1 3GPP

To generate the SIMO and the MIMO channels, we use a spatial channel model [35,36] where channels are modeled as $(\boldsymbol{h} \mid \boldsymbol{d} = \boldsymbol{d}) \sim \mathcal{N}_{\mathbb{C}}(\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{d}})$. That is, given a realization $\boldsymbol{d}$ of a random variable $\boldsymbol{d}$ which models the angles of arrival/departure and the path gains of the main propagation clusters between a mobile terminal and the base station, the channel is assumed to be Gaussian with a covariance matrix $\boldsymbol{C}_{\boldsymbol{d}}$ which depends on the realized angles and gains. To generate T channel samples $\{\boldsymbol{h}_t\}_{t=1}^{T}$, we realize T different $\boldsymbol{d}_t$ and corresponding $\boldsymbol{C}_{\boldsymbol{d}_t}$. For every $\boldsymbol{C}_{\boldsymbol{d}_t}$, we then realize one Gaussian vector $\boldsymbol{h}_t$ from $\mathcal{N}_{\mathbb{C}}(\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{d}_t})$.

If both the mobile terminal and the base station employ ULAs, the transmit and receive side spatial channel covariance matrices are given by

$$\boldsymbol{C}_{\{\mathrm{tx,rx}\},\boldsymbol{d}} = \int_{-\pi}^{\pi} g_{\{\mathrm{tx,rx}\}}(\theta; \boldsymbol{d}) \boldsymbol{a}_{\{\mathrm{tx,rx}\}}(\theta) \boldsymbol{a}_{\{\mathrm{tx,rx}\}}(\theta)^{\mathrm{H}} d\theta. \tag{3.28}$$

Here,

$$\boldsymbol{a}_{\{\mathrm{tx,rx}\}}(\theta) = [1, e^{\mathrm{j}\,\pi \sin(\theta)}, \ldots, e^{\mathrm{j}\,\pi (N_{\{\mathrm{tx,rx}\}}-1) \sin(\theta)}]^{\mathrm{T}} \tag{3.29}$$

is the array steering vector for an angle of arrival/departure θ and $g_{\{\mathrm{tx,rx}\}}$ is a power density consisting of a sum of weighted Laplace densities whose standard deviations describe the angle spread of the propagation clusters [35]. Assuming independent scattering in the vicinity of the transmitter and the receiver, cf., e.g., [33], the full covariance matrices are then given as $\boldsymbol{C}_{\boldsymbol{d}} = \boldsymbol{C}_{\mathrm{tx},\boldsymbol{d}} \otimes \boldsymbol{C}_{\mathrm{rx},\boldsymbol{d}}$. In the SIMO case, we have $\boldsymbol{C}_{\boldsymbol{d}} = \boldsymbol{C}_{\mathrm{rx},\boldsymbol{d}}$.

3.4.2 QuaDRiGa

We use version 2.4 of the QuaDRiGa channel simulator [37, 38]. The base station has a height of 25 meters and covers a 120° sector in an urban macrocell scenario. It is equipped with an N_{rx}-antenna ULA with "3GPP-3D" antennas. Mobile terminals are equipped with an N_{tx}-antenna ULA with "omni-directional" antennas. The mobile terminals have a minimum and a maximum distance to the base station of 35 meters and 500 meters, respectively. In 80% of the cases, the mobile terminals are located indoors at different floor levels. The mobile terminals' height is 1.5 meters in the case of outdoor locations. For the SIMO and the MIMO simulations, we generate single-carrier channels at a center frequency of 2.53 GHz. For the wideband SISO simulations, we have a typical 5G frame structure (cf. [39]) with $N_c = 24$ carriers over a bandwidth of 360 kHz and with $N_t = 14$ time symbols over slots of 1 ms duration.

The center frequency is again 2.53 GHz. Each mobile terminal moves at a certain velocity v in a random direction.

The QuaDRiGa simulator models a channel corresponding to a carrier c (with frequency f_c) and a time symbol t as $\boldsymbol{H}_{c,t} = \sum_{l=1}^{L} \boldsymbol{G}_l e^{-2\pi \mathrm{j}\, f_c \tau_{l,t}}$, where l is the path number. The lth path delay corresponding to the time symbol t is $\tau_{l,t}$. The number of multi-path components L depends on whether there is line of sight (LOS), non-line of sight (NLOS), or outdoor-to-indoor (O2I) propagation: $L_{\mathrm{LOS}} = 37$, $L_{\mathrm{NLOS}} = 61$, or $L_{\mathrm{O2I}} = 37$, cf. [40]. The attenuation of the lth path, the antenna radiation pattern weighting, and the polarization are collected in the coefficients matrix $\boldsymbol{G}_l$, cf. [40]. The generated channels are post-processed to remove the path gain as described in the QuaDRiGa manual [38].

3.5 Other Channel Estimators

We now discuss the algorithms which we compare the GMM estimator $\hat{\boldsymbol{h}}^{(K)}$ to. The numerical evaluation is presented in Section 3.6.

Using the Moore-Penrose pseudoinverse $\boldsymbol{A}^{\dagger}$, a simple baseline algorithm is given by the least squares (LS) channel estimation:

$$\boldsymbol{y} \mapsto \hat{\boldsymbol{h}}_{\mathrm{LS}}(\boldsymbol{y}) = \boldsymbol{A}^{\dagger}\boldsymbol{y}. \tag{3.30}$$

Further, using a set $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$ of training data, we can compute a sample covariance matrix $\hat{\boldsymbol{C}} = \frac{1}{T_{\mathrm{tr}}} \sum_{t=1}^{T_{\mathrm{tr}}} \boldsymbol{h}_t \boldsymbol{h}_t^{\mathrm{H}}$ and employ this matrix together with the LMMSE formula such that we have the following channel estimator:

$$\boldsymbol{y} \mapsto \hat{\boldsymbol{h}}_{\text{sample cov.}}(\boldsymbol{y}) = \hat{\boldsymbol{C}}\boldsymbol{A}^{\mathrm{H}}(\boldsymbol{A}\hat{\boldsymbol{C}}\boldsymbol{A}^{\mathrm{H}} + \boldsymbol{\Sigma})^{-1}\boldsymbol{y}. \tag{3.31}$$

When using the 3GPP channel model from Section 3.4.1, every channel is generated from a Gaussian PDF $\mathcal{N}_{\mathbb{C}}(\boldsymbol{0}, \boldsymbol{C_d})$ so that we have a channel covariance matrix $\boldsymbol{C_d}$ for every channel sample. Given this matrix (or, equivalently, given the realization $\boldsymbol{d}$), the LMMSE estimator

$$\boldsymbol{y} \mapsto \hat{\boldsymbol{h}}_{\text{gen. LMMSE}}(\boldsymbol{y}) = \mathrm{E}[\boldsymbol{h} \mid \boldsymbol{y} = \boldsymbol{y}, \boldsymbol{d} = \boldsymbol{d}] = \boldsymbol{C_d}\boldsymbol{A}^{\mathrm{H}}(\boldsymbol{A}\boldsymbol{C_d}\boldsymbol{A}^{\mathrm{H}} + \boldsymbol{\Sigma})^{-1}\boldsymbol{y} \tag{3.32}$$

is optimal. Note, however, that this estimator is not equal to the CME $\hat{\boldsymbol{h}}$ from (3.5) because of the additional knowledge of the realization $\boldsymbol{d} = \boldsymbol{d}$. Since, in practice, the channel covariance matrix $\boldsymbol{C_d}$ is not given, we speak of a *genie-aided LMMSE estimator* (gen. LMMSE). For this reason, $\hat{\boldsymbol{h}}_{\text{gen. LMMSE}}$ yields a performance bound for all estimators in this chapter and we can generally not expect to reach this bound with any of the other algorithms.

Another channel estimator is based on a convolutional neural network (CNN). The estimator is introduced in [36] for the SIMO case (cf. (2.4)) and [41] presents our generalization of it to the MIMO case (cf. (2.5)). For example, assuming the noise covariance matrix is $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I} \in \mathbb{C}^{N \times N}$, the SIMO CNN estimator is given by [36]

$$\boldsymbol{y} \mapsto \hat{\boldsymbol{h}}_{\text{CNN}}(\boldsymbol{y}) = \boldsymbol{F}_2^{\text{H}} \operatorname{diag}\left(\boldsymbol{a}^{(2)} \star \text{ReLU}\left(\boldsymbol{a}^{(1)} \star \frac{1}{\sigma^2}|\boldsymbol{F}_2\boldsymbol{y}|^2 + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right) \boldsymbol{F}_2 \boldsymbol{y} \tag{3.33}$$

where the columns of the matrix $\boldsymbol{F}_2 \in \mathbb{C}^{2N \times N}$ consist of the first N columns of the $2N \times 2N$ DFT matrix, ReLU denotes the rectified linear unit, the absolute value in $|\boldsymbol{F}_2\boldsymbol{y}|^2$ is applied elementwise, $\star$ denotes circular convolution, and $\boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}, \boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)} \in \mathbb{R}^{2N}$ are learnable parameters which are optimized based on an MSE loss function. We use this estimator for SIMO channel estimation. For the MIMO case, we use the estimator as described in [41] where the activation function is also the rectified linear unit. In both cases, the computational complexity is $\mathcal{O}(N \log(N))$. During the training phase, samples corresponding to the channel model on which the estimator is later tested are used. Note that the CNN estimator depends on σ^2 and thereby on the SNR such that a new estimator needs to be trained for every SNR value.

Many modern channel estimation algorithms focus on compressive sensing approaches, cf., e.g., the surveys [42,43]. For channel estimation by means of compressive sensing algorithms, every channel realization is modeled as (approximately) sparse: $\boldsymbol{h} \approx \boldsymbol{\Psi}\boldsymbol{s}$. The *dictionary* $\boldsymbol{\Psi} \in \mathbb{C}^{N \times L}$ is typically an oversampled DFT matrix (e.g., [44]), and $\boldsymbol{s} \in \mathbb{C}^L$ is a sparse vector. Given an observation $\boldsymbol{y}$, compressive sensing algorithms then introduce the auxiliary model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{\Psi}\boldsymbol{s} + \boldsymbol{n}$ and recover an estimate $\hat{\boldsymbol{s}}$ of $\boldsymbol{s}$ so that the channel can be estimated as $\hat{\boldsymbol{h}}_{\text{CS}} = \boldsymbol{\Psi}\hat{\boldsymbol{s}}$. One example algorithm is orthogonal matching pursuit (OMP) [14–16] which is detailed in Algorithm 1 (in Section 2.3) and would be used with $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{\Psi}$. Since OMP requires the sparsity order (number of nonzero elements in $\boldsymbol{s}$) and since this is not known for the two channel models described in Section 3.4, we provide the true channel realization to the OMP algorithm so that it can find the optimal sparsity order, see Algorithm 3. This genie-aided approach yields a performance bound for OMP. Another compressive sensing algorithm we compare our results with is approximate message passing (AMP) [45, 46], which does not need to know the sparsity order.

For wideband channel estimation (cf. Section 2.2.1), we compare our results with the concrete autoencoder (CAE) algorithm from [47]. The vectorized wideband channels have a dimension of $N_c N_t$ and the pilot selection matrix $\boldsymbol{A} \in \{0,1\}^{N_p \times N_c N_t}$ allows to observe $N_p < N_c N_t$ of the channel elements. The idea of the CAE is to replace the encoder of an autoencoder by a *concrete selector layer* which extracts

Algorithm 3 Genie-Aided Orthogonal Matching Pursuit (gen. OMP)

Require: observation matrix $\boldsymbol{A}$, dictionary $\boldsymbol{\Psi} \in \mathbb{C}^{N \times L}$
Require: observation $\boldsymbol{y}$ and corresponding true channel $\boldsymbol{h}$
$S^{(0)} \leftarrow \emptyset$, $\boldsymbol{s}^{(0)} \leftarrow \boldsymbol{0}$, $e \leftarrow \infty$
for $i = 1$ to L **do**
  $j^* \leftarrow \arg\max_{j \in \{1,\dots,L\}} \{[(\boldsymbol{A\Psi})^{\mathrm{H}}(\boldsymbol{y} - \boldsymbol{A\Psi s}^{(i-1)})]_j\}$
  $S^{(i)} \leftarrow S^{(i-1)} \cup \{j^*\}$
  $\boldsymbol{s}^{(i)} \leftarrow \arg\min_{\tilde{\boldsymbol{s}}} \{\|\boldsymbol{y} - \boldsymbol{A\Psi}\tilde{\boldsymbol{s}}\|, \operatorname{support}(\tilde{\boldsymbol{s}}) \subset S^{(i)}\}$
  if $\|\boldsymbol{h} - \boldsymbol{\Psi s}^{(i)}\| < e$ **then**
    $e \leftarrow \|\boldsymbol{h} - \boldsymbol{\Psi s}^{(i)}\|$ *// $\boldsymbol{\Psi s}^{(i)}$ is closer to the true $\boldsymbol{h}$ than $\boldsymbol{\Psi s}^{(i-1)}$*
  else
    return $\boldsymbol{\Psi s}^{(i-1)}$ *// the new estimate $\boldsymbol{\Psi s}^{(i)}$ is worse than the previous one*
  end if
end for
return $\boldsymbol{\Psi s}^{(L)}$

the N_p most important features of the $N_c N_t$-dimensional input. This corresponds to designing the matrix $\boldsymbol{A}$. The decoder of the CAE then serves as the channel estimator. The training phase uses noisy channel data so that the CAE performs both a denoising of the input as well as a reconstruction of the $N_c N_t$-dimensional channels. Since the training data is noisy, a new CAE needs to be trained for every SNR. We do not employ further denoising networks after the decoder of the CAE.

Another wideband channel estimator, called *ChannelNet*, can be found in [48]. It consists of 2D CNNs and combines both an image super-resolution network and an image restoration network. The super-resolution network has three layers, the image restoration network has twenty layers. In this context, the receive signal is interpreted as a low-resolution image and the channel is the image that needs to be restored. Again, a new ChannelNet needs to be trained for every SNR.

3.6 Numerical Evaluation

We look at SIMO, MIMO, and wideband channel estimation scenarios (cf. Section 2.2.1) involving the GMM estimator $\hat{\boldsymbol{h}}^{(K)}$ from (3.22). To obtain the GMM estimator, we fit one GMM on the given training data via an EM algorithm, see Section 3.3.2. Thereafter, the inverses and the determinants in (3.22) are precomputed for every SNR. In comparison, the neural network-based estimators from Section 3.5 all require new estimators to be trained for every SNR. In particular, for every SNR, a

hyperparameter search (cf. Section 2.7) is necessary.

Channel estimation performance is measured in terms of the normalized MSE (nMSE). To this end, $T_{\mathrm{tst}} = 10^4$ N-dimensional test channel samples $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tst}}}$ are generated and used to get a corresponding number of noisy observations of the form (2.3) from which channel estimates $\{\hat{\boldsymbol{h}}_t\}_{t=1}^{T_{\mathrm{tst}}}$ are computed with every algorithm of interest. Then, we calculate $\text{nMSE} = \frac{1}{NT_{\mathrm{tst}}} \sum_{t=1}^{T_{\mathrm{tst}}} \|\boldsymbol{h}_t - \hat{\boldsymbol{h}}_t\|^2$. The channels are normalized such that $\mathrm{E}[\|\boldsymbol{h}\|^2] = N$ holds. The noise covariance matrix is $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I} \in \mathbb{C}^{N \times N}$ which leads us to define the SNR as $\frac{1}{\sigma^2}$. For algorithms which require training data (including the sample covariance estimator (3.31)), we generate $T_{\mathrm{tr}} = 10^5$ samples unless another T_{tr} is stated. Generally, we choose T_{tr} large enough such that increasing it does not lead to better results in the testing phase.

3.6.1 SIMO Channel Estimation

In the SIMO case (Figures 3.2 to 3.6), the observation matrix is the identity matrix $\boldsymbol{A} = \boldsymbol{I} \in \mathbb{C}^{N \times N}$ and the dictionaries of OMP and AMP are $L = 4N$ and $L = 2N$ times oversampled DFT dictionaries, respectively. These numbers lead to the best performance in our simulations. We use a training data size of $T_{\mathrm{tr}} = 19 \cdot 10^4$ to fit the GMM unless another T_{tr} is stated.

The first experiment, displayed in Figure 3.2, uses the 3GPP channel model from Section 3.4.1 with one propagation cluster and the number N of antennas at the base station is varied. The genie-aided LMMSE estimator from (3.32) provides a utopian performance bound for all estimators. The GMM estimator with $K = 128$ components comes close to this performance bound. At around $N = 96$ antennas, the CNN estimator (cf. (3.33)) starts to become better than the genie-aided OMP estimator. The reason is that the assumptions which are made in [36] to derive the CNN estimator are better fulfilled for larger N. Generally, the qualitative behavior of all estimators does not change much as N varies. For this reason, the number of antennas is fixed at $N = 128$ in the other experiments.

In Figure 3.3, we again use the 3GPP channel model from Section 3.4.1 with one propagation cluster. From the mid- to high-SNR range, the two compressive sensing algorithms show a similar performance and they are among the better estimators. For most SNRs, the GMM estimator with $K = 128$ components comes close to the genie-aided LMMSE performance bound. In Figure 3.4, where we have three propagation clusters, the CNN estimator's strong performance in the mid-SNR range is prominent. In the high-SNR range, all estimators tend to show a similar performance to the LS estimator. The only exception is the GMM estimator which still improves

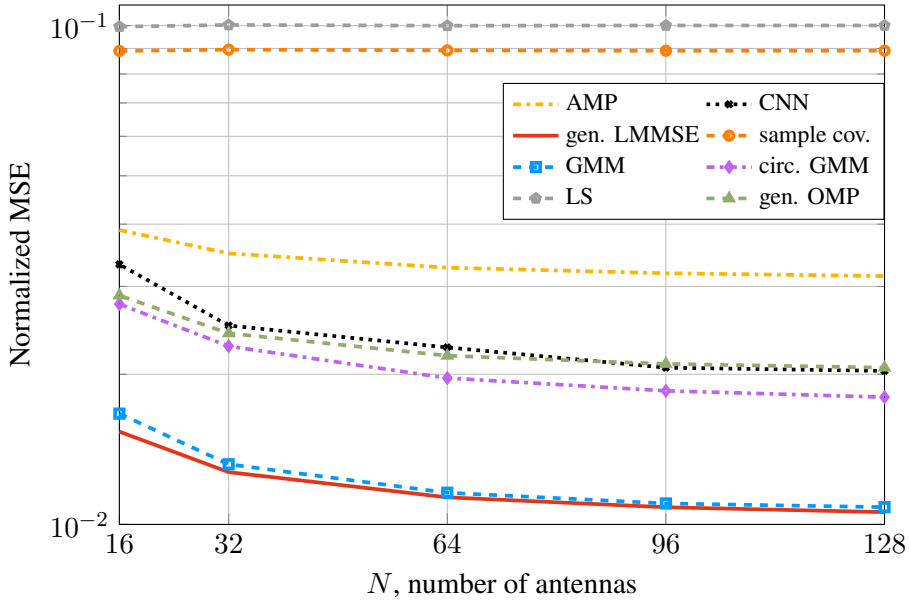

Figure 3.2: SIMO signal model and 3GPP channel model (Section 3.4.1) with **one** propagation cluster at 10 dB SNR. The performance of the circulant GMM estimator ("circ. GMM", -✦-, Section 3.3.3.1) is shown, too. In both cases, $K = 128$ components are used.

upon the LS estimator. Recall that we generally cannot expect any estimator to reach the genie LMMSE curve because it has more knowledge (the true covariance matrix for every sample, cf. (3.32)) than is usually available.

In Figure 3.5, we consider the QuaDRiGa channel model (cf. Section 3.4.2) where the channel covariance matrices and therefore the genie LMMSE curve are no longer available. The two compressive sensing algorithms' performances now differ considerably which may be attributed to a lack of sparsity of the channels. Both the CNN and the GMM estimators generally show a strong performance mainly differing in the low-SNR range where the GMM estimator is better.

In Figure 3.6, we investigate the interplay between the number K of GMM components and the number T_{tr} of training data used to fit the K-component GMM. We concentrate on the 3GPP channel model with three propagation clusters at an SNR of 10 dB (see also Figure 3.4). As expected, since increasing K leads to a larger number of GMM parameters, more training data is necessary for a good fit. As long as T_{tr} is large enough, the behavior indicated by Theorem 2 can be observed: increasing K improves the estimator's performance. Theorem 2 can be applied because we have

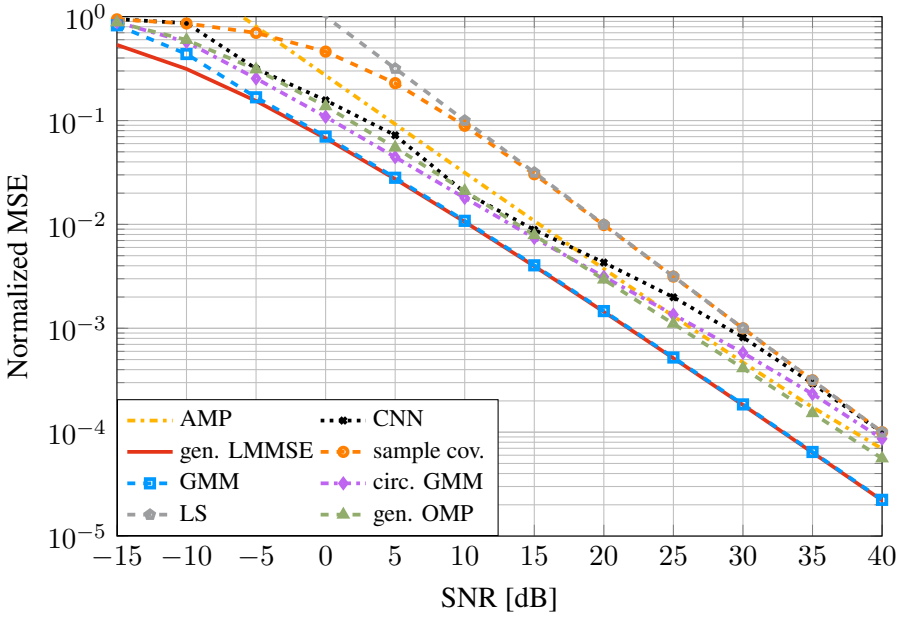

Figure 3.3: SIMO signal model and 3GPP channel model (Section 3.4.1) with **one** propagation cluster and $N = 128$ antennas. The performance of the circulant GMM estimator ("circ. GMM", -✦-, Section 3.3.3.1) is shown, too. In both cases, $K = 128$ components are used.

$\boldsymbol{A} = \boldsymbol{I}$ which is invertible. At this point, note that we generally cannot expect the GMM estimator to converge to the genie LMMSE estimator (3.32) (which is displayed in Figure 3.4). The genie LMMSE estimator has more knowledge (namely the true channel covariance matrix $\boldsymbol{C_d}$ for every channel realization) and is therefore not the CME $\boldsymbol{y} \mapsto \hat{\boldsymbol{h}}(\boldsymbol{y}) = \mathrm{E}[\boldsymbol{h} \mid \boldsymbol{y} = \boldsymbol{y}]$ which we want to approximate in Theorem 2. The CME $\hat{\boldsymbol{h}}$ cannot be computed (because $f_{\boldsymbol{h}}$ is not known) which is the motivation to study the GMM estimator in the first place.

