universität
wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

"A Global Matrix Factorization Problem for Interference Alignment"

verfasst von / submitted by

Markus Levonyak, B.Sc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2016 / Vienna, 2016

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 940
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Scientific Computing
Betreut von / Supervisor:	Assoz. Prof. Dr. Wilfried Gansterer, Privatdoz.

Abstract

Interference alignment on the K-user interference channel is a technique from information theory that aims to avoid electromagnetic interference in wireless networks by employing precoding matrices $\boldsymbol{V}_j$ at the transmitters and postcoding matrices $\boldsymbol{U}_k$ at the receivers such that $\forall j \neq k \colon \boldsymbol{U}_k^{\mathrm{H}} \boldsymbol{H}_{kj} \boldsymbol{V}_j = \boldsymbol{0}$ for given channel matrices $\boldsymbol{H}_{kj}$. An algorithm by Gomadam et al. solves the interference alignment problem by iteratively optimizing $\boldsymbol{V}_j$ and $\boldsymbol{U}_k$ at all transmitters and receivers. However, it has not yet been proved that this iterative algorithm converges to a global optimum.

For the purpose of finding a direct and optimal solution, we reformulate the interference alignment problem as the equivalent problem to factorize a given global channel matrix $\boldsymbol{H}$ such that $\boldsymbol{H} = \bar{\boldsymbol{U}} \bar{\boldsymbol{\Sigma}} \bar{\boldsymbol{V}}^{\mathrm{H}}$, where $\bar{\boldsymbol{U}}$, $\bar{\boldsymbol{\Sigma}}$, and $\bar{\boldsymbol{V}}$ are matrices of specific sparsity patterns. As a first step towards a direct solution to this global matrix factorization problem, we focus on constructing $\bar{\boldsymbol{\Sigma}}$ by applying unitary Householder and Givens transformations to $\boldsymbol{H}$. We propose several variants of a direct algorithm that create $\bar{\boldsymbol{\Sigma}}$ with full, tridiagonal, and bidiagonal main diagonal blocks of a block-diagonal submatrix. The bidiagonal blocks algorithm variant accepts all input parameter values in conformance with the feasibility criteria for interference alignment from the literature.

In consideration of the main diagonal elements that have to be equal to zero, we argue that $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ cannot be products of Householder and Givens matrices, and hence, there is no direct solution to the general global matrix factorization problem solely based on Householder reflections and Givens rotations. On the basis of numerical experiments with a prototype implementation of the iterative algorithm, we discover that a prototype implementation of our direct algorithm requires substantially less operations, which shows the relevance and potential of a direct solution to the interference alignment problem.

Zusammenfassung

Interferenzausrichtung auf dem K-Teilnehmer-Interferenzkanal ist ein Verfahren der Informationstheorie zur Vermeidung elektromagnetischer Interferenz in drahtlosen Netzwerken durch Anwendung von Präkodierungsmatrizen $\boldsymbol{V}_j$ auf Senderseite und Postkodierungsmatrizen $\boldsymbol{U}_k$ auf Empfängerseite, sodass $\forall j \neq k\colon \boldsymbol{U}_k^{\mathrm{H}} \boldsymbol{H}_{kj} \boldsymbol{V}_j = \boldsymbol{0}$ für gegebene Kanalmatrizen $\boldsymbol{H}_{kj}$. Ein Algorithmus von Gomadam u. a. löst das Interferenzausrichtungsproblem durch iterative Optimierung von $\boldsymbol{V}_j$ und $\boldsymbol{U}_k$ bei allen Sendern und Empfängern. Es wurde jedoch bisher nicht bewiesen, dass dieser iterative Algorithmus gegen ein globales Optimum konvergiert.

Um eine direkte und optimale Lösung zu finden, reformulieren wir das Interferenzausrichtungsproblem als das äquivalente Problem, eine gegebene globale Kanalmatrix $\boldsymbol{H}$ so zu faktorisieren, dass $\boldsymbol{H} = \bar{\boldsymbol{U}} \bar{\boldsymbol{\Sigma}} \bar{\boldsymbol{V}}^{\mathrm{H}}$, wobei $\bar{\boldsymbol{U}}$, $\bar{\boldsymbol{\Sigma}}$ und $\bar{\boldsymbol{V}}$ Matrizen mit bestimmten dünnbesetzten Strukturen sind. Als ersten Schritt in Richtung einer direkten Lösung dieses globalen Matrizenfaktorisierungsproblems konzentrieren wir uns darauf, $\bar{\boldsymbol{\Sigma}}$ durch Anwendung von unitären Householder- und Givenstransformationen auf $\boldsymbol{H}$ zu konstruieren. Wir schlagen mehrere Varianten eines direkten Algorithmus vor, die $\bar{\boldsymbol{\Sigma}}$ mit vollen, tridiagonalen und bidiagonalen Hauptdiagonalblöcken einer blockdiagonalen Untermatrix erzeugen. Die Algorithmenvariante, die bidiagonale Blöcke hervorbringt, akzeptiert alle Eingabeparameterwerte, die mit den Durchführbarkeitskriterien für Interferenzausrichtung aus der Literatur übereinstimmen.

Unter Betrachtung der Hauptdiagonalelemente, die gleich null sein müssen, argumentieren wir, dass $\bar{\boldsymbol{U}}$ und $\bar{\boldsymbol{V}}$ keine Produkte von Householder- und Givensmatrizen sein können und es daher keine direkte Lösung des allgemeinen globalen Matrizenfaktorisierungsproblems ausschließlich basierend auf Householderspiegelungen und Givensdrehungen gibt. Auf Grundlage von numerischen Experimenten mit einer prototypischen Implementierung des iterativen Algorithmus erkennen wir, dass eine prototypische Implementierung unseres direkten Algorithmus beträchtlich weniger Operationen benötigt, wodurch die Bedeutung und das Potenzial einer direkten Lösung des Interferenzausrichtungsproblems ersichtlich ist.

Contents

List of figures

List of tables

List of definitions and theorems

List of algorithms

Chapter 1

Introduction

The electromagnetic spectrum—and thus also the radio frequency spectrum—is a limited natural resource. Both human utilization and natural phenomena commonly cause disturbance of signals carried by radio waves. This radio frequency interference is of particular interest for engineering applications, especially in radio communications. In the context of wireless networks, there have been several attempts to manage radio frequency interference in order to minimize its effect on the quality of the transmitted signals.

One of these approaches has been *interference alignment*, a technique from information theory that strives to avoid interference by determining and employing appropriate transmit and receive filters. In the literature, an *iterative* optimization algorithm for computing matrix representations of these transmit and receive filters has been proposed. Although this algorithm in practice seems to converge to an optimal solution, this convergence has not yet been proved.

This thesis aims to find a *direct* and hence optimal solution to the interference alignment problem by applying specific unitary transformations—namely *Householder reflections* and *Givens rotations*. For this purpose, a global matrix formulation of the interference alignment problem is adopted, which leads to a dense matrix factorization problem that resembles the singular value decomposition (SVD) [GV13, pp. 76–81] in some aspects.

1.1 Research objective

The primary research objective is to develop a direct (constructive) algorithm for solving this global matrix factorization problem for interference alignment, or to show that Householder and Givens transformations are insufficient for achieving this goal. The secondary research objective is to compare the computational cost of the existing iterative and a possible novel direct algorithm based on numerical experiments conducted for the iterative solution.

1.2 Chapter overview

Chapter 2 outlines necessary background from information theory and illustrates basic principles of interference alignment. In particular, the communication channel, the interference channel, degrees of freedom, interference management, and interference alignment on both the MIMO X channel and the K-user interference channel are introduced. Most importantly, the conditions and feasibility criteria for interference alignment on the K-user interference channel are specified.

Chapter 3 concisely explains fundamentals and methods which are essential for understanding the remainder of the thesis. This includes direct methods in contrast to iterative methods, Householder reflections, Givens rotations, and the quantification of computational cost.

Chapter 4 precisely formulates the interference alignment problem and presents the iterative optimization algorithm by Gomadam et al. for solving the interference alignment problem. Besides the original algorithm, a variant for interference alignment on the symmetric K-user interference channel is described.

Chapter 5 reformulates the interference alignment problem as the equivalent interference alignment decomposition problem by introducing global matrices. As a first step towards a direct solution, the relaxed interference alignment decomposition problem, a simplified variant of the interference alignment decomposition problem, is defined. On this basis, various possibilities how to apply Householder reflections and Givens rotations are investigated.

Chapter 6 suggests a direct solution to the relaxed matrix factorization problem. Three tridiagonal blocks variants, a full blocks variant, and a bidiagonal blocks variant of a constructive algorithm for solving the relaxed interference alignment decomposition problem are listed and discussed. The correctness of the algorithm is proved for the bidiagonal blocks variant.

Chapter 7 shows that the application of Householder and Givens transformations is not sufficient for solving the general matrix factorization problem. Furthermore, some alternative strategies for approaching a direct solution to the interference alignment decomposition problem are mentioned.

Chapter 8 describes prototype Matlab/Octave implementations of the previously presented iterative and constructive algorithms. Apart from that, the results of two series of numerical experiments for determining the numbers of iterations required by the iterative implementation are illustrated, and the operations of both the iterative and the direct implementation are counted and compared.

Chapter 9 concludes by reviewing the findings of this thesis and providing an outlook on future work.

Appendix A lists the source code of the prototype Matlab/Octave implementations of the iterative and constructive algorithms.

1.3 Related work

Cadambe and Jafar introduce and describe interference alignment on the K-user time-varying interference channel [CJ08]. They argue that this method achieves $\frac{K}{2}$ degrees of freedom.

Gomadam, Cadambe, and Jafar present a numerical approach to interference alignment that only requires local channel knowledge at each node [GCJ11]. They propose two iterative algorithms, one that minimizes leakage interference and one that maximizes the signal-to-interference-plus-noise ratio at the receivers.

González, Beltrán, and Santamaría examine the feasibility of interference alignment on the K-user MIMO interference channel and suggest a polynomial-time feasibility test [GBS14].

Guillaud proposes a global matrix formulation of the interference alignment problem for developing a new algorithm [Gui12].

Trefethen and Bau as well as Golub and Van Loan provide comprehensive introductions to numerical linear algebra [TB97; GV13]. Particularly, they explain Householder reflections, Givens rotations, and matrix factorization techniques like the LU decomposition, the QR decomposition, and the singular value decomposition (SVD).

To the best of our knowledge, a direct factorization-based approach to solving the global interference alignment problem has not been investigated so far.

1.4 Notation and conventions

$\mathbb{N}$ is the set of natural numbers without zero, and $\mathbb{N}_0 \coloneqq \mathbb{N} \cup \{0\}$. $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$ denote the sets of integers, real numbers, and complex numbers, respectively. $\mathbb{C}^m$ and $\mathbb{C}^{m \times n}$ are the sets of complex vectors of length m and complex matrices of size $m \times n$. By convention, $\mathbb{C}^m = \mathbb{C}^{m \times 1}$, i.e., the elements of $\mathbb{C}^m$ are column vectors.

π and e are the familiar mathematical constants. Depending on the context, i denotes the imaginary unit or an index. $\Re(z)$, $\Im(z)$, $|z|$, $\arg(z)$, and $\overline{z}$ are the real part, the imaginary part, the absolute value, the argument, and the complex conjugate of $z \in \mathbb{C}$.

A lowercase letter, e.g., a, denotes a scalar, a bold lowercase letter, e.g., $\boldsymbol{a}$, denotes a vector, and a bold uppercase letter, e.g., $\boldsymbol{A}$, denotes a matrix. However, there are exceptions for variables well established in the literature, e.g., K, M, and N in the context of interference alignment. The notations $a = a(x)$, $\boldsymbol{a} = \boldsymbol{a}(x)$, and $\boldsymbol{A} = \boldsymbol{A}(x)$ mean that a, $\boldsymbol{a}$, and $\boldsymbol{A}$ are functions of x, where x may be one or more scalars, vectors, and matrices. $\langle x, y \rangle$ and $\langle x, y, z \rangle$ are pairs and triples of scalars, vectors, and matrices.

$\boldsymbol{A}(i_1 \colon i_2, :)$ and $\boldsymbol{A}(:, j_1 \colon j_2)$ denote the rows i_1 to i_2 and the columns j_1 to j_2 of matrix $\boldsymbol{A}$, respectively. $\boldsymbol{A}(i_1 \colon i_2, j_1 \colon j_2)$ is the submatrix of $\boldsymbol{A}$ at the intersection of rows i_1 to i_2 and columns j_1 to j_2. For a matrix $\boldsymbol{A} = (a_{ij})$, $a_{ij} = \boldsymbol{A}(i \colon i, j \colon j)$ denotes the element in row i and column j of $\boldsymbol{A}$.

$\boldsymbol{I}_n \in \mathbb{C}^{n\times n}$ or simply $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{e}_j \coloneqq \boldsymbol{I}(:, j:j)$ is a unit vector, $\mathbf{0}_{mn} \in \mathbb{C}^{m\times n}$ or simply $\mathbf{0}$ is the zero matrix, and $\mathbf{1}_m \in \mathbb{C}^m$ is the vector of all ones. $\boldsymbol{a}^{\mathrm{T}}$ and $\boldsymbol{a}^{\mathrm{H}}$ are the transpose and conjugate transpose of vector $\boldsymbol{a}$. Similarly, $\boldsymbol{A}^{\mathrm{T}}$ and $\boldsymbol{A}^{\mathrm{H}}$ are the transpose and conjugate transpose of matrix $\boldsymbol{A}$. $\|\boldsymbol{a}\|_p$ denotes the p-norm of $\boldsymbol{a}$. $\mathrm{tr}(\boldsymbol{A})$ and $\mathrm{rank}(\boldsymbol{A})$ are the trace and rank of $\boldsymbol{A}$.

$\mathcal{O}$ and o are the big O and small o Landau symbols. $\textsc{Function}(x)$ denotes a function call, where x stands for the function arguments.

Chapter 2

Interference alignment

This chapter aims to provide an introduction to the basic principles of *interference alignment*, in particular, interference alignment on the K-user interference channel:

> Interference alignment on the K user interference channel refers to the idea of constructing signals in such a way that they cast overlapping shadows over one half of the signal space observed by each receiver where they constitute interference, leaving the other half of the signal space free of interference for the desired signal. [GCJ11, p. 3309]

Before we can fully understand the meaning of the above quotation, we have to look into some information-theoretic fundamentals first. Section 2.1 briefly introduces the concept of the *communication channel*, which is the foundation for all subsequently presented terms and definitions. In Section 2.2 we will explain what we understand by the *interference channel*, a type of communication channel with specific properties, and compare the MIMO X channel to the K-user interference channel. Section 2.3 summarizes different approaches for coping with interference. Sections 2.4 and 2.5 describe the main ideas of interference alignment on the MIMO X channel and on the K-user interference channel. Finally, we will see in Section 2.6 which parameter combinations allow a solution to the interference alignment problem.

2.1 Communication channel

In his fundamental work from 1949, *The Mathematical Theory of Communication*, Claude E. Shannon identifies the *communication channel*, or *channel*, as one of five parts of a *communication system* (along with the *information source*, the *transmitter*, the *receiver*, and the *destination*) [SW49, pp. 33–34]:

> The *channel* is merely the medium used to transmit the signal from transmitter to receiver. It may be a pair of wires, a coaxial cable, a band of radio frequen-

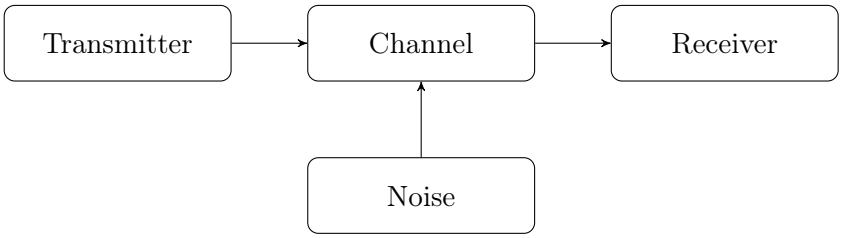


Figure 2.1: The parts of a communication system around the channel.

> cies, a beam of light, etc. During transmission, or at one of the terminals, the signal may be perturbed by noise. [SW49, p. 34]

A similar model is introduced in [Ash65, p. 1]. Figure 2.1 visualizes the parts of a communication system around the channel. In conformance with [Cio09, pp. 402–405] and [EC80, pp. 1467–1470], the following types of communication channels can be distinguished:

- the *single-user channel*,
- the *multiple-access channel*,
- the *broadcast channel*, and
- the *interference channel*.

Figure 2.2 shows how transmitters and receivers communicate over each of these different types of channels. The multiple-access channel, the broadcast channel and the interference channel together form the group of *multiple-user channels*. The following definition of the multiple-user channel is derived from [Cio09, pp. 404–405].

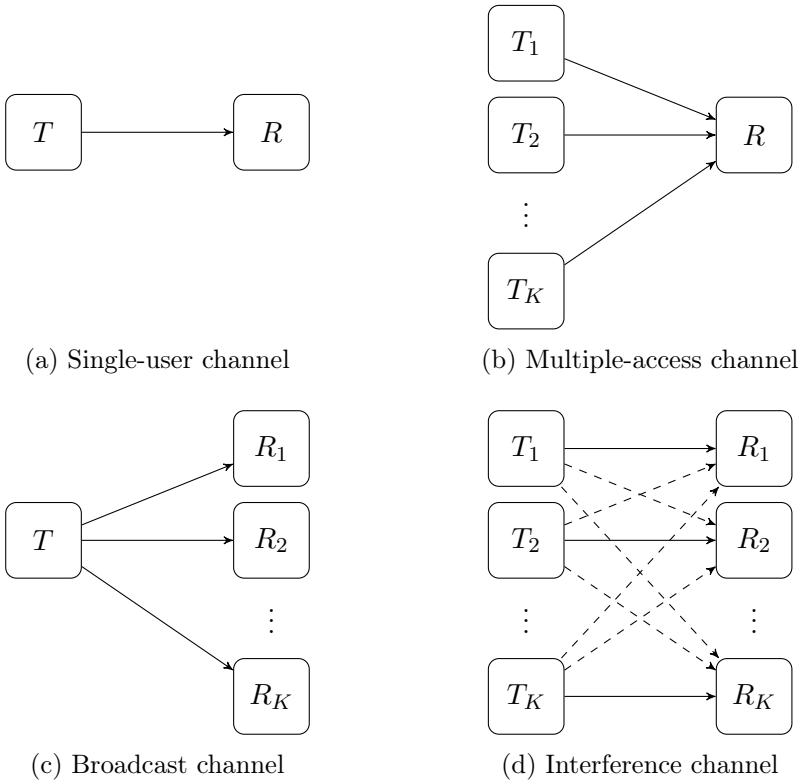
Definition 2.1 (Multiple-user channel)**.** Assume there are $K_T > 0$ transmitters and $K_R > 0$ receivers. Let

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{K_T} \end{pmatrix}, \ \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{K_R} \end{pmatrix}, \ \boldsymbol{H} = \begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1K_T} \\ H_{21} & H_{22} & \cdots & H_{2K_T} \\ \vdots & \vdots & \ddots & \vdots \\ H_{K_R1} & H_{K_R2} & \cdots & H_{K_RK_T} \end{pmatrix}, \text{ and } \boldsymbol{n} = \begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_{K_R} \end{pmatrix}.$$

$\boldsymbol{x}$ is referred to as the *transmit vector*, $\boldsymbol{y}$ as the *receive vector*, $\boldsymbol{H}$ as the *channel fade matrix*, or *channel matrix*, and $\boldsymbol{n}$ as the *noise vector*. Then, the output of the *multiple-user channel* is described as

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}.$$

Hereinafter, we will assume $K_T = K_R$, i.e., the number of transmitters equals the number of receivers. Table 2.1 lists the transmit vector $\boldsymbol{x}$, receive vector $\boldsymbol{y}$, channel matrix $\boldsymbol{H}$, and noise vector $\boldsymbol{n}$ for the different types of multiple-user channels.



(a) Single-user channel (b) Multiple-access channel

(c) Broadcast channel (d) Interference channel

Figure 2.2: Different types of communication channels with a single transmitter T or multiple transmitters T_j, $j \in \{1, 2, \ldots, K\}$, and a single receiver R or multiple receivers R_k, $k \in \{1, 2, \ldots, K\}$.

2.2 Interference channel

The concept of the interference channel was first introduced by Aydano B. Carleial. In 1978, he provided the following definition:

> The situation often occurs where several sender-receiver pairs share a common communication channel so that transmission of information from one sender to its corresponding receiver interferes with communications between the other senders and their receivers. In radio communications, for example, since the electromagnetic spectrum is a limited resource, frequency bands are often simultaneously used by several radio links that are not completely isolated. A communication channel that is shared in this manner is called an *interference channel*. [Car78, p. 60]

Different variants of interference channels have been introduced over time. One distinguishing feature has been the number of antennas at each transmitter and receiver. While the channel input is affected by the number of antennas at each transmitter, the channel output is determined by the number of antennas at each receiver. We can differentiate between the following types of interference channels [AV11, p. 2565]:

Table 2.1: The transmit vector $\boldsymbol{x}$, receive vector $\boldsymbol{y}$, channel matrix $\boldsymbol{H}$, and noise vector $\boldsymbol{n}$ for the different types of multiple-user channels with K users.

	$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{H}$	$\boldsymbol{n}$
Multiple-access channel	$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix}$	y	$\begin{pmatrix} H_1 & H_2 & \cdots & H_K \end{pmatrix}$	n
Broadcast channel	x	$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{pmatrix}$	$\begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_K \end{pmatrix}$	$\begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_K \end{pmatrix}$
Interference channel	$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix}$	$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{pmatrix}$	$\begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1K} \\ H_{21} & H_{22} & \cdots & H_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ H_{K1} & H_{K2} & \cdots & H_{KK} \end{pmatrix}$	$\begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_K \end{pmatrix}$

- *single-input single-output* (SISO),
- *multiple-input single-output* (MISO),
- *single-input multiple-output* (SIMO), and
- *multiple-input multiple-output* (MIMO).

Figure 2.3 visualizes these four types of interference channels based on the number of antennas at each transmitter and receiver.

2.2.1 MIMO X channel

A specific kind of interference channel that has been studied extensively in the literature over the last years, inter alia, within the context of interference alignment (cf. Section 2.4), is the X *channel* [JS08; MMK08; HCJ12; AGK13; LAS14]. Lashgari et al. describe the X channel as follows:

> The X-channel is a canonical setting for the information-theoretic study of interference management in wireless networks. This channel consists of two transmitters causing interference at two receivers, and each transmitter aims to communicate intended messages to both receivers. [LAS14, p. 2180]

Maddah-Ali et al. concretize a similar description in [MMK08] by adding the following clarification:

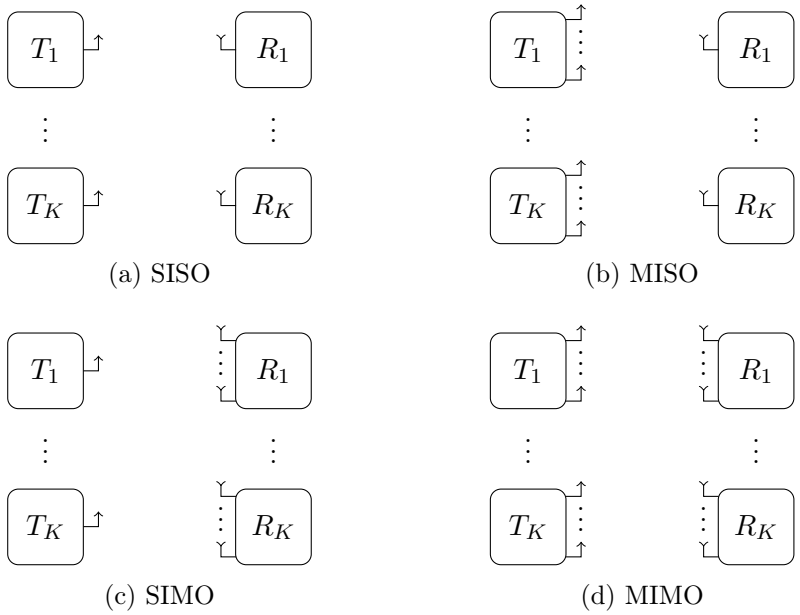


Figure 2.3: The types of interference channels based on the number of antennas at each transmitter T_j, $j \in \{1, 2, \ldots, K\}$, and each receiver R_k, $k \in \{1, 2, \ldots, K\}$.

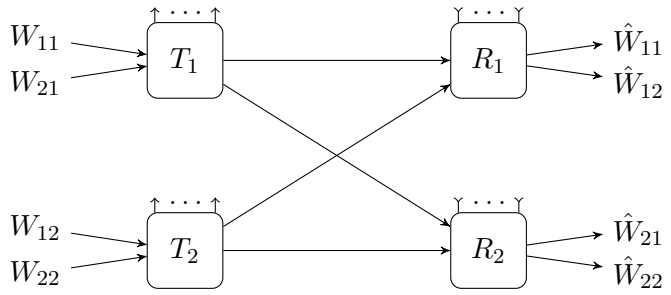


Figure 2.4: The MIMO X channel with transmitter T_j's message W_{kj} intended for receiver R_k, $j, k \in \{1, 2\}$. This figure is based on [JS08, fig. 1].

> In this scenario, it is assumed that each transmitter is unaware of the other transmitter's data (noncooperative scenario). This system can be considered as a combination of two broadcast channels (from the transmitters' points of view) and two multiple-access channels (from the receivers' points of view). [MMK08, p. 3457]

If each of the two transmitters and two receivers is equipped with multiple antennas, we speak of the *MIMO X channel* [JS08, p. 151], as shown in Figure 2.4. The subsequent definition is based on [MMK08, pp. 3458–3459].

Definition 2.2 (MIMO X channel)**.** Let M_j and N_k, $j, k \in \{1, 2\}$, be the number of antennas at transmitter T_j and receiver R_k, respectively. Without loss of generality, let $M_1 \geq M_2$ and $N_1 \geq N_2$. Furthermore, let $\boldsymbol{x}_j \in \mathbb{C}^{M_j \times 1}$ be the transmit vector

of transmitter T_j, $\boldsymbol{y}_k \in \mathbb{C}^{N_k \times 1}$ the receive vector of receiver R_k, $\boldsymbol{H}_{kj} \in \mathbb{C}^{N_k \times M_j}$ the channel matrix that represents the channel between transmitter T_j and receiver R_k, and $\boldsymbol{n}_k \in \mathbb{C}^{N_k \times 1}$ the white Gaussian noise vector at receiver R_k with zero mean and identity covariance matrix. Then the output of the *MIMO X channel* is described as

$$\boldsymbol{y}_1 = \boldsymbol{H}_{11}\boldsymbol{x}_1 + \boldsymbol{H}_{12}\boldsymbol{x}_2 + \boldsymbol{n}_1,$$

$$\boldsymbol{y}_2 = \boldsymbol{H}_{21}\boldsymbol{x}_1 + \boldsymbol{H}_{22}\boldsymbol{x}_2 + \boldsymbol{n}_2.$$

It is assumed that the channel matrices $\boldsymbol{H}_{kj}$, $j, k \in \{1, 2\}$, are known by both transmitters and both receivers.

2.2.2 K-user interference channel

Another well-studied type of interference channel is the K-user interference channel, an interference channel with K transmitters and K receivers [CJ08; GJ10; BCT11; GCJ11; RLL12; AGK13; GBS14; TAV14]. Each transmitter and receiver is equipped with one or multiple antennas. Thus, we distinguish

- the *K-user SISO interference channel*,
- the *K-user MISO interference channel*,
- the *K-user SIMO interference channel*, and
- the *K-user MIMO interference channel*.

Subsequently, we will focus on time-varying variants of the K-user interference channel, as introduced in [CJ08]. The following three definitions of the K-user time-varying interference channel are based on [CJ08, p. 3426] and [GJ10, p. 6042].

Definition 2.3 (K-user time-varying SISO interference channel)**.** Let $k \in \{1, 2, \ldots, K\}$ be the user index and $t \in \mathbb{N}$ the time slot index. Moreover, let $x_k(t) \in \mathbb{C}$ be the channel input signal of transmitter T_k, $y_k(t) \in \mathbb{C}$ the channel output signal of receiver R_k, $H_{kj}(t) \in \mathbb{C}$ the channel fade coefficient from transmitter T_j, $j \in \{1, 2, \ldots, K\}$, to receiver R_k over the t^{th} time slot, and $n_k(t) \in \mathbb{C}$ the additive white Gaussian noise term at receiver R_k. Then the output of the *K-user time-varying SISO interference channel* at receiver R_k over the t^{th} time slot is described as

$$y_k(t) = \sum_{j=1}^{K} H_{kj}(t)x_j(t) + n_k(t).$$

A simplified variant of the K-user time-varying SISO interference channel is the *constant* K-user SISO interference channel defined below.

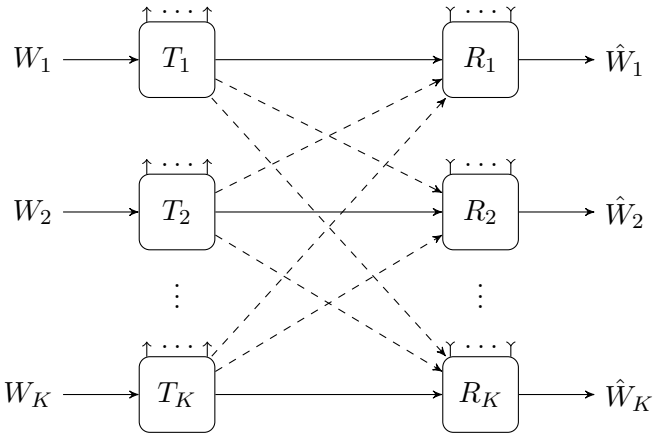


Figure 2.5: The K-user MIMO interference channel with transmitter T_k's message W_k intended for receiver R_k, $k \in \{1, 2, \ldots, K\}$.

Definition 2.4 (Constant K-user time-varying SISO interference channel)**.** Let k, t, $x_k(t)$, $y_k(t)$, and $n_k(t)$ be given as in Definition 2.3. Then the output of the *constant K-user time-varying SISO interference channel* at receiver R_k over the t^{th} time slot is described as

$$y_k(t) = x_k(t) + \sqrt{-1} \sum_{\substack{j=1 \\ j \neq k}}^{K} x_j(t) + n_k(t).$$

Hence, for the constant K-user time-varying SISO interference channel the direct channel coefficients, i.e., $H_{kk}(t)$ in Definition 2.3, are always equal to 1, while the coefficients for the interference carrying cross channels, i.e. $H_{kj}(t)$, $j \neq k$, are equal to $\sqrt{-1}$. The subsequent definition of the symmetric K-user MIMO interference channel generalizes Definition 2.3 for more than one antenna at each transmitter and receiver.

Definition 2.5 (Symmetric K-user time-varying MIMO interference channel)**.** Let $k \in \{1, 2, \ldots, K\}$ be the user index, $t \in \mathbb{N}$ the time slot index, M the number of antennas at a transmitter, and N the number of antennas at a receiver. Furthermore, let $\boldsymbol{x}_k(t) \in \mathbb{C}^{M \times 1}$ be the channel input signal vector of transmitter T_k, $\boldsymbol{y}_k(t) \in \mathbb{C}^{N \times 1}$ the channel output signal vector of receiver R_k, $\boldsymbol{H}_{kj}(t) \in \mathbb{C}^{N \times M}$ the channel matrix from transmitter T_j, $j \in \{1, 2, \ldots, K\}$, to receiver R_k over the t^{th} time slot, and $\boldsymbol{n}_k(t) \in \mathbb{C}$ the additive white Gaussian noise vector at receiver R_k. Then the output of the *symmetric K-user time-varying MIMO interference channel* at receiver R_k over the t^{th} time slot is described as

$$\boldsymbol{y}_k(t) = \sum_{j=1}^{K} \boldsymbol{H}_{kj}(t) \boldsymbol{x}_j(t) + \boldsymbol{n}_k(t).$$

Figure 2.5 visualizes the K-user MIMO interference channel. In the literature, non-symmetric variants of the K-user MIMO interference channel, which allow a different number of antennas for each transmitter and each receiver as well as a different number of degrees of freedom for each user (cf. Section 2.2.3), are also studied [GBS14], but are out of scope for this thesis.

2.2.3 Degrees of freedom

As stated by González et al., the *degrees of freedom* of the interference channel, also called the *multiplexing gain* [CJ08, p. 3425], correspond to "the maximum number of independent data streams that can be transmitted without interference in the channel" [GBS14, p. 1842]. More formally, if the capacity C of the interference channel can be written as

$$C(\mathrm{SNR}) = d \log(\mathrm{SNR}) + o(\log(\mathrm{SNR})),$$

where SNR stands for the *signal-to-noise ratio*, the interference channel has d degrees of freedom [CJ08, p. 3425].

2.3 Interference management

Over the last decades, different approaches have been pursued for managing interference, particularly with regard to wireless networks. Cadambe et al. summarize some of these *interference management* schemes in [CJ08, p. 3425] and classify them in three groups. They differentiate between techniques that

1. *decode* the interfering signal,
2. *treat* the interfering signal *as noise*, and
3. avoid interference by *orthogonalizing* channel access.

For the first group of interference management schemes, strong interference is assumed. If this precondition is met, interference can be decoded together with the desired signal. Whereas decoding the interfering signal may enhance the quality of the desired signal, the decodability limits the rates for other users. Another drawback of this approach is its usual limitation to two users. Carleial, for instance, shows in [Car75] for the two-user interference channel that under certain conditions it is possible to achieve the same rates with strong interference as without any interference. Other notable works in this area have been [HK81] and [Sat81].

On the other hand, if interference is weak, the requirement for the second approach to interference management is fulfilled. Then, the interfering signal simply can be treated as noise, which has been widely used in practical applications. Information theorists have

shown that introducing structure into the interference signals is of no benefit in these cases [ETW08; MK09; SKC09; AV09].

For the third approach to interference management to be feasible, the desired signal is required to be approximately as strong as the interfering signal. Under this condition, channel access can be orthogonalized, and time or frequency division medium access schemes that divide the available spectrum for the purpose of interference avoidance may be applied. Interference alignment belongs to this group of interference management approaches (cf. Sections 2.4 and 2.5).

2.4 Interference alignment on the MIMO X channel

According to [JS08, p. 154] and [BCT11, p. 1], the concept of *interference alignment* was first introduced within the context of the MIMO X channel (cf. Section 2.2.1). More specifically, Cadambe et al. state that it "evolved out of the degrees of freedom investigations on the two-user MIMO X channel" [CJ08, p. 3427]. While Maddah-Ali et al. initially proposed iterative optimization schemes that implicitly align interference [MMK06a; MMK06b], Jafar et al. were the first to describe an approach that explicitly achieves interference alignment on the MIMO X channel [JS08]. Other notable works in this area include [MMK08] and [HCJ12].

In the setting of the MIMO X channel (cf. Definition 2.2 and Figure 2.4), transmitter T_j, $j \in \{1, 2\}$, transmits an independent code word for an independent message to receiver R_k, $k \in \{1, 2\}$. Thus, four independent code words are transmitted at the same time, and each receiver R_k receives all of them. Since only two thereof are intended for a particular R_k, the receiver shall be able to distinguish the desired signals from the unwanted ones. The aim of interference alignment is to align the non-intended signals at each receiver for the purpose of treating them as interference, and to separate the desired signals. Jafar et al. summarize the essence of interference alignment as follows:

> Interference alignment refers to the careful choice of beamforming directions in such a manner that the desired signals are separable at their respective receivers while the interference signals are aligned, i.e., the interference vectors cast *overlapping shadows*. [JS08, p. 154]

Figure 2.6 visualizes how interference alignment on the MIMO X channel can be achieved. We can see that transmitter T_j, $j \in \{1, 2\}$, transmits the code word $\boldsymbol{x}_{kj}$ along the beamforming direction $\boldsymbol{V}_{kj}$ to receiver R_k, $k \in \{1, 2\}$. The channel matrices $\boldsymbol{H}_{kj}$ linearly transform the code words during transmission. At receiver R_1, the vectors of the desired signals, $\boldsymbol{H}_{11}\boldsymbol{V}_{11}\boldsymbol{x}_{11}$ and $\boldsymbol{H}_{12}\boldsymbol{V}_{12}\boldsymbol{x}_{12}$, are linearly independent, hence separable, whereas the vectors representing the interference, $\boldsymbol{H}_{11}\boldsymbol{V}_{21}\boldsymbol{x}_{21}$ and $\boldsymbol{H}_{12}\boldsymbol{V}_{22}\boldsymbol{x}_{22}$, are linearly dependent, thus aligned. The situation at receiver R_2 is similar. Details on how to choose the beamforming directions $\boldsymbol{V}_{kj}$ are provided in [JS08].