3.6.2 GMM Estimator with Structured Covariance Matrices

We briefly study the circulant and Kronecker GMM estimators from Section 3.3.3.1 and Section 3.3.3.2, respectively. The circulant GMM estimator uses circulant covariance matrices in the GMM. Figures 3.3 to 3.5 contain the corresponding curves. The circulant GMM estimator has a significantly reduced number of parameters and a computational complexity of $\mathcal{O}(N \log(N))$. Its performance is not as good as that of

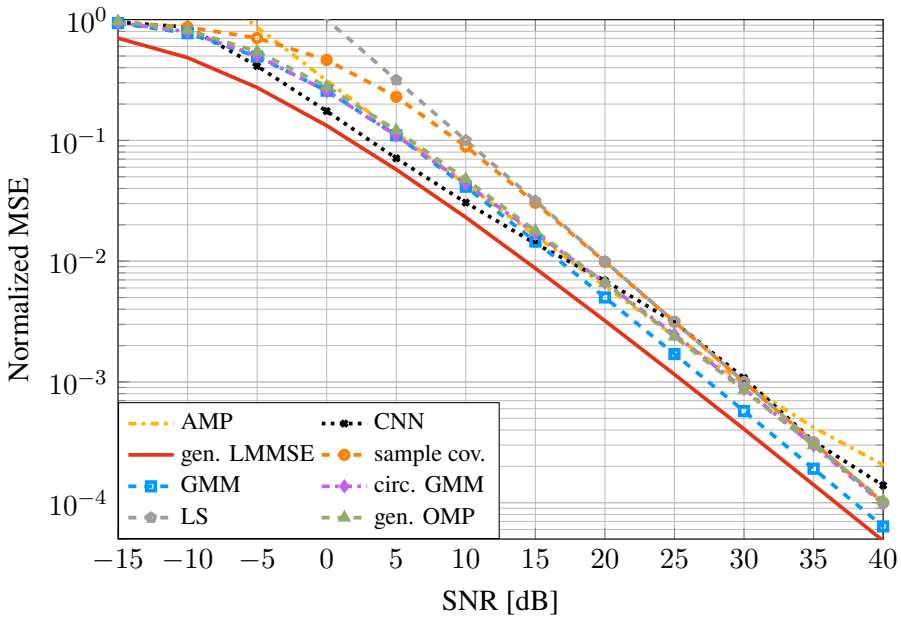

Figure 3.4: SIMO signal model and 3GPP channel model (Section 3.4.1) with **three** propagation clusters and $N = 128$ antennas. The performance of the circulant GMM estimator ("circ. GMM", -·◆·-, Section 3.3.3.1) is shown, too. In both cases, $K = 128$ components are used.

the GMM estimator with unconstrained covariance matrices but it can compete with the other depicted estimators. In particular, in Figure 3.5 the circulant GMM estimator performs well and is even comparable to the unconstrained GMM estimator.

For a brief discussion of the Kronecker GMM estimator, we consider a MIMO simulation with a scaled DFT matrix as pilot matrix $\boldsymbol{P}$, cf. (2.5). Our main interest is to compare the Kronecker GMM estimator and the GMM estimator with unconstrained covariance matrices. First, we generate $T_{\mathrm{tr}} = 10^5$ training data $\{\boldsymbol{H}_t\}_{t=1}^{T_{\mathrm{tr}}}$. Then, in order to obtain the GMM estimator with unconstrained covariance matrices, we fit a GMM with $K = 32$ components on the vectorized data $\{\mathrm{vec}(\boldsymbol{H}_t)\}_{t=1}^{T_{\mathrm{tr}}}$. In contrast, in order to get the Kronecker GMM estimator, we fit two GMMs: a GMM with $K_{\mathrm{tx}} = 4$ components using all of the rows of $\{\boldsymbol{H}_t\}_{t=1}^{T_{\mathrm{tr}}}$ and a GMM with $K_{\mathrm{rx}} = 8$ components using all of the columns of $\{\boldsymbol{H}_t\}_{t=1}^{T_{\mathrm{tr}}}$. These two GMMs are then combined to obtain the Kronecker GMM estimator with $K_{\mathrm{tx}}K_{\mathrm{rx}} = 32$ components. The details are explained in Section 3.3.3.2.

A MIMO simulation with $(N_{\mathrm{rx}}, N_{\mathrm{tx}}) = (32, 4)$ can be found in Figure 3.7. It can

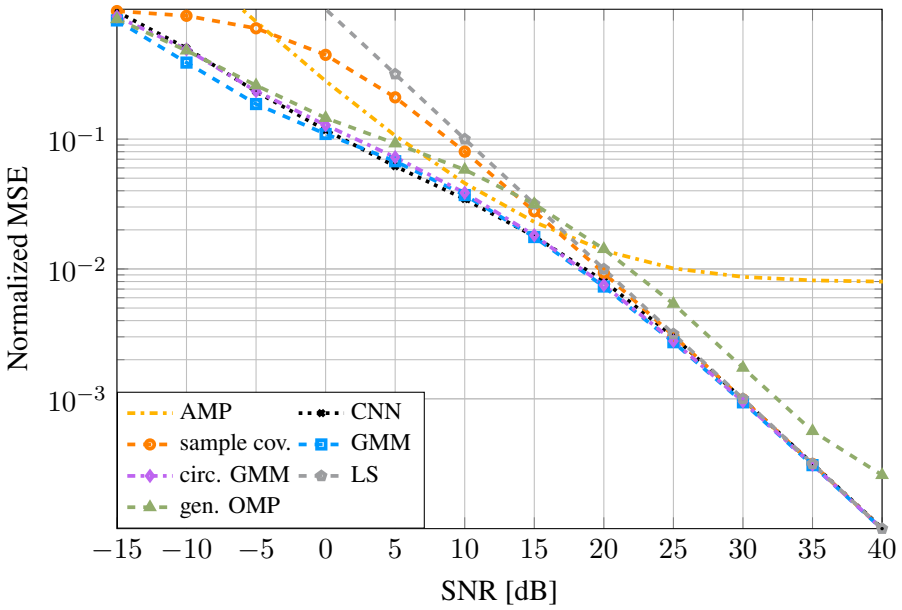

Figure 3.5: SIMO signal model and QuaDRiGa channel model (Section 3.4.2) with $N = 128$ antennas. The performance of the circulant GMM estimator ("circ. GMM", -·-, Section 3.3.3.1) is shown, too. In both cases, $K = 128$ components are used.

be seen that the two GMM estimators show a similar performance. This is interesting because these two estimators have a different number of parameters. In particular, the GMM estimator with unconstrained covariance matrices has $K\frac{N(N+1)}{2} = 264{,}192$ covariance parameters with $N = N_{\text{rx}}N_{\text{tx}} = 128$. By comparison, the Kronecker GMM estimator has only $K_{\text{rx}}\frac{N_{\text{rx}}(N_{\text{rx}}+1)}{2} + K_{\text{tx}}\frac{N_{\text{tx}}(N_{\text{tx}}+1)}{2} = 4{,}224 + 40 = 4{,}264$ covariance parameters. As the number of parameters differs, we can expect the Kronecker GMM estimator to show a better performance if the amount of training data is relatively small. This behavior can be seen in Figure 3.8: The performance of the two different GMM estimators is very different for a small amount of training data and the unconstrained GMM estimator starts to outperform the Kronecker GMM estimator when enough training data are available. Another observation in Figure 3.7 is that also the CNN estimator performs similarly to the GMM estimators with a minor performance loss in the high-SNR regime. All three mentioned estimators outperform the channel covariance matrix-based estimator as well as the genie-aided OMP algorithm. The OMP algorithm uses a Kronecker product of two two-times oversampled DFT matrices as a dictionary in this simulation.

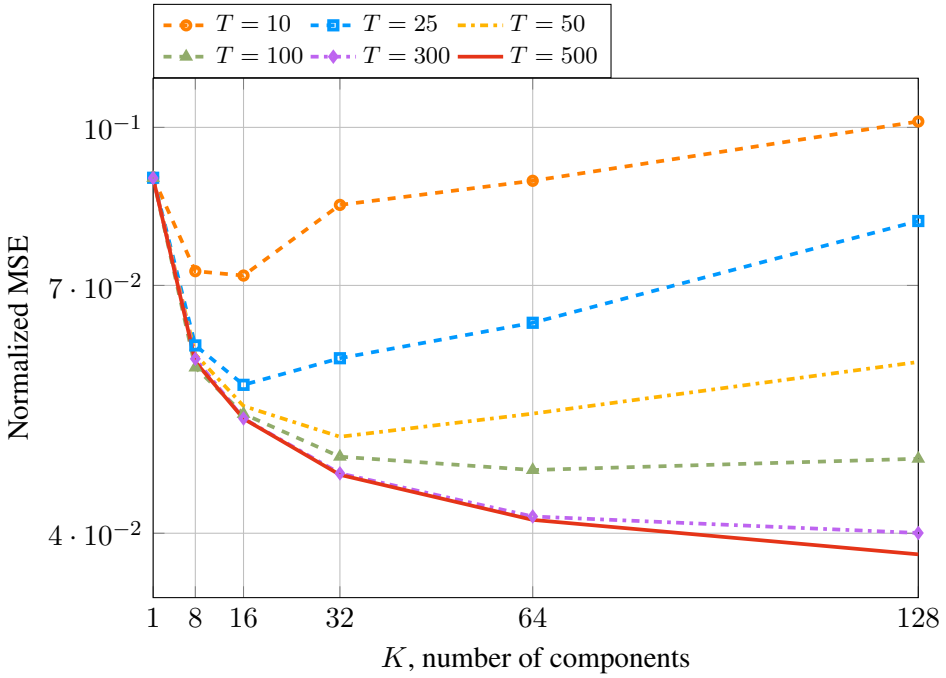

Figure 3.6: SIMO signal model and 3GPP channel model (Section 3.4.1) with **three** propagation clusters and $N = 128$ antennas. The SNR is 10 dB. The GMM estimator is fit using $T_{\text{tr}} = T \cdot 10^3$ training data.

3.6.3 Wideband Channel Estimation

For wideband simulations, we focus on the 5G frame structure described in Section 3.4.2. The number of pilots is $N_p = 50$ so that 50 out of the $24 \cdot 14 = 336$ resource elements are occupied with pilots. Thus, here, the observation matrix $\boldsymbol{A} \in \{0, 1\}^{50 \times 336}$ is not invertible so that Theorem 2 does not apply. Nonetheless, the following Figures 3.9 to 3.11 show the desired behavior of an improving GMM estimator if the number K of components is increased. In this subsection, the considered algorithms are the GMM estimator, the sample covariance matrix-based estimator in (3.31), ChannelNet, and the CAE estimator (see Section 3.5). All estimators are trained on $T_{\text{tr}} = 10^5$ channel samples. There are three commonly used types of pilot arrangements: block, comb, lattice (see, e.g., [49, 50]). We indicate in the figures which of these is used.

Figure 3.9 shows results of a simulation where the mobile terminals move at a constant velocity of $v = 3$ km/h. The block-type pilot arrangement is used with the CAE estimator as the only exception because it uses its own optimized pilot

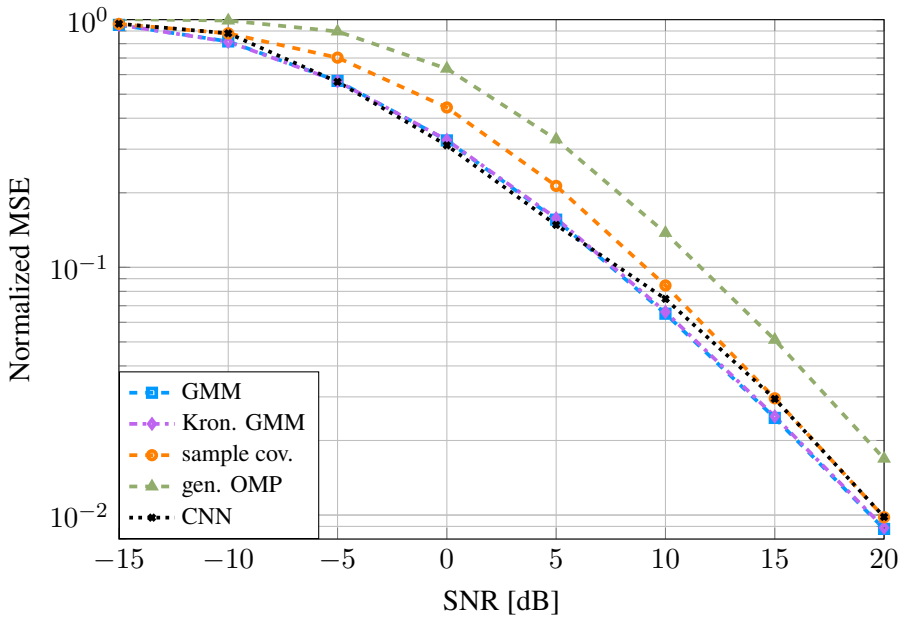

Figure 3.7: MIMO signal model and QuaDRiGa channel model (Section 3.4.2) with $(N_{\text{rx}}, N_{\text{tx}}) = (32, 4)$. The GMM estimator ("GMM", - ■ -) uses $K = 32$ components and the Kronecker GMM estimator ("Kron. GMM", -♦-, Section 3.3.3.2) uses $K = 4 \cdot 8$ components.

pattern. Both ChannelNet and the CAE estimator behave similarly and outperform the sample covariance matrix-based estimator. The GMM estimator can compete with or outperforms both of those estimators. Noticably, an increase of the number of components from $K = 8$ to $K = 128$ leads to a performance improvement even though the observation matrix $\boldsymbol{A}$ is not invertible.

In Figure 3.10, the mobile terminals' velocities are chosen uniformly at random from the interval $[0, 300]$ km/h. In this setting, the lattice-type pilot arrangement is used. Overall, channel estimation is more difficult now and the gap between the sample covariance matrix-based estimator and the other estimators increases. Further, choosing a larger number K of components for the GMM estimator is necessary for a competitive performance. Again, the GMM estimator shows the desired behavior of an improved performance for increased K. The last figure, Figure 3.11, investigates the GMM estimator's performance for different numbers of components and highlights the importance of a large enough amount of training data. Overall, the considerations in this subsection provide some numerical evidence for the GMM estimator's convergence

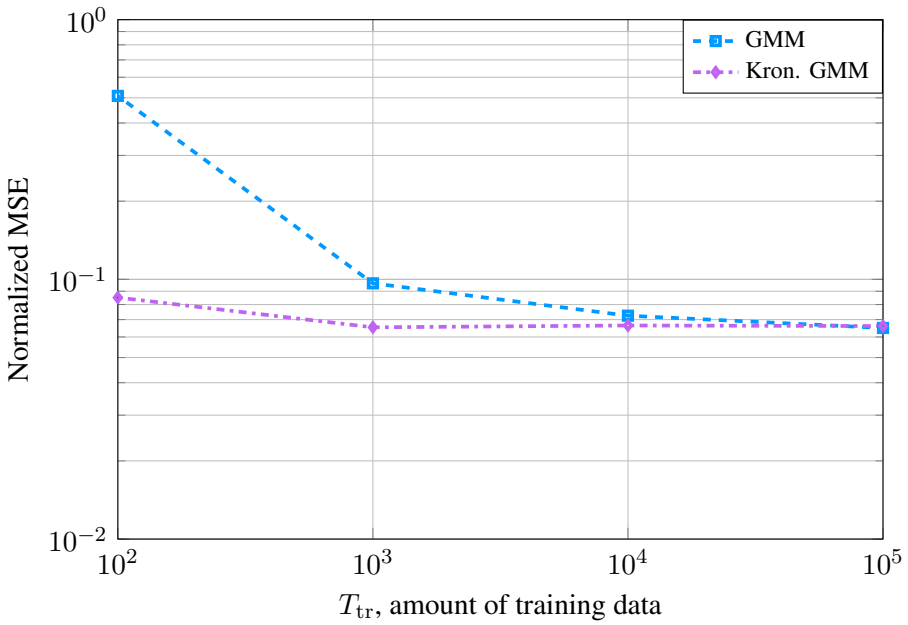

Figure 3.8: MIMO signal model and QuaDRiGa channel model (Section 3.4.2) with $(N_{\mathrm{rx}}, N_{\mathrm{tx}}) = (32, 4)$. The SNR is 10 dB. The Kronecker product GMM estimator ("Kron. GMM", -◆-, Section 3.3.3.2) has fewer parameters than the GMM estimator with unconstrained covariance matrices. Both estimators have $K = 32$ components.

in case of noninvertible observation matrices.

3.6.4 Conclusion

The main contribution of this chapter is a strong motivation to employ the GMM estimator even if the channel PDF $f_{\boldsymbol{h}}$ is not a GMM. On the one hand, we have a theretical motivation in terms of Theorem 2. On the other hand, we have a motivation in terms of the numerical simulations. Overall, the numerical simulations seem to show the performance promised by Theorem 2. In the SIMO and in the MIMO settings, Theorem 2 can be applied directly and the desired behavior is evident. In the wideband setting, the observation matrix is not invertible so that we do not know whether Theorem 2 holds. Nonetheless, the GMM estimator's performance improves when the number of components is increased.

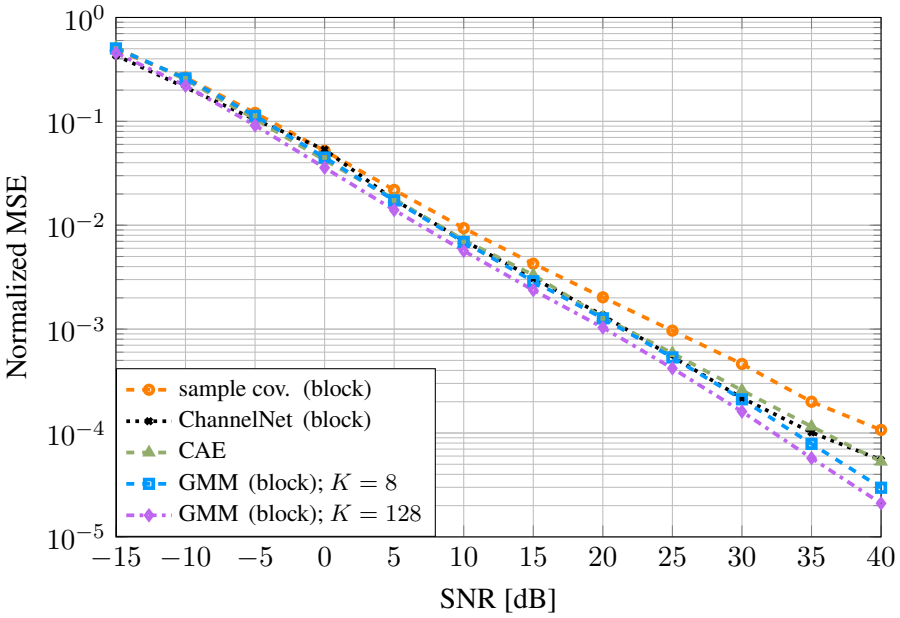

Figure 3.9: Wideband signal model and QuaDRiGa channel model (Section 3.4.2) with $N_p = 50$ pilots for $N_c = 24$ carriers and $N_t = 14$ time symbols for $v = 3$ km/h.

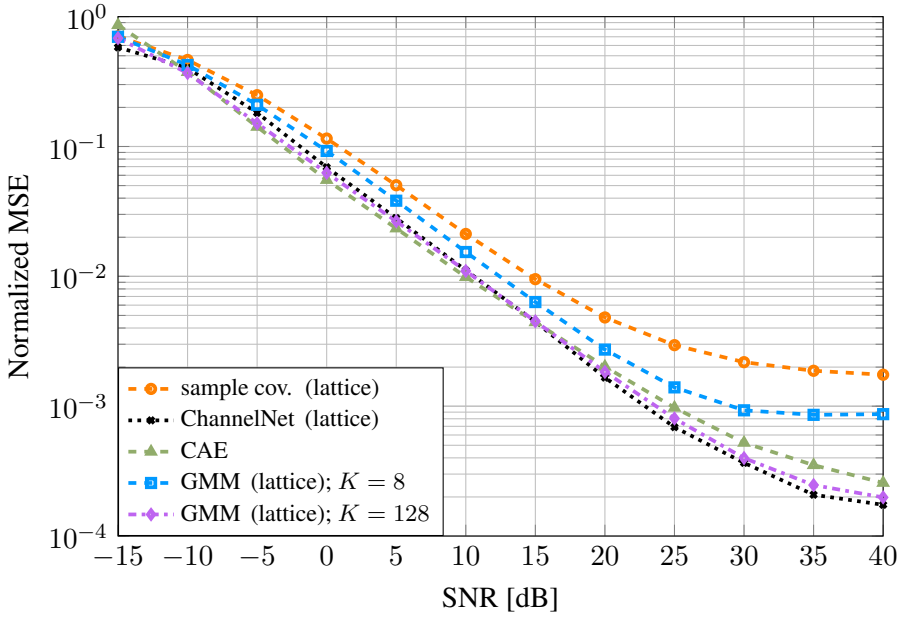

Figure 3.10: Wideband signal model and QuaDRiGa channel model (Section 3.4.2) with $N_p = 50$ pilots for $N_c = 24$ carriers and $N_t = 14$ time symbols for $v \in [0, 300]$ km/h.

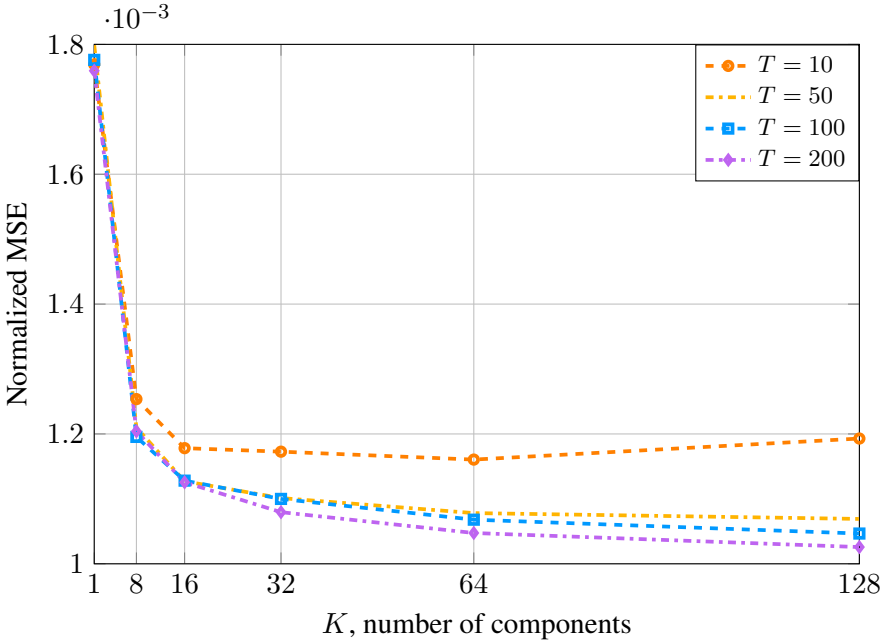

Figure 3.11: Wideband signal model and QuaDRiGa channel model (Section 3.4.2) with $N_p = 50$ pilots for $N_c = 24$ carriers and $N_t = 14$ time symbols for $v = 3$ km/h. The block-type pilot arrangement is used. The SNR is 20 dB. The GMM estimator is fit using $T_{\text{tr}} = T \cdot 10^3$ training data.