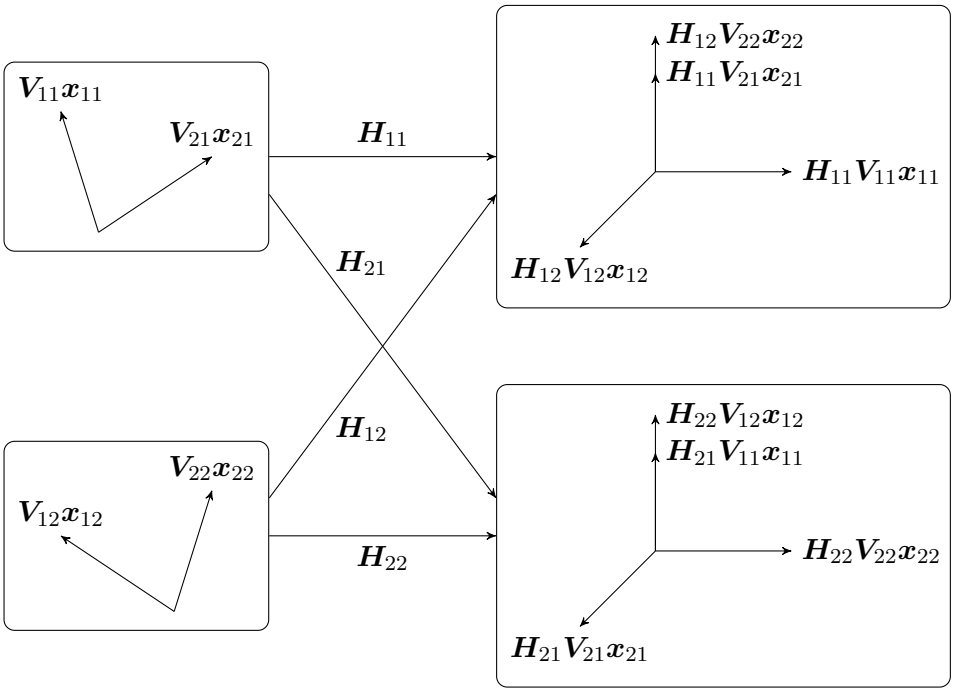


Figure 2.6: Interference alignment on the MIMO X channel. This figure is based on [JS08, fig. 2].

Based on their interference alignment schemes, Jafar et al. show that the MIMO X channel with $M > 1$ antennas at each transmitter and receiver has exactly $\frac{4}{3}M$ degrees of freedom (cf. Section 2.2.3) [JS08, pp. 159–160].

2.5 Interference alignment on the K-user interference channel

Shortly after interference alignment on the MIMO X channel was introduced (cf. Section 2.4), Cadambe et al. proposed interference alignment schemes also for the K-user (time-varying) interference channel (cf. Section 2.2.2) [CJ08]. Since then, the problem has been extensively studied in the literature, and the interference alignment approach has been applied to other scenarios [GJ10; BCT11; GCJ11; RLL12; AGK13; GBS14; TAV14].

Interference alignment on the K-user interference channel aims to "restrict all interference at every receiver to approximately half of the received signal space, leaving the other half interference-free for the desired signal" [CJ08, p. 3426]. This concept is illustrated in [CJ08, pp. 3426–3427] by means of the constant K-user time-varying SISO interference channel (cf. Definition 2.4). For a power constraint P, the channel capacity in absence of interference is $\log(1 + P)$. In this setting, interference alignment can be achieved by applying the following scheme:

- When we recall that all symbols are *complex*, each receiver relinquishes half the signal

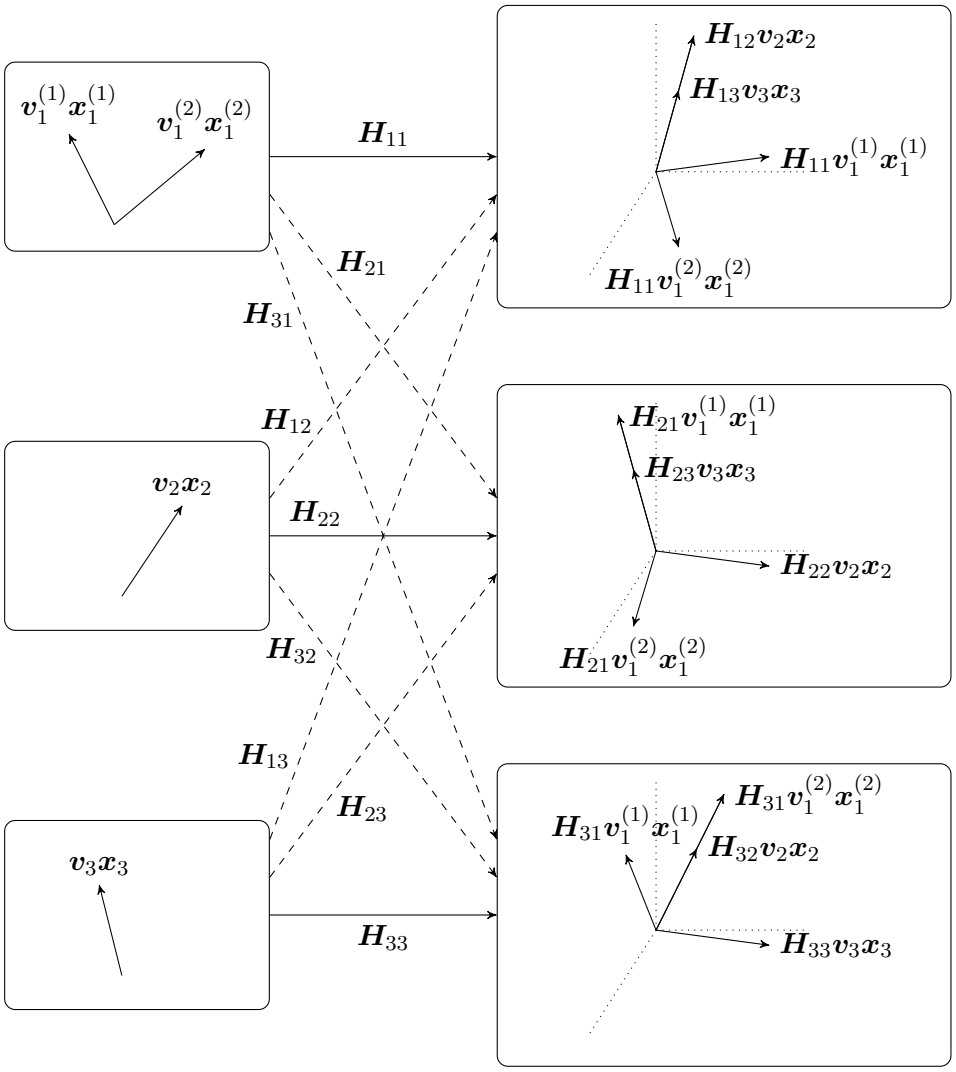


Figure 2.7: Interference alignment on the three-user SISO interference channel. This figure is based on [CJ08, fig. 1].

space by transmitting only a *real* Gaussian signal with power P.

- Then, every receiver can discard the interference-carrying imaginary part of the received signal and decode the real part at a rate $\frac{1}{2}\log(1+2P)$ [CJ08, p. 3426].

When applying this interference alignment scheme, both the sum rate of all K users and the channel capacity equal to $\frac{K}{2}\log(1+2P)$ [CJ08, p. 3427]. Therefore, the constant K-user time-varying SISO interference channel has $\frac{K}{2}$ degrees of freedom (cf. Section 2.2.3).

In contrast to the conjecture in [HN05], this result also holds for the (non-constant) K-user time-varying SISO interference channel (cf. Definition 2.3) [CJ08, pp. 3427–3434]. For example, for $K = 3$ single-antenna users and $n > 0$, $3n + 1$ degrees of freedom over a $2n + 1$ symbol extension can be achieved. Figure 2.7 visualizes how to accomplish interference alignment on the three-user SISO interference channel for the simple case

$n = 1$. Transmitter T_1 achieves two degrees of freedom by transmitting two independent code words $\boldsymbol{x}_1^{(1)}$, $\boldsymbol{x}_1^{(2)}$ along the beamforming directions $\boldsymbol{v}_1^{(1)}$, $\boldsymbol{v}_1^{(2)}$ to receiver R_1, whereas transmitter T_k, $k \in \{2, 3\}$, achieves one degree of freedom by transmitting the independent code word $\boldsymbol{x}_k$ along the beamforming direction $\boldsymbol{v}_k$ to receiver R_k.

For alignment of the interference vectors at the receivers, the following beamforming vectors can be chosen [CJ08, pp. 3428–3429]:

$$\boldsymbol{v}_2 = \mathbf{1}_3 \tag{2.1}$$

$$\boldsymbol{H}_{12}\boldsymbol{v}_2 = \boldsymbol{H}_{13}\boldsymbol{v}_3 \;\Rightarrow\; \boldsymbol{v}_3 = \big(\boldsymbol{H}_{13}\big)^{-1}\boldsymbol{H}_{12}\mathbf{1}_3 \tag{2.2}$$

$$\boldsymbol{H}_{23}\boldsymbol{v}_3 = \boldsymbol{H}_{21}\boldsymbol{v}_1^{(1)} \;\Rightarrow\; \boldsymbol{v}_1{}^{(1)} = \big(\boldsymbol{H}_{21}\big)^{-1}\boldsymbol{H}_{23}\big(\boldsymbol{H}_{13}\big)^{-1}\boldsymbol{H}_{12}\mathbf{1}_3 \tag{2.3}$$

$$\boldsymbol{H}_{32}\boldsymbol{v}_2 = \boldsymbol{H}_{31}\boldsymbol{v}_1^{(2)} \;\Rightarrow\; \boldsymbol{v}_1{}^{(2)} = \big(\boldsymbol{H}_{31}\big)^{-1}\boldsymbol{H}_{32}\mathbf{1}_3 \tag{2.4}$$

So far, we have only considered beamforming at the transmitters. A generalization of the concept of beamforming in the context of the symmetric K-user time-varying MIMO interference channel (cf. Definition 2.5) is *precoding* at the transmitters and *postcoding* at the receivers. The subsequent definition is based on [GCJ11, pp. 3311–3312].

Definition 2.6 (Precoding and postcoding)**.** Let M and N be given as in Definition 2.5. Furthermore, let $d \leq \min(M, N)$ denote the degrees of freedom for each user. Then, $\boldsymbol{V}_j \in \mathbb{C}^{M\times d}$ is referred to as the *precoding matrix* of transmitter T_j, $j \in \{1, 2, \ldots, K\}$, and $\boldsymbol{U}_k \in \mathbb{C}^{N\times d}$ as the *postcoding matrix* of receiver R_k, $k \in \{1, 2, \ldots, K\}$.

Using precoding and postcoding matrices, we can finally formulate the conditions that have to be satisfied for interference alignment on the symmetric K-user time-varying MIMO interference channel [GCJ11, p. 3312]:

$$\forall j \neq k\colon\ \boldsymbol{U}_k^{\mathrm{H}}\boldsymbol{H}_{kj}\boldsymbol{V}_j = \mathbf{0},\ j, k \in \{1, 2, \ldots, K\}, \tag{2.5}$$

$$\operatorname{rank}\big(\boldsymbol{U}_k^{\mathrm{H}}\boldsymbol{H}_{kk}\boldsymbol{V}_k\big) = d,\ k \in \{1, 2, \ldots, K\}. \tag{2.6}$$

Based on [CJ08, p. 3427], Gomadam et al. summarize the main benefit of interference alignment on the K-user interference channel as follows:

> At high [signal-to-noise ratio], every user in a wireless interference network is (simultaneously and almost surely) able to achieve approximately [...] one half of the capacity that he could achieve in the absence of all interference. [GCJ11, p. 3309]

2.6 Existence of a solution

If the system of bilinear equations given by Equation 2.5 is generic in the sense of algebraic geometry, it is possible to decide the existence of a solution to the interference alignment problem based merely on the number of users K, the number of antennas at each transmitter M, the number of antennas at each receiver N, and the degrees of freedom for each user d [Gui12; BCT11; RLL12; GBS14]. González et al. define the following sets for the formulation of their feasibility criteria [GBS14, pp. 1841–1842]:

$$\Phi \coloneqq \{(k,l) \in \{1,2,\ldots,K\} \times \{1,2,\ldots,K\} \mid k \neq l\}, \tag{2.7}$$

$$\Phi_R \coloneqq \{k \in \{1,2,\ldots,K\} \mid \exists l \in \{1,2,\ldots,K\} \colon (k,l) \in \Phi\}, \tag{2.8}$$

$$\Phi_T \coloneqq \{l \in \{1,2,\ldots,K\} \mid \exists k \in \{1,2,\ldots,K\} \colon (k,l) \in \Phi\}. \tag{2.9}$$

For the general (non-symmetric) K-user MIMO interference channel, the subsequent (necessary but not sufficient) conditions have to be satisfied for the interference alignment problem to be feasible [GBS14, pp. 1842–1844]:

$$\forall k \in \Phi_R \; 1 \leq d_k \leq N_k \;\wedge\; \forall l \in \Phi_T \; 1 \leq d_l \leq M_l \tag{2.10}$$

$$\forall (k,l) \in \Phi \; N_k M_l > d_k d_l \tag{2.11}$$

$$\left(\sum_{k \in \Phi_R} N_k d_k - d_k^2 \right) + \left(\sum_{l \in \Phi_T} M_l d_l - d_l^2 \right) - \sum_{(k,l) \in \Phi} d_k d_l \geq 0 \tag{2.12}$$

From the above conditions, we can infer the following corollary for the problem of interference alignment on the symmetric K-user MIMO interference channel (cf. Section 2.5).

Corollary 2.7 (Necessary conditions for interference alignment on the symmetric K-user MIMO interference channel)**.** For the existence of a solution to the interference alignment problem on the symmetric K-user MIMO interference channel, the following necessary but not sufficient conditions have to be satisfied:

(a) $1 \leq d \leq N \;\wedge\; 1 \leq d \leq M$,

(b) $NM > d^2$,

(c) $N + M \geq (K+1)\, d$.

Proof. Since M, N and d by definition are the same for all users on the symmetric K-user MIMO interference channel, conditions (a) and (b) follow directly from Equations 2.10 and 2.11, respectively. For condition (c), we get from Equation 2.12

$$
\begin{aligned}
& |\Phi_R|(Nd - d^2) + |\Phi_T|(Md - d^2) - |\Phi|d^2 \geq 0 \\
\Rightarrow \quad & (K-1)(Nd - d^2) + (K-1)(Md - d^2) - (K-1)^2 d^2 \geq 0 \\
\Rightarrow \quad & Nd - d^2 + Md - d^2 - (K-1)d^2 \geq 0 \\
\Rightarrow \quad & N - d + M - d - (K-1)d \geq 0 \\
\Rightarrow \quad & N + M - (K+1)d \geq 0 \\
\Rightarrow \quad & N + M \geq (K+1)d. \qquad \square
\end{aligned}
$$

Note that conditions (a) and (b) of Corollary 2.7 together require either $M > d$ and $N \geq d$ or $M \geq d$ and $N > d$.

Chapter 3

Fundamentals and methods

This thesis should be accessible to anyone familiar with essential concepts in computer science and scientific computing. Nonetheless, in this chapter, we are going to concisely recapitulate some ideas and methods, notably from numerical linear algebra, which are fundamental for understanding the subsequent chapters.

In Section 3.1, we will characterize *direct* and *iterative methods* for solving a given problem numerically. Afterwards, in Section 3.2, we are going to define and explain *Householder reflections* and *Givens rotations*, two particular unitary operators that will be central to our considerations in Chapters 5 to 8. Finally, in Section 3.3, we are going to outline how to quantify the cost of a computation by counting elementary floating-point operations.

3.1 Direct and iterative methods

In numerical analysis and numerical linear algebra, direct and iterative methods can be distinguished [SY78; JR82; TB97; Tyr97; GV13]:

> The term *direct method* refers to a numerical procedure that can be executed in a finite number of steps. Direct methods are in contrast to *iterative methods*, which generate an infinite sequence of approximations that [...] converge to the solution. [JR82, p. 22]

More specifically, *iterative algorithms*, i.e., numerical algorithms of the iterative method, start with an initial guess and improve the approximate result in iterations, until a convergence criterion is satisfied and the result is considered to be sufficiently accurate. In general, iterative algorithms do not find the exact result, not even in arbitrary-precision arithmetic.

On the other hand, *direct* or *constructive algorithms*, i.e., numerical algorithms of the direct method, "[...] provide the exact answer [...] within a predetermined number of steps." [SY78, p. 180] However, Trefethen et al. underline that "[...] even direct methods

are inexact when carried out on a computer: one hopes for answers accurate to machine precision, no better." [TB97, p. 247]

3.2 Householder and Givens transformations

Both *Householder reflections*—also referred to as *Householder transformations* or *Householder matrices*—and *Givens rotations*—also known as *Givens transformations* or *Givens matrices*—are unitary matrices [TB97; Tyr97; GV13], i.e., matrix representations of unitary transformations that are automorphisms on finite-dimensional Hilbert spaces. For a unitary matrix $\boldsymbol{\mathcal{U}} \in \mathbb{C}^{n\times n}$, it holds that

$$\boldsymbol{\mathcal{U}}^{-1} = \boldsymbol{\mathcal{U}}^{\mathrm{H}}, \tag{3.1}$$

and thus

$$\boldsymbol{\mathcal{U}}^{\mathrm{H}}\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{U}}\boldsymbol{\mathcal{U}}^{\mathrm{H}} = \boldsymbol{I}_n. \tag{3.2}$$

Through multiplying a matrix $\boldsymbol{A}$ by an appropriate Householder or Givens matrix $\boldsymbol{\mathcal{U}}$, it is possible to *annihilate* elements in the resulting matrix $\boldsymbol{B} = \boldsymbol{\mathcal{U}}\boldsymbol{A}$ or $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{\mathcal{U}}$, i.e., to introduce zeros into $\boldsymbol{B} = \boldsymbol{\mathcal{U}}\boldsymbol{A}$ or $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{\mathcal{U}}$.

3.2.1 Householder reflections

Householder matrices have been suggested by Alston S. Householder in [Hou58b], based on a more general unitary matrix used in [Hou58a]. The following definition for the Householder vector $\boldsymbol{\hbar}$ and the Householder matrix $\boldsymbol{\mathcal{H}}$ is inspired by [GV13, pp. 234–243].