4 Learning an Observation Matrix

This chapter is mainly based on our work in [51]. In Chapter 3, we study a channel estimator for the signal model

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{n}. \tag{4.1}$$

In this chapter, we propose a method to find a suitable observation matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ to recover $\boldsymbol{h}$ given this model. The focus lies on constant modulus matrices, see Section 2.2.1. For ease of exposition, we define the set of constant modulus matrices in $\mathbb{C}^{m\times N}$ as $\mathcal{M}_{\text{const}}^{m\times N} = \{\boldsymbol{A} \in \mathbb{C}^{m\times N} : |[\boldsymbol{A}]_{k,l}| = \frac{1}{\sqrt{m}}\}$. We assume that all channel realizations are an element of a set $\mathcal{H} \subset \mathbb{C}^N$. For example, $\mathcal{H}$ could be the set of all p-sparse vectors for some $p < N$; or, $\mathcal{H}$ could, e.g., contain vectors which have a sparse representation in some unknown basis, see also Section 2.3. The goal in this chapter is to obtain a fixed constant modulus observation matrix suitable for recovering vectors in $\mathcal{H}$ from observations of the form (4.1).

4.1 Proposed Method

Compressive sensing theory provides answers to the question what properties of an observation matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ with $m < N$ can facilitate the recovery of $\boldsymbol{h}$ from (4.1). One widely used property is the restricted isometry property (RIP) in (2.7). Since $\boldsymbol{0} \in \mathbb{C}^N$ fulfills (2.7) for any $\delta \geq 0$ and any $\boldsymbol{A} \in \mathbb{C}^{m\times N}$, we exclude it from the set $\mathcal{H}$ and then write the RIP condition as follows:

$$(1-\delta) \leq \left\| \boldsymbol{A} \frac{\boldsymbol{h}}{\|\boldsymbol{h}\|} \right\|^2 \leq (1+\delta) \quad \text{for all } \boldsymbol{h} \in \mathcal{H}. \tag{4.2}$$

Matrices which fulfill (4.2) with δ as small as possible are desired.

The vector $\overline{\boldsymbol{h}} = \frac{\boldsymbol{h}}{\|\boldsymbol{h}\|} \in \mathbb{C}^N$ in (4.2) lies on the (unit) hypersphere in $\mathbb{C}^N$ and according to (4.2), $\boldsymbol{A}$ maps $\overline{\boldsymbol{h}}$ to a vector $\boldsymbol{A}\overline{\boldsymbol{h}} \in \mathbb{C}^m$ which lies (approximately) on the (unit) hypersphere in $\mathbb{C}^m$—the Euclidean distance between $\boldsymbol{A}\overline{\boldsymbol{h}}$ and the closest point to it on the hypersphere in $\mathbb{C}^m$ is at most δ. In this analogy, finding an $\boldsymbol{A}$ with a small δ corresponds to determining a matrix which brings all $\boldsymbol{A}\overline{\boldsymbol{h}}$ close to the hypersphere in $\mathbb{C}^m$. In this chapter, we aim to find a fixed matrix which has this

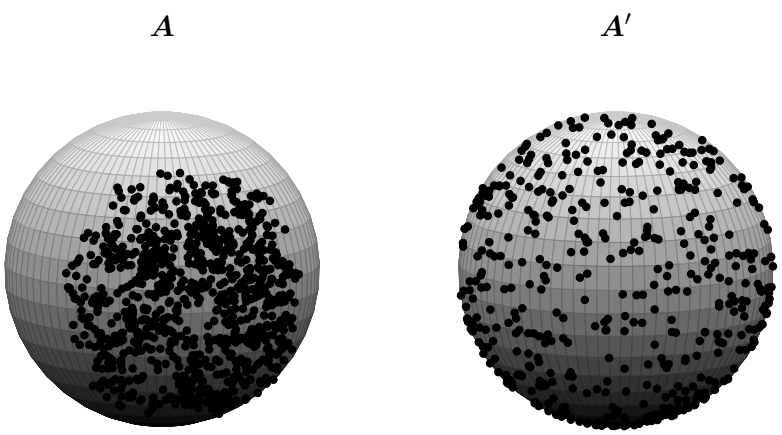

Figure 4.1: The image of 500 points $\boldsymbol{h}_i/\|\boldsymbol{h}_i\|$ with $\boldsymbol{h}_i \in \mathcal{H}$ under two different matrices $\boldsymbol{A}$ (left) and $\boldsymbol{A}'$ (right).

property. Construction methods to obtain a matrix with the RIP deterministically are generally unknown and one typically resorts to drawing random matrices which have the RIP with high probability, cf. Section 2.3. In view of this, it seems advisable to design a probabilistic method to achieve our goal.

To motivate the design of our proposed method, consider the sketch in Figure 4.1. With the hyperspheres interpretation of (4.2) in mind, imagine two matrices are given, $\boldsymbol{A}$ and $\boldsymbol{A}'$ with the RIP, and assume $\delta = 0$ holds for both matrices. That is, both $\boldsymbol{A}$ and $\boldsymbol{A}'$ map all $\overline{\boldsymbol{h}}$ perfectly onto the hypersphere in $\mathbb{C}^m$. Further, imagine $\boldsymbol{A}$ maps all $\overline{\boldsymbol{h}}$ in such a way onto the hypersphere in $\mathbb{C}^m$ that the points $\boldsymbol{A}\overline{\boldsymbol{h}}$ are all clustered around the point $\boldsymbol{e}_1 = (1, 0, 0, \dots)^{\mathrm{T}} \in \mathbb{C}^m$ (i.e., all $\boldsymbol{A}\overline{\boldsymbol{h}}$ are clustered around a "pole"). In contrast, assume $\boldsymbol{A}'$ maps all $\overline{\boldsymbol{h}}$ in such a way onto the hypersphere in $\mathbb{C}^m$ that the points $\boldsymbol{A}'\overline{\boldsymbol{h}}$ are isotropically distributed around the origin.

We would now like to argue that the matrix $\boldsymbol{A}'$ offers more robustness against measurement noise than the matrix $\boldsymbol{A}$. To this end, let $\boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathcal{H}$ be two vectors with $\|\boldsymbol{h}_1\| \neq \|\boldsymbol{h}_2\|$ so that we also have $\|\boldsymbol{A}\boldsymbol{h}_1\| \neq \|\boldsymbol{A}\boldsymbol{h}_2\|$ as well as $\|\boldsymbol{A}'\boldsymbol{h}_1\| \neq \|\boldsymbol{A}'\boldsymbol{h}_2\|$ (because $\delta = 0$ is assumed). This means that both $\boldsymbol{A}$ and $\boldsymbol{A}'$ map $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ onto two different noiseless observations —$\boldsymbol{A}\boldsymbol{h}_1 \neq \boldsymbol{A}\boldsymbol{h}_2$ and $\boldsymbol{A}'\boldsymbol{h}_1 \neq \boldsymbol{A}'\boldsymbol{h}_2$— so that there is a chance of uniquely recovering $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ in both cases. At the same time, note that we have $\|\boldsymbol{A}\boldsymbol{h}_1\| = \|\boldsymbol{A}'\boldsymbol{h}_1\|$ and $\|\boldsymbol{A}\boldsymbol{h}_2\| = \|\boldsymbol{A}'\boldsymbol{h}_2\|$ (because $\delta = 0$ is assumed). Therefore, the main difference between the pairs of noiseless observations $(\boldsymbol{A}\boldsymbol{h}_1, \boldsymbol{A}\boldsymbol{h}_2)$ and $(\boldsymbol{A}'\boldsymbol{h}_1, \boldsymbol{A}'\boldsymbol{h}_2)$ are their relative directions. That is, given the "pole clustering property" of $\boldsymbol{A}$, the vectors $\boldsymbol{A}\boldsymbol{h}_1$ and $\boldsymbol{A}\boldsymbol{h}_2$ tend to point in the same direction (in the

direction of $\boldsymbol{e}_1$) whereas the vectors $\boldsymbol{A}'\boldsymbol{h}_1$ and $\boldsymbol{A}'\boldsymbol{h}_2$ can point in more dissimilar directions and are thus farther apart from one another. In the presence of noise, i.e., if we have observations $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{h}_i + \boldsymbol{n}_i$ and $\boldsymbol{y}_i' = \boldsymbol{A}'\boldsymbol{h}_i + \boldsymbol{n}_i$, it can occur that the noise realizations map $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ close to each other, making them numerically indistinguishable, which, in turn, makes recovering distinct estimates of $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ difficult. At the same time, $\boldsymbol{y}_1'$ and $\boldsymbol{y}_2'$ might still be distinguishable because $\boldsymbol{A}'\boldsymbol{h}_1$ and $\boldsymbol{A}'\boldsymbol{h}_2$ are likely to be farther apart from one another. Generally, we believe that observations under $\boldsymbol{A}'$ are more robust with respect to noise than under $\boldsymbol{A}$, making $\boldsymbol{A}'$ preferable.

In view of this, we aim to find a matrix $\boldsymbol{A}$ which maps all $\overline{\boldsymbol{h}}$ to the hypersphere in $\mathbb{C}^m$ and which at the same time distributes the points isotropically around the origin. The first requirement corresponds to satisfying the RIP condition with a small δ and the latter requirement aims to combat measurement noise. Therefore, in summary, we want all $\boldsymbol{A}\overline{\boldsymbol{h}}$ to be uniformly distributed on the hypersphere in $\mathbb{C}^m$ to achieve both requirements. To express this goal in terms of an optimization problem, we let all vectors in $\mathcal{H}$ be realizations of a random variable $\boldsymbol{h}$ so that also $\overline{\boldsymbol{h}}$ and $\boldsymbol{A}\overline{\boldsymbol{h}}$ are random variables. Note that $\boldsymbol{A}\overline{\boldsymbol{h}}$ is a random variable because $\overline{\boldsymbol{h}}$ is a random variable whereas $\boldsymbol{A}$ is not random. Further, let $\boldsymbol{u}$ be a random variable with a uniform distribution on the (unit) hypersphere in $\mathbb{C}^m$. That is, we have $\|\boldsymbol{u}\| = 1$ and $\boldsymbol{u}$ is isotropically distributed around the origin. If we find a matrix which achieves the stated goal, then $\boldsymbol{A}\overline{\boldsymbol{h}}$ has the same distribution as $\boldsymbol{u}$. Generally, we aim to find an $\boldsymbol{A}$ which brings the distribution of $\boldsymbol{A}\overline{\boldsymbol{h}}$ close to the distribution of $\boldsymbol{u}$. Using the maximum mean discrepancy (MMD) as a metric between probability distributions (cf. Section 2.5), we can encapsulate our goal in the following distribution matching (cf. Section 2.6) optimization problem:

$$\min_{\boldsymbol{A}\in\mathbb{C}^{m\times N}} \mathrm{MMD}_k^2(p_{\boldsymbol{u}}, q_{\boldsymbol{A}\overline{\boldsymbol{h}}}) \tag{4.3}$$

where we write $p_{\boldsymbol{u}}$ and $q_{\boldsymbol{A}\overline{\boldsymbol{h}}}$ for the distributions of $\boldsymbol{u}$ and $\boldsymbol{A}\overline{\boldsymbol{h}}$, respectively. Further, k in (4.3) denotes a kernel (cf. Section 2.5).

A main challenge with (4.3) is that in order to evaluate the objective function, the distribution $q_{\boldsymbol{A}\overline{\boldsymbol{h}}}$ needs to be available in closed form. Even if that is the case, the evaluation can be too difficult. This is a challenge which MMD shares with alternative distances like, e.g., the Kullback-Leibler divergence. An advantage of MMD however is the possibility to estimate $\mathrm{MMD}_k^2(p_{\boldsymbol{u}}, q_{\boldsymbol{A}\overline{\boldsymbol{h}}})$ based on samples of $\boldsymbol{u}$ and $\overline{\boldsymbol{h}}$ (and thereby $\boldsymbol{A}\overline{\boldsymbol{h}}$), which can easily be generated (in case of $\boldsymbol{u}$) or are assumed to be available (in case of $\overline{\boldsymbol{h}}$). This, together with differentiability, makes MMD charming and enables us to employ machine learning techniques to solve the optimization problem (4.3).

4.1.1 Learning a Constant Modulus Matrix

Coming back to the goal of finding a suitable constant modulus matrix, we want to solve

$$\min_{\substack{\boldsymbol{A}\in\mathbb{C}^{m\times N}\\ |[\boldsymbol{A}]_{k,l}|=\frac{1}{\sqrt{m}}}} \mathrm{MMD}_k^2(p_{\boldsymbol{u}}, q_{\boldsymbol{A}\overline{\boldsymbol{h}}}). \tag{4.4}$$

This optimization problem aims to find a constant modulus matrix $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times N}$ that brings $q_{\boldsymbol{A}\overline{\boldsymbol{h}}}$ as close to a uniform distribution on the (unit) hypersphere in $\mathbb{C}^m$ as possible. Obtaining an analytic solution seems difficult in general, for example, because an analytic expression of $q_{\boldsymbol{A}\overline{\boldsymbol{h}}}$ is not available.

However, as detailed in Section 2.6 where we go from problem (2.17) to problem (2.18), we can use samples $\{\overline{\boldsymbol{h}}_t\}_{t=1}^{T_{\text{tr}}}$ and $\{\boldsymbol{u}_t\}_{t=1}^{T_{\text{tr}}}$ of both $\overline{\boldsymbol{h}}$ and $\boldsymbol{u}$ to solve an approximation of (4.4). A random variable $\boldsymbol{u}$ with uniform distribution on the (unit) hypersphere in $\mathbb{C}^m$ is given by normalization of an isotropic Gaussian random variable, e.g., $\boldsymbol{u} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$ with $\boldsymbol{v} \sim \mathcal{N}_{\mathbb{C}}(\boldsymbol{0}, \boldsymbol{I})$, which means that sampling is simple. If

$$\mathrm{stk} : \mathbb{C}^m \to \mathbb{R}^{2m},\ \boldsymbol{z} \mapsto \mathrm{stk}(\boldsymbol{z}) = [\Re(\boldsymbol{z})^{\mathrm{T}}, \Im(\boldsymbol{z})^{\mathrm{T}}]^{\mathrm{T}} \tag{4.5}$$

denotes stacking real and imaginary parts of a complex vector into a real vector, we can formulate an approximation of (4.4):

$$\min_{\substack{\boldsymbol{A}\in\mathbb{C}^{m\times N}\\ |[\boldsymbol{A}]_{k,l}|=\frac{1}{\sqrt{m}}}} \mathrm{MMD}_k^2\left(\{\mathrm{stk}(\boldsymbol{u}_t)\}_{t=1}^{T_{\text{tr}}}, \{\mathrm{stk}(\boldsymbol{A}\overline{\boldsymbol{h}}_t)\}_{t=1}^{T_{\text{tr}}}\right). \tag{4.6}$$

We would further like to express (4.6) in terms of real-valued optimization variables. Every entry of $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times N}$ has the form $[\boldsymbol{A}]_{k,l} = \frac{1}{\sqrt{m}}\exp(\mathrm{j}\,\phi_{k,l})$ with a phase $\phi_{k,l} \in \mathbb{R}$ so that $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times N} \subset \mathbb{C}^{m\times N}$ is parameterized by mN real-valued parameters which we collect in the matrix $\boldsymbol{\Phi} \in \mathbb{R}^{m\times N}$. Applying $\sin$ and $\cos$ elementwise, we can write

$$\boldsymbol{A} = \Re(\boldsymbol{A}) + \mathrm{j}\,\Im(\boldsymbol{A}) = \frac{1}{\sqrt{m}}\cos(\boldsymbol{\Phi}) + \mathrm{j}\,\frac{1}{\sqrt{m}}\sin(\boldsymbol{\Phi}) \tag{4.7}$$

and $\boldsymbol{A}\overline{\boldsymbol{h}} = (\Re(\boldsymbol{A}) + \mathrm{j}\,\Im(\boldsymbol{A}))(\Re(\overline{\boldsymbol{h}}) + \mathrm{j}\,\Im(\overline{\boldsymbol{h}}))$ yields

$$\mathrm{stk}(\boldsymbol{A}\overline{\boldsymbol{h}}) = \begin{bmatrix}\Re(\boldsymbol{A}\overline{\boldsymbol{h}})\\ \Im(\boldsymbol{A}\overline{\boldsymbol{h}})\end{bmatrix} = \underbrace{\begin{bmatrix}\frac{\cos(\boldsymbol{\Phi})}{\sqrt{m}} & -\frac{\sin(\boldsymbol{\Phi})}{\sqrt{m}}\\ \frac{\sin(\boldsymbol{\Phi})}{\sqrt{m}} & \frac{\cos(\boldsymbol{\Phi})}{\sqrt{m}}\end{bmatrix}}_{=\boldsymbol{A}(\boldsymbol{\Phi})} \begin{bmatrix}\Re(\overline{\boldsymbol{h}})\\ \Im(\overline{\boldsymbol{h}})\end{bmatrix} = \boldsymbol{A}(\boldsymbol{\Phi})\,\mathrm{stk}(\overline{\boldsymbol{h}}) \tag{4.8}$$

Algorithm 4 Learning $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m \times N}$

Require: training data $\{(\overline{\boldsymbol{h}}_t, \boldsymbol{u}_t)\}_{t=1}^{T_{\text{tr}}}$

1: randomly initialize $\boldsymbol{\Phi} \in \mathbb{R}^{m \times N}$
2: **while** termination criterion not met **do**
3: draw a batch of T samples uniformly from the training data: $\{(\overline{\boldsymbol{h}}_t, \boldsymbol{u}_t)\}_{t=1}^{T} \subset \{(\overline{\boldsymbol{h}}_t, \boldsymbol{u}_t)\}_{t=1}^{T_{\text{tr}}}$
4: compute the stochastic gradient

$$\boldsymbol{g}_{\boldsymbol{\Phi}} = \frac{\partial}{\partial \boldsymbol{\Phi}} \operatorname{MMD}_k^2 \left(\{\operatorname{stk}(\boldsymbol{u}_t)\}_{t=1}^{T}, \{\boldsymbol{A}(\boldsymbol{\Phi}) \operatorname{stk}(\overline{\boldsymbol{h}}_t)\}_{t=1}^{T} \right)$$

5: update $\boldsymbol{\Phi}$ with a gradient algorithm using $\boldsymbol{g}_{\boldsymbol{\Phi}}$
6: **end while**

where the notation $\boldsymbol{A}(\boldsymbol{\Phi}) \in \mathbb{R}^{2m \times 2N}$ emphasizes that $\boldsymbol{A}$ only depends on the mN real parameters in $\boldsymbol{\Phi}$. The optimization problem with real-valued optimization parameters is thus

$$\min_{\boldsymbol{\Phi} \in \mathbb{R}^{m \times N}} \operatorname{MMD}_k^2 \left(\{\operatorname{stk}(\boldsymbol{u}_t)\}_{t=1}^{T_{\text{tr}}}, \{\boldsymbol{A}(\boldsymbol{\Phi}) \operatorname{stk}(\overline{\boldsymbol{h}}_t)\}_{t=1}^{T_{\text{tr}}} \right). \tag{4.9}$$

As explained in Section 2.6, such a problem is typically solved via stochastic gradient descent, see Algorithm 4. An example of how the kernel k can be chosen and of how good hyperparameters of the gradient algorithm in Algorithm 4 can be found is presented in Section 4.6.3.

Calculating the gradient in Algorithm 4 has the same order of computational complexity as evaluating MMD_k^2 in the forward pass if backpropagation is used [52]. One evaluation of MMD_k^2 for a batch of T samples requires $\mathcal{O}(T^2)$ evaluations of the kernel, see (2.16), where the vectors have dimension $2m$ due to stacked real and imaginary parts. Evaluating the kernel can be done in $\mathcal{O}(m)$, see (2.15). Computing $\boldsymbol{A}(\boldsymbol{\Phi}) \operatorname{stk}(\overline{\boldsymbol{h}})$ can be done in $\mathcal{O}(mN)$. Therefore, the computational complexity of evaluating MMD_k^2 is $\mathcal{O}(mNT + mT^2)$. To save computation time, it can be interesting to use a linear (in T) version of MMD which can be found in [2, Lemma 14]. This would result in a complexity of $\mathcal{O}(mNT)$.

4.2 Related Literature

The following reviews different approaches to design an observation matrix. It is worth pointing out that the approaches focus on real-valued observation matrices without

constraints, which differs from the approach presented in Section 4.1 where we are looking for a complex-valued constant modulus matrix in $\mathcal{M}_{\text{const}}^{m\times N}$. The only exception is the very recently introduced algorithm in [53] which also studies constant modulus matrices, see Section 4.4. The comparison of Algorithm 4 to the method in [53] is new in this dissertation and not present in [51]. Further, we modify an algorithm from [54] such that it also yields constant modulus matrices, see Section 4.3.1.

The proposed Algorithm 4 and all reviewed approaches have a data dependency in common. This is the case because either data samples are used explicitly or because the obtained matrix is based on a dictionary which, in turn, is typically suitable for a particular data set. In contrast, random matrices, which are the default choice in compressive sensing applications (cf. Section 2.3), enjoy a universality property which makes them applicable with almost any dictionary and thus data [10]. The data dependency, therefore, introduces a tradeoff where universality is traded for a potentially better performance on a given data set.

4.2.1 Literature Review

The real-valued signal model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{n}$ with $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ is studied in [55]. Given a set $\{\boldsymbol{h}_t\}_{t=1}^{T_{\text{tr}}}$ of samples, the proposed *Uncertain Component Analysis* determines

$$\boldsymbol{A} = \underset{\tilde{\boldsymbol{A}}\in\mathbb{R}^{m\times N},\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^{\mathrm{T}}=\boldsymbol{I}}{\arg\max} \prod_{t=1}^{T_{\text{tr}}} \Pr(\boldsymbol{h}_t \mid \boldsymbol{y}_t; \tilde{\boldsymbol{A}}) \tag{4.10}$$

with $\boldsymbol{y}_t = \tilde{\boldsymbol{A}}\boldsymbol{h}_t$ and where $\Pr(\boldsymbol{h}_t \mid \boldsymbol{y}_t; \tilde{\boldsymbol{A}})$ is the posterior probability of the data. Two fixed-point equations are used to solve the optimization problem algorithmically.

The author of [56] works with the signal model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{\Psi}\boldsymbol{s}$ with $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ and where $\boldsymbol{\Psi} \in \mathbb{R}^{N\times L}$ $(N < L)$ is a dictionary such that $\boldsymbol{s}$ is sparse. The author proposes minimizing the t-averaged mutual coherence $\mu_t(\boldsymbol{A}\boldsymbol{\Psi})$ with respect to $\boldsymbol{A}$. To this end, the columns of $\boldsymbol{A}\boldsymbol{\Psi}$ are normalized and then the Gram matrix $\boldsymbol{G} = (\boldsymbol{A}\boldsymbol{\Psi})^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{\Psi})$ is computed. The t-averaged mutual coherence is now given by the average of all entries with $|[\boldsymbol{G}]_{k,l}| \geq t$. An iterative algorithm which alternates between two stages is developed to achieve the goal. The algorithm chooses a *down-scaling factor* $\gamma \in (0, 1)$ and starts with an initial $\boldsymbol{A}^{(1)}$. In iteration i, the entries of $\boldsymbol{G}^{(i)} = (\boldsymbol{A}^{(i)}\boldsymbol{\Psi})^{\mathrm{T}}(\boldsymbol{A}^{(i)}\boldsymbol{\Psi})$, which have an absolute value larger than t, are multiplied by γ to *shrink* them. This yields a matrix $\hat{\boldsymbol{G}}^{(i)}$ which is decomposed in the algorithm's second stage. The decomposition aims to find $\boldsymbol{A}^{(i+1)}$ such that $(\boldsymbol{A}^{(i+1)}\boldsymbol{\Psi})^{\mathrm{T}}(\boldsymbol{A}^{(i+1)}\boldsymbol{\Psi})$ is close to $\hat{\boldsymbol{G}}^{(i)}$. The two stages are alternated for a predetermined number of iterations.