Definition 3.1 (Householder vector $\boldsymbol{\hbar}$ and Householder matrix $\boldsymbol{\mathcal{H}}$)**.** Let $\boldsymbol{z} = (z_1\ z_2\ \ldots\ z_m)^{\mathrm{T}} \in \mathbb{C}^m$ be fixed but arbitrary, with $z_1 = re^{i\varphi}$ and $r, \varphi \in \mathbb{R}$. Then, the *Householder vector* $\boldsymbol{\hbar} = \boldsymbol{\hbar}(\boldsymbol{z}) \in \mathbb{C}^m$ is defined as

$$\boldsymbol{\hbar} \coloneqq \boldsymbol{z} \pm \|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1,$$

and the *Householder matrix* $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}(\boldsymbol{z}) \in \mathbb{C}^{m\times m}$ is defined as

$$\boldsymbol{\mathcal{H}} \coloneqq \boldsymbol{I}_m - \beta\boldsymbol{\hbar}\boldsymbol{\hbar}^{\mathrm{H}} \text{ with } \beta \coloneqq \frac{2}{\boldsymbol{\hbar}^{\mathrm{H}}\boldsymbol{\hbar}}.$$

For a fixed but arbitrary vector $\boldsymbol{z} \in \mathbb{C}^m$ and the Householder matrix $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}(\boldsymbol{z}) \in \mathbb{C}^{m\times m}$, it holds that

$$\boldsymbol{\mathcal{H}}\boldsymbol{z} = \mp\|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1 \tag{3.3}$$

[GV13, p. 243] and

$$\boldsymbol{z}^{\mathrm{H}}\boldsymbol{\mathcal{H}} = \mp\|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1^{\mathrm{T}}. \tag{3.4}$$

Hence, all but the first components of the vectors $\mathcal{H}\boldsymbol{z}$ and $\boldsymbol{z}^{\mathrm{H}}\mathcal{H}$ are equal to zero. For improving numerical stability, the sign can be chosen such that $\|\boldsymbol{\hbar}\|_2$ is maximized for the corresponding Householder vector $\boldsymbol{\hbar}$ [GV13, p. 243]. As an example, let

$$\boldsymbol{A} \coloneqq \begin{pmatrix} 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 \\ 14 & 15 & 16 & 17 \end{pmatrix} \tag{3.5}$$

and $\boldsymbol{z} = (z_1 \;\; z_2 \;\; z_3 \;\; z_4)^{\mathrm{T}} = (re^{i\varphi} \;\; z_2 \;\; z_3 \;\; z_4)^{\mathrm{T}} \coloneqq \boldsymbol{A}(:, 1:1)$. Because of $\|\boldsymbol{z}\|_2 = \sqrt{2^2 + 6^2 + 10^2 + 14^2} = \sqrt{336} = 4\sqrt{21}$ and $\varphi = \arg(z_1) = \arg(2) = 0$, and thus $e^{i\varphi} = e^0 = 1$, it follows that

$$\boldsymbol{\hbar} = \boldsymbol{z} \pm \|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1 = \begin{pmatrix} 2 \pm 4\sqrt{21} \\ 6 \\ 10 \\ 14 \end{pmatrix},$$

and therefore

$$\boldsymbol{\hbar} = \boldsymbol{\hbar}_+ = \begin{pmatrix} 2 + 4\sqrt{21} \\ 6 \\ 10 \\ 14 \end{pmatrix} \text{ or } \boldsymbol{\hbar} = \boldsymbol{\hbar}_- = \begin{pmatrix} 2 - 4\sqrt{21} \\ 6 \\ 10 \\ 14 \end{pmatrix}.$$

Since $\|\boldsymbol{\hbar}_+\|_2 = 4\sqrt{42 + \sqrt{21}} > 4\sqrt{42 - \sqrt{21}} = \|\boldsymbol{\hbar}_-\|_2$, we choose $\boldsymbol{\hbar} = \boldsymbol{\hbar}_+$. Consequently, it holds that

$$\beta = \frac{2}{\boldsymbol{\hbar}^{\mathrm{H}}\boldsymbol{\hbar}} = \frac{2}{(2 + 4\sqrt{21})^2 + 6^2 + 10^2 + 14^2} = \frac{2}{16(42 + \sqrt{21})} = \frac{1}{8(42 + \sqrt{21})}$$

and

$$\begin{aligned} \mathcal{H} &= \boldsymbol{I}_4 - \beta\boldsymbol{\hbar}\boldsymbol{\hbar}^{\mathrm{H}} \\ &= \boldsymbol{I}_4 - \beta \begin{pmatrix} (2 + 4\sqrt{21})^2 & 6(2 + 4\sqrt{21}) & 10(2 + 4\sqrt{21}) & 14(2 + 4\sqrt{21}) \\ 6(2 + 4\sqrt{21}) & 36 & 60 & 84 \\ 10(2 + 4\sqrt{21}) & 60 & 100 & 140 \\ 14(2 + 4\sqrt{21}) & 84 & 140 & 196 \end{pmatrix} \\ &= \begin{pmatrix} -\frac{1}{2\sqrt{21}} & -\frac{\sqrt{3}}{2\sqrt{7}} & -\frac{5}{2\sqrt{21}} & -\frac{\sqrt{7}}{2\sqrt{3}} \\ -\frac{\sqrt{3}}{2\sqrt{7}} & 1 - \frac{9}{2(42+\sqrt{21})} & -\frac{15}{2(42+\sqrt{21})} & -\frac{21}{2(42+\sqrt{21})} \\ -\frac{5}{2\sqrt{21}} & -\frac{15}{2(42+\sqrt{21})} & 1 - \frac{25}{2(42+\sqrt{21})} & -\frac{35}{2(42+\sqrt{21})} \\ -\frac{\sqrt{7}}{2\sqrt{3}} & -\frac{21}{2(42+\sqrt{21})} & -\frac{35}{2(42+\sqrt{21})} & 1 - \frac{49}{2(42+\sqrt{21})} \end{pmatrix}. \end{aligned}$$

Hence, we obtain

$$\mathcal{H}\boldsymbol{A} = \begin{pmatrix} -4\sqrt{21} & -\frac{92}{\sqrt{21}} & -\frac{100}{\sqrt{21}} & -\frac{36\sqrt{3}}{\sqrt{7}} \\ 0 & \frac{38}{83} - \frac{34\sqrt{3}}{83\sqrt{7}} & \frac{76}{83} - \frac{68\sqrt{3}}{83\sqrt{7}} & -\frac{6(-133+17\sqrt{21})}{581} \\ 0 & \frac{8}{83} - \frac{170}{83\sqrt{21}} & \frac{4(84-85\sqrt{21})}{1743} & -\frac{2(-84+85\sqrt{21})}{581} \\ 0 & -\frac{2(33+17\sqrt{21})}{249} & -\frac{4(33+17\sqrt{21})}{249} & -\frac{2(33+17\sqrt{21})}{83} \end{pmatrix},$$

and thus, in conformance with Equation 3.3, $\mathcal{H}\boldsymbol{z} = (-4\sqrt{21} \quad 0 \quad 0 \quad 0)^{\mathrm{T}} = -\|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1$. If $\boldsymbol{z} \coloneqq \boldsymbol{A}(1:1,:)^{\mathrm{H}}$, a similar calculation yields $\|\boldsymbol{z}\|_2 = 3\sqrt{6}$ and, in agreement with Equation 3.4, $\boldsymbol{z}^{\mathrm{H}}\mathcal{H} = (\boldsymbol{A}\mathcal{H})(1:1,:) = (-3\sqrt{6} \quad 0 \quad 0 \quad 0) = -\|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1^{\mathrm{T}}$.

In many applications, only parts of a matrix shall be affected by a Householder transformation (cf. Chapter 5). The following definition introduces the *extended Householder matrix* for this purpose.

Definition 3.2 (Extended Householder matrix $\tilde{\mathcal{H}}$)**.** Let $\boldsymbol{z} \in \mathbb{C}^m$ be fixed but arbitrary and $\mathcal{H} = \mathcal{H}(\boldsymbol{z}) \in \mathbb{C}^{m\times m}$ the Householder matrix for $\boldsymbol{z}$ (cf. Definition 3.1). Furthermore, let $m \leq n$, $1 \leq j_1 < j_2 \leq n$, and $j_2 - j_1 + 1 = m$. Then, the *extended Householder matrix* $\tilde{\mathcal{H}} = \tilde{\mathcal{H}}(\boldsymbol{z}, j_1, j_2) \in \mathbb{C}^{n\times n}$ is defined as

$$\tilde{\mathcal{H}} \coloneqq \boldsymbol{I}_n \text{ with } \tilde{\mathcal{H}}(j_1:j_2, j_1:j_2) \coloneqq \mathcal{H}.$$

Since the Householder matrix $\mathcal{H}$ and—trivially—also the identity matrix are unitary matrices, it readily follows from Definition 3.2 that the extended Householder matrix $\tilde{\mathcal{H}}$ is a unitary matrix as well. Let $\boldsymbol{A}$ again be given as in Equation 3.5. If $\boldsymbol{z} \coloneqq \boldsymbol{A}(2:4,1:1)$, it holds that $\|\boldsymbol{z}\|_2 = 2\sqrt{83}$ and, for $\tilde{\mathcal{H}} = \tilde{\mathcal{H}}(\boldsymbol{z}, 2, 4)$, $(\tilde{\mathcal{H}}\boldsymbol{A})(2:4,1:1) = (-2\sqrt{83} \quad 0 \quad 0)^{\mathrm{T}} = -\|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1$. Likewise, if $\boldsymbol{z} \coloneqq \boldsymbol{A}(1:1,2:4)^{\mathrm{H}}$, it follows that $\|\boldsymbol{z}\|_2 = 5\sqrt{2}$ and, for $\tilde{\mathcal{H}} = \tilde{\mathcal{H}}(\boldsymbol{z}, 2, 4)$, $(\boldsymbol{A}\tilde{\mathcal{H}})(1:1,2:4) = (-5\sqrt{2} \quad 0 \quad 0) = -\|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1^{\mathrm{T}}$.

Algorithm 3.1 computes an (extended) Householder matrix $\tilde{\mathcal{H}}$ according to Definition 3.2. Let $\boldsymbol{A} = (a_{ij})$ and $\varphi = \arg(a_{i_1,j_1})$. Then, if $j_1 = j_2$, the algorithm returns $\tilde{\mathcal{H}}$ such that $(\tilde{\mathcal{H}}\boldsymbol{A})(i_1:i_2, j_1:j_1) = \mp\|\boldsymbol{A}(i_1:i_2, j_1:j_1)\|_2 e^{i\varphi}\boldsymbol{e}_1$. Otherwise, i.e., if $i_1 = i_2$, the algorithm returns $\tilde{\mathcal{H}}$ such that $(\boldsymbol{A}\tilde{\mathcal{H}})(i_1:i_1, j_1:j_2) = \mp\|\boldsymbol{A}(i_1:i_1, j_1:j_2)\|_2 e^{i\varphi}\boldsymbol{e}_1^{\mathrm{T}}$. Note that Algorithm 3.1—though correct—can be improved to avoid cancellation errors [GV13, pp. 235–236].

3.2.2 Givens rotations

Golub et al. introduce Givens rotations, which have first been proposed by J. Wallace Givens [Giv54; Giv58], as follows:

> Householder reflections are exceedingly useful for introducing zeros on a grand scale, e.g., the annihilation of all but the first component of a vector. However,

Algorithm 3.1 Algorithm for computing an (extended) Householder matrix.

Require: $\boldsymbol{A} \in \mathbb{C}^{\mu\times\nu}$, $i_1, i_2, j_1, j_2 \in \mathbb{N}$, $(1 \leq i_1 < i_2 \leq \mu \ \wedge \ 1 \leq j_1 = j_2 \leq \nu) \ \vee \ (1 \leq i_1 = i_2 \leq \mu \ \wedge \ 1 \leq j_1 < j_2 \leq \nu)$

function HOUSEHOLDER($\boldsymbol{A}, \langle i_1, j_1\rangle, \langle i_2, j_2\rangle$)
  if $j_1 = j_2$ **then** ▷ Premultiplication by Householder matrix
    $k_1 \leftarrow i_1$
    $k_2 \leftarrow i_2$
    $m \leftarrow i_2 - i_1 + 1$
    $n \leftarrow \mu$
    $\boldsymbol{z} = (z_1 \ z_2 \ \dots \ z_m)^{\mathrm{T}} \leftarrow \boldsymbol{A}(i_1 : i_2, j_1 : j_1)$
  else ▷ Postmultiplication by Householder matrix
    $k_1 \leftarrow j_1$
    $k_2 \leftarrow j_2$
    $m \leftarrow j_2 - j_1 + 1$
    $n \leftarrow \nu$
    $\boldsymbol{z} = (z_1 \ z_2 \ \dots \ z_m)^{\mathrm{T}} \leftarrow \boldsymbol{A}(i_1 : i_1, j_1 : j_2)^{\mathrm{H}}$
  end if

  $\varphi \leftarrow \arg(z_1)$
  $\boldsymbol{\hbar} \leftarrow \boldsymbol{z} \pm \|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1$
  $\beta \leftarrow \frac{2}{\boldsymbol{\hbar}^{\mathrm{H}}\boldsymbol{\hbar}}$
  $\mathcal{H} \leftarrow \boldsymbol{I}_m - \beta\boldsymbol{\hbar}\boldsymbol{\hbar}^{\mathrm{H}}$

  $\tilde{\mathcal{H}} \leftarrow \boldsymbol{I}_n$
  $\tilde{\mathcal{H}}(k_1 : k_2, k_1 : k_2) \leftarrow \mathcal{H}$

  return $\tilde{\mathcal{H}}$
end function

> in calculations where it is necessary to zero elements more selectively, *Givens rotations* are the transformation of choice. [GV13, p. 239]

The application of one Givens matrix $\mathcal{G}$ introduces exactly one zero. As an illustration, consider the vector $\boldsymbol{z} \in \mathbb{C}^2$. An appropriate Givens rotation produces $\mathcal{G}^{\mathrm{H}}\boldsymbol{z} = (r \ 0)^{\mathrm{T}}$ and $\boldsymbol{z}^{\mathrm{H}}\mathcal{G} = (\overline{r} \ 0)$, where r is a nonzero component. This principle can be generalized such that a single element of an arbitrary vector $\boldsymbol{z} \in \mathbb{C}^n$, $n \geq 2$, is annihilated. The following definition, which is motivated by [GV13, pp. 239–244], specifies how such a Givens matrix $\mathcal{G}$ is constructed.

Definition 3.3 (Givens matrix $\mathcal{G}$)**.** Let $z_1, z_2 \in \mathbb{C}$ be fixed but arbitrary, with $z_1 = r_1 e^{i\varphi_1}$, $z_2 = r_2 e^{i\varphi_2}$, and $r_1, r_2, \varphi_1, \varphi_2 \in \mathbb{R}$. Furthermore, let $\theta, \varphi \in \mathbb{R}$, $\varphi = \varphi_1 - \varphi_2$, $c \coloneqq \cos(\theta) = \frac{r_1}{\sqrt{r_1^2+r_2^2}}$, $s \coloneqq \sin(\theta)e^{i\varphi} = -\frac{r_2}{\sqrt{r_1^2+r_2^2}}e^{i\varphi}$, and $1 \leq j_1 \neq j_2 \leq n$. Then, the

Givens matrix $\mathcal{G} = \mathcal{G}(z_1, z_2, j_1, j_2) = \mathcal{G}(\theta, \varphi, j_1, j_2) \in \mathbb{C}^{n\times n}$ is defined as

$$\mathcal{G} \coloneqq \begin{array}{c} \\ \\ j_1 \\ \\ j_2 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & j_1 & & j_2 & & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\overline{s} & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array}.$$

It is apparent from Definition 3.3 that only two rows or two columns of a matrix $\boldsymbol{A}$ are affected when $\boldsymbol{A}$ is multiplied by a Givens matrix $\mathcal{G}$. For a fixed but arbitrary vector $\boldsymbol{z} = (z_1 \ \dots \ z_{j_1} \ \dots \ z_{j_2} \ \dots \ z_n)^{\mathrm{T}} \in \mathbb{C}^n$ and the Givens matrix $\mathcal{G} = \mathcal{G}(z_{j_1}, z_{j_2}, j_1, j_2) \in \mathbb{C}^{n\times n}$, it holds that

$$\mathcal{G}^{\mathrm{H}}\boldsymbol{z} = (z_1 \ \dots \ z_{j_1-1} \ \ r \ \ z_{j_1+1} \ \dots \ z_{j_2-1} \ \ 0 \ \ z_{j_2+1} \ \dots \ z_n)^{\mathrm{T}} \tag{3.6}$$

and

$$\boldsymbol{z}^{\mathrm{H}}\mathcal{G} = (\overline{z_1} \ \dots \ \overline{z_{j_1-1}} \ \ \overline{r} \ \ \overline{z_{j_1+1}} \ \dots \ \overline{z_{j_2-1}} \ \ 0 \ \ \overline{z_{j_2+1}} \ \dots \ \overline{z_n}), \tag{3.7}$$

where r again is a nonzero component. To complete the picture, let us consider an example matrix

$$\boldsymbol{A} = (a_{ij}) \coloneqq \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}.$$

There are four possibilities to apply Givens matrices $\mathcal{G} = \mathcal{G}(z_1, z_2, 1, 2) \in \mathbb{C}^{2\times 2}$ to $\boldsymbol{A}$, namely

$$\mathcal{G}(a_{11}, a_{21}, 1, 2)^{\mathrm{H}}\boldsymbol{A} = \begin{pmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} = \begin{pmatrix} 2\sqrt{5} & \frac{13}{\sqrt{5}} \\ 0 & -\frac{1}{\sqrt{5}} \end{pmatrix},$$

$$\mathcal{G}(a_{12}, a_{22}, 1, 2)^{\mathrm{H}}\boldsymbol{A} = \begin{pmatrix} \frac{3}{\sqrt{34}} & \frac{5}{\sqrt{34}} \\ -\frac{5}{\sqrt{34}} & \frac{3}{\sqrt{34}} \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} = \begin{pmatrix} 13\sqrt{\frac{2}{17}} & \sqrt{34} \\ \sqrt{\frac{2}{17}} & 0 \end{pmatrix},$$

$$\boldsymbol{A}\mathcal{G}(\overline{a_{11}}, \overline{a_{12}}, 1, 2) = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} \begin{pmatrix} \frac{2}{\sqrt{13}} & -\frac{3}{\sqrt{13}} \\ \frac{3}{\sqrt{13}} & \frac{2}{\sqrt{13}} \end{pmatrix} = \begin{pmatrix} \sqrt{13} & 0 \\ \frac{23}{\sqrt{13}} & -\frac{2}{\sqrt{13}} \end{pmatrix},$$

and

$$\boldsymbol{A}\mathcal{G}(\overline{a_{21}}, \overline{a_{22}}, 1, 2) = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} \begin{pmatrix} \frac{4}{\sqrt{41}} & -\frac{5}{\sqrt{41}} \\ \frac{5}{\sqrt{41}} & \frac{4}{\sqrt{41}} \end{pmatrix} = \begin{pmatrix} \frac{23}{\sqrt{41}} & \frac{2}{\sqrt{41}} \\ \sqrt{41} & 0 \end{pmatrix}.$$

As we have seen, these four applications of Givens rotations annihilate three different elements of $\boldsymbol{A}$. Since Definition 3.3 also allows to swap indices, a Givens matrix that

Algorithm 3.2 Algorithm for computing a Givens matrix.

Require: $\boldsymbol{A} = (a_{ij}) \in \mathbb{C}^{\mu\times\nu}$, $i_1, i_2, j_1, j_2 \in \mathbb{N}$, $(1 \leq i_1 \neq i_2 \leq \mu \wedge 1 \leq j_1 = j_2 \leq \nu) \vee (1 \leq i_1 = i_2 \leq \mu \wedge 1 \leq j_1 \neq j_2 \leq \nu)$

function GIVENS($\boldsymbol{A}, \langle i_1, j_1\rangle, \langle i_2, j_2\rangle$)
  if $j_1 = j_2$ **then** ▷ Premultiplication by Givens matrix
    $k_1 \leftarrow i_1$
    $k_2 \leftarrow i_2$
    $n \leftarrow \mu$
    $z_1 \leftarrow a_{i_1,j_1}$
    $z_2 \leftarrow a_{i_2,j_2}$
  else ▷ Postmultiplication by Givens matrix
    $k_1 \leftarrow j_1$
    $k_2 \leftarrow j_2$
    $n \leftarrow \nu$
    $z_1 \leftarrow \overline{a_{i_1,j_1}}$
    $z_2 \leftarrow \overline{a_{i_2,j_2}}$
  end if

  if $z_2 = 0$ **then**
    $c \leftarrow 1$
    $s \leftarrow 0$
  else
    $r_1 \leftarrow |z_1|$
    $r_2 \leftarrow |z_2|$
    $r \leftarrow \sqrt{r_1^2 + r_2^2}$
    $\varphi \leftarrow \arg(z_1) - \arg(z_2)$
    $c \leftarrow \frac{r_1}{r}$
    $s \leftarrow -\frac{r_2}{r} e^{i\varphi}$
  end if

  $\mathcal{G} = (g_{ij}) \leftarrow \boldsymbol{I}_n$
  $g_{k_1,k_1} \leftarrow c$
  $g_{k_1,k_2} \leftarrow s$
  $g_{k_2,k_1} \leftarrow -\overline{s}$
  $g_{k_2,k_2} \leftarrow c$

  if $j_1 = j_2$ **then**
    return $\mathcal{G}^{\mathrm{H}}$
  else
    return $\mathcal{G}$
  end if
end function

zeroes element a_{11} may be applied as well, for example

$$\mathcal{G}(a_{21}, a_{11}, 2, 1)^{\mathrm{H}} \boldsymbol{A} = \begin{pmatrix} \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{\sqrt{5}} \\ 2\sqrt{5} & \frac{13}{\sqrt{5}} \end{pmatrix}.$$

In conformance with Definition 3.3, Algorithm 3.2 computes a Givens matrix $\mathcal{G}$ (if $i_1 = i_2$, i.e., if the matrix $\boldsymbol{A} = (a_{ij})$ is to be postmultiplied by the Givens matrix) or $\mathcal{G}^{\mathrm{H}}$ (if $j_1 = j_2$, i.e., if $\boldsymbol{A}$ is to be premultiplied by the Givens matrix). The application of the computed Givens matrix to $\boldsymbol{A}$ in both cases annihilates element a_{i_2,j_2}. When $\boldsymbol{A}$ is premultiplied by $\mathcal{G}^{\mathrm{H}}$, only the rows $\boldsymbol{A}(i_1 : i_1, :)$ and $\boldsymbol{A}(i_2 : i_2, :)$ are affected. Likewise, when $\boldsymbol{A}$ is postmultiplied by $\mathcal{G}$, solely the columns $\boldsymbol{A}(:, j_1 : j_1)$ and $\boldsymbol{A}(:, j_2 : j_2)$ are altered.

3.3 Computational cost

There are several possibilities to quantify the cost of a computation. In many cases, floating-point operations (*flops*), i.e., additions, subtractions, multiplications, and divisions, are the dominant factor. Therefore, counting these mathematical operations is a common method to evaluate the *computational cost*, and we refer to it also as the *operation count*.

Golub et al. explain that the "[...] number of flops in a given matrix computation is usually obtained by summing the amount of arithmetic associated with the most deeply nested statements." [GV13, p. 12] As an example, let $\boldsymbol{A} \in \mathbb{R}^{m \times r}$, $\boldsymbol{B} \in \mathbb{R}^{r \times n}$, and $\boldsymbol{C} \in \mathbb{R}^{m \times n}$. Then, the operation count of

$$\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{AB}$$

is $2mnr$ [GV13, p. 12] or, in Landau notation, $\mathcal{O}(mnr)$.

In this thesis, we are going to count *complex operations* (cf. Chapter 8). Notwithstanding the fact that one complex addition requires two real additions, but one complex multiplication necessitates four real multiplications and two real additions, we define one complex operation as either a complex addition or a complex multiplication.

Chapter 4

Problem formulation and iterative solution

In Chapter 2, the problem of interference alignment on the K-user interference channel has been briefly introduced from the perspective of information theory. This chapter presents an established computational solution to this problem, a distributed iterative optimization algorithm first proposed by Gomadam et al. in [GCJ11, pp. 3313–3315].

Before the actual algorithm is discussed, the problem to be solved, hereinafter denoted as the *interference alignment problem*, is precisely formulated in Section 4.1. Subsequently, in Section 4.2, we will list and analyze the distributed iterative optimization algorithm by Gomadam et al. We are going to distinguish two flavors of the algorithm, one for the general (nonsymmetric) case of the interference alignment problem in Section 4.2.1, and one derived therefrom for the (to us more relevant) symmetric case in Section 4.2.2. Finally, we will indicate some of the algorithm's characteristics in Section 4.3.

4.1 Problem formulation

Based on the definitions and explanations in Chapter 2, especially in Section 2.5, we now shall appropriately define the interference alignment problem for solving it algorithmically. The parameters K, d, M, and N, as introduced in Chapter 2, are central to the formulation of this problem and will be input parameters for both the iterative and constructive (cf. Chapter 6) algorithms we are going to study. Table 4.1 provides a terse summary of the meanings of these four parameters.

Mathematically, systems of bilinear equations (cf. Equation 2.5) have to be satisfied in order to find a solution to the interference alignment problem. The precoding and postcoding matrices (cf. Definition 2.6) appearing in these bilinear systems are *truncated unitary matrices* [Gui12, p. 1]. The concept of truncated unitary matrices particularly proves to be expedient for reformulating the interference alignment problem as the *interference alignment decomposition problem* in view of finding a direct problem solution (cf.

Table 4.1: The relevant parameters for the interference alignment problem.

	Description	Further details
K	K users, i.e., K transmitters and K receivers, communicate over the interference channel.	Sections 2.1 and 2.2
d	The interference channel has d degrees of freedom.	Section 2.2.3
M	Each transmitter has M antennas.	Section 2.2.2
N	Each receiver has N antennas.	Section 2.2.2

Chapter 5).

Definition 4.1 (Truncated unitary matrix)**.** Let $\boldsymbol{A}$ be a unitary matrix of size $m \times m$. Then, a matrix $\boldsymbol{B}$ consisting of the first $n < m$ columns of $\boldsymbol{A}$ is referred to as a *truncated unitary matrix* of size $m \times n$.

As noted in Chapter 2, a valid solution to the interference alignment problem has to satisfy, besides the bilinear systems in Equation 2.5, the condition in Equation 2.6. However, since, according to Gomadam et al., the condition in Equation 2.6 is automatically fulfilled [GCJ11, p. 3313], the relevant condition to satisfy is Equation 2.5. For this reason, and in agreement with [Gui12, p. 1], we can formulate the interference alignment problem based on the condition in Equation 2.5 along with fundamental definitions from Chapter 2 and Definition 4.1.

Definition 4.2 (Interference alignment problem)**.** Let $K \in \mathbb{N}$ users, $d \in \mathbb{N}$ degrees of freedom, $M \in \mathbb{N}$ antennas at each transmitter, $N \in \mathbb{N}$ antennas at each receiver (cf. Table 4.1), and K^2 channel matrices $\boldsymbol{H}_{kj} \in \mathbb{C}^{N \times M}$, $j, k \in \{1, 2, \ldots, K\}$ (cf. Definition 2.1), be given such that $K \geq 2$, $1 \leq d \leq N$, $1 \leq d \leq M$, $NM > d^2$, and $N + M \geq (K + 1)\, d$ (cf. Corollary 2.7). Then, the *interference alignment problem* is defined as the problem to find K truncated unitary matrices $\boldsymbol{U}_k \in \mathbb{C}^{N \times d}$, $k \in \{1, 2, \ldots, K\}$, i.e., K postcoding matrices, and K truncated unitary matrices $\boldsymbol{V}_j \in \mathbb{C}^{M \times d}$, $j \in \{1, 2, \ldots, K\}$, i.e., K precoding matrices (cf. Definition 2.6), such that

$$\forall j \neq k\colon\ \boldsymbol{U}_k^{\mathrm{H}} \boldsymbol{H}_{kj} \boldsymbol{V}_j = \boldsymbol{0}$$

(cf. Equation 2.5) is satisfied.

4.2 Algorithm description

The distributed iterative optimization algorithm, as described in [GCJ11, pp. 3313–3314], considers the general (nonsymmetric) K-user time-varying MIMO interference channel

[GCJ11, p. 3311] (cf. Section 2.2.2), where transmitter T_j and receiver R_k are equipped with M_j and N_k antennas and have d_j and d_k degrees of freedom, respectively.

Since we want to focus on the symmetric case, where each transmitter (receiver) is equipped with M (N) antennas and has d degrees of freedom, we will not only discuss the original algorithm by Gomadam et al. (cf. Section 4.2.1) but also its adaption to the symmetric K-user time-varying MIMO interference channel (cf. Section 4.2.2). In Section 4.3, we will list some of the algorithm's characteristics and briefly explain why the algorithm can be considered a distributed algorithm.

4.2.1 General variant

To be consistent with the condition in Definition 4.2, it is required that at each receiver all interference is suppressed. For being able to achieve that suppression, the algorithm follows the subsequent procedure:

- Initially, truncated unitary transmit filters (precoding matrices) $\boldsymbol{V}_j$, $j \in \{1, 2, \ldots, K\}$, as well as arbitrary receive filters (postcoding matrices) $\boldsymbol{U}_k$, $k \in \{1, 2, \ldots, K\}$, are chosen.
- Then, these filters are iteratively updated such that the *leakage interference* at the receivers, i.e., the power remaining in the received signals after the filters are applied, is minimized.

The total leakage interference I_k at receiver R_k due to interference from transmitters T_j, $j \neq k$, is given by

$$I_k = \operatorname{tr}(\boldsymbol{U}_k^{\mathrm{H}} \boldsymbol{Q}_k \boldsymbol{U}_k), \tag{4.1}$$

where

$$\boldsymbol{Q}_k = \sum_{\substack{j=1 \\ j \neq k}}^{K} \frac{P_j}{d_j} \boldsymbol{H}_{kj} \boldsymbol{V}_j \boldsymbol{V}_j^{\mathrm{H}} \boldsymbol{H}_{kj}^{\mathrm{H}} \tag{4.2}$$

is the interference covariance matrix at receiver R_k [GCJ11, p. 3313]. For the symmetric case, $\frac{P_j}{d_j}$ is substituted by $\frac{P_j}{d}$ in Equation 4.2. If I_k converges to zero at each receiver R_k, $k \in \{1, 2, \ldots, K\}$, interference alignment is feasible [GCJ11, p. 3313] (cf. Section 2.6).

For optimization, the algorithm considers both the *original network* (interference channel) with transmitters T_j, $j \in \{1, 2, \ldots, K\}$, and receivers R_k, $k \in \{1, 2, \ldots, K\}$, as well as a *reciprocal network* with transmitters $\overleftarrow{T}_j = R_j$, $j \in \{1, 2, \ldots, K\}$, and receivers $\overleftarrow{R}_k = T_k$, $k \in \{1, 2, \ldots, K\}$. Hence, in the reciprocal network, the actual transmitters act as receivers and the actual receivers act as transmitters.

To minimize the total leakage interference, only the receivers R_k and $\overleftarrow{R}_k$ in both the original and reciprocal network update their interference suppression filters $\boldsymbol{U}_k$ and $\overleftarrow{\boldsymbol{U}}_k$, respectively. Since the algorithm alternates between the original and the reciprocal

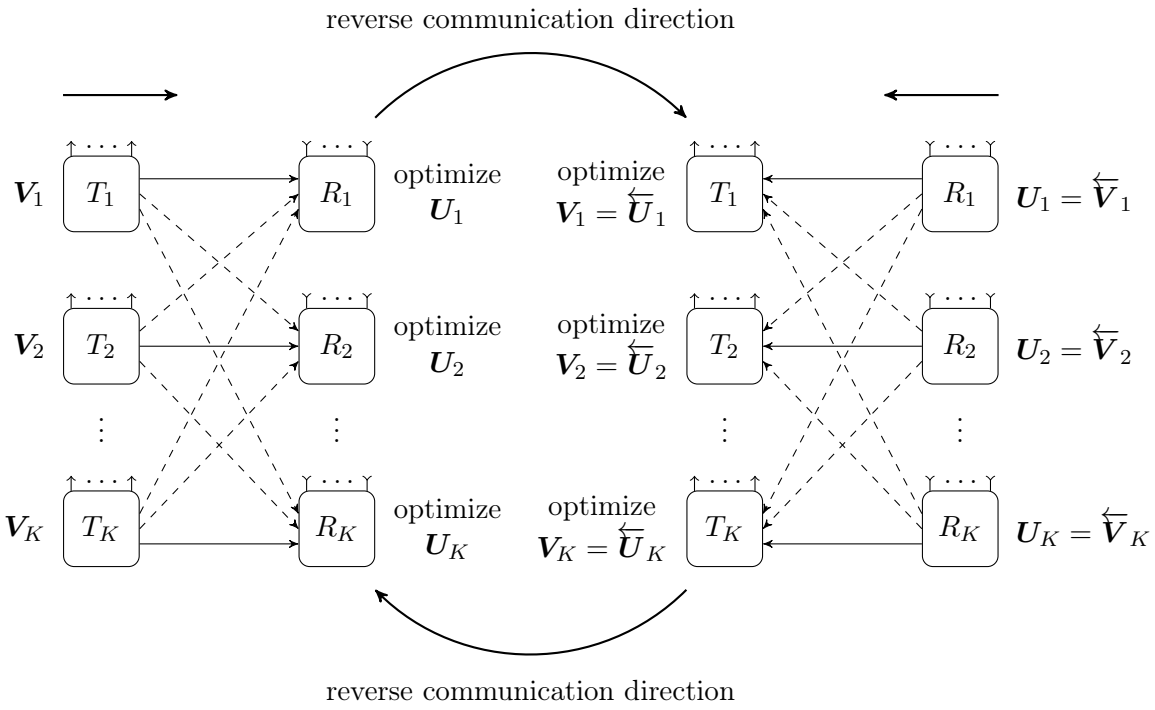


Figure 4.1: Distributed algorithm for iterative interference alignment. This figure is based on [GCJ11, fig. 2].

network, i.e., the algorithm constantly reverses the communication direction, not only the receive filters but also the transmit filters are optimized. Figure 4.1 visualizes the relation between the original and the reciprocal network and shows which filters are updated in the particular network.

Algorithm 4.1 concretizes the aforegoing description of the distributed iterative optimization algorithm. In Lines 1 to 3, the transmit filters $\boldsymbol{V}_j$, $j \in \{1, 2, \ldots, K\}$, are initialized. Considering that the receive filters $\boldsymbol{U}_k$, $k \in \{1, 2, \ldots, K\}$, are not used before values are assigned to them in Line 9, they do not need to be initialized. After the initialization, the transmit and receive filters are iteratively optimized (cf. Lines 5 to 21).

In a first step (cf. Lines 6 to 12), the optimization in the original network is performed, i.e., the minimization problem

$$\forall k \in \{1, 2, \ldots, K\}: \quad \min_{\boldsymbol{U}_k \in \mathbb{C}^{N_k \times d_k}, \boldsymbol{U}_k^{\mathrm{H}} \boldsymbol{U}_k = \boldsymbol{I}_{d_k}} I_k \tag{4.3}$$

is solved. Put differently, receiver R_k "chooses its interference suppression filter $\boldsymbol{U}_k$ to minimize the leakage interference due to all undesired transmitters. The d_k-dimensional received signal subspace that contains the least interference is the space spanned by the eigenvectors corresponding to the d_k smallest eigenvalues of the interference covariance matrix $\boldsymbol{Q}_k$." [GCJ11, p. 3313]

Algorithm 4.1 Distributed iterative optimization algorithm for solving the interference alignment problem (general variant). This algorithm is based on [GCJ11, alg. 1].

1: **for all** $j \in \{1, 2 \ldots, K\}$ **do** ▷ Initialization
2: let $\boldsymbol{V}_j \in \mathbb{C}^{M_j \times d_j}$ be arbitrary such that $\boldsymbol{V}_j^{\mathrm{H}} \boldsymbol{V}_j = \boldsymbol{I}_{d_j}$
3: **end for**
4:
5: **repeat**
6: **for all** $k \in \{1, 2, \ldots, K\}$ **do** ▷ Optimization original network
7: $\boldsymbol{Q}_k \leftarrow \sum_{j=1, j \neq k}^{K} \frac{P_j}{d_j} \boldsymbol{H}_{kj} \boldsymbol{V}_j \boldsymbol{V}_j^{\mathrm{H}} \boldsymbol{H}_{kj}^{\mathrm{H}}$
8: **for all** $i \in \{1, 2, \ldots, d_k\}$ **do**
9: $\boldsymbol{U}_k(:, i\!:\!i) \leftarrow \textsc{Eigenvector}(\boldsymbol{Q}_k, i)$
10: **end for**
11: $\overleftarrow{\boldsymbol{V}}_k \leftarrow \boldsymbol{U}_k$
12: **end for**
13:
14: **for all** $j \in \{1, 2, \ldots, K\}$ **do** ▷ Optimization reciprocal network
15: $\overleftarrow{\boldsymbol{Q}}_j \leftarrow \sum_{k=1, k \neq j}^{K} \frac{\overleftarrow{P}_k}{d_k} \overleftarrow{\boldsymbol{H}}_{jk} \overleftarrow{\boldsymbol{V}}_k \overleftarrow{\boldsymbol{V}}_k^{\mathrm{H}} \overleftarrow{\boldsymbol{H}}_{jk}^{\mathrm{H}}$
16: **for all** $i \in \{1, 2, \ldots, d_j\}$ **do**
17: $\overleftarrow{\boldsymbol{U}}_j(:, i\!:\!i) \leftarrow \textsc{Eigenvector}(\overleftarrow{\boldsymbol{Q}}_j, i)$
18: **end for**
19: $\boldsymbol{V}_j \leftarrow \overleftarrow{\boldsymbol{U}}_j$
20: **end for**
21: **until** convergence

Therefore, the d_k columns of $\boldsymbol{U}_k$, $k \in \{1, 2, \ldots, K\}$, can be obtained by

$$\forall k \in \{1, 2, \ldots, K\}\colon\ \forall i \in \{1, 2, \ldots, d_k\}\colon\ \boldsymbol{U}_k(:, i\!:\!i) = \textsc{Eigenvector}(\boldsymbol{Q}_k, i), \tag{4.4}$$

where $\textsc{Eigenvector}(\boldsymbol{A}, i)$ denotes the eigenvector corresponding to the i^{th} smallest eigenvalue of a matrix $\boldsymbol{A}$. After the computation of $\boldsymbol{Q}_k$ in Line 7 of Algorithm 4.1, the just characterized method is used to get the columns of $\boldsymbol{U}_k$ in Lines 8 to 10. In the final statement of the first step, the updated receive filter $\boldsymbol{U}_k$ is assigned to $\overleftarrow{\boldsymbol{V}}_k$ as input for the second step (cf. Line 11).