In the same setting as [56], the Gram matrix $\boldsymbol{G} = (\boldsymbol{A}\boldsymbol{\Psi})^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{\Psi})$ is utilized to find $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ in [57]. There, the goal is to bring $\boldsymbol{G}$ close to the identity matrix. After a random initialization, a KSVD-like algorithm determines $\boldsymbol{A}$. Further, [57] proposes a modification of the approach where also the dictionary $\boldsymbol{\Psi}$ can be optimized.

Also the authors of [58] work with a Gram matrix. Their goal is to determine $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ such that a bi-Lipschitz condition is fulfilled. Specifically, given a set $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$ of samples, the goal is to establish the RIP for all differences $\boldsymbol{h}_t - \boldsymbol{h}_s$. To this end, the secant set

$$\mathcal{S} = \left\{ \frac{\boldsymbol{h}_t - \boldsymbol{h}_s}{\|\boldsymbol{h}_t - \boldsymbol{h}_s\|} : 1 \leq t < s \leq T_{\mathrm{tr}} \right\} \tag{4.11}$$

is defined and the bi-Lipschitz criterion is expressed as

$$|\|\boldsymbol{A}\boldsymbol{v}\|^2 - 1| = |\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v} - 1| = |\boldsymbol{v}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{v} - 1| \leq \delta \tag{4.12}$$

for all $\boldsymbol{v} \in \mathcal{S}$ and for $\boldsymbol{G} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$. The motivating optimization problem then is

$$\min_{\boldsymbol{G}\in\mathbb{R}^{N\times N}} \sup_{\boldsymbol{v}\in\mathcal{S}} |\boldsymbol{v}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{v} - 1| \quad \text{s.t.} \quad \boldsymbol{G} \succeq 0, \quad \operatorname{rank}(\boldsymbol{G}) = r, \quad \operatorname{trace}(\boldsymbol{G}) = b \tag{4.13}$$

and a relaxation thereof is solved. The related problem

$$\min_{\boldsymbol{G}\in\mathbb{R}^{N\times N}, \boldsymbol{G}=\boldsymbol{G}^{\mathrm{T}}, \boldsymbol{G}\succeq 0} \operatorname{trace}(\boldsymbol{G}) \quad \text{s.t.} \quad \sup_{\boldsymbol{v}\in\mathcal{S}} |\boldsymbol{v}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{v} - 1| \leq \delta \tag{4.14}$$

is studied in [59]. In both cases, an eigendecomposition of the optimal $\boldsymbol{G}$ yields the desired $\boldsymbol{A}$.

Neural networks are used in [60] in the context of the signal model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{s}$ with nonnegative sparse vectors $\boldsymbol{s} \in \mathbb{R}_+^N$ and $\boldsymbol{A} \in \mathbb{R}^{m\times N}$. The matrix $\boldsymbol{A}$ is learned by *unrolling* the update rule of a projected subgradient method. The authors of [61] then employ the same unrolling method for channel state information feedback. There, the matrix $\boldsymbol{A} \in \mathbb{R}^{m\times 2N}$ is used to compress a channel $\boldsymbol{h} \in \mathbb{C}^N$ as $\boldsymbol{y} = \boldsymbol{A}[\Re(\boldsymbol{h})^{\mathrm{T}}, \Im(\boldsymbol{h})^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^m$, allowing the base station to recover $\boldsymbol{h}$ from the fed back signal $\boldsymbol{y}$ using compressive sensing methods. The approach is modified in [62] for channel estimation. To this end, real and imaginary parts of channels $\boldsymbol{h} \in \mathbb{C}^N$ are treated as independent data points such that a real-valued matrix $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ can be learned.

Algorithm 5 Learning-Based Compressive Subsampling (LBCS) [54]

Require: training data $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$
Require: matrix $\boldsymbol{V} \in \mathbb{C}^{N\times N}$, number m of rows to select
1: normalize the training data: $\{\overline{\boldsymbol{h}}_t\}_{t=1}^{T_{\mathrm{tr}}} \leftarrow \left\{\frac{\boldsymbol{h}_t}{\|\boldsymbol{h}_t\|}\right\}_{t=1}^{T_{\mathrm{tr}}}$
2: **for** $r = 1$ to N **do**
3: $\quad \alpha_r \leftarrow \sum_{t=1}^{T_{\mathrm{tr}}} |\boldsymbol{v}_r^* \overline{\boldsymbol{h}}_t|^2$ *// $\boldsymbol{v}_r$ is row r of $\boldsymbol{V}$*
4: **end for**
5: $\Omega \leftarrow$ indices r which correspond to m largest α_r
6: **return** $\boldsymbol{P}_\Omega \boldsymbol{V}$ *// learned observation matrix*

4.3 Learning-Based Compressive Subsampling

The methods summarized in Section 4.2 lead to an unconstrained real-valued matrix $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ whereas we are looking for a complex-valued constant modulus matrix $\boldsymbol{A} \in \mathcal{M}_{\mathrm{const}}^{m\times N}$. The methods which are based on a decomposition $\boldsymbol{G} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ of a Gram matrix seem ill-suited for our task because such a decomposition will generally not exist if the entries are constrained to have a constant modulus. However, it seems simple to adapt the algorithm in [54] to the constant modulus constraint. After introducing the algorithm, we present a modified version in Section 4.3.1.

The signal model in [54] is

$$\boldsymbol{y} = \boldsymbol{P}_\Omega \boldsymbol{V} \boldsymbol{h} \tag{4.15}$$

for a unitary matrix $\boldsymbol{V} \in \mathbb{C}^{N\times N}$. Here, $\Omega \subset \{1, 2, \ldots, N\}$ is an index set and $\boldsymbol{P}_\Omega \in \mathbb{C}^{m\times N}$ selects the rows of $\boldsymbol{V}$ corresponding to the indices in Ω. The observation matrix is then given by $\boldsymbol{A} = \boldsymbol{P}_\Omega \boldsymbol{V}$. Selecting the indices randomly is a known strategy to obtain a suitable $\boldsymbol{A}$. However, the authors of [54] propose to select the row indices deterministically based on a set $\{\overline{\boldsymbol{h}}_t\}_{t=1}^{T_{\mathrm{tr}}}$ of normalized samples since this strategy can improve upon random indices. The authors of [54] select the m rows $\boldsymbol{v}_r \in \mathbb{C}^{1\times N}$ of $\boldsymbol{V}$ which maximize the average captured energy

$$\frac{1}{T_{\mathrm{tr}}} \sum_{t=1}^{T_{\mathrm{tr}}} \sum_{r\in\Omega} |\boldsymbol{v}_r^* \overline{\boldsymbol{h}}_t|^2 \tag{4.16}$$

of the training set. We call the algorithm "learning-based compressive subsampling (LBCS)" after the paper's title. The algorithm is summarized in Algorithm 5.

Algorithm 6 Monte Carlo LBCS

Require: training data $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}$, validation data $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{val}}}$
Require: evaluation parameters par, number m of rows to select, iterations I
1: $\boldsymbol{A}_{\mathrm{MC}} \leftarrow \boldsymbol{0}$, $\mathrm{error}_{\mathrm{MC}} \leftarrow \infty$
2: **for** $i = 1$ to I **do**
3: draw a random unitary matrix $\boldsymbol{V} \in \mathbb{C}^{N \times N}$
4: divide all entries of $\boldsymbol{V}$ by their absolute values
5: normalize the rows of $\boldsymbol{V}$
6: $\boldsymbol{A} \leftarrow \mathrm{LBCS}(\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tr}}}, \boldsymbol{V}, m)$
7: $\mathrm{error} \leftarrow \mathrm{Evaluation}(\boldsymbol{A}, \{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{val}}}, \mathrm{par})$
8: **if** $\mathrm{error} < \mathrm{error}_{\mathrm{MC}}$ **then**
9: $\mathrm{error}_{\mathrm{MC}} \leftarrow \mathrm{error}$, $\boldsymbol{A}_{\mathrm{MC}} \leftarrow \boldsymbol{A}$
10: **end if**
11: **end for**
12: **return** $\boldsymbol{A}_{\mathrm{MC}}$

4.3.1 Modified LBCS Algorithm

Since Algorithm 5 selects rows of a given matrix $\boldsymbol{V} \in \mathbb{C}^{N \times N}$, we get a constant modulus matrix whenever $\boldsymbol{V}$ is a constant modulus matrix. Thus, the first modification is to initialize Algorithm 5 with a constant modulus matrix. The second modification is due to the fact that the algorithm was derived for noiseless observations whereas we work with the signal model (4.1). The main idea is to run Algorithm 5 multiple times (I times in Algorithm 6) for multiple random initializations, to then evaluate all the obtained matrices on a held out set $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{val}}}$ of samples, and to return the best performing matrix. The modified algorithm is outlined in Algorithm 6. An algorithm for generating random unitary matrices (cf. Line 3) can be found, e.g., in [63]. The evaluation (cf. Line 7) depends on the desired goal. Since we are ultimately interested in channel estimation, in Section 4.6, we evaluate the current observation matrix by running a channel estimation algorithm on the data set and by returning the corresponding mean square error (MSE). In this case, we have $\mathrm{error} = \mathrm{MSE}$. Generally, the evaluation needs to return an error and the matrix which achieves the smallest error is returned by Algorithm 6. An example of this part of the algorithm is explained in Section 4.6.

Algorithm 7 PGDM [53]

Require: initial $\boldsymbol{A}^{(0)} \in \mathbb{C}^{m\times N}$ with $|[\boldsymbol{A}^{(0)}]_{k,l}| = 1$, dictionary $\boldsymbol{\Psi} \in \mathbb{C}^{N\times L}$

Require: $\alpha \geq 1$, step size ζ, iterations I

$\beta \leftarrow \sqrt{\frac{L-m}{m(L-1)}}$

for $i = 1$ to I **do**

$\boldsymbol{Q} \leftarrow \boldsymbol{A}^{(i-1)}\boldsymbol{\Psi}$

$\boldsymbol{D} \leftarrow \operatorname{diag}\left(\frac{1}{\|\boldsymbol{q}_1\|}, \ldots, \frac{1}{\|\boldsymbol{q}_L\|}\right)$ *// $\boldsymbol{q}_j$ is column j of $\boldsymbol{Q}$*

$\boldsymbol{E} \leftarrow \boldsymbol{D}\boldsymbol{Q}^{\mathrm{H}}\boldsymbol{Q}\boldsymbol{D} - \boldsymbol{I}$

compute $\tilde{\boldsymbol{E}}$ with elements

$$[\tilde{\boldsymbol{E}}]_{k,l} \leftarrow \begin{cases} 0, & |[\boldsymbol{E}]_{k,l}| < \alpha\beta \\ \frac{[\boldsymbol{E}]_{k,l}}{|[\boldsymbol{E}]_{k,l}|}\left(|[\boldsymbol{E}]_{k,l}| - \alpha\beta\right), & \text{otherwise} \end{cases}$$

$\boldsymbol{C} \leftarrow 2\tilde{\boldsymbol{E}}\boldsymbol{D}\boldsymbol{Q}^{\mathrm{H}}\boldsymbol{Q}\boldsymbol{D}^3$

compute the gradient

$$\boldsymbol{g} \leftarrow 4\boldsymbol{Q}\boldsymbol{D}\tilde{\boldsymbol{E}}\boldsymbol{D}\boldsymbol{\Psi}^{\mathrm{H}} - 2\boldsymbol{A}^{(i-1)}\boldsymbol{\Psi}\operatorname{diag}(\boldsymbol{C})\boldsymbol{\Psi}^{\mathrm{H}}$$

$\tilde{\boldsymbol{A}} \leftarrow \boldsymbol{A}^{(i-1)} - \zeta\boldsymbol{g}$ *// gradient descent*

compute $\boldsymbol{A}^{(i)}$ with elements $[\boldsymbol{A}^{(i)}]_{k,l} \leftarrow \frac{[\tilde{\boldsymbol{A}}]_{k,l}}{|[\tilde{\boldsymbol{A}}]_{k,l}|}$ *// projection*

end for

return $\frac{1}{\sqrt{m}}\boldsymbol{A}^{(I)}$

4.4 Projected Gradient Descent Method

In [53], the setting of this chapter is studied and the goal is to obtain a constant modulus matrix. A dictionary $\boldsymbol{\Psi} \in \mathbb{C}^{N\times L}$ for the data of interest is assumed to be given. Using the abbreviation $\boldsymbol{\Phi} = \boldsymbol{A}\boldsymbol{\Psi}$, the authors want to minimize the mutual coherence

$$\mu(\boldsymbol{\Phi}) = \max_{k\neq l} \frac{|\boldsymbol{\phi}_k^{\mathrm{H}}\boldsymbol{\phi}_l|}{\|\boldsymbol{\phi}_k\|\|\boldsymbol{\phi}_l\|} \tag{4.17}$$

where $\boldsymbol{\phi}_k$ is column k of $\boldsymbol{\Phi}$. Since the corresponding optimization problem is difficult, the authors instead solve

$$\min_{\boldsymbol{A}\in\mathbb{C}^{m\times N}} \|\boldsymbol{\Phi}^{\mathrm{H}}\boldsymbol{\Phi} - \boldsymbol{I}\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{\phi}_k\| = 1 \quad \text{and} \quad |\boldsymbol{A}_{k,l}| = 1 \tag{4.18}$$

Algorithm	Goal	Method	Data Dependency
Alg. 4 (MMD)	spherical uniform distribution	distribution matching with maximum mean discrepancy	via training data
Alg. 6 (LBCS)	maximize average captured energy	selecting rows	via training data
Alg. 7 (PGDM)	minimize coherence	projected gradient descent on surrogate objective	via dictionary

Table 4.1: Three algorithms to obtain a constant modulus matrix. Algorithm 4 is proposed in this chapter, Algorithm 6 is our modification of [54], and Algorithm 7 is from [53].

via a projected gradient descent method (PGDM), where $\|\cdot\|_F$ is the Frobenius norm.

The details of PGDM are summarized in Algorithm 7. The algorithm's hyperparameters α, ζ, I need to be determined via a grid or a random search (cf. Section 2.7). The hyperparameters which lead to the smallest coherence are considered to be the best. The initialization of $\boldsymbol{A}^{(0)}$ is another hyperparameter. In [53], the m leading eigenvalues and corresponding eigenvectors of the eigenvalue decomposition $\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{H}} = \boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathrm{H}}$ are used to compute $\tilde{\boldsymbol{A}} = \boldsymbol{\Lambda}_m^{-1/2}\boldsymbol{U}_m^{\mathrm{H}}$ which is then projected as in Line 10 of Algorithm 7. However, since $\tilde{\boldsymbol{A}}$ can contain zeros, the projection and therefore this initialization is not always possible. In that case, we initialize $\boldsymbol{A}^{(0)}$ with independent random elements of the form $e^{\mathrm{j}\,\phi}$ with ϕ uniformly drawn in the interval $[0, 2\pi]$ in the numerical experiments.

4.5 Comparison

With Algorithm 4, Algorithm 6, and Algorithm 7, we have three different methods to obtain a constant modulus matrix. Table 4.1 briefly summarizes the different approaches.

All algorithms share a grid search or a random search aspect: Algorithms 4 and 7 depend on a good choice of their hyperparameters (e.g., learning rate) which are found by running the algorithms multiple times with different hyperparameter values, and Algorithm 6 searches over multiple random matrix initializations. Another similarity of the algorithms is data dependency: Algorithms 4 and 6 require training data and Algorithm 7 requires a dictionary which is typically tailored for a particular data set.

The necessity of a dictionary can be viewed as an advantage and as a disadvantage. On the one hand, if the obtained matrix is used in conjunction with a recovery algorithm which also requires a dictionary, then the matrix and the recovery algorithm might harmonize well. On the other hand, if the recovery algorithm does not need a dictionary or if no dictionary for a given data set is known, it might not be beneficial or it might be difficult to employ Algorithm 7.

A difference between Algorithms 4 and 7 and Algorithm 6 is that the latter requires an evaluation function to decide for the best matrix (Line 7 in Algorithm 6). In the following numerical simulations, we evaluate the current matrix via a channel estimation algorithm. As such, the obtained matrix depends on this algorithm (and on the signal-to-noise ratio (SNR)) which might not be desirable. We discuss the dependency of Algorithm 6 on the evaluation function in more detail in Chapter 5 where we combine the matrix design algorithms with the channel estimation algorithm from Chapter 3. In the remainder of this chapter, we compare the three algorithms in Table 4.1 in compressive sensing recovery tasks.

4.6 Numerical Evaluation

We investigate three data models which have an (approximate) sparsity in common and are thus typical for compressive sensing applications. We continue to use the nomenclature established in the dissertation so far. Only the third data model can be considered a channel model but we speak of channel models and channel estimation nonetheless. We are interested in assessing how well-suited the matrices determined via Algorithm 4, Algorithm 6, or Algorithm 7 are for the problem of channel estimation from observations of the form (4.1). To this end, we employ the well-known compressive sensing algorithm orthogonal matching pursuit (OMP) (see Algorithm 1 in Section 2.3) to perform channel estimation using different matrices and evaluate the corresponding MSE as the figure of merit. Ultimately, the choice to use OMP as a channel estimation algorithm is arbitrary. It is, however, a widely used and fast algorithm. Our main interest lies in observing the effect of different matrices on the channel estimation performance. In Chapter 5, we discuss the performance of the Gaussian mixture model (GMM) estimator from Chapter 3 when it is used with matrices learned via the algorithms in this chapter.

4.6.1 Channel Models and Estimation

The first channel model is

$$\boldsymbol{h} = \boldsymbol{s} \tag{4.19}$$

where $\boldsymbol{s} \in \mathbb{C}^N$ is a sparse vector with p nonzero entries. Second, we consider channels

$$\boldsymbol{h} = \boldsymbol{F}\boldsymbol{s} \tag{4.20}$$

which are p-sparse in the discrete Fourier transform (DFT) basis $\boldsymbol{F} \in \mathbb{C}^{N \times N}$. In both cases, the nonzero entries are drawn independently from $\mathcal{N}_{\mathbb{C}}(0, \frac{1}{p})$. Third, channels

$$\boldsymbol{h} = \sum_{k=1}^{p} s_k \boldsymbol{a}(\theta_k) \tag{4.21}$$

are modeled as a weighted sum of steering vectors

$$\boldsymbol{a}(\theta) = [1, e^{\mathrm{j}\,\pi \sin(\theta)}, \ldots, e^{\mathrm{j}\,\pi(N-1)\sin(\theta)}]^{\mathrm{T}}. \tag{4.22}$$

The path angles θ_k are drawn uniformly in $[0, 2\pi]$ and the path gains s_k are drawn from $\mathcal{N}_{\mathbb{C}}(0, \frac{1}{p})$.

As explained in Section 2.3, OMP makes use of a dictionary for channel estimation. For the channel models (4.19) and (4.20), we can use the dictionaries $\boldsymbol{\Psi} = \boldsymbol{I}_N$ and $\boldsymbol{\Psi} = \boldsymbol{F}$, respectively, to express the channel as $\boldsymbol{h} = \boldsymbol{\Psi}\boldsymbol{s}$. For the channel model (4.21), we define a dictionary

$$\boldsymbol{\Psi}_L = \begin{bmatrix} \boldsymbol{a}(\hat{\theta}_1) & \ldots & \boldsymbol{a}(\hat{\theta}_L) \end{bmatrix} \in \mathbb{C}^{N \times L} \tag{4.23}$$

of L steering vectors which correspond to L equidistantly sampled angles $\hat{\theta}_l$ in $[0, 2\pi]$. We can then approximate a channel by means of a linear combination of the steering vectors in $\boldsymbol{\Psi}_L$: $\boldsymbol{h} \approx \boldsymbol{\Psi}_L \boldsymbol{s}$ where $\boldsymbol{s} \in \mathbb{C}^L$ is a p-sparse vector, cf., e.g., [44]. In all three cases, we write

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{\Psi}\boldsymbol{s} + \boldsymbol{n} \tag{4.24}$$

with either $\boldsymbol{\Psi} = \boldsymbol{I}_N$, $\boldsymbol{\Psi} = \boldsymbol{F}$, or $\boldsymbol{\Psi} = \boldsymbol{\Psi}_L$, and we use OMP from Algorithm 1 with $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{\Psi}$ to recover a p-sparse estimate $\hat{\boldsymbol{s}}$ of $\boldsymbol{s}$ so that the channel estimate is given by $\hat{\boldsymbol{h}} = \boldsymbol{\Psi}\hat{\boldsymbol{s}}$, see also Section 2.3. In summary, the three channel models represent: (i) sparse vectors, (ii) vectors which are sparse in a nontrivial basis, and (iii) vectors which are approximately sparse.

4.6.2 Simulation Setup

The noise in (4.1) is Gaussian: $\boldsymbol{n} \sim \mathcal{N}_{\mathbb{C}}(\boldsymbol{0}_m, \sigma^2 \boldsymbol{I}_m)$. We define the SNR as

$$\mathrm{SNR} = \frac{\|\boldsymbol{A}\boldsymbol{h}\|^2}{\mathrm{E}[\|\boldsymbol{n}\|^2]} = \frac{\|\boldsymbol{A}\boldsymbol{h}\|^2}{m\sigma^2} \tag{4.25}$$

where σ^2 is adjusted for every pair $(\boldsymbol{A}, \boldsymbol{h})$ to achieve the desired SNR. To measure the channel estimation performance, we generate $T_{\mathrm{tst}} = 10^4$ test channel samples $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tst}}}$, generate corresponding observations according to (4.1), and compute channel estimates $\{\hat{\boldsymbol{h}}_t\}_{t=1}^{T_{\mathrm{tst}}}$ using OMP. While the channel estimation algorithm is always OMP, there is a difference in the matrix $\boldsymbol{A}$ that is used to obtain the observations. We distinguish five cases for a given channel model:

- *MMD*: A single observation matrix is learned via Algorithm 4 and all observations at all SNRs are computed using this one matrix.
- *PGDM*: A single observation matrix is obtained via Algorithm 7 and all observations at all SNRs are computed using this one matrix.
- *MMD with PGDM init.*: Algorithm 4 is initialized with the matrix obtained via Algorithm 7 and all observations at all SNRs are computed using the resulting matrix.
- *LBCS*: For every SNR, a matrix is obtained via Algorithm 6 and all observations at a given SNR are computed using the corresponding matrix.
- *random*: For every SNR and every channel $\boldsymbol{h}_t$, a new random constant modulus matrix with entries according to (2.9) is drawn and used to compute a corresponding observation.

The channel estimation performance is always measured in terms of relative MSE: $\sum_{t=1}^{T_{\mathrm{tst}}} \|\boldsymbol{h}_t - \hat{\boldsymbol{h}}_t\|^2 / \sum_{t=1}^{T_{\mathrm{tst}}} \|\boldsymbol{h}_t\|^2$. The whole procedure is summarized in Algorithm 8 for a given SNR. For the *random* case, a new random constant modulus matrix is drawn between Line 2 and Line 3 in Algorithm 8. It is worth pointing out that the random case has a higher computational complexity because the product $\boldsymbol{A}\boldsymbol{\Psi}$ has to be computed for every channel, see Line 5 in Algorithm 8, whereas it can be precomputed in all other cases.

4.6.3 Simulation Parameters

When OMP is used in conjunction with the channel model in (4.21), we use the dictionary $\boldsymbol{\Psi} = \boldsymbol{\Psi}_L$ from (4.23) with $L = 16N$.