The second step of the optimization iteration (cf. Lines 14 to 20) follows the same procedure for the reciprocal as the first step for the original network. First, $\overleftarrow{\boldsymbol{Q}}_j$ is computed (cf. Line 15), then the columns of $\overleftarrow{\boldsymbol{U}}_j$ are obtained (cf. Lines 16 to 18), and finally the updated $\overleftarrow{\boldsymbol{U}}_j$ is assigned to $\boldsymbol{V}_j$ (cf. Line 19). The iteration ends when a defined convergence criterion, e.g., a certain leakage interference precision, is fulfilled, or a maximum number of iterations is reached.

4.2.2 Symmetric variant

As previously stated, our concern in the context of this thesis is interference alignment on the symmetric K-user (time-varying) MIMO interference channel rather than the general

Algorithm 4.2 Distributed iterative optimization algorithm for solving the interference alignment problem (symmetric variant). This algorithm is based on Algorithm 4.1.

for all $j \in \{1, 2, \ldots, K\}$ **do** ▷ Initialization
  let $\boldsymbol{V}_j \in \mathbb{C}^{M \times d}$ be arbitrary such that $\boldsymbol{V}_j^{\mathrm{H}} \boldsymbol{V}_j = \boldsymbol{I}_d$
end for

repeat
  for all $k \in \{1, 2, \ldots, K\}$ **do** ▷ Optimization original network
    $\boldsymbol{Q}_k \leftarrow \sum_{j=1, j \neq k}^{K} \frac{P_j}{d} \boldsymbol{H}_{kj} \boldsymbol{V}_j \boldsymbol{V}_j^{\mathrm{H}} \boldsymbol{H}_{kj}^{\mathrm{H}}$
    for all $i \in \{1, 2, \ldots, d\}$ **do**
      $\boldsymbol{U}_k(:, i\!:\!i) \leftarrow \textsc{Eigenvector}(\boldsymbol{Q}_k, i)$
    end for
    $\overleftarrow{\boldsymbol{V}}_k \leftarrow \boldsymbol{U}_k$
  end for

  for all $j \in \{1, 2 \ldots, K\}$ **do** ▷ Optimization reciprocal network
    $\overleftarrow{\boldsymbol{Q}}_j \leftarrow \sum_{k=1, k \neq j}^{K} \frac{\overleftarrow{P}_k}{d} \overleftarrow{\boldsymbol{H}}_{jk} \overleftarrow{\boldsymbol{V}}_k \overleftarrow{\boldsymbol{V}}_k^{\mathrm{H}} \overleftarrow{\boldsymbol{H}}_{jk}^{\mathrm{H}}$
    for all $i \in \{1, 2, \ldots, d\}$ **do**
      $\overleftarrow{\boldsymbol{U}}_j(:, i\!:\!i) \leftarrow \textsc{Eigenvector}(\overleftarrow{\boldsymbol{Q}}_j, i)$
    end for
    $\boldsymbol{V}_j \leftarrow \overleftarrow{\boldsymbol{U}}_j$
  end for
until convergence

(nonsymmetric) K-user MIMO interference channel. In Chapter 8, we will compare an implementation of our constructive algorithm to one of the iterative algorithm for the symmetric case as described in this section.

The adaption of Algorithm 4.1 to the symmetric K-user MIMO interference channel requires only minor modifications, as we can see in Algorithm 4.2. In essence, the description in Section 4.2.1 is also valid for the symmetric case. Since now all transmitters T_j, $j \in \{1, 2, \ldots, K\}$, and receivers R_k, $k \in \{1, 2, \ldots, K\}$, have M (instead of M_j) and N (instead of N_k) antennas, respectively, and d (instead of d_j and d_k) degrees of freedom, the dimensions of the matrices $\boldsymbol{H}_{kj} = \overleftarrow{\boldsymbol{H}}_{jk}^{\mathrm{H}} \in \mathbb{C}^{N \times M}$, $\boldsymbol{U}_k = \overleftarrow{\boldsymbol{V}}_k \in \mathbb{C}^{N \times d}$, $\boldsymbol{V}_j = \overleftarrow{\boldsymbol{U}}_j \in \mathbb{C}^{M \times d}$, $\boldsymbol{Q}_k \in \mathbb{C}^{N \times N}$, and $\overleftarrow{\boldsymbol{Q}}_j \in \mathbb{C}^{M \times M}$ are the same for all users. Apart from that, the algorithm works the same as before.

4.3 Algorithm characteristics

To complete our discussion of the distributed iterative optimization algorithm by Gomadam et al., we want to take a brief look at some of its properties. One aspect we have excluded so far is to clarify in what way the algorithm can be considered *distributed*:

> The reciprocal property of the wireless channels, combined with the fact that

> the interference covariance matrices can be naturally learnt at the receivers, enables a distributed implementation of the above algorithm. [GCJ11, p. 3315]

Accordingly, although the interference covariance matrices $\boldsymbol{Q}_k$ and $\overleftarrow{\boldsymbol{Q}}_j$, $j,k \in \{1,2,\ldots,K\}$, depend on the channel and precoding matrices of all users (cf. Equation 4.2 and Lines 7 and 15 of Algorithms 4.1 and 4.2), they can be estimated in a distributed manner [GCJ11, p. 3315].

Another important property is the algorithm's convergence characteristics. Gomadam et al. show that the algorithm reduces the leakage interference at every iteration and, for this reason, is guaranteed to converge [GCJ11, pp. 3314–3315], but due to the nonconvex nature of the interference optimization problem not necessarily to a global minimum [GCJ11, p. 3315]. Consequently, solutions computed by the algorithm may not be optimal.

Chapter 5

Problem reformulation and direct solution approaches

After we have studied an iterative solution to the interference alignment problem in Chapter 4, we want to approach a direct solution in this chapter. In contrast to the iterative solution, a direct solution guarantees to find a global optimum. For this purpose, we will analyze global matrix factorization schemes based on Householder and Givens transformations (cf. Chapter 3) and illustrate how they can be of avail for developing a direct solution to the problem at hand.

In order to obtain a global matrix formulation appropriate for applying Householder reflections and Givens rotations, we will reformulate the interference alignment problem as the equivalent *interference alignment decomposition problem* in Section 5.1. Subsequently, in Section 5.2, we will define a related problem that removes some constraints from the interference alignment decomposition problem. This simplified problem will help us to understand the matrix factorization techniques introduced in Section 5.3.

5.1 Problem reformulation

The original formulation of the interference alignment problem (cf. Definition 4.2) has been well-suited for finding an iterative solution like the distributed iterative optimization algorithm we have discussed in Chapter 4. For developing a direct solution on the basis of Householder and Givens transformations, we need to adapt this formulation such that we obtain a single system of bilinear equations to be solved by applying appropriate matrix factorization schemes. In this context, the meanwhile well-known parameters K, d, M, and N (cf. Table 4.1) will again be fundamental to our considerations.

For a proper reformulation of the interference alignment problem, the following definition introduces the *global channel matrix* $\boldsymbol{H}$ as the concatenation of all channel matrices $\boldsymbol{H}_{kj}$, $j,k \in \{1,2,\ldots,K\}$, as well as the *global postcoding matrix* $\boldsymbol{U}$ and the *global precoding matrix* $\boldsymbol{V}$ as concatenations of all postcoding matrices $\boldsymbol{U}_k$, $k \in \{1,2,\ldots,K\}$, and all

precoding matrices $\boldsymbol{V}_j$, $j \in \{1, 2, \ldots, K\}$, respectively [Gui12, p. 1].

Definition 5.1 (Global channel matrix $\boldsymbol{H}$, global postcoding matrix $\boldsymbol{U}$, and global precoding matrix $\boldsymbol{V}$)**.** Let $\boldsymbol{H}_{kj} \in \mathbb{C}^{N \times M}$, be the channel matrix from transmitter T_j, $j \in \{1, 2, \ldots, K\}$, to receiver R_k, $k \in \{1, 2, \ldots, K\}$ (cf. Definition 2.1). Furthermore, let $\boldsymbol{U}_k \in \mathbb{C}^{N \times d}$ be the postcoding matrix at receiver R_k, $k \in \{1, 2, \ldots, K\}$, and let $\boldsymbol{V}_j \in \mathbb{C}^{M \times d}$ be the precoding matrix at transmitter T_j, $j \in \{1, 2, \ldots, K\}$ (cf. Definition 2.6). Then, the *global channel matrix* $\boldsymbol{H} \in \mathbb{C}^{KN \times KM}$, the *global postcoding matrix* $\boldsymbol{U} \in \mathbb{C}^{KN \times Kd}$, and the *global precoding matrix* $\boldsymbol{V} \in \mathbb{C}^{KM \times Kd}$ are defined as

$$\boldsymbol{H} \coloneqq \begin{pmatrix} \boldsymbol{H}_{11} & \boldsymbol{H}_{12} & \cdots & \boldsymbol{H}_{1K} \\ \boldsymbol{H}_{21} & \boldsymbol{H}_{22} & \cdots & \boldsymbol{H}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{H}_{K1} & \boldsymbol{H}_{K2} & \cdots & \boldsymbol{H}_{KK} \end{pmatrix},$$

$$\boldsymbol{U} \coloneqq \begin{pmatrix} \boldsymbol{U}_1 & & & \boldsymbol{0} \\ & \boldsymbol{U}_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \boldsymbol{U}_K \end{pmatrix}, \text{ and } \boldsymbol{V} \coloneqq \begin{pmatrix} \boldsymbol{V}_1 & & & \boldsymbol{0} \\ & \boldsymbol{V}_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \boldsymbol{V}_K \end{pmatrix}.$$

Based on the global channel matrix $\boldsymbol{H}$, the global postcoding matrix $\boldsymbol{U}$, and the global precoding matrix $\boldsymbol{V}$, we now define the *global interference alignment matrix* $\boldsymbol{\Sigma}$, which will be central to the reformulation of the interference alignment problem.

Definition 5.2 (Global interference alignment matrix $\boldsymbol{\Sigma}$)**.** Let $\boldsymbol{H} \in \mathbb{C}^{KN \times KM}$ be the global channel matrix, $\boldsymbol{U} \in \mathbb{C}^{KN \times Kd}$ the global postcoding matrix, and $\boldsymbol{V} \in \mathbb{C}^{KM \times Kd}$ the global precoding matrix (cf. Definition 5.1). Then, the *global interference alignment matrix* $\boldsymbol{\Sigma} \in \mathbb{C}^{Kd \times Kd}$ is defined as

$$\boldsymbol{\Sigma} \coloneqq \boldsymbol{U}^{\mathrm{H}} \boldsymbol{H} \boldsymbol{V}.$$

Like the postcoding matrices $\boldsymbol{U}_k$, $k \in \{1, 2, \ldots, K\}$, and the precoding matrices $\boldsymbol{V}_j$, $j \in \{1, 2, \ldots, K\}$, the global postcoding matrix $\boldsymbol{U}$ and the global precoding matrix $\boldsymbol{V}$ are truncated unitary matrices [Gui12, p. 1] (cf. Definition 4.1). By concatenation of any orthonormal bases of their null column spaces, $\boldsymbol{U}$ and $\boldsymbol{V}$ can be extended to (square) unitary matrices $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ [Gui12, p. 2], which have the important property of being invertible.

Definition 5.3 (Extended global postcoding matrix $\bar{\boldsymbol{U}}$ and extended global precoding matrix $\bar{\boldsymbol{V}}$)**.** Let $\boldsymbol{U} \in \mathbb{C}^{KN \times Kd}$ be the global postcoding matrix and $\boldsymbol{V} \in \mathbb{C}^{KM \times Kd}$ the global precoding matrix (cf. Definition 5.1). Moreover, let $\tilde{\boldsymbol{U}} \in \mathbb{C}^{KN \times K(N-d)}$ denote any orthonormal basis of the null column space of $\boldsymbol{U}$ and $\tilde{\boldsymbol{V}} \in \mathbb{C}^{KM \times K(M-d)}$ any

orthonormal basis of the null column space of $\boldsymbol{V}$. Then, the *extended global postcoding matrix* $\bar{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$ and the *extended global precoding matrix* $\bar{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ are defined as

$$\bar{\boldsymbol{U}} \coloneqq \begin{pmatrix} \boldsymbol{U} & \tilde{\boldsymbol{U}} \end{pmatrix} \text{ and}$$
$$\bar{\boldsymbol{V}} \coloneqq \begin{pmatrix} \boldsymbol{V} & \tilde{\boldsymbol{V}} \end{pmatrix}.$$

Similar to the definition of the global interference alignment matrix $\boldsymbol{\Sigma}$ based on the global postcoding matrix $\boldsymbol{U}$ and the global precoding matrix $\boldsymbol{V}$ (cf. Definition 5.2), we define the *extended global interference alignment matrix* $\bar{\boldsymbol{\Sigma}}$ based on the extended global postcoding matrix $\bar{\boldsymbol{U}}$ and the extended global precoding matrix $\bar{\boldsymbol{V}}$.

Definition 5.4 (Extended global interference alignment matrix $\bar{\boldsymbol{\Sigma}}$)**.** Let $\boldsymbol{H} \in \mathbb{C}^{KN \times KM}$ be the global channel matrix (cf. Definition 5.1), $\bar{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$ the extended global postcoding matrix, and $\bar{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ the extended global precoding matrix (cf. Definition 5.3). Then, the *extended global interference alignment matrix* $\bar{\boldsymbol{\Sigma}} \in \mathbb{C}^{KN \times KM}$ is defined as

$$\bar{\boldsymbol{\Sigma}} \coloneqq \bar{\boldsymbol{U}}^{\mathrm{H}} \boldsymbol{H} \bar{\boldsymbol{V}}.$$

Due to the close relations of $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ to $\boldsymbol{U}$ and $\boldsymbol{V}$ (cf. Definition 5.3), there is a close relationship between $\bar{\boldsymbol{\Sigma}}$ and $\boldsymbol{\Sigma}$ as well. The comparison of Definitions 5.2 and 5.4 readily unveils

$$\bar{\boldsymbol{\Sigma}} = \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{A} \\ \boldsymbol{B} & \boldsymbol{C} \end{pmatrix}, \tag{5.1}$$

where $\boldsymbol{A} = \boldsymbol{U}^{\mathrm{H}} \boldsymbol{H} \tilde{\boldsymbol{V}} \in \mathbb{C}^{Kd \times K(M-d)}$, $\boldsymbol{B} = \tilde{\boldsymbol{U}}^{\mathrm{H}} \boldsymbol{H} \boldsymbol{V} \in \mathbb{C}^{K(N-d) \times Kd}$, and $\boldsymbol{C} = \tilde{\boldsymbol{U}}^{\mathrm{H}} \boldsymbol{H} \tilde{\boldsymbol{V}} \in \mathbb{C}^{K(N-d) \times K(M-d)}$.

In Definitions 5.1 to 5.4, we have introduced all global matrices we need for reformulating the interference alignment problem. Since $\boldsymbol{U}$, $\boldsymbol{H}$, and $\boldsymbol{V}$ are concatenations of $\boldsymbol{U}_k$, $\boldsymbol{H}_{kj}$, and $\boldsymbol{V}_j$, respectively (cf. Definition 5.1), $j, k \in \{1, 2, \ldots, K\}$, and because of the condition

$$\forall j \neq k\colon\ \boldsymbol{U}_k^{\mathrm{H}} \boldsymbol{H}_{kj} \boldsymbol{V}_j = \boldsymbol{0} \tag{5.2}$$

(cf. Definition 4.2), it follows that $\boldsymbol{\Sigma} = \boldsymbol{U}^{\mathrm{H}} \boldsymbol{H} \boldsymbol{V} \in \mathbb{C}^{Kd \times Kd}$ (cf. Definition 5.2) has to be a block-diagonal matrix, i.e., a matrix with all but the K diagonal blocks of size $d \times d$ being equal to $\boldsymbol{0}$ [Gui12, p. 1] (note that $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ in Equation 5.1 are matrices of no particular structure [Gui12, p. 2]). This insight eventually enables us to reformulate the interference alignment problem (cf. Definition 4.2) as the equivalent *interference alignment decomposition problem*.

Definition 5.5 (Interference alignment decomposition problem)**.** Let $K \in \mathbb{N}$ users, $d \in \mathbb{N}$ degrees of freedom, $M \in \mathbb{N}$ antennas at each transmitter, $N \in \mathbb{N}$ antennas at each receiver (cf. Table 4.1), and a global channel matrix $\boldsymbol{H} \in \mathbb{C}^{KN \times KM}$ (cf. Definition 5.1) be given such that $K \geq 2$, $1 \leq d \leq N$, $1 \leq d \leq M$, $NM > d^2$, and $N + M \geq (K+1)\,d$ (cf. Corollary 2.7). Then, the *interference alignment decomposition problem* is defined as the problem to find an extended global postcoding matrix $\bar{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$, an extended global precoding matrix $\bar{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ (cf. Definition 5.3), and an extended global interference alignment matrix $\bar{\boldsymbol{\Sigma}} \in \mathbb{C}^{KN \times KM}$ (cf. Definition 5.4) such that

$$\bar{\boldsymbol{U}}^{\mathrm{H}} \boldsymbol{H} \bar{\boldsymbol{V}} = \bar{\boldsymbol{\Sigma}}$$

and

$$\boldsymbol{\Sigma} = \bar{\boldsymbol{\Sigma}}(1\!:\!Kd, 1\!:\!Kd) = \begin{pmatrix} \boldsymbol{\Sigma}_1 & & & \boldsymbol{0} \\ & \boldsymbol{\Sigma}_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \boldsymbol{\Sigma}_K \end{pmatrix},$$

where $\boldsymbol{\Sigma}_k \in \mathbb{C}^{d \times d}$, $k \in \{1, 2, \ldots, K\}$.

Since the extended global postcoding and precoding matrices $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ are unitary and thus invertible matrices, Definition 5.5 implies that a given global channel matrix $\boldsymbol{H}$ can be factorized as

$$\boldsymbol{H} = \bar{\boldsymbol{U}} \bar{\boldsymbol{\Sigma}} \bar{\boldsymbol{V}}^{\mathrm{H}} \tag{5.3}$$

for finding a direct solution to the interference alignment decomposition problem.

5.2 Problem relaxation

According to Definition 5.5, we have to find both $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ as well as $\bar{\boldsymbol{\Sigma}}$ in order to solve the interference alignment decomposition problem. While $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ have to be sparse block matrices (cf. Definitions 5.1 and 5.3), the matrix $\boldsymbol{\Sigma} = \bar{\boldsymbol{\Sigma}}(1\!:\!Kd, 1\!:\!Kd)$ must be of block-diagonal form (cf. Definition 5.5). However, in Section 5.3 and Chapter 6, we will be content with constructing the block-diagonal submatrix $\boldsymbol{\Sigma}$ of the matrix $\bar{\boldsymbol{\Sigma}}$.

For this reason, we are going to define the *relaxed interference alignment decomposition problem*, which is a first step towards a direct solution to the interference alignment decomposition problem. Although solutions to this simplified problem cannot be considered solutions to the problem described in Chapter 2, a direct solution to the relaxed interference alignment decomposition problem offers valuable insights for a direct solution to the interference alignment decomposition problem. Therefore, we will, within the context of the relaxed interference alignment decomposition problem, still talk about K users, d degrees of freedom, M antennas at each transmitter, N antennas at each receiver, the

global channel matrix $\boldsymbol{H}$, and even the global interference alignment matrix $\boldsymbol{\Sigma}$ and the extended global interference alignment matrix $\bar{\boldsymbol{\Sigma}}$, but always keep in mind that these are just names for mathematical objects.

Definition 5.6 (Relaxed interference alignment decomposition problem)**.** Let $K \in \mathbb{N}$ users, $d \in \mathbb{N}$ degrees of freedom, $M \in \mathbb{N}$ antennas at each transmitter, $N \in \mathbb{N}$ antennas at each receiver, and a global channel matrix $\boldsymbol{H} \in \mathbb{C}^{KN \times KM}$ be given such that $K \geq 2$, $1 \leq d \leq N$, $1 \leq d \leq M$, $NM > d^2$, and $N + M \geq (K+1)\,d$. Then, the *relaxed interference alignment decomposition problem* is defined as the problem to find an extended global interference alignment matrix $\bar{\boldsymbol{\Sigma}} \in \mathbb{C}^{KN \times KM}$ and unitary matrices $\hat{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$ and $\hat{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ such that

$$\hat{\boldsymbol{U}}^{\mathrm{H}} \boldsymbol{H} \hat{\boldsymbol{V}} = \bar{\boldsymbol{\Sigma}}$$

and

$$\boldsymbol{\Sigma} = \bar{\boldsymbol{\Sigma}}(1\!:\!Kd, 1\!:\!Kd) = \begin{pmatrix} \boldsymbol{\Sigma}_1 & & & \boldsymbol{0} \\ & \boldsymbol{\Sigma}_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \boldsymbol{\Sigma}_K \end{pmatrix},$$

where $\boldsymbol{\Sigma}_k \in \mathbb{C}^{d \times d}$, $k \in \{1, 2, \ldots, K\}$.

Because the newly introduced matrices $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ by definition are unitary matrices, a given global channel matrix $\boldsymbol{H}$, analogous to Equation 5.3, can be factorized as

$$\boldsymbol{H} = \hat{\boldsymbol{U}} \bar{\boldsymbol{\Sigma}} \hat{\boldsymbol{V}}^{\mathrm{H}} \tag{5.4}$$

for finding a direct solution to the relaxed interference alignment decomposition problem.

5.3 Direct solution approaches

As we already know from Chapter 1, the objective of this thesis is to find, if possible, a direct solution, i.e., a constructive algorithm, for the interference alignment decomposition problem by applying only Householder reflections and Givens rotations (cf. Chapter 3). Considering our reformulation of the problem in Section 5.1, this means that our algorithm has to factorize a given global channel matrix $\boldsymbol{H}$ by applying a composition of Householder and Givens transformations such that Equation 5.3 is satisfied. However, in this section, we will discuss approaches towards finding a direct solution to the relaxed interference alignment decomposition problem, i.e., we will examine the possibilities of applying Householder reflections and Givens rotations such that $\boldsymbol{H}$ is decomposed according to Equation 5.4.

For being able to systematically analyze the effects of Householder and Givens transfor-

mations, we are going to introduce the concept of *decomposition steps*, which is applicable to both the interference alignment decomposition problem and the relaxed interference alignment decomposition problem.

Definition 5.7 (Decomposition steps)**.** A *decomposition step* is defined as exactly one application of a Householder or Givens transformation (cf. Section 3.2).

$$\bar{\boldsymbol{\Sigma}}^{(0)} \coloneqq \boldsymbol{H}$$

denotes the matrix to be decomposed before the first decomposition step and $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$ the matrix to be decomposed after the λ^{th} decomposition step. Furthermore, $\Lambda \in \mathbb{N}_0$ is defined as the total number of decomposition steps. If there is a direct solution to the interference alignment decomposition problem (or the relaxed interference alignment decomposition problem), Λ is a finite number and

$$\bar{\boldsymbol{\Sigma}}^{(\Lambda)} = \bar{\boldsymbol{\Sigma}}$$

(cf. Definitions 5.5 and 5.6).

Both an (extended) Householder matrix $\tilde{\mathcal{H}}$ (cf. Definition 3.2) and a Givens matrix $\mathcal{G}$ (cf. Definition 3.3) can be multiplied from the left side, i.e., premultiplied, and from the right side, i.e., postmultiplied. It follows that there are, in accordance with Definition 5.7, exactly four alternatives for the λ^{th} decomposition step:

$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \tilde{\mathcal{H}}\bar{\boldsymbol{\Sigma}}^{(\lambda-1)}, \tag{5.5}$$

$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \bar{\boldsymbol{\Sigma}}^{(\lambda-1)}\tilde{\mathcal{H}}, \tag{5.6}$$

$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \mathcal{G}\bar{\boldsymbol{\Sigma}}^{(\lambda-1)}, \text{ and} \tag{5.7}$$

$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \bar{\boldsymbol{\Sigma}}^{(\lambda-1)}\mathcal{G}. \tag{5.8}$$

The following lemma asserts that $\hat{\boldsymbol{U}}^{\text{H}}$ and $\hat{\boldsymbol{V}}$ (cf. Definition 5.6) are products of (extended) Householder and Givens matrices.

Lemma 5.8 ($\hat{\boldsymbol{U}}^{\text{H}}$ and $\hat{\boldsymbol{V}}$ are products of (extended) Householder and Givens matrices)**.** Let $\Delta = \{1, 2, \ldots, \Lambda\}$ be the set of decomposition step indices with $\bar{\boldsymbol{\Sigma}}^{(\Lambda)} = \bar{\boldsymbol{\Sigma}}$ (cf. Definition 5.7) and $\mathcal{U}^{(\lambda)}$, $\lambda \in \Delta$, the (extended) Householder or Givens matrix multiplied by $\bar{\boldsymbol{\Sigma}}^{(\lambda-1)}$ in the λ^{th} decomposition step. Furthermore, let $\Delta' = \{\delta'_1, \delta'_2, \ldots, \delta'_m\} \subseteq \Delta$ with $\delta'_k < \delta'_{k+1}$ be the set of decomposition step indices where $\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \mathcal{U}^{(\lambda)}\bar{\boldsymbol{\Sigma}}^{(\lambda-1)}$, $\lambda \in \Delta$, and let $\Delta'' = \{\delta''_1, \delta''_2, \ldots, \delta''_n\} \subseteq \Delta$ with $\delta''_k < \delta''_{k+1}$ be the set of decomposition step indices where $\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \bar{\boldsymbol{\Sigma}}^{(\lambda-1)}\mathcal{U}^{(\lambda)}$, $\lambda \in \Delta$. Then

$$\hat{\boldsymbol{U}}^{\text{H}} = \mathcal{U}^{(\delta'_m)}\mathcal{U}^{(\delta'_{m-1})} \ldots \mathcal{U}^{(\delta'_2)}\mathcal{U}^{(\delta'_1)} \text{ and}$$

$$\hat{\boldsymbol{V}} = \boldsymbol{\mathcal{U}}^{(\delta_1'')}\boldsymbol{\mathcal{U}}^{(\delta_2'')}\dots\boldsymbol{\mathcal{U}}^{(\delta_{n-1}'')}\boldsymbol{\mathcal{U}}^{(\delta_n'')}.$$

Proof. By definition, the only two allowed distinct types of decomposition steps are

$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \boldsymbol{\mathcal{U}}^{(\lambda)}\bar{\boldsymbol{\Sigma}}^{(\lambda-1)}, \lambda \in \Delta, \text{ and}$$
$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \bar{\boldsymbol{\Sigma}}^{(\lambda-1)}\boldsymbol{\mathcal{U}}^{(\lambda)}, \lambda \in \Delta.$$

Hence, $\Delta' \cap \Delta'' = \varnothing$ and $\Delta' \cup \Delta'' = \Delta$. Since $\bar{\boldsymbol{\Sigma}}^{(0)} = \boldsymbol{H}$ (cf. Definition 5.7), the first decomposition step is

$$\bar{\boldsymbol{\Sigma}}^{(1)} = \boldsymbol{\mathcal{U}}^{(1)}\boldsymbol{H} = \boldsymbol{\mathcal{U}}^{(\delta_1')}\boldsymbol{H} \text{ or}$$
$$\bar{\boldsymbol{\Sigma}}^{(1)} = \boldsymbol{H}\boldsymbol{\mathcal{U}}^{(1)} = \boldsymbol{H}\boldsymbol{\mathcal{U}}^{(\delta_1'')}.$$

Because of

$$\bar{\boldsymbol{\Sigma}}^{(\Lambda)} = \bar{\boldsymbol{\Sigma}} = \hat{\boldsymbol{U}}^{\mathrm{H}}\boldsymbol{H}\hat{\boldsymbol{V}}$$

(cf. Definition 5.6) and either

$$\bar{\boldsymbol{\Sigma}}^{(\Lambda)} = \boldsymbol{\mathcal{U}}^{(\Lambda)}\bar{\boldsymbol{\Sigma}}^{(\Lambda-1)} = \boldsymbol{\mathcal{U}}^{(\delta_m')}\bar{\boldsymbol{\Sigma}}^{(\Lambda-1)} \text{ or}$$
$$\bar{\boldsymbol{\Sigma}}^{(\Lambda)} = \bar{\boldsymbol{\Sigma}}^{(\Lambda-1)}\boldsymbol{\mathcal{U}}^{(\Lambda)} = \bar{\boldsymbol{\Sigma}}^{(\Lambda-1)}\boldsymbol{\mathcal{U}}^{(\delta_n'')},$$

it holds that

$$\hat{\boldsymbol{U}}^{\mathrm{H}}\boldsymbol{H}\hat{\boldsymbol{V}} = \boldsymbol{\mathcal{U}}^{(\delta_m')}\bar{\boldsymbol{\Sigma}}^{(\Lambda-1)} \text{ or}$$
$$\hat{\boldsymbol{U}}^{\mathrm{H}}\boldsymbol{H}\hat{\boldsymbol{V}} = \bar{\boldsymbol{\Sigma}}^{(\Lambda-1)}\boldsymbol{\mathcal{U}}^{(\delta_n'')}$$

and thus by induction

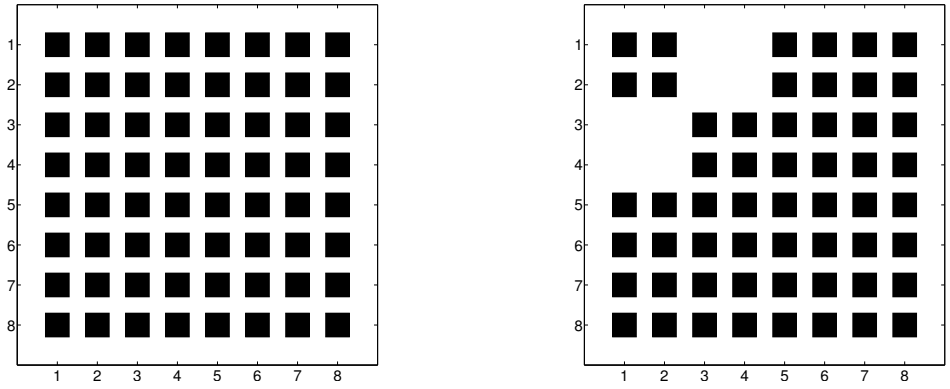
$$\hat{\boldsymbol{U}}^{\mathrm{H}}\boldsymbol{H}\hat{\boldsymbol{V}} = \boldsymbol{\mathcal{U}}^{(\delta_m')}\boldsymbol{\mathcal{U}}^{(\delta_{m-1}')}\dots\boldsymbol{\mathcal{U}}^{(\delta_2')}\boldsymbol{\mathcal{U}}^{(\delta_1')}\boldsymbol{H}\boldsymbol{\mathcal{U}}^{(\delta_1'')}\boldsymbol{\mathcal{U}}^{(\delta_2'')}\dots\boldsymbol{\mathcal{U}}^{(\delta_{n-1}'')}\boldsymbol{\mathcal{U}}^{(\delta_n'')}.$$

It follows the assertion. □

Definition 5.6 requires $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ to be unitary matrices. The following corollary confirms that $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ are unitary matrices when solely Householder and Givens transformations are applied for solving the relaxed interference alignment decomposition problem.

Corollary 5.9 ($\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ are unitary matrices)**.** Let there be Λ decomposition steps such that $\bar{\boldsymbol{\Sigma}}^{(\Lambda)} = \bar{\boldsymbol{\Sigma}}$ (cf. Definition 5.7). Then, $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ (cf. Definition 5.6) are unitary matrices.

Proof. Since a product of unitary matrices is unitary itself, and (extended) Householder and Givens matrices are unitary matrices (cf. Chapter 3), it readily follows from Lemma 5.8



(a) $\boldsymbol{H}$ (b) Targeted $\bar{\boldsymbol{\Sigma}}$

Figure 5.1: The global channel matrix $\boldsymbol{H}$ and $\bar{\boldsymbol{\Sigma}}$ as targeted during the decomposition of $\boldsymbol{H}$ for $K = 2$, $d = 2$, $M = 4$, and $N = 4$.

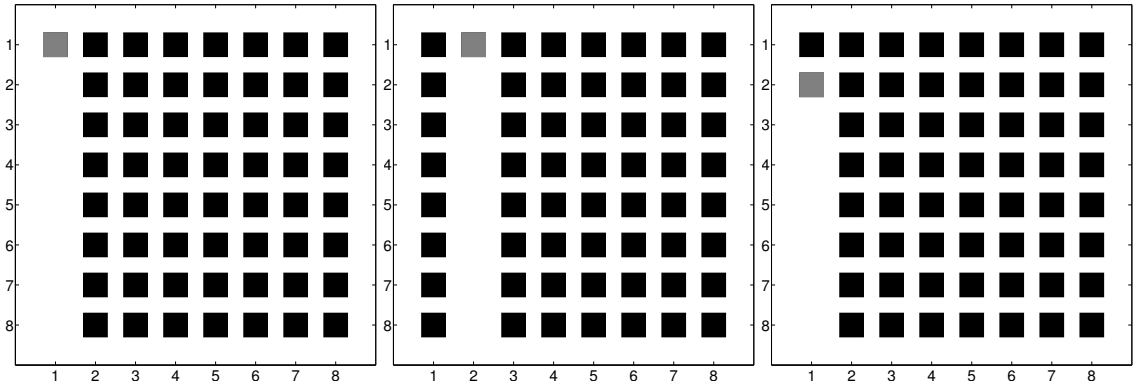
that $\hat{\boldsymbol{U}}^{\mathrm{H}}$ and $\hat{\boldsymbol{V}}$ are unitary matrices. As the conjugate transpose of a unitary matrix is unitary, $\hat{\boldsymbol{U}}$ is a unitary matrix, too. □

Figure 5.1a visualizes the global channel matrix $\boldsymbol{H}$ for $K = 2$, $d = 2$, $M = 4$, and $N = 4$. In this example, $\boldsymbol{H}$ is of dimension $KN \times KM = 8 \times 8$. Each of the black squares represents one (not necessarily, but potentially) nonzero matrix element $h_{ij} \in \mathbb{C}$ of $\boldsymbol{H}$, $i \in \{1, 2, \ldots, KN\}$, $j \in \{1, 2, \ldots, KM\}$. The corresponding matrix $\bar{\boldsymbol{\Sigma}}$, also of dimension $KN \times KM = 8 \times 8$, that is targeted during the decomposition of $\boldsymbol{H}$, i.e., by multiplying $\boldsymbol{H}$ with (extended) Householder and Givens matrices, is shown in Figure 5.1b. Again, the black squares represent nonzero matrix elements $\bar{\sigma}_{ij} \in \mathbb{C}$ of $\bar{\boldsymbol{\Sigma}}$, $i \in \{1, 2, \ldots, KN\}$, $j \in \{1, 2, \ldots, KM\}$, while the white space represents (strictly) zero elements. As we can see, $\boldsymbol{\Sigma}$, the top-left $Kd \times Kd = 4 \times 4$ submatrix of $\bar{\boldsymbol{\Sigma}}$, is a block-diagonal matrix with blocks of size $d \times d = 2 \times 2$. For a correct solution, the elements of $\bar{\boldsymbol{\Sigma}}$ shown as nonzero elements may (partly) also be zero elements.

5.3.1 Application of Householder reflections

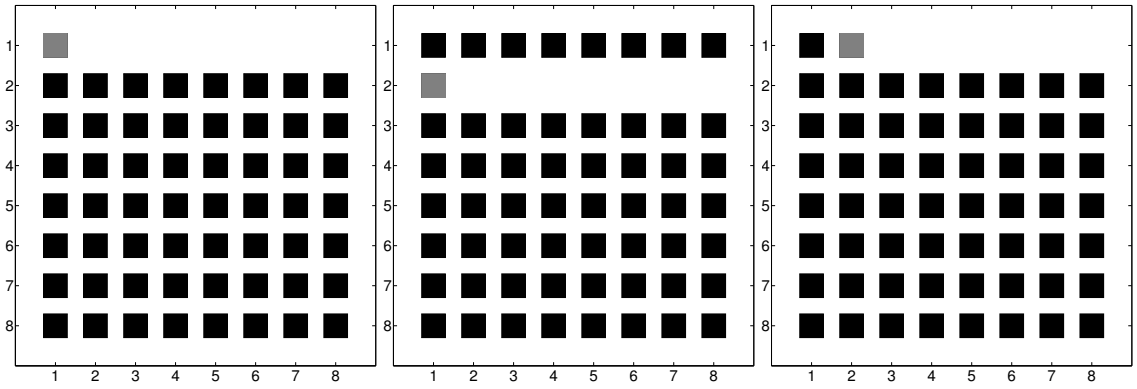
Since $\boldsymbol{H} = \bar{\boldsymbol{\Sigma}}^{(0)}$ is a dense matrix, and Householder reflections in contrast to Givens rotations are able to introduce several zeros at once into a matrix (cf. Chapter 3), we are first going to examine decomposition steps in which (extended) Householder matrices are premultiplied or postmultiplied. Figure 5.2 depicts some examples for $\bar{\boldsymbol{\Sigma}}^{(1)} = \tilde{\mathcal{H}}\bar{\boldsymbol{\Sigma}}^{(0)}$. In each of the three illustrations, a different (extended) Householder matrix $\tilde{\mathcal{H}}$ is premultiplied, resulting in different columns or parts of columns being annihilated. The annihilated (part of a) column of $\bar{\boldsymbol{\Sigma}}^{(1)}$, including the nonzero top element (highlighted in gray), is equal to $\tilde{\mathcal{H}}\boldsymbol{z}$, where $\boldsymbol{z}$ denotes the column vector of corresponding elements of $\bar{\boldsymbol{\Sigma}}^{(0)}$ (cf. Definition 3.1).