Algorithm 6 is used with $T_{\mathrm{tr}} = 10^5$ and $T_{\mathrm{val}} = 10^3$. The OMP algorithm is used for the evaluation in Line 7. The evaluation parameters par are the SNR, the sparsity p, and the OMP dictionary $\boldsymbol{\Psi}$. We always run $I = 100$ iterations.

Algorithm 8 Evaluation

Require: observation matrix $\boldsymbol{A}$, data $\{\boldsymbol{h}_t\}_{t=1}^{T_{\mathrm{tst}}}$
Require: SNR, sparsity p, dictionary $\boldsymbol{\Psi}$
$e \leftarrow 0$, $\Delta \leftarrow 0$
for $t = 1$ to T_{tst} **do**
  using $(\mathrm{SNR}, \boldsymbol{A}, \boldsymbol{h}_t)$, get σ via (4.25), draw $\boldsymbol{n} \sim \mathcal{N}_{\mathbb{C}}(\boldsymbol{0}_m, \sigma^2 \boldsymbol{I}_m)$
  $\boldsymbol{y} \leftarrow \boldsymbol{A}\boldsymbol{h}_t + \boldsymbol{n}$ *// compute the observation*
  $\hat{\boldsymbol{h}} \leftarrow \boldsymbol{\Psi}\mathrm{OMP}(\boldsymbol{A}\boldsymbol{\Psi}, \boldsymbol{y}, p)$ *// estimate the channel $\boldsymbol{h}_t$ using OMP*
  $e \leftarrow e + \|\boldsymbol{h}_t\|^2$, $\Delta \leftarrow \Delta + \|\hat{\boldsymbol{h}} - \boldsymbol{h}_t\|^2$ *// square error*
end for
return $\frac{\Delta}{e}$ *// relative mean square error*

Algorithm 7 is used with the OMP dictionary $\boldsymbol{\Psi}$ and with an initial $\boldsymbol{A}^{(0)}$ as described in Section 4.4. The number of iterations is always $I = 500$ and the remaining hyperparameters α and ζ are determined as follows. First, we run Algorithm 7 with all combinations of

$$\alpha \in \{1.0 + i \cdot 0.1 : i \in \{0, 1, \ldots, 10\}\} \cup \{5.0, 10.0, 50.0, 100.0\} \tag{4.26}$$

and $\zeta \in \{10^i : i \in \{-7, \ldots, 1\}\}$ and determine the best (smallest coherence, see Section 4.4) combination $(\alpha^\star, \zeta^\star)$. Then, we run Algorithm 7 another 10 times with $\alpha \in [0.9\alpha^\star, 1.1\alpha^\star]$ and $\zeta \in [0.9\zeta^\star, 1.1\zeta^\star]$ chosen randomly (random search, cf. Section 2.7). Among all these 145 evaluations of Algorithm 7, the matrix which leads to the smallest coherence is used for the numerical experiments.

For Algorithm 4, we use MMD_k in (2.16) with the kernel

$$k : \mathbb{R}^{2m} \times \mathbb{R}^{2m},\ (\boldsymbol{x}, \boldsymbol{y}) \mapsto k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\sigma \in \mathcal{S}} k_\sigma(\boldsymbol{x}, \boldsymbol{y}) \tag{4.27}$$

where k_σ is the Gaussian kernel from (2.15). The kernel parameter $\sigma > 0$ is difficult to choose optimally [20]. However, a mixture of kernels [20] can lead to a satisfying performance. Here, we use $\mathcal{S} = \{2, 5, 10, 20, 40, 80\}$.

We make use of Pytorch [64] to implement the stochastic gradient optimization in Algorithm 4. The employed optimizer is Adam [65]. The stochastic gradient algorithm has three hyperparameters: batch size T, learning rate (gradient step size) l_r, and exponential learning rate decay β. We determine these hyperparameters via random search [24]. As explained in Section 2.7, the strategy is to draw $(T, l_r, \beta) \in [150, 1500] \times [10^{-6}, 5 \cdot 10^{-3}] \times [0.94, 1]$ randomly and run Algorithm 4. In our case,

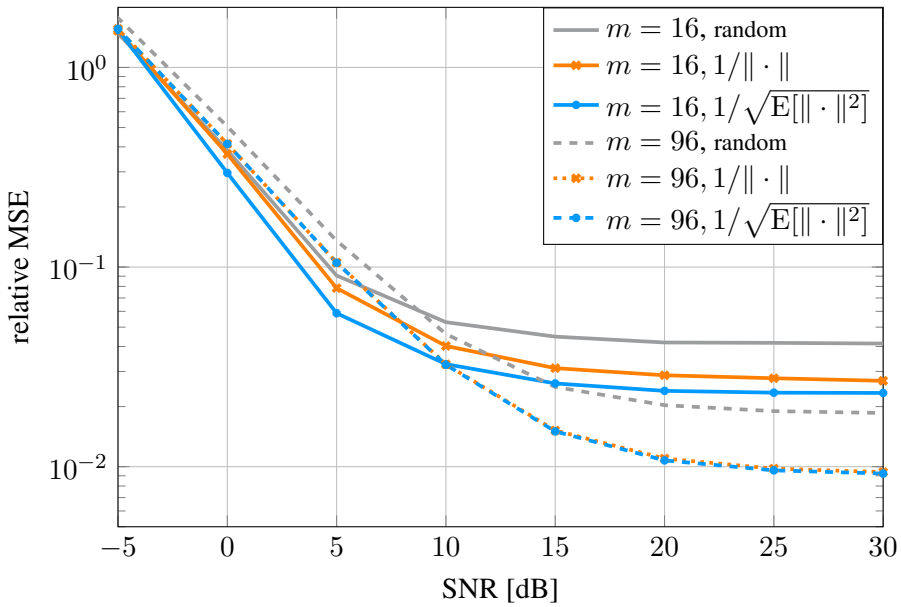

Figure 4.2: We have $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times 128}$. Dashed curves show evaluation with model (4.21) for $m = 96$ and $p = 10$, solid curves for $m = 16$ and $p = 1$. Evaluation with random matrices is displayed in dark gray ("random"). Blue curves refer to learning a matrix with training data normalization $\boldsymbol{h}/\|\boldsymbol{h}\|$, orange curves refer to normalization $\boldsymbol{h}/\sqrt{\mathrm{E}[\|\boldsymbol{h}\|^2]}$.

this is repeated 100 times and validation data is then used to pick the best of the 100 so-obtained matrices. The termination criterion for Algorithm 4 is early stopping [66] with a patience of 150. We always have $T_{\text{tr}} = 5 \cdot 10^4$ training data in all cases and settings. However, instead of normalizing the data such that we work with $\{\frac{\boldsymbol{h}_t}{\|\boldsymbol{h}_t\|}\}_{t=1}^{T_{\text{tr}}}$, we normalize the training data as follows:

$$\overline{\boldsymbol{h}}_t = \frac{\boldsymbol{h}_t}{\sqrt{\frac{1}{T_{\text{tr}}}\sum_{i=1}^{T_{\text{tr}}} \|\boldsymbol{h}_i\|^2}}, \quad \text{for } t = 1, 2, \ldots, T_{\text{tr}}. \tag{4.28}$$

The reason for this is explained in Section 4.6.4.

4.6.4 Training Data Normalization

The motivation to propose Algorithm 4 is to write the RIP condition as

$$(1-\delta) \leq \left\| \boldsymbol{A} \frac{\boldsymbol{h}}{\|\boldsymbol{h}\|} \right\|^2 \leq (1+\delta) \tag{4.29}$$

for all $\boldsymbol{h} \in \mathcal{H}$. The proposed optimization problem

$$\min_{\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m \times N}} \text{MMD}_k^2(p_{\boldsymbol{u}}, q_{\boldsymbol{A}\overline{\boldsymbol{h}}}) \tag{4.30}$$

aims to match the distributions $q_{\boldsymbol{A}\overline{\boldsymbol{h}}}$ to $p_{\boldsymbol{u}}$, where $p_{\boldsymbol{u}}$ denotes the uniform distribution on the unit hypersphere in $\mathbb{C}^m$ and where $\overline{\boldsymbol{h}} = \boldsymbol{h}/\|\boldsymbol{h}\|$. We can generally not expect to achieve a minimum of zero. This is because both $\|\overline{\boldsymbol{h}}\| = 1$ and $\|\boldsymbol{u}\| = 1$ hold so that a minimum of zero implies $\|\boldsymbol{A}\overline{\boldsymbol{h}}\| = 1$ which corresponds to a matrix with restricted isometry constant $\delta = 0$. However, in particular, when the number m of rows is small, we can expect to require a larger $\delta > 0$.

It might now be beneficial to introduce some variation in the norm of $\overline{\boldsymbol{h}}$ to allow $\|\boldsymbol{A}\overline{\boldsymbol{h}}\|$ to be closer to one. Since the optimization problem not only aims to match the norm but also aims for isotropy, varying the norm may allow for a better isotropy. These considerations lead us to compare the training of Algorithm 4 (MMD) using two different training data normalizations: (i) all training channels $\boldsymbol{h}_t$ are normalized as $\boldsymbol{h}_t/\|\boldsymbol{h}_t\|$, and (ii) all training channels are normalized as described in (4.28). In the latter case, the training data only has "on average" a norm equal to one.

Figure 4.2 displays the result of running Algorithm 4 (MMD) with the two different normalizations. We consider two edge cases: (i) a small number m of rows and only one path, and (ii) a large number m of rows and many paths. In both cases, the channel model from (4.21) is used. We observe that the training data normalization (4.28) is at least as good as or better than the normalization $\boldsymbol{h}_t/\|\boldsymbol{h}_t\|$. The difference is particularly pronounced when the number of rows is small. This suits the intuition that the restricted isometry constant δ is potentially larger for smaller m. In these cases, alleviating the norm one constraint can be beneficial. On the other hand, as m increases, the matrix $\boldsymbol{A}$ gets closer to a square matrix where an isometry with $\delta = 0$ is possible. Due to the observed effect, in all other experiments involving Algorithm 4 (MMD), the training data normalization (4.28) is employed.

4.6.5 Simulation Results

In all simulations, we consider constant modulus matrices $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m \times N}$ with $m \in \{16, 32, 64, 96\}$ and $N = 128$ and we vary the number of paths (i.e., the sparsity value) $p \in \{1, 5, 10\}$. First, results with the channel models (4.19) and (4.20) are discussed, where the channels have a sparse representation. Thereafter, we discuss the case of approximate sparsity with channel model (4.21).

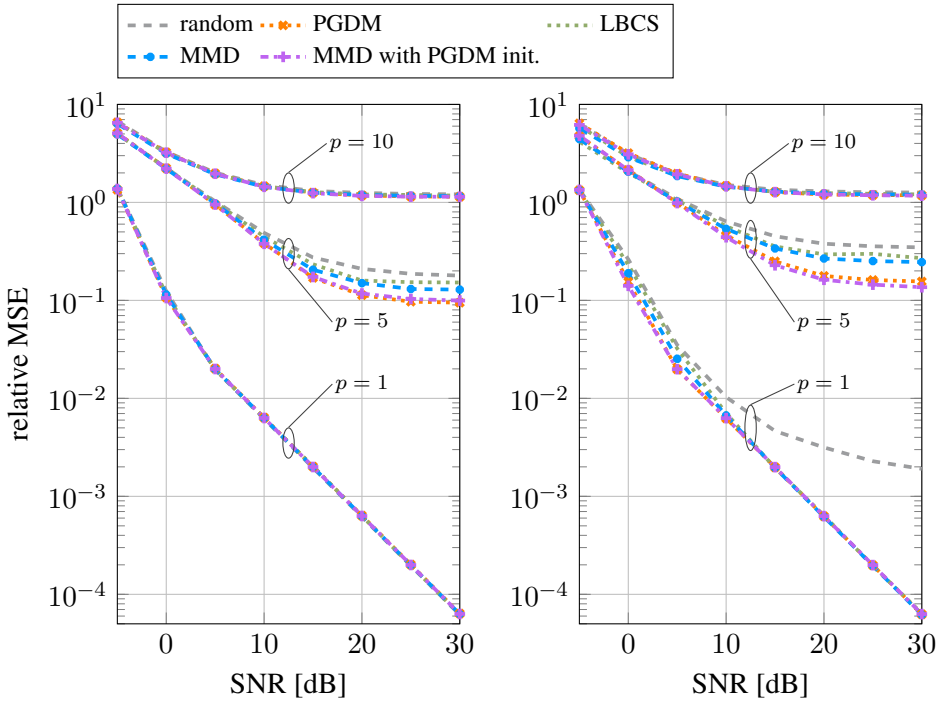

Figure 4.3: Evaluation with channel models (4.19) (left) and (4.20) (right). We have $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{16\times 128}$ and $p \in \{1, 5, 10\}$. The legend is explained in Section 4.6.2.

4.6.5.1 Sparse channels

The left plot in Figure 4.3 shows evaluation with the channel model from (4.19) and the right plot shows evaluation with the channel model from (4.20). In both cases, the number of rows is $m = 16$. Already $p = 5$ seems to be a difficult setting and for $p = 10$, the relative MSE is above 1.0 which means that channel estimation is not possible. The channel models in Figure 4.3 have true sparsity with respect to an orthonormal basis ($\boldsymbol{\Psi} = \boldsymbol{I}$ and $\boldsymbol{\Psi} = \boldsymbol{F}$, respectively) in common so that the performance of random matrices should be similar in both cases. This is because Subgaussian random matrices (and constant modulus matrices are Subgaussian) are universal which means that they can be used for sparse recovery in an arbitrary orthonormal basis [10]. However, in particular, the case $p = 1$ behaves very differently in the two plots. One reason might be that both the random observation matrix as well as the basis $\boldsymbol{\Psi} = \boldsymbol{F}$ have entries of the form $e^{\mathrm{j}\,\phi}$ which can lead to a too large coherence between $\boldsymbol{A}$ and $\boldsymbol{\Psi}$. Interestingly, all non-random algorithms are able to find matrices which do not suffer from this effect. This emphasizes that universality can be traded for a performance improvement.

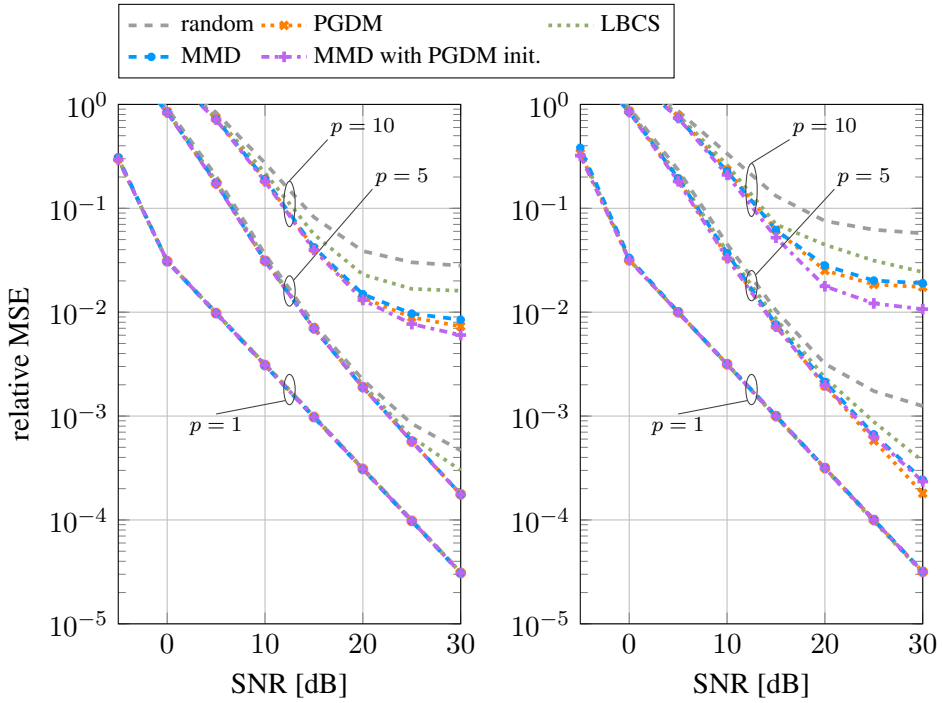

Figure 4.4: Evaluation with channel models (4.19) (left) and (4.20) (right). We have $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{32\times 128}$ and $p \in \{1, 5, 10\}$. The legend is explained in Section 4.6.2.

For $p = 5$, all algorithms in Figure 4.3 outperform random matrices. Algorithm 7 (PGDM) is particularly strong.

Figure 4.4 considers the same setting as Figure 4.3 but with $m = 32$. The number m of rows appears to be large enough to facilitate channel estimation also for $p = 10$. Furthermore, the non-random algorithms yield good matrices for the cases $p = 1$ and $p = 5$ where for higher SNRs the remaining estimation error seems to be due to the presence of noise only. However, random matrices do not quite show this behavior in the case where $p = 5$ and the coherence problem seemingly still exists. In almost all cases, Algorithm 4 (MMD) and Algorithm 7 (PGDM) achieve a similar performance with Algorithm 7 (PGDM) in a slight advantage in some settings. Interestingly, the combination of using Algorithm 7 (PGDM) as an initialization of Algorithm 4 (MMD) can lead to a performance which is superior to the one which is obtained if both algorithms are used independently, see $p = 10$ in the left and right plot of Figure 4.4. Unfortunately, as the case $p = 5$ in the right plot of Figure 4.4 highlights for high SNRs, the combination is not guaranteed to yield the best matrix but it is at least as

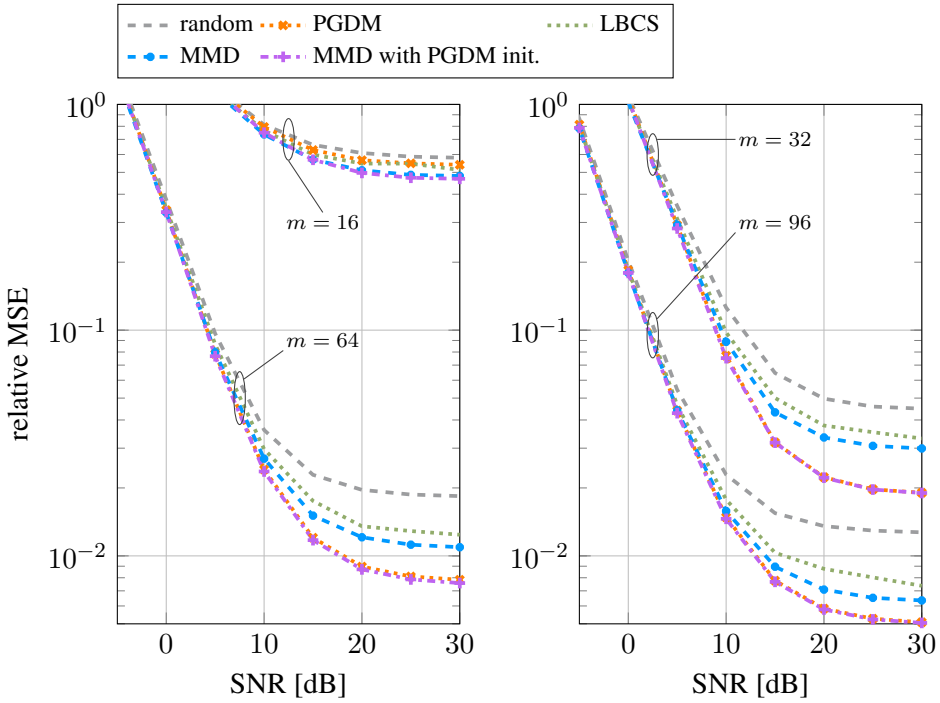

Figure 4.5: Evaluation with channel model (4.21). We have $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times 128}$ for $m \in \{16, 32, 64, 96\}$ and $p = 5$. The legend is explained in Section 4.6.2.

good as the result of initializing Algorithm 4 (MMD) randomly.

The cases $m = 64$ and $m = 96$ lead to overlapping curves for all algorithms and for both channel models, and all three settings $p \in \{1, 5, 10\}$ show straight lines. These results are not visualized.

4.6.5.2 Approximately sparse channels

Figures 4.5 and 4.6 focus on the channel model in (4.21). A first observation is that in all of the considered settings, the curves saturate even at high SNRs. This is a result of the approximate sparsity: Whenever one of the steering vectors $\boldsymbol{a}(\theta_k)$ in the channel (4.21) corresponds to an angle $\theta_k \notin \{\hat{\theta}_1, \dots, \hat{\theta}_L\}$ which is not present in the dictionary $\boldsymbol{\Psi}_L$ from (4.23), then this channel cannot be estimated perfectly. There exist algorithms which focus on *grid-less* channel estimation (see, e.g., [67, Chapter 11] for an overview). However, since we are only interested in the behavior of different observation matrices, working with the on-grid OMP algorithm suffices for our purpose.

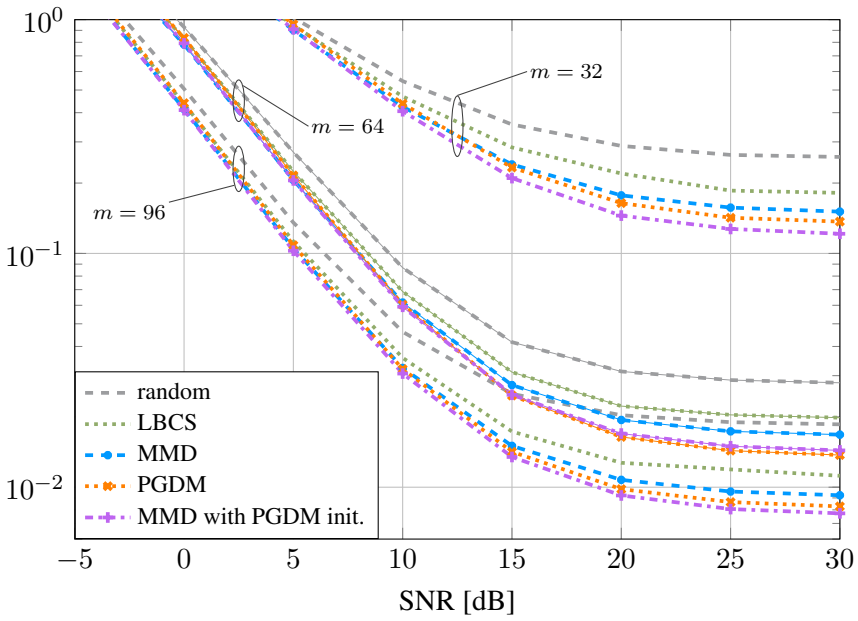

Figure 4.6: Evaluation with channel model (4.21). We have $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times 128}$ for $m \in \{32, 64, 96\}$ and $p = 10$. The legend is explained in Section 4.6.2.

It is interesting to see how large the gap between the random matrices approach and all non-random algorithms is. For example, a comparison of the left and right plot in Figure 4.5 reveals that the best matrix with $m = 32$ rows (right plot) performs almost as well as random matrices with $m = 64$ rows (left plot). What is more, the best matrix with $m = 64$ rows (left plot) outperforms random matrices with $m = 96$ rows (right plot). In that sense, computing non-random matrices can save 32 rows (32 receiver chains) in both cases. Figure 4.6 displays a similar effect (random with $m = 96$ rows vs. non-random with $m = 64$ rows). In Figure 4.6, the combination of Algorithm 4 (MMD) and Algorithm 7 (PGDM) can improve upon the individual algorithms.

4.6.5.3 Conclusion

Generally, in all considered experiments, Algorithm 4 (MMD) and, in particular, the recently introduced Algorithm 7 (PGDM) show a strong performance. It seems to be a good strategy to initialize Algorithm 4 (MMD) with Algorithm 7 (PGDM) in the random search training procedure: In one half of the random searches, the initialization

would be random, in the other half, the initialization would be based on Algorithm 7 (PGDM). It is worth pointing out that Algorithm 7 (PGDM) makes use of a dictionary for the considered channels whereas this is not required for the training of Algorithm 4 (MMD). Thus, in contrast to Algorithm 7 (PGDM), in cases where no dictionary is available, Algorithm 4 (MMD) can still be used.

In all experiments, also Algorithm 6 (LBCS) improves upon random matrices. It lags behind the other non-random approaches. However, one advantage of the method is that it does not explicitly require a dictionary for the training procedure as well. Of course, this is only true if the evaluation algorithm used in Algorithm 6 (LBCS) does not require a dictionary. We consider such a setting in Chapter 5. As a consequence, if Algorithm 7 (PGDM) cannot be employed, it is interesting to initialize Algorithm 4 (MMD) with Algorithm 6 (LBCS) in one half of the random searches.

Learned Observation Matrix and the Gaussian Mixture Estimator

5

One motivation to prefer a deterministic observation matrix over random matrices is to save computational complexity during the channel estimation. For example, as already pointed out in Chapter 4, if the orthogonal matching pursuit (OMP) algorithm is used, employing random matrices requires the computation of the product $\boldsymbol{A}\boldsymbol{\Psi}$ between a random observation matrix $\boldsymbol{A}$ and the dictionary $\boldsymbol{\Psi}$ for every channel estimate, see Line 5 in Algorithm 8. In contrast, this product can be precomputed if a matrix from one of the three algorithms in Chapter 4 is used to learn an observation matrix.

Being able to precompute quantities involving $\boldsymbol{A}$ is also interesting if the Gaussian mixture model (GMM) estimator from Chapter 3 is used. The GMM estimator computes a weighted sum of linear minimum mean square error (LMMSE) channel estimates (cf. (3.22)), and to compute an LMMSE channel estimate, the matrices $\tilde{\boldsymbol{C}}_k = \boldsymbol{A}\boldsymbol{C}_k\boldsymbol{A}^{\mathrm{H}} + \boldsymbol{\Sigma}$ from (3.16) need to be inverted, cf. (3.21b). Further, to compute the weights (or responsibilities), the determinants of all $\tilde{\boldsymbol{C}}_k$ are necessary, too, cf. (3.20). The inverses and determinants have to be computed for every channel estimate if random matrices are used. Again, in contrast, these quantities can be precomputed if $\boldsymbol{A}$ is not a random matrix.

In Section 5.3, we combine the GMM estimator from Chapter 3 with the matrices obtained via the algorithms from Chapter 4. Further, we study what influence the evaluation function which is employed during the iterations of Algorithm 6 (LBCS) (see Line 7) has on the resulting observation matrix. To this end, we run Algorithm 6 (LBCS) using three different evaluation functions, cf. Section 5.4.

5.1 Simulation Setup

It seems natural to also study the matrices obtained via one of the algorithms from Chapter 4 in the context of a channel model which does not admit a sparse representation by construction. One such setting is presented in Figure 3.3 in Chapter 3 where we evaluate various channel estimators with a 3GPP channel model with one propagation

cluster and where we have $\boldsymbol{A} = \boldsymbol{I}$. In this setting, (i) the displayed compressive sensing algorithms show a considerable gap to the genie lower bound and (ii) the GMM estimator comes close to the genie lower bound. It is now interesting to investigate what happens if $\boldsymbol{A} = \boldsymbol{I}$ is replaced with a constant modulus observation matrix. In particular, it is interesting to see whether (i) the gap between the estimators remains when a constant modulus observation matrix is present, (ii) the GMM estimator still comes close to the genie lower bound, and (iii) we can again improve upon random matrices.

In the following simulations, the signal model is $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{n}$ with a constant modulus matrix $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m \times N}$ with $N = 128$ and the channels are generated with one propagation cluster as explained in Section 3.4.1. We work with three channel estimation algorithms: the genie LMMSE estimator from (3.32), the genie OMP estimator (Algorithm 3), and the GMM estimator from (3.22). The GMM estimator always uses $K = 128$ components and $19 \cdot 10^4$ training data. The channel estimators are evaluated as described in Algorithm 8. The only difference is that the OMP algorithm in Line 5 is replaced with one of the three mentioned estimators. The algorithms from Chapter 4 are used as explained in Section 4.6.3. As in Chapter 4, when random observation matrices are used, a new random constant modulus matrix is drawn for every channel.

Algorithm 6 (LBCS) evaluates the current observation matrix on a validation set of channel samples, see Line 7. In Chapter 4, OMP (Algorithm 1) is used for channel estimation during the evaluation of Algorithm 6 (LBCS). In this chapter, we instead employ either the genie OMP estimator, the genie LMMSE estimator, or the GMM estimator during the evaluation in Line 7 in order to gauge the estimators' effect on the resulting observation matrix. We indicate this as follows:

- *LBCS(algo)*: Algorithm 6 (LBCS) uses the estimator

$$\text{algo} \in \{\text{gen. OMP}, \text{gen. LMMSE}, \text{GMM}\}$$

  for the evaluation in Line 7 of its iterations.

5.2 Random Matrices

Figure 5.1 shows the behavior of the three channel estimators genie OMP, genie LMMSE, and GMM estimator if random constant modulus matrices with $m \in \{16, 32, 64, 96\}$ rows are used. All cases display a considerable gap between genie OMP and the other two estimators. Further, with $m = 32$ or more rows, the GMM

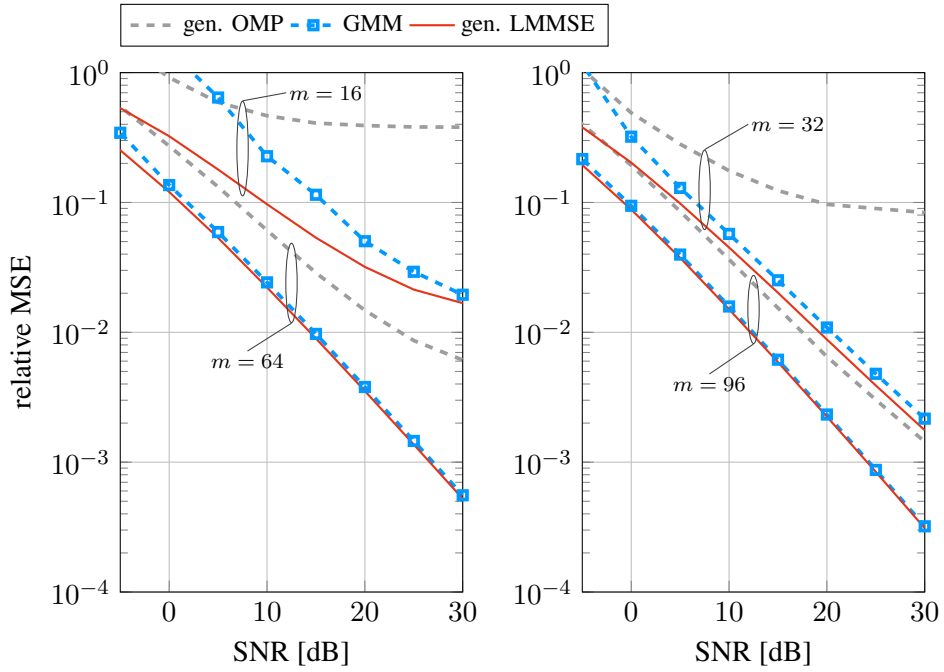

Figure 5.1: Channel estimation with the genie OMP, the genie LMMSE, and the GMM estimator. In all cases, random observation matrices $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m \times 128}$ with $m \in \{16, 64\}$ (left) and $m \in \{32, 96\}$ (right) are used.

estimator starts to come close to the genie LMMSE estimator. This re-emphasizes the strength of the GMM estimator even for cases where the observation matrix is not invertible. Note, of the three algorithms, only the GMM estimator can be implemented in practice as is whereas the other two require genie knowledge.

5.3 Learned Matrices

Figure 5.2 compares channel estimation using the GMM estimator when different observation matrices designed according to one of the algorithms from Chapter 4 are used, cf. Section 4.6.2. As another reference curve, the genie LMMSE estimator in conjunction with random observation matrices is displayed as well. As a first observation, only in the cases $m = 16$ and $m = 32$ there is a noteworthy gap between genie LMMSE estimation and GMM estimation. Further, for $m = 64$ and $m = 96$, the GMM estimator with learned matrices can slightly outperform the genie LMMSE estimator with random matrices. In all cases, (slight) improvements upon random

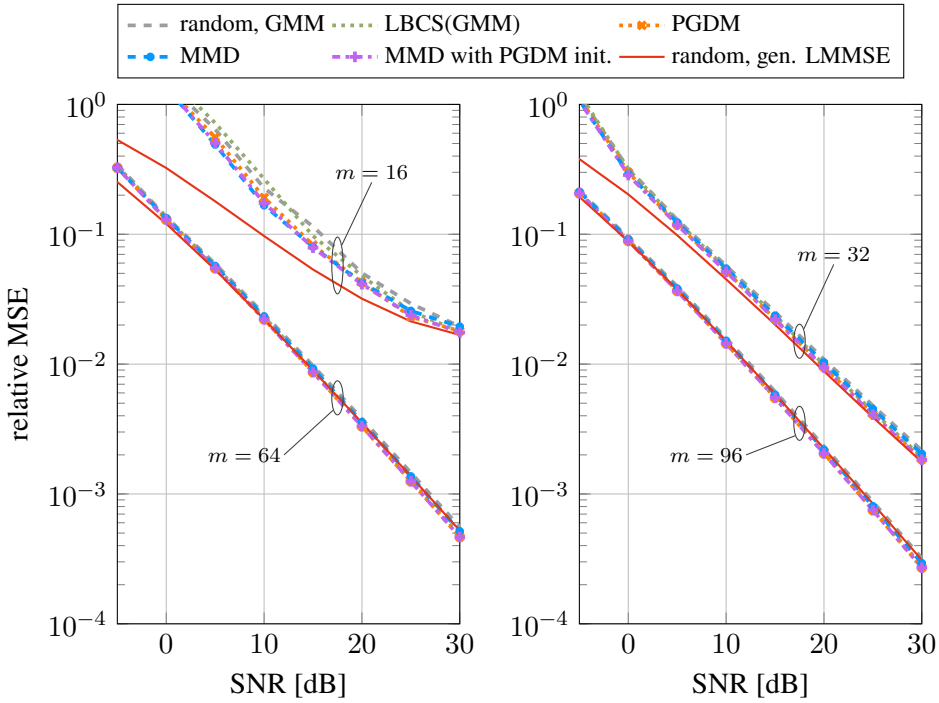

Figure 5.2: Channel estimation using the GMM estimator. In all cases, observation matrices $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times 128}$ with $m \in \{16, 64\}$ (left) and $m \in \{32, 96\}$ (right) are used. "random" refers to employing random observation matrices with either the GMM estimator ("random, GMM") or the genie LMMSE estimator ("random, gen. LMMSE"). The rest of the legend is explained in Section 4.6.2.

matrices are visible with only a little difference between the matrix design algorithms.

In Figure 5.3, we see the same experiment but with the genie OMP estimator instead of the GMM estimator. Generally, there is a significant gap between the genie OMP estimator and the genie LMMSE estimator. Further, learning an observation matrix can considerably improve the estimation performance in comparison to employing random matrices. Interestingly, for $m = 64$ and $m = 96$, the genie OMP does not yet show a saturation effect in the high signal-to-noise ratio (SNR) region if learned matrices are used. Figure 5.4 shows the same experiment but with the genie LMMSE estimator. In this case, there is not much difference between all methods but minor improvements can still be seen compared to random matrices.

Lastly, as a summary, Figure 5.5 is similar to Figure 5.1 in that it compares the three different estimators. In Figure 5.1, the observation matrices are always random

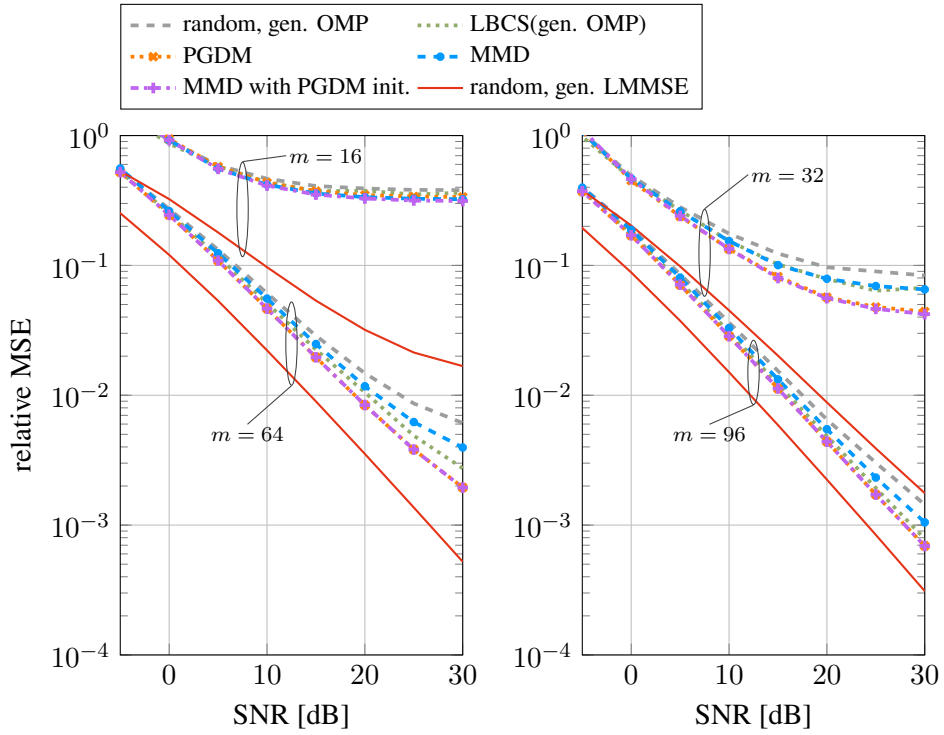

Figure 5.3: Channel estimation using the genie OMP estimator. In all cases, observation matrices $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m \times 128}$ with $m \in \{16, 64\}$ (left) and $m \in \{32, 96\}$ (right) are used. "random" refers to employing random observation matrices with either the genie OMP estimator ("random, gen. OMP") or the genie LMMSE estimator ("random, gen. LMMSE"). The rest of the legend is explained in Section 4.6.2.

whereas in Figure 5.5, the observation matrices are always determined via Algorithm 4 (MMD) initialized with Algorithm 7 (PGDM). The red curves in both figures are identical: the genie LMMSE estimator is employed with random observation matrices. It is interesting to see that essentially all curves improve from Figure 5.1 to Figure 5.5 and they come closer to the "random, gen. LMMSE" baseline, which highlights the importance of a good observation matrix.

Overall, the difference between the performance of random matrices and the learned matrices seems to be smaller for better estimators. In the shown plots, the genie LMMSE estimator is as good as or better than the GMM estimator which, in turn, is as good as or better than the genie OMP estimator. At the same time, the gap between random matrices and learned matrices increases from genie LMMSE estimation

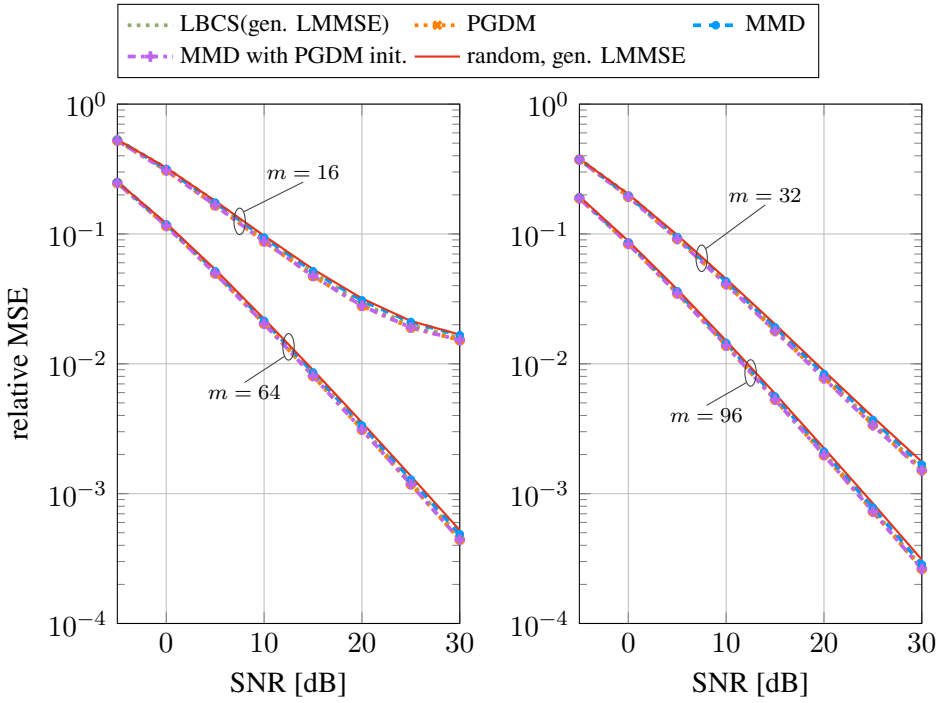

Figure 5.4: Channel estimation using the genie LMMSE estimator. In all cases, observation matrices $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times 128}$ with $m \in \{16, 64\}$ (left) and $m \in \{32, 96\}$ (right) are used. "random, gen. LMMSE" refers to employing random observation matrices with the genie LMMSE estimator. The rest of the legend is explained in Section 4.6.2.

through GMM estimation to genie OMP estimation. Nonetheless, all estimators benefit from learned matrices. The recently introduced projected gradient descent method (PGDM) (Algorithm 7) from [53] is strong, indicating that the codebook $\boldsymbol{\Psi}_L$ from (4.23) is well-suited for the considered data set and problem setting. Generally, a sensible strategy seems to be to initialize Algorithm 4 (MMD) with Algorithm 7 (PGDM) in half of the random searches as long as a dictionary for the considered data is available. If this is not the case, Algorithm 5 (LBCS) should serve as an initializer.

5.4 Different Evaluation Functions for Algorithm 6

Lastly, we look at the influence of the evaluation function on the final result of Algorithm 6 (LBCS). We use the notation described in Section 5.1. The left plot of

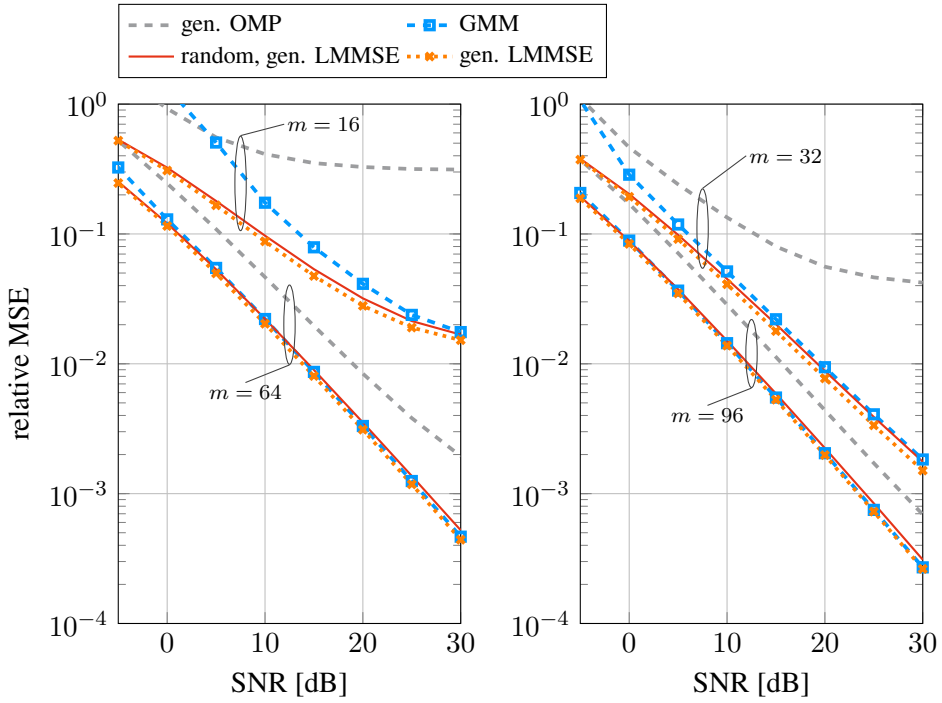

Figure 5.5: Channel estimation with the genie OMP, the genie LMMSE, and the GMM estimator. In all cases, observation matrices $\boldsymbol{A} \in \mathcal{M}_{\text{const}}^{m\times 128}$ with $m \in \{16, 64\}$ (left) and $m \in \{32, 96\}$ (right) are used. "random, gen. LMMSE" refers to employing random observation matrices with the genie LMMSE estimator. In the remaining cases, the observation matrices are determined via *MMD with PGDM init.*, cf. Section 4.6.2.

Figure 5.6 shows the channel estimation with the genie OMP estimator. While the difference between the three options LBCS(gen. OMP), LBCS(gen. LMMSE), and LBCS(GMM) is small, LBCS(gen. OMP) tends to perform best. In the right plot of Figure 5.6, the channel estimation is performed with the genie LMMSE estimator. In this case, the difference between the three options is even smaller but LBCS(gen. LMMSE) tends to perform best. Finally, Figure 5.7 shows the channel estimation with the GMM estimator. Here, for $m = 16$, no clear preference can be seen, and in the other cases, the three options hardly differ. In summary, we can observe the tendency that it is beneficial to use the same channel estimator during the iterations of Algorithm 6 (LBCS) as is applied afterwards when the resulting matrix is used for channel estimation. Hence, the appropriate version of Algorithm 6 (LBCS) is displayed in Figures 5.1 to 5.4. However, the differences in the experiments are minute.

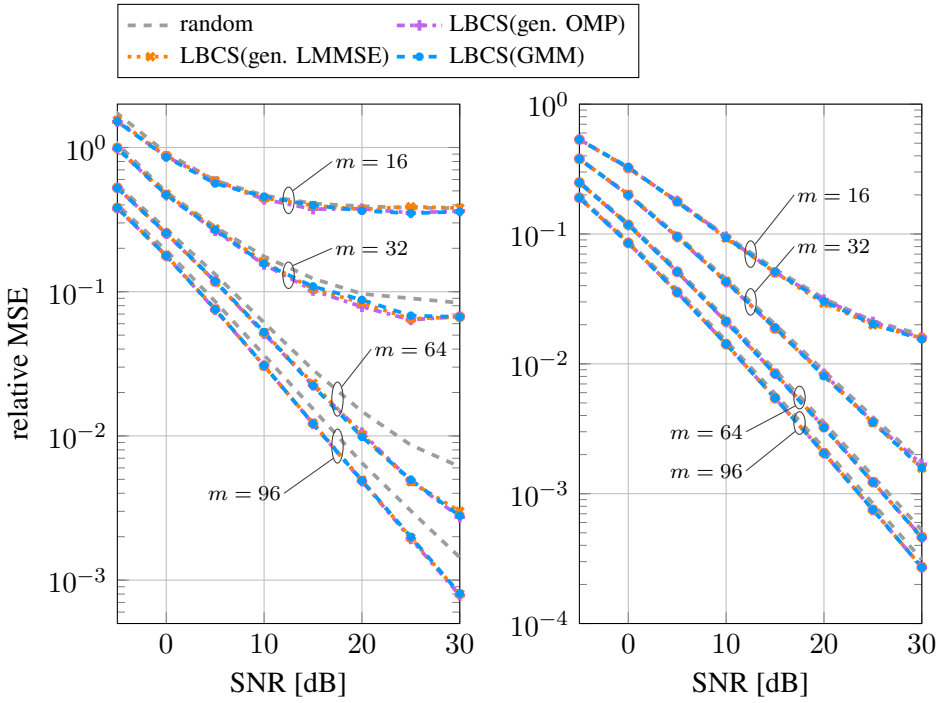

Figure 5.6: Algorithm 6 (LBCS) uses the channel estimator shown in brackets during its iterations. Thereafter, for the evaluation in the plots, channels are estimated with genie OMP (left plot) or genie LMMSE (right plot).