According to Figure 5.3, the situation for $\bar{\boldsymbol{\Sigma}}^{(1)} = \bar{\boldsymbol{\Sigma}}^{(0)}\tilde{\mathcal{H}}$ is very similar. The only difference is that rows instead of columns are annihilated, which means that the annihilated (part of a) row of $\bar{\boldsymbol{\Sigma}}^{(1)}$, including the nonzero leftmost element (again highlighted in gray),



(a) (b) (c)

Figure 5.2: The global channel matrix $\boldsymbol{H}$ premultiplied by (extended) Householder matrices that annihilate different columns or parts of columns.



(a) (b) (c)

Figure 5.3: The global channel matrix $\boldsymbol{H}$ postmultiplied by (extended) Householder matrices that annihilate different rows or parts of rows.

is equal to $\boldsymbol{z}^{\mathrm{H}}\tilde{\mathcal{H}}$, where $\boldsymbol{z}^{\mathrm{H}}$ denotes the row vector of corresponding elements of $\bar{\boldsymbol{\Sigma}}^{(0)}$. For this reason, we can focus on the case $\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \tilde{\mathcal{H}}\bar{\boldsymbol{\Sigma}}^{(\lambda-1)}$, our conclusions will be equally valid for the case $\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \bar{\boldsymbol{\Sigma}}^{(\lambda-1)}\tilde{\mathcal{H}}$.

As we have seen, we can annihilate any column or row of $\bar{\boldsymbol{\Sigma}}^{(0)}$ (except for the top or leftmost element) by premultiplying or postmultiplying an appropriate (extended) Householder matrix. Since we want $\bar{\boldsymbol{\Sigma}}^{(\Lambda)}(1:Kd, 1:Kd) = \bar{\boldsymbol{\Sigma}}(1:Kd, 1:Kd) = \boldsymbol{\Sigma}$ to be of block-diagonal form (cf. Definition 5.6), we need to know which exact matrix elements have to be annihilated.

Lemma 5.10 (Zero elements of $\bar{\boldsymbol{\Sigma}}$)**.** Let $\bar{\boldsymbol{\Sigma}} = (\bar{\sigma}_{ij}) \in \mathbb{C}^{KN\times KM}$ be the extended global interference alignment matrix with $\boldsymbol{\Sigma} = \bar{\boldsymbol{\Sigma}}(1:Kd, 1:Kd) \in \mathbb{C}^{Kd\times Kd}$ being a block-diagonal matrix with the main diagonal blocks $\boldsymbol{\Sigma}_k \in \mathbb{C}^{d\times d}$, $k \in \{1, 2, \ldots, K\}$

(cf. Definitions 5.5 and 5.6). Then, it holds that

$$\forall \kappa \in \{1, 2, \ldots, K-1\}\colon\ \forall \delta \in \{1, 2, \ldots, d\}\colon$$
$$\bar{\sigma}_{\kappa d+1,(\kappa-1)d+\delta} = \bar{\sigma}_{\kappa d+2,(\kappa-1)d+\delta} = \cdots = \bar{\sigma}_{Kd,(\kappa-1)d+\delta} = 0$$

and

$$\forall \kappa \in \{1, 2, \ldots, K-1\}\colon\ \forall \delta \in \{1, 2, \ldots, d\}\colon$$
$$\bar{\sigma}_{(\kappa-1)d+\delta,\kappa d+1} = \bar{\sigma}_{(\kappa-1)d+\delta,\kappa d+2} = \cdots = \bar{\sigma}_{(\kappa-1)d+\delta,Kd} = 0,$$

and there is no other element of $\bar{\boldsymbol{\Sigma}}$ that is required to be equal to zero.

Proof. For $\boldsymbol{\Sigma}$ to be a block-diagonal matrix, all elements of $\boldsymbol{\Sigma}$ except for the elements of the main diagonal blocks $\boldsymbol{\Sigma}_k$, $k \in \{1, 2, \ldots, K\}$, are required to be equal to zero. It follows that the elements of $\boldsymbol{\Sigma}$ below and right of each main diagonal block $\boldsymbol{\Sigma}_k$, $k \in \{1, 2, \ldots, K\}$, need to be equal to zero. Since each element left of or above $\boldsymbol{\Sigma}_{k'}$, $k' \in \{2, 3, \ldots, K\}$, is an element below or right of one $\boldsymbol{\Sigma}_{k''}$, $k'' \in \{1, 2, \ldots, k'-1\}$, it is sufficient to consider the elements of $\boldsymbol{\Sigma}$ below and right of each main diagonal block $\boldsymbol{\Sigma}_k$, $k \in \{1, 2, \ldots, K\}$.

The elements of $\boldsymbol{\Sigma}$ below $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}(1\colon d, 1\colon d) = \bar{\boldsymbol{\Sigma}}(1\colon d, 1\colon d)$ are the elements of $\boldsymbol{\Sigma}(d+1\colon Kd, 1\colon d) = \bar{\boldsymbol{\Sigma}}(d+1\colon Kd, 1\colon d)$, i.e.,

$$\forall \delta \in \{1, 2, \ldots, d\}\colon\ \bar{\sigma}_{d+1,\delta}, \bar{\sigma}_{d+2,\delta}, \ldots, \bar{\sigma}_{Kd,\delta},$$

and the elements of $\boldsymbol{\Sigma}$ right of $\boldsymbol{\Sigma}_1$ are the elements of $\boldsymbol{\Sigma}(1\colon d, d+1\colon Kd) = \bar{\boldsymbol{\Sigma}}(1\colon d, d+1\colon Kd)$, i.e.,

$$\forall \delta \in \{1, 2, \ldots, d\}\colon\ \bar{\sigma}_{\delta,d+1}, \bar{\sigma}_{\delta,d+2}, \ldots, \bar{\sigma}_{\delta,Kd}.$$

Similarly, the elements of $\boldsymbol{\Sigma}$ below $\boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}(d+1\colon 2d, d+1\colon 2d) = \bar{\boldsymbol{\Sigma}}(d+1\colon 2d, d+1\colon 2d)$ are the elements of $\boldsymbol{\Sigma}(2d+1\colon Kd, d+1\colon 2d) = \bar{\boldsymbol{\Sigma}}(2d+1\colon Kd, d+1\colon 2d)$, i.e.,

$$\forall \delta \in \{1, 2, \ldots, d\}\colon\ \bar{\sigma}_{2d+1,d+\delta}, \bar{\sigma}_{2d+2,d+\delta}, \ldots, \bar{\sigma}_{Kd,d+\delta},$$

and the elements of $\boldsymbol{\Sigma}$ right of $\boldsymbol{\Sigma}_2$ are the elements of $\boldsymbol{\Sigma}(d+1\colon 2d, 2d+1\colon Kd) = \bar{\boldsymbol{\Sigma}}(d+1\colon 2d, 2d+1\colon Kd)$, i.e.,

$$\forall \delta \in \{1, 2, \ldots, d\}\colon\ \bar{\sigma}_{d+\delta,2d+1}, \bar{\sigma}_{d+\delta,2d+2}, \ldots, \bar{\sigma}_{d+\delta,Kd}.$$

By induction, a similar argument applies for all $\boldsymbol{\Sigma}_\kappa = \boldsymbol{\Sigma}((\kappa-1)d+1\colon \kappa d, (\kappa-1)d+1\colon \kappa d) = \bar{\boldsymbol{\Sigma}}((\kappa-1)d+1\colon \kappa d, (\kappa-1)d+1\colon \kappa d)$, $\kappa \in \{1, 2, \ldots, K-1\}$. Since there are no elements of $\boldsymbol{\Sigma}$ below and right of $\boldsymbol{\Sigma}_K = \boldsymbol{\Sigma}((K-1)d+1\colon Kd, (K-1)d+1\colon Kd)$, the assertion of the lemma is true. □

Lemma 5.10 specifies the minimum set of zero elements for $\bar{\boldsymbol{\Sigma}}$ being in conformance

with Definitions 5.5 and 5.6. As previously mentioned, other elements of $\bar{\boldsymbol{\Sigma}}$ may also be zero for a valid result. Let us now discuss if there is a preferred sequence of column annihilations. For this purpose, it is advantageous to agree on a notation for (extended) Householder and Givens matrices that annihilate specific elements when applied to a $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{0, 1, \ldots, \Lambda - 1\}$.

Definition 5.11 (Specific Householder and Givens transformations in decomposition steps)**.** Let $\bar{\boldsymbol{\Sigma}}^{(\lambda-1)} \in \mathbb{C}^{KN \times KM}$, $\lambda \in \{1, 2, \ldots, \Lambda\}$, be the matrix to be decomposed in the λ^{th} decomposition step (cf. Definition 5.7) and $\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \left(\bar{\sigma}_{ij}^{(\lambda)}\right) \in \mathbb{C}^{KN \times KM}$ the matrix resulting from the λ^{th} decomposition step. Then, $\mathcal{H}_{i_2,j}^{i_1,j} \in \mathbb{C}^{KN \times KN}$, $1 \leq i_1 < i_2 \leq KN$, $1 \leq j \leq KM$, denotes the (extended) Householder matrix (cf. Definition 3.2) that affects the rows $\bar{\boldsymbol{\Sigma}}^{(\lambda)}(i_1 : i_2, :)$ for annihilating the elements $\bar{\sigma}_{i_1+1,j}^{(\lambda)}, \bar{\sigma}_{i_1+2,j}^{(\lambda)}, \ldots, \bar{\sigma}_{i_2,j}^{(\lambda)}$ in the decomposition step

$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \mathcal{H}_{i_2,j}^{i_1,j} \bar{\boldsymbol{\Sigma}}^{(\lambda-1)},$$

while $\mathcal{H}_{i,j_2}^{i,j_1} \in \mathbb{C}^{KM \times KM}$, $1 \leq i \leq KN$, $1 \leq j_1 < j_2 \leq KM$, is defined as the (extended) Householder matrix that affects the columns $\bar{\boldsymbol{\Sigma}}^{(\lambda)}(:, j_1 : j_2)$ for annihilating the elements $\bar{\sigma}_{i,j_1+1}^{(\lambda)}, \bar{\sigma}_{i,j_1+2}^{(\lambda)}, \ldots, \bar{\sigma}_{i,j_2}^{(\lambda)}$ in the decomposition step

$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \bar{\boldsymbol{\Sigma}}^{(\lambda-1)} \mathcal{H}_{i,j_2}^{i,j_1}.$$

Likewise, $\mathcal{G}_{i_2,j}^{i_1,j} \in \mathbb{C}^{KN \times KN}$, $1 \leq i_1 \neq i_2 \leq KN$, $1 \leq j \leq KM$, denotes the Givens matrix (cf. Definition 3.3) that affects the rows $\bar{\boldsymbol{\Sigma}}^{(\lambda)}(i_1 : i_1, :)$ and $\bar{\boldsymbol{\Sigma}}^{(\lambda)}(i_2 : i_2, :)$ for annihilating the element $\bar{\sigma}_{i_2,j}^{(\lambda)}$ in the decomposition step

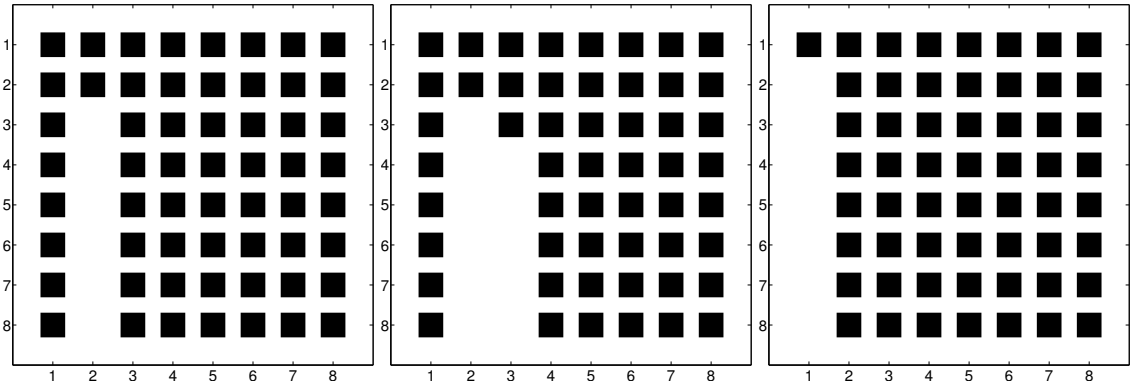
$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \mathcal{G}_{i_2,j}^{i_1,j} \bar{\boldsymbol{\Sigma}}^{(\lambda-1)},$$

while $\mathcal{G}_{i,j_2}^{i,j_1} \in \mathbb{C}^{KM \times KM}$, $1 \leq i \leq KN$, $1 \leq j_1 \neq j_2 \leq KM$, is defined as the Givens matrix that affects the columns $\bar{\boldsymbol{\Sigma}}^{(\lambda)}(:, j_1 : j_1)$ and $\bar{\boldsymbol{\Sigma}}^{(\lambda)}(:, j_2 : j_2)$ for annihilating the element $\bar{\sigma}_{i,j_2}^{(\lambda)}$ in the decomposition step

$$\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \bar{\boldsymbol{\Sigma}}^{(\lambda-1)} \mathcal{G}_{i,j_2}^{i,j_1}.$$

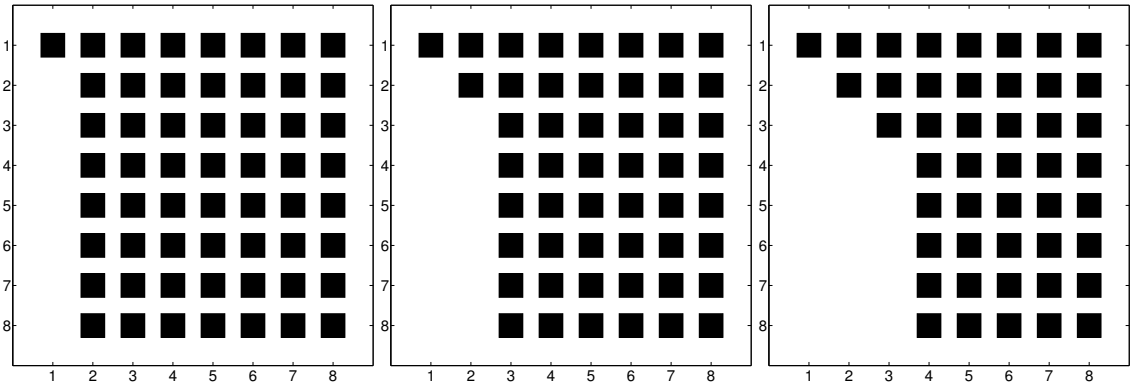
Based on the notation introduced in Definition 5.11, we now want to demonstrate how the order of Householder matrix applications influences the resulting matrix. Figure 5.4 illustrates $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{1, 2, 3\}$, for the simple sequence

$$\begin{aligned}
\bar{\boldsymbol{\Sigma}}^{(1)} &= \mathcal{H}_{8,2}^{2,2} \bar{\boldsymbol{\Sigma}}^{(0)}, \\
\bar{\boldsymbol{\Sigma}}^{(2)} &= \mathcal{H}_{8,3}^{3,3} \bar{\boldsymbol{\Sigma}}^{(1)}, \text{ and} \\
\bar{\boldsymbol{\Sigma}}^{(3)} &= \mathcal{H}_{8,1}^{1,1} \bar{\boldsymbol{\Sigma}}^{(2)},
\end{aligned}$$



(a) $\bar{\boldsymbol{\Sigma}}^{(1)} = \mathcal{H}_{8,2}^{2,2}\bar{\boldsymbol{\Sigma}}^{(0)}$ (b) $\bar{\boldsymbol{\Sigma}}^{(2)} = \mathcal{H}_{8,3}^{3,3}\bar{\boldsymbol{\Sigma}}^{(1)}$ (c) $\bar{\boldsymbol{\Sigma}}^{(3)} = \mathcal{H}_{8,1}^{1,1}\bar{\boldsymbol{\Sigma}}^{(2)}$

Figure 5.4: $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{1, 2, 3\}$, for a sequence of Householder transformations with inappropriate element annihilations.



(a) $\bar{\boldsymbol{\Sigma}}^{(1)} = \mathcal{H}_{8,1}^{1,1}\bar{\boldsymbol{\Sigma}}^{(0)}$ (b) $\bar{\boldsymbol{\Sigma}}^{(2)} = \mathcal{H}_{8,2}^{2,2}\bar{\boldsymbol{\Sigma}}^{(1)}$ (c) $\bar{\boldsymbol{\Sigma}}^{(3)} = \mathcal{H}_{8,3}^{3,3}\bar{\boldsymbol{\Sigma}}^{(2)}$

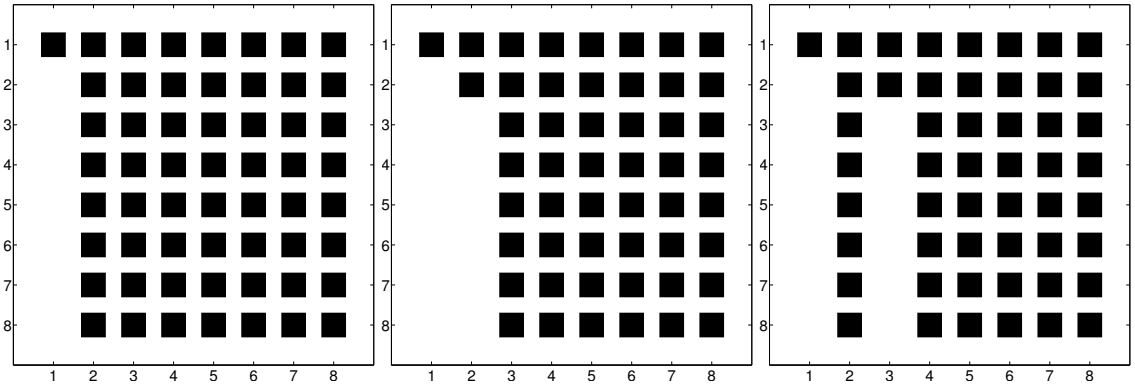
Figure 5.5: $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{1, 2, 3\}$, for a sequence of Householder transformations with appropriate element annihilations.

while Figure 5.5 visualizes $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{1, 2, 3\}$, for

$$\begin{aligned}\bar{\boldsymbol{\Sigma}}^{(1)} &= \mathcal{H}_{8,1}^{1,1}\bar{\boldsymbol{\Sigma}}^{(0)},\\ \bar{\boldsymbol{\Sigma}}^{(2)} &= \mathcal{H}_{8,2}^{2,2}\bar{\boldsymbol{\Sigma}}^{(1)}, \text{ and}\\ \bar{\boldsymbol