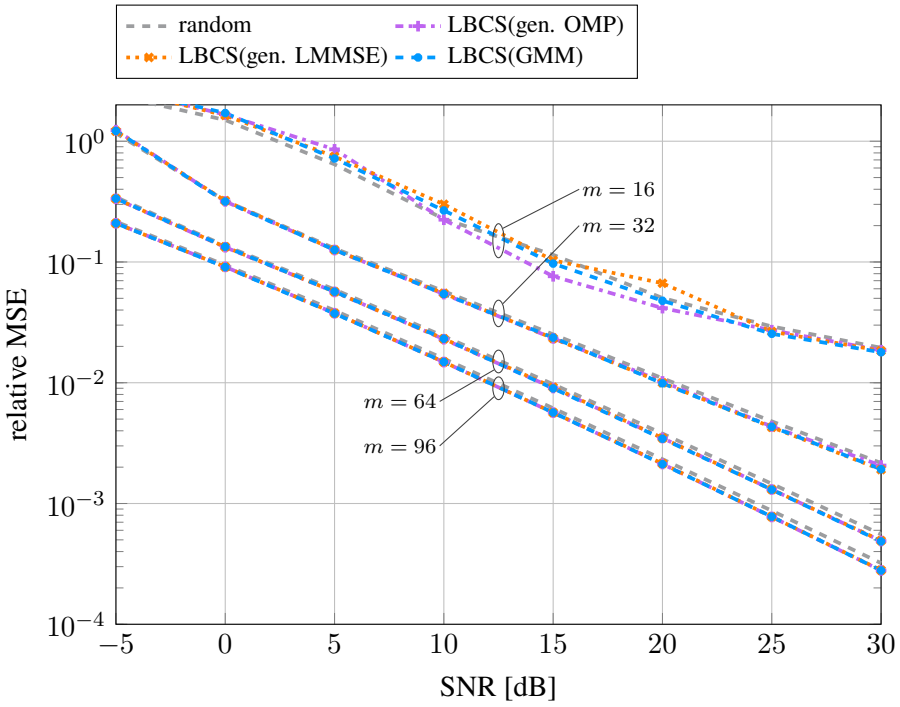

Figure 5.7: Algorithm 6 (LBCS) uses the channel estimator shown in brackets during its iterations. Thereafter, for the evaluation in the plot, channels are estimated with the GMM estimator.

6 Outlook

The following topics related to Chapters 3 and 4 might be worth studying in the future.

6.1 Asymptotically Optimal Channel Estimation

The approach in Chapter 3 is to approximate the channel probability density function (PDF) by means of a uniformly convergent sequence of PDFs. Thereafter, for every sequence element, a conditional mean estimator (CME) is defined, which yields a sequence of CMEs. Theorem 2 then shows under which conditions the sequence of CMEs converges to the optimal CME.

Interestingly, in Theorem 2, there are no further conditions on the sequence of PDFs. One example of such a sequence is given by Gaussian mixture models (GMMs). However, no properties specific to GMMs are used in the proof. Thus, it is possible and interesting to investigate other universal approximators. This could potentially lead to estimators with, e.g., a lower computational complexity or with a better performance or which require a smaller amount of channel training data. Other universal approximators are mentioned in [1]. The challenge lies in fitting the corresponding PDF based on channel training data or in computing the corresponding CME $\hat{\boldsymbol{h}}^{(K)}$ in closed form.

If the computation of the CME $\hat{\boldsymbol{h}}^{(K)}$ is based on GMMs, a weighted sum of K linear minimum mean square error (LMMSE) estimators is computed (cf. (3.22)):

$$\hat{\boldsymbol{h}}^{(K)} : \boldsymbol{y} \mapsto \hat{\boldsymbol{h}}^{(K)}(\boldsymbol{y}) = \sum_{k=1}^{K} p(k \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}) \hat{\boldsymbol{h}}_{\mathrm{LMMSE},k}(\boldsymbol{y}). \tag{6.1}$$

Here, the responsibility $p(k \mid \boldsymbol{y}^{(K)} = \boldsymbol{y})$ (cf. (3.20)) is the probability that the current observation $\boldsymbol{y}$ was sampled from the kth GMM component. In view of the computational complexity, we could compute only the sum of those $K' < K$ summands in (6.1) which correspond to the K' largest probabilities. It is, in particular, interesting to investigate the estimator's behavior when $K' \in \{1, 2, \ldots, K\}$ is varied. Another question in this context is how sensitive the estimator (6.1) is with respect to the responsibilities. For instance, in order to avoid evaluating K Gaussian PDFs to

compute the exact responsibilities (cf. (3.20)), we could try to find a classifier which aims to compute

$$\boldsymbol{y} \mapsto \begin{pmatrix} p(1 \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}) \\ \vdots \\ p(K \mid \boldsymbol{y}^{(K)} = \boldsymbol{y}) \end{pmatrix} \tag{6.2}$$

with less complexity. For example, using a neural network-based classifier, all responsibilities could be computed simultaneously in a single forward pass. This might save computation time while hardly affecting the sum in (6.1).

Abstractly, in Chapter 3, we study a function $\hat{\boldsymbol{h}}$ whose computation requires an analytic expression of a PDF. Since this PDF is not given, we approximate it by means of a uniformly convergent sequence and compute the corresponding approximation-based functions $\hat{\boldsymbol{h}}^{(K)}$. Then we ask whether the sequence $(\hat{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ converges to $\hat{\boldsymbol{h}}$. This approach could work with other functions (not necessarily channel estimators) as well. It is particularly interesting when GMMs are used as approximating sequence. In the case of computing the CME, we can directly make use of the well-known LMMSE formula because conditioned on a GMM component, we are in a Gaussian setting. More generally, the conditional Gaussianity of GMMs can make it possible to use available closed-form solutions in order to obtain a practically useful approximation of the function $\hat{\boldsymbol{h}}$ of interest.

6.2 Learning a Compressive Sensing Matrix

In Chapter 4, we study a distribution matching problem which yields a constant modulus observation matrix. For an arbitrary (not necessarily constant modulus) matrix $\boldsymbol{A} \in \mathbb{C}^{m \times N}$, the optimization problem reads

$$\min_{\boldsymbol{A} \in \mathbb{C}^{m \times N}} \mathrm{MMD}_k^2 \left(\{\mathrm{stk}(\boldsymbol{u}_t)\}_{t=1}^{T_{\mathrm{tr}}}, \{\mathrm{stk}(\boldsymbol{A}\overline{\boldsymbol{h}}_t)\}_{t=1}^{T_{\mathrm{tr}}} \right). \tag{6.3}$$

Using

$$\mathrm{stk}(\boldsymbol{A}\overline{\boldsymbol{h}}) = \begin{bmatrix} \Re(\boldsymbol{A}\overline{\boldsymbol{h}}) \\ \Im(\boldsymbol{A}\overline{\boldsymbol{h}}) \end{bmatrix} = \underbrace{\begin{bmatrix} \Re(\boldsymbol{A}) & -\Im(\boldsymbol{A}) \\ \Im(\boldsymbol{A}) & \Re(\boldsymbol{A}) \end{bmatrix}}_{=\tilde{\boldsymbol{A}}} \begin{bmatrix} \Re(\overline{\boldsymbol{h}}) \\ \Im(\overline{\boldsymbol{h}}) \end{bmatrix} \tag{6.4}$$

it can again be expressed by means of real quantities. Interestingly, structural constraints like, e.g., a Toeplitz structure, a circulant structure, or a block diagonal structure can easily be incorporated into the learning algorithm which solves (6.3). To this end, the number of optimization variables in $\tilde{\boldsymbol{A}}$ is reduced by forcing certain elements to be equal, as dictated by the sought structure. This technique is known as *weight tying*

in machine learning, see, e.g., [66]. A motivation for doing this is, for example, [68] where Toeplitz matrices are used in the context of channel estimation and where it is shown that randomly drawing such matrices can lead to matrices with the restricted isometry property (RIP). Here, we might again be interested in obtaining one fixed matrix instead of drawing random ones.

More generally, the idea in Chapter 4 is to match the distribution of $\boldsymbol{A}\overline{\boldsymbol{h}}$ to the distribution of $\boldsymbol{u}$ where $\overline{\boldsymbol{h}} = \boldsymbol{h}/\|\boldsymbol{h}\|$ are normalized channels and where $\boldsymbol{u}$ is uniformly distributed on the m-dimensional unit hypersphere. It might be interesting to investigate other distribution matching methods as well. One example is the generative adversarial network (GAN) [69]. In the context of Chapter 4, the normalized channel samples $\overline{\boldsymbol{h}}_t$ are the GAN's noise samples and the observation matrix $\boldsymbol{A}$ is the generator. The GAN's discriminator then aims to distinguish between fake samples $\boldsymbol{A}\overline{\boldsymbol{h}}_t$ and real samples $\boldsymbol{u}_t$.

Proof of Theorem 2 A

The proof of Theorem 2 makes use of and is presented after the following lemma and can also be found in [26].

Lemma 1. *For an arbitrary $\boldsymbol{y} \in \mathbb{R}^N$, it holds*

$$\int \|\boldsymbol{h}\| f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) d\boldsymbol{h} \leq \sqrt{\det(\boldsymbol{A}^{-1}\boldsymbol{A}^{-\mathrm{T}})} \sqrt{\|\boldsymbol{A}^{-1}\boldsymbol{y}\|^2 + \operatorname{trace}(\boldsymbol{A}^{-1}\boldsymbol{\Sigma}\boldsymbol{A}^{-\mathrm{T}})}.$$

Proof. Recall that $f_{\boldsymbol{n}} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ denotes a Gaussian probability density function (PDF). We have

$$f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) = \frac{\exp(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}))}{\sqrt{(2\pi)^N \det(\boldsymbol{\Sigma})}} \tag{A.1}$$

$$= \frac{\exp(-\frac{1}{2}(\boldsymbol{A}^{-1}\boldsymbol{y} - \boldsymbol{h})^{\mathrm{T}} \boldsymbol{A}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{A} (\boldsymbol{A}^{-1}\boldsymbol{y} - \boldsymbol{h}))}{\sqrt{(2\pi)^N \det(\boldsymbol{\Sigma})}} \tag{A.2}$$

$$= \frac{\exp(-\frac{1}{2}(\boldsymbol{h} - \boldsymbol{A}^{-1}\boldsymbol{y})^{\mathrm{T}} (\boldsymbol{A}^{-1}\boldsymbol{\Sigma}\boldsymbol{A}^{-\mathrm{T}})^{-1} (\boldsymbol{h} - \boldsymbol{A}^{-1}\boldsymbol{y}))}{\sqrt{(2\pi)^N \det(\boldsymbol{\Sigma})}}. \tag{A.3}$$

Therefore,

$$\frac{f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h})}{\sqrt{\det(\boldsymbol{A}^{-1}\boldsymbol{A}^{-\mathrm{T}})}} = \mathcal{N}(\boldsymbol{h}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \tag{A.4}$$

is a Gaussian PDF with mean vector $\tilde{\boldsymbol{\mu}} = \boldsymbol{A}^{-1}\boldsymbol{y}$ and covariance matrix $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{A}^{-1}\boldsymbol{\Sigma}\boldsymbol{A}^{-\mathrm{T}}$. Let $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ be a standard Gaussian random variable and let $\tilde{\boldsymbol{\Sigma}}^{\frac{1}{2}}$ be a square root of the covariance matrix $\tilde{\boldsymbol{\Sigma}} = \tilde{\boldsymbol{\Sigma}}^{\frac{\mathrm{T}}{2}} \tilde{\boldsymbol{\Sigma}}^{\frac{1}{2}}$. Then, we are interested in computing

$$\int \|\boldsymbol{h}\| f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) d\boldsymbol{h} = \sqrt{\det(\boldsymbol{A}^{-1}\boldsymbol{A}^{-\mathrm{T}})} \int \|\boldsymbol{h}\| \frac{f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h})}{\sqrt{\det(\boldsymbol{A}^{-1}\boldsymbol{A}^{-\mathrm{T}})}} d\boldsymbol{h} \tag{A.5}$$

$$= \sqrt{\det(\boldsymbol{A}^{-1}\boldsymbol{A}^{-\mathrm{T}})} \int \|\boldsymbol{h}\| \mathcal{N}(\boldsymbol{h}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) d\boldsymbol{h} \tag{A.6}$$

$$= \sqrt{\det(\boldsymbol{A}^{-1}\boldsymbol{A}^{-\mathrm{T}})}\, \mathrm{E}_{\boldsymbol{w}}[\|\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\Sigma}}^{\frac{1}{2}} \boldsymbol{w}\|] \tag{A.7}$$

where we take the expectation with respect to the standard Gaussian random variable $\boldsymbol{w}$. To see the last equality, note that $(\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\Sigma}}^{\frac{1}{2}}\boldsymbol{w}) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}})$ is a Gaussian random variable with mean $\tilde{\boldsymbol{\mu}}$ and covariance matrix $\tilde{\boldsymbol{\Sigma}}$. Jensen's inequality yields:

$$\left(\mathrm{E}_{\boldsymbol{w}}[\|\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\Sigma}}^{\frac{1}{2}}\boldsymbol{w}\|]\right)^2 \leq \mathrm{E}_{\boldsymbol{w}}[\|\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\Sigma}}^{\frac{1}{2}}\boldsymbol{w}\|^2] \tag{A.8}$$

$$= \|\tilde{\boldsymbol{\mu}}\|^2 + 2\tilde{\boldsymbol{\mu}}^{\mathrm{T}}\tilde{\boldsymbol{\Sigma}}^{\frac{1}{2}}\,\mathrm{E}_{\boldsymbol{w}}[\boldsymbol{w}] + \mathrm{E}_{\boldsymbol{w}}[\mathrm{trace}(\boldsymbol{w}^{\mathrm{T}}\tilde{\boldsymbol{\Sigma}}^{\frac{\mathrm{T}}{2}}\tilde{\boldsymbol{\Sigma}}^{\frac{1}{2}}\boldsymbol{w})] \tag{A.9}$$

$$= \|\tilde{\boldsymbol{\mu}}\|^2 + \mathrm{trace}(\tilde{\boldsymbol{\Sigma}}\,\mathrm{E}_{\boldsymbol{w}}[\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}]) = \|\tilde{\boldsymbol{\mu}}\|^2 + \mathrm{trace}(\tilde{\boldsymbol{\Sigma}}) \tag{A.10}$$

$$= \|\boldsymbol{A}^{-1}\boldsymbol{y}\|^2 + \mathrm{trace}(\boldsymbol{A}^{-1}\boldsymbol{\Sigma}\boldsymbol{A}^{-\mathrm{T}}). \tag{A.11}$$

Lastly, we use the square root of this bound in (A.7). □

Proof of Theorem 2. First, we show that the uniform convergence of $(f_{\boldsymbol{h}}^{(K)})_{K=1}^{\infty}$ to $f_{\boldsymbol{h}}$ implies the uniform convergence of $(f_{\boldsymbol{y}}^{(K)})_{K=1}^{\infty}$ to $f_{\boldsymbol{y}}$. The PDF of $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{h}$ is

$$f_{\boldsymbol{x}}(\boldsymbol{x}) = \frac{1}{|\det(\boldsymbol{A})|} f_{\boldsymbol{h}}(\boldsymbol{A}^{-1}\boldsymbol{x}) \tag{A.12}$$

because $\boldsymbol{A}$ is invertible. Analogously, we obtain

$$f_{\boldsymbol{x}}^{(K)}(\boldsymbol{x}) = \frac{1}{|\det(\boldsymbol{A})|} f_{\boldsymbol{h}}^{(K)}(\boldsymbol{A}^{-1}\boldsymbol{x}) \tag{A.13}$$

as the PDF of $\boldsymbol{A}\boldsymbol{h}^{(K)}$. Since the random variable $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$ is a sum of two stochastically independent random variables, its PDF can be computed via convolution:

$$f_{\boldsymbol{y}}(\boldsymbol{y}) = \int f_{\boldsymbol{n}}(\boldsymbol{s}) f_{\boldsymbol{x}}(\boldsymbol{y} - \boldsymbol{s}) d\boldsymbol{s}. \tag{A.14}$$

Similarly, $f_{\boldsymbol{y}}^{(K)}$ is obtained by replacing $f_{\boldsymbol{x}}$ with $f_{\boldsymbol{x}}^{(K)}$ in (A.14). For later reference, note, because $f_{\boldsymbol{n}}$ is positive ($f_{\boldsymbol{n}}(\boldsymbol{s}) > 0$ for all $\boldsymbol{s} \in \mathbb{R}^N$) and $f_{\boldsymbol{x}}$ as well as $f_{\boldsymbol{x}}^{(K)}$ are continuous PDFs, the convolution results $f_{\boldsymbol{y}}^{(K)}$ and $f_{\boldsymbol{y}}$ are positive, too.

We have

$$|f_{\boldsymbol{y}}(\boldsymbol{y}) - f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})| = \left| \int f_{\boldsymbol{n}}(\boldsymbol{s}) \left(f_{\boldsymbol{x}}(\boldsymbol{y} - \boldsymbol{s}) - f_{\boldsymbol{x}}^{(K)}(\boldsymbol{y} - \boldsymbol{s}) \right) d\boldsymbol{s} \right| \tag{A.15}$$

$$\leq \int \left| f_{\boldsymbol{n}}(\boldsymbol{s}) \frac{f_{\boldsymbol{h}}(\boldsymbol{A}^{-1}(\boldsymbol{y} - \boldsymbol{s})) - f_{\boldsymbol{h}}^{(K)}(\boldsymbol{A}^{-1}(\boldsymbol{y} - \boldsymbol{s}))}{|\det(\boldsymbol{A})|} \right| d\boldsymbol{s} \tag{A.16}$$

$$\leq \frac{\|f_{\boldsymbol{h}} - f_{\boldsymbol{h}}^{(K)}\|_\infty}{|\det(\boldsymbol{A})|} \int |f_{\boldsymbol{n}}(\boldsymbol{s})| d\boldsymbol{s} = \frac{\|f_{\boldsymbol{h}} - f_{\boldsymbol{h}}^{(K)}\|_\infty}{|\det(\boldsymbol{A})|}. \tag{A.17}$$

The last integral is equal to one because $f_{\boldsymbol{n}}$ is a PDF. Since $\lim_{K\to\infty} \|f_{\boldsymbol{h}} - f_{\boldsymbol{h}}^{(K)}\|_\infty = 0$ holds by the uniform convergence assumption, we have

$$\lim_{K\to\infty} \|f_{\boldsymbol{y}} - f_{\boldsymbol{y}}^{(K)}\|_\infty = 0 \tag{A.18}$$

which means that the sequence $(f_{\boldsymbol{y}}^{(K)})_{K=1}^{\infty}$ converges uniformly to $f_{\boldsymbol{y}}$.

Let $r > 0$ be arbitrary. In the following, we show that (3.11) holds uniformly for all $\boldsymbol{y} \in \mathcal{B}_r$. But because r is arbitrary, the pointwise convergence in (3.11) follows immediately.

Let $\boldsymbol{y} \in \mathcal{B}_r$ be arbitrary. With (3.10) and (3.9) in mind, we find the following upper bound:

$$\|\hat{\boldsymbol{h}}(\boldsymbol{y}) - \hat{\boldsymbol{h}}^{(K)}(\boldsymbol{y})\| \leq \int \|\boldsymbol{h}\| |f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h})| \left| \frac{f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} - \frac{f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} \right| d\boldsymbol{h} \tag{A.19}$$

$$\leq \sup_{\boldsymbol{h}\in\mathbb{R}^N} \left| \frac{f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} - \frac{f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} \right| \int \|\boldsymbol{h}\| f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) d\boldsymbol{h}. \tag{A.20}$$

The last integral is independent of K and by Lemma 1, it is finite for any $\boldsymbol{y} \in \mathbb{R}^N$. It is, in particular, bounded by

$$\sqrt{\det(\boldsymbol{A}^{-1}\boldsymbol{A}^{-\mathrm{T}})} \sqrt{\|\boldsymbol{A}^{-1}\|^2 r^2 + \operatorname{trace}(\boldsymbol{A}^{-1}\boldsymbol{\Sigma}\boldsymbol{A}^{-\mathrm{T}})} < \infty$$

for all $\boldsymbol{y} \in \mathcal{B}_r$, where $\|\boldsymbol{A}^{-1}\|$ is the spectral norm of the matrix $\boldsymbol{A}^{-1}$. Hence, as soon as

$$\lim_{K\to\infty} \sup_{\boldsymbol{h}\in\mathbb{R}^N} \left| \frac{f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} - \frac{f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} \right| = 0, \quad \forall \boldsymbol{y} \in \mathcal{B}_r \tag{A.21}$$

is shown, (3.11) is confirmed for all $\boldsymbol{y} \in \mathcal{B}_r$.

To prove (A.21), we write

$$\left| \frac{f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} - \frac{f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} \right| = \left| \frac{f_{\boldsymbol{h}}(\boldsymbol{h}) f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) - f_{\boldsymbol{y}}(\boldsymbol{y}) f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y}) f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} \right| \tag{A.22}$$

for an arbitrary $\boldsymbol{h} \in \mathbb{R}^N$. Now, we add $0 = f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h}) - f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})$ in

the numerator on the right-hand side and apply the triangle inequality to get

$$\left| \frac{f_{\boldsymbol{h}}(\boldsymbol{h}) f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) - f_{\boldsymbol{y}}(\boldsymbol{y}) f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y}) f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} \right| \tag{A.23}$$

$$\leq \frac{\left| \left(f_{\boldsymbol{h}}(\boldsymbol{h}) - f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h}) \right) f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) \right|}{f_{\boldsymbol{y}}(\boldsymbol{y}) f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})} + \frac{\left| \left(f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) - f_{\boldsymbol{y}}(\boldsymbol{y}) \right) f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h}) \right|}{f_{\boldsymbol{y}}(\boldsymbol{y}) f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})}$$

$$\leq \frac{\| f_{\boldsymbol{h}} - f_{\boldsymbol{h}}^{(K)} \|_\infty \| f_{\boldsymbol{y}}^{(K)} \|_\infty + \| f_{\boldsymbol{y}}^{(K)} - f_{\boldsymbol{y}} \|_\infty \| f_{\boldsymbol{h}}^{(K)} \|_\infty}{f_{\boldsymbol{y}}(\boldsymbol{y}) f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})}. \tag{A.24}$$

The goal is now to find upper bounds for all terms in the numerator and lower bounds for all terms in the denominator which in the end allow us to see that (A.22) converges to zero as K approaches infinity.

By the compactness of $\mathcal{B}_r$ and the continuity of $f_{\boldsymbol{y}}$, there exists a $\boldsymbol{y}_{\min} \in \mathcal{B}_r$ at which $f_{\boldsymbol{y}}$ attains a minimum value $f_{\boldsymbol{y}}(\boldsymbol{y}_{\min}) > 0$ over $\mathcal{B}_r$. Due to the uniform convergence (A.18), there exists an index $N_1 \in \mathbb{N}$ such that $|f_{\boldsymbol{y}}(\boldsymbol{y}) - f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})| \leq \frac{1}{2} f_{\boldsymbol{y}}(\boldsymbol{y}_{\min})$ holds for all $K \geq N_1$ and for all $\boldsymbol{y} \in \mathcal{B}_r$. The reverse triangle inequality then shows that $f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) \geq f_{\boldsymbol{y}}(\boldsymbol{y}) - |f_{\boldsymbol{y}}(\boldsymbol{y}) - f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})| \geq f_{\boldsymbol{y}}(\boldsymbol{y}_{\min}) - \frac{1}{2} f_{\boldsymbol{y}}(\boldsymbol{y}_{\min}) = \frac{1}{2} f_{\boldsymbol{y}}(\boldsymbol{y}_{\min})$ is true. Hence, with $M_1 = \frac{1}{2} f_{\boldsymbol{y}}(\boldsymbol{y}_{\min}) > 0$, the inequality

$$f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y}) \geq M_1 \tag{A.25}$$

holds for all $\boldsymbol{y} \in \mathcal{B}_r$ and for all $K \geq N_1$. Further, since $\| f_{\boldsymbol{h}} - f_{\boldsymbol{h}}^{(K)} \|_\infty \to 0$ and $\| f_{\boldsymbol{h}} \|_\infty < \infty$, there exist an $M_2 > 0$ and an $N_2 \in \mathbb{N}$ such that

$$\| f_{\boldsymbol{h}}^{(K)} \|_\infty \leq M_2 \quad \text{for all} \quad K \geq N_2. \tag{A.26}$$

Analogously, there exist an $M_3 > 0$ and an $N_3 \in \mathbb{N}$ such that

$$\| f_{\boldsymbol{y}}^{(K)} \|_\infty \leq M_3 \quad \text{for all} \quad K \geq N_3. \tag{A.27}$$

Let $\varepsilon > 0$ be arbitrary. Due to $\| f_{\boldsymbol{h}} - f_{\boldsymbol{h}}^{(K)} \|_\infty \to 0$, there exists an index $N_4 \geq \max\{N_1, N_3\}$ such that

$$\| f_{\boldsymbol{h}} - f_{\boldsymbol{h}}^{(K)} \|_\infty \leq \frac{f_{\boldsymbol{y}}(\boldsymbol{y}_{\min}) M_1}{2 M_3} \varepsilon \quad \text{for all} \quad K \geq N_4. \tag{A.28}$$

Similarly, there exists an index $N_5 \geq \max\{N_1, N_2\}$ with

$$\| f_{\boldsymbol{y}} - f_{\boldsymbol{y}}^{(K)} \|_\infty \leq \frac{f_{\boldsymbol{y}}(\boldsymbol{y}_{\min}) M_1}{2 M_2} \varepsilon \quad \text{for all} \quad K \geq N_5. \tag{A.29}$$

We can use the last five inequalities to bound (A.24). To this end, $f_{\boldsymbol{y}}(\boldsymbol{y}_{\min})$ and (A.25) provide lower bounds on the terms in the denominator, (A.28) and (A.27) bound the first summand in the numerator, and (A.29) and (A.26) bound the second summand in the numerator. In total, this yields an upper bound on (A.22):

$$\left|\frac{f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} - \frac{f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})}\right| \leq \frac{f_{\boldsymbol{y}}(\boldsymbol{y}_{\min})M_1}{2M_3}\varepsilon \cdot \frac{M_3}{f_{\boldsymbol{y}}(\boldsymbol{y}_{\min})M_1} + \frac{f_{\boldsymbol{y}}(\boldsymbol{y}_{\min})M_1}{2M_2}\varepsilon \cdot \frac{M_2}{f_{\boldsymbol{y}}(\boldsymbol{y}_{\min})M_1} \tag{A.30}$$

for all $K \geq \max\{N_4, N_5\}$ and for all $\boldsymbol{y} \in \mathcal{B}_r$. We conclude

$$\sup_{\boldsymbol{h}\in\mathbb{R}^N} \left|\frac{f_{\boldsymbol{h}}(\boldsymbol{h})}{f_{\boldsymbol{y}}(\boldsymbol{y})} - \frac{f_{\boldsymbol{h}}^{(K)}(\boldsymbol{h})}{f_{\boldsymbol{y}}^{(K)}(\boldsymbol{y})}\right| \leq \varepsilon \tag{A.31}$$

for all $K \geq \max\{N_4, N_5\}$ and all $\boldsymbol{y} \in \mathcal{B}_r$, and because ε was arbitrary, (A.21) is confirmed, which concludes the proof. □

B Discussion for Noninvertible Matrices

This appendix provides more details for the discussion in Section 3.2.1. As mentioned there, two problems arise in the proof of Theorem 2 if $\boldsymbol{A}$ is not invertible. First, Lemma 1 (see Appendix A) might no longer hold. That is, the integral $\int \|\boldsymbol{h}\| f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) d\boldsymbol{h}$ might not be finite for all $\boldsymbol{y} \in \mathbb{R}^m$. This is discussed in Appendix B.1. Second, a crucial step in the proof of Theorem 2 is showing that $(f_{\boldsymbol{y}}^{(K)})_{K=1}^{\infty}$ converges uniformly to $f_{\boldsymbol{y}}$, cf. (A.18). This is possible because the sequence $(f_{\boldsymbol{x}}^{(K)})_{K=1}^{\infty}$ of probability density functions (PDFs) corresponding to the auxiliary variables $\boldsymbol{x}^{(K)} = \boldsymbol{A}\boldsymbol{h}^{(K)}$ converges uniformly to the PDF $f_{\boldsymbol{x}}$ of $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{h}$ if $\boldsymbol{A}$ is invertible. Unfortunately, if $\boldsymbol{A}$ is not invertible, this will likely not hold for an arbitrary $f_{\boldsymbol{h}}$. This is discussed in Appendix B.2.

B.1 Integral in Lemma 1

To see why the integral $\int \|\boldsymbol{h}\| f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) d\boldsymbol{h}$ in Lemma 1 might not be finite if $\boldsymbol{A}$ is not invertible, consider the matrix $\boldsymbol{A} = [\boldsymbol{I}, \boldsymbol{0}] \in \mathbb{R}^{m \times N}$ with $m < N$ where $\boldsymbol{I} \in \mathbb{R}^{m \times m}$ is the identity matrix and the remaining elements of $\boldsymbol{A}$ are zero. Let us write $\boldsymbol{h} = [\boldsymbol{h}_m^{\mathrm{T}}, \boldsymbol{h}_{N-m}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^m \times \mathbb{R}^{N-m}$. Define the set

$$\mathcal{D} = \{\boldsymbol{h} \in \mathbb{R}^N : \|\boldsymbol{h}_m\| \geq 1, \|\boldsymbol{h}_{N-m}\| \geq 1\} \tag{B.1}$$

where the norm of both subvectors $\boldsymbol{h}_m$ and $\boldsymbol{h}_{N-m}$ is at least one such that we always have $\|\boldsymbol{h}\| \geq 1$ on $\mathcal{D}$. We can now compute:

$$\int_{\mathbb{R}^N} \|\boldsymbol{h}\| f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) d\boldsymbol{h} \geq \int_{\mathcal{D}} f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) d\boldsymbol{h} \tag{B.2}$$

$$= \int_{\|\boldsymbol{h}_{N-m}\| \geq 1} \int_{\|\boldsymbol{h}_m\| \geq 1} f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{h}_m) d\boldsymbol{h}_m d\boldsymbol{h}_{N-m}. \tag{B.3}$$

Since $f_{\boldsymbol{n}}$ is an m-dimensional Gaussian PDF, the inner integral is equal to some constant c with $0 < c < 1$ and it follows that $\int \|\boldsymbol{h}\| f_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}) d\boldsymbol{h}$ is not finite.

B.2 Uniform Convergence

We want to find an expression for the PDF of $\mathsf{x} = \boldsymbol{A}\mathsf{h}$ when $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ is a wide matrix with full rank $m < N$. We can assume that the first m columns are linearly independent (otherwise we introduce a permutation matrix for the following argument). This allows us to partition $\boldsymbol{A} = [\boldsymbol{A}_i, \boldsymbol{A}_n]$ into an invertible part $\boldsymbol{A}_i \in \mathbb{R}^{m\times m}$ and a noninvertible part $\boldsymbol{A}_n \in \mathbb{R}^{m\times N-m}$. With a corresponding partitioning of $\boldsymbol{h} = [\boldsymbol{h}_i^{\mathrm{T}}, \boldsymbol{h}_n^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^m \times \mathbb{R}^{N-m}$, we can write

$$\mathsf{x} = \boldsymbol{A}\mathsf{h} = \boldsymbol{A}_i\mathsf{h}_i + \boldsymbol{A}_n\mathsf{h}_n. \tag{B.4}$$

Defining a transformation

$$t : \mathbb{R}^m \times \mathbb{R}^{N-m} \to \mathbb{R}^m \times \mathbb{R}^{N-m},\ (\boldsymbol{h}_i, \boldsymbol{h}_n) \mapsto (\boldsymbol{A}_i\boldsymbol{h}_i + \boldsymbol{A}_n\boldsymbol{h}_n, \boldsymbol{h}_n) \tag{B.5}$$

with the inverse

$$t^{-1} : (\boldsymbol{x}, \boldsymbol{x}') \mapsto (\boldsymbol{A}_i^{-1}(\boldsymbol{x} - \boldsymbol{A}_n\boldsymbol{x}'), \boldsymbol{x}') \tag{B.6}$$

we can now compute the joint density $f_{\mathsf{x},\mathsf{x}'}$ via the usual transformation formula:

$$(\boldsymbol{x}, \boldsymbol{x}') \mapsto f_{\mathsf{x},\mathsf{x}'}(\boldsymbol{x}, \boldsymbol{x}') = \frac{f_{\mathsf{h}}(t^{-1}(\boldsymbol{x}, \boldsymbol{x}'))}{\left|\det\left(\frac{\partial t}{\partial \boldsymbol{h}}(t^{-1}(\boldsymbol{x}, \boldsymbol{x}'))\right)\right|}. \tag{B.7}$$

Together with $\left|\det\left(\frac{\partial t}{\partial \boldsymbol{h}}(t^{-1}(\boldsymbol{x}, \boldsymbol{x}'))\right)\right| = |\det(\boldsymbol{A}_i)|$, we can express the PDF of $\mathsf{x} = \boldsymbol{A}\mathsf{h}$ via marginalization:

$$\boldsymbol{x} \mapsto f_{\mathsf{x}}(\boldsymbol{x}) = \int_{\mathbb{R}^{N-m}} f_{\mathsf{x},\mathsf{x}'}(\boldsymbol{x}, \boldsymbol{x}')d\boldsymbol{x}' = \int_{\mathbb{R}^{N-m}} \frac{f_{\mathsf{h}}(t^{-1}(\boldsymbol{x}, \boldsymbol{x}'))}{|\det(\boldsymbol{A}_i)|}d\boldsymbol{x}'. \tag{B.8}$$

Analogously, we obtain the PDF $f_{\mathsf{x}}^{(K)}$ of $\mathsf{x}^{(K)} = \boldsymbol{A}\mathsf{h}^{(K)}$. Given

$$\left|f_{\mathsf{x}}^{(K)}(\boldsymbol{x}) - f_{\mathsf{x}}(\boldsymbol{x})\right| = \frac{\left|\int_{\mathbb{R}^{N-m}} \left(f_{\mathsf{h}}^{(K)}(t^{-1}(\boldsymbol{x}, \boldsymbol{x}')) - f_{\mathsf{h}}(t^{-1}(\boldsymbol{x}, \boldsymbol{x}'))\right) d\boldsymbol{x}'\right|}{|\det(\boldsymbol{A}_i)|} \tag{B.9}$$

for $\boldsymbol{x} \in \mathbb{R}^m$, we can conjecture that due to the integral over $\mathbb{R}^{N-m}$ the uniform convergence of $(f_{\mathsf{h}}^{(K)})_{K=1}^{\infty}$ to f_{h} alone is generally not sufficient to infer the uniform convergence of $(f_{\mathsf{x}}^{(K)})_{K=1}^{\infty}$ to f_{x}.

Bibliography

[1] T. T. Nguyen, H. D. Nguyen, F. Chamroukhi, and G. J. McLachlan, "Approximation by finite mixtures of continuous density functions that vanish at infinity," *Cogent Math. Statist.*, vol. 7, no. 1, p. 1750861, 2020.

[2] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 723–773, Mar. 2012.

[3] E. Björnson, J. Hoydis, and L. Sanguinetti, *Massive MIMO Networks: Spectral, Energy, and Hardware Efficiency*. Now Publishers Inc., Nov. 2017, vol. 11, no. 3–4.

[4] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[5] O. E. Ayach, S. Rajagopal, S. Abu-Surra, Z. Pi, and R. W. Heath, "Spatially sparse precoding in millimeter wave MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1499–1513, Mar. 2014.

[6] L. Liang, W. Xu, and X. Dong, "Low-complexity hybrid precoding in massive multiuser MIMO systems," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 653–656, Dec. 2014.

[7] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 831–846, Oct. 2014.

[8] R. Méndez-Rial, C. Rusu, N. González-Prelcic, A. Alkhateeb, and R. W. Heath, "Hybrid MIMO architectures for millimeter wave communications: Phase shifters or switches?" *IEEE Access*, vol. 4, pp. 247–267, 2016.

[9] K. Ardah, G. Fodor, Y. C. B. Silva, W. C. Freitas, and F. R. P. Cavalcanti, "A unifying design of hybrid beamforming architectures employing phase shifters or switches," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 243–11 247, Nov. 2018.

[10] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[11] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 5302–5316, Nov. 2009.

[12] T. Blumensath, "Sampling and reconstructing signals from a union of linear subspaces," *IEEE Trans. Inf. Theory*, vol. 57, no. 7, pp. 4660–4671, Jul. 2011.

[13] T. Wiese, L. Weiland, and W. Utschick, "Inexact projected gradients on unions of subspaces," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 236–240.

[14] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constr. Approx.*, vol. 13, no. 1, pp. 57–98, Mar. 1997.

[15] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. Asilomar Conf. on Signals, Syst., and Comput.*, Nov. 1993, pp. 40–44.

[16] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[18] C. F. J. Wu, "On the convergence properties of the EM algorithm," *The Ann. of Statist.*, vol. 11, no. 1, pp. 95–103, 1983.

[19] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet, "Hilbert space embeddings and metrics on probability measures," *J. Mach. Learn. Res.*, vol. 11, pp. 1517–1561, Aug. 2010.

[20] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *Proc. Int. Conf. on Mach. Learn. (ICML)*, 2015, pp. 1718–1727.

[21] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via maximum mean discrepancy optimization," in *Proc. Conf. on Uncertainty in Artif. Intell. (UAI)*, 2015, pp. 258–267.

[22] Y. Ren, J. Li, Y. Luo, and J. Zhu, "Conditional generative moment-matching networks," in *Proc. Adv. in Neural Inf. Process. Syst. (NeurIPS)*, 2016, pp. 2928–2936.

[23] H. Gao and H. Huang, "Joint generative moment-matching network for learning structural latent code," in *Proc. Int. Joint Conf. on Artif. Intell. (IJCAI)*, Jul. 2018, pp. 2121–2127.

[24] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[25] M. Koller, B. Fesl, N. Turan, and W. Utschick, "An asymptotically optimal approximation of the conditional mean channel estimator based on Gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. (ICASSP)*, 2022, pp. 5268–5272.

[26] ——, "An asymptotically MSE-optimal estimator based on Gaussian mixture models," *IEEE Trans. Signal Process.*, vol. 70, pp. 4109–4123, 2022.

[27] G. Schay, *Introduction to Probability with Statistical Applications*. Birkhäuser Boston, Jun. 2016.

[28] S. Resnick, *A Probability Path*. Birkhäuser Boston, 2005.

[29] A. Klenke, *Probability Theory: A Comprehensive Course*. Springer, 2008.

[30] E. M. Goggin, "Convergence in distribution of conditional expectations," *The Ann. of Probability*, vol. 22, no. 2, pp. 1097–1114, 1994.

[31] Y. Gu and Y. D. Zhang, "Information-theoretic pilot design for downlink channel estimation in FDD massive MIMO systems," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2334–2346, May 2019.

[32] R. M. Gray, "Toeplitz and circulant matrices: A review," *Found. and Trends in Commun. and Inf. Theory*, no. 3, pp. 155–239, 2006.

[33] J. Kermoal, L. Schumacher, K. Pedersen, P. Mogensen, and F. Frederiksen, "A stochastic MIMO radio channel model with experimental validation," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 6, pp. 1211–1226, 2002.

[34] M. Šimko, C. Mehlführer, M. Wrulich, and M. Rupp, "Doubly dispersive channel estimation with scalable complexity," in *Proc. Int. ITG Workshop on Smart Antennas (WSA)*, Feb. 2010, pp. 251–256.

[35] 3GPP, "Spatial channel model for multiple input multiple output (MIMO) simulations," 3rd Generation Partnership Project (3GPP), Tech. Rep. 25.996 (V16.0.0), Jul. 2020.

[36] D. Neumann, T. Wiese, and W. Utschick, "Learning the MMSE channel estimator," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 2905–2917, Jun. 2018.

[37] S. Jaeckel, L. Raschkowski, K. Börner, and L. Thiele, "QuaDRiGa: A 3-D multi-cell channel model with time evolution for enabling virtual field trials," *IEEE Trans. Antennas Propag.*, vol. 62, no. 6, pp. 3242–3256, 2014.

[38] S. Jaeckel, L. Raschkowski, K. Börner, L. Thiele, F. Burkhardt, and E. Eberlein, "QuaDRiGa: Quasi deterministic radio channel generator, user manual and documentation," Fraunhofer Heinrich Hertz Institute, Tech. Rep., v2.2.0, 2019.

[39] 3GPP, "NR; physical channels and modulation," 3rd Generation Partnership Project (3GPP), Tech. Spec. 38.211 (V16.7.0), Sep. 2021.

[40] M. Kurras, S. Dai, S. Jaeckel, and L. Thiele, "Evaluation of the spatial consistency feature in the 3GPP geometry-based stochastic channel model," in *Proc. IEEE Wireless Commun. and Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–6.

[41] B. Fesl, N. Turan, M. Koller, and W. Utschick, "A low-complexity MIMO channel estimator with implicit structure of a convolutional neural network," in *Proc. IEEE Int. Workshop on Signal Process. Adv. in Wireless Commun. (SPAWC)*, 2021.

[42] S. A. Busari, K. M. S. Huq, S. Mumtaz, L. Dai, and J. Rodriguez, "Millimeter-wave massive MIMO communication for future wireless systems: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 836–869, Secondquarter 2018.

[43] K. Hassan, M. Masarra, M. Zwingelstein, and I. Dayoub, "Channel estimation techniques for millimeter-wave communication systems: Achievements and challenges," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 1336–1363, 2020.

[44] A. Alkhateeb, G. Leus, and R. W. Heath, "Compressed sensing based multi-user millimeter wave systems: How many measurements are needed?" in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. (ICASSP)*, 2015, pp. 2909–2913.

[45] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. IEEE Inf. Theory Workshop on Inf. Theory (ITW)*, Jan. 2010, pp. 1–5.

[46] A. Maleki, L. Anitori, Z. Yang, and R. G. Baraniuk, "Asymptotic analysis of complex LASSO via complex approximate message passing (CAMP)," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4290–4308, Jul. 2013.

[47] M. Soltani, V. Pourahmadi, and H. Sheikhzadeh, "Pilot pattern design for deep learning-based channel estimation in OFDM systems," *IEEE Wireless Commun. Lett.*, vol. 9, no. 12, pp. 2173–2176, Dec. 2020.

[48] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 652–655, Apr. 2019.

[49] S. Coleri, M. Ergen, A. Puri, and A. Bahai, "Channel estimation techniques based on pilot arrangement in OFDM systems," *IEEE Trans. Broadcast.*, vol. 48, no. 3, pp. 223–229, 2002.

[50] L. Tong, B. Sadler, and M. Dong, "Pilot-assisted wireless transmissions: General model, design criteria, and signal processing," *IEEE Signal Process. Mag.*, vol. 21, no. 6, pp. 12–25, Nov. 2004.

[51] M. Koller and W. Utschick, "Learning a compressive sensing matrix with structural constraints via maximum mean discrepancy optimization," *Signal Process.*, p. 108553, 2022.

[52] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, 2008.

[53] K. Ardah and M. Haardt, "Compressed sensing constant modulus constrained projection matrix design and high-resolution DoA estimation methods," in *Proc. Int. ITG Workshop on Smart Antennas (WSA)*, Nov. 2021, pp. 1–5.

[54] L. Baldassarre, Y. Li, J. Scarlett, B. Gözcü, I. Bogunovic, and V. Cevher, "Learning-based compressive subsampling," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 4, pp. 809–822, Jun. 2016.

[55] Y. Weiss, H. S. Chang, and W. T. Freeman, "Learning compressed sensing," in *Proc. Allerton Conf. Commun., Control, and Comput.*, 2007.

[56] M. Elad, "Optimized projections for compressed sensing," *IEEE Trans. Signal Process.*, vol. 55, no. 12, pp. 5695–5702, Dec. 2007.

[57] J. M. Duarte-Carvajalino and G. Sapiro, "Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1395–1408, Jul. 2009.

[58] A. Sadeghian, B. Bah, and V. Cevher, "Energy-aware adaptive bi-Lipschitz embeddings," in *Proc. Int. Conf. Sampling Theory and Appl. (SampTA)*, Jul. 2013.

[59] C. Hegde, A. C. Sankaranarayanan, W. Yin, and R. G. Baraniuk, "NuMax: A convex approach for learning near-isometric linear embeddings," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 6109–6121, Nov. 2015.

[60] S. Wu, A. Dimakis, S. Sanghavi, F. Yu, D. Holtmann-Rice, D. Storcheus, A. Rostamizadeh, and S. Kumar, "Learning a compressed sensing measurement matrix via gradient unrolling," in *Proc. Int. Conf. on Mach. Learn. (ICML)*, vol. 97, Jun. 2019, pp. 6828–6839.

[61] P. Wu, Z. Liu, and J. Cheng, "Compressed CSI feedback with learned measurement matrix for mmWave massive MIMO," *arXiv*, 2019. [Online]. Available: http://arxiv.org/abs/1903.02127

[62] P. Wu and J. Cheng, "Acquiring measurement matrices via deep basis pursuit for sparse channel estimation in mmWave massive MIMO systems," *arXiv*, 2020. [Online]. Available: http://arxiv.org/abs/2007.05177

[63] F. Mezzadri, "How to generate random matrices from the classical compact groups," *Notices of the Amer. Math. Society*, vol. 54, Oct. 2006.

[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. in Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019, pp. 8026–8037.

[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[67] Z. Yang, J. Li, P. Stoica, and L. Xie, "Sparse methods for direction-of-arrival estimation," in *Academic Press Library in Signal Process.*, 2018, pp. 509–581.

[68] J. Haupt, W. U. Bajwa, G. Raz, and R. Nowak, "Toeplitz compressed sensing matrices with applications to sparse channel estimation," *IEEE Trans. Inf. Theory*, vol. 56, no. 11, pp. 5862–5875, Nov. 2010.

[69] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. in Neural Inf. Process. Syst. (NeurIPS)*, 2014, pp. 2672–2680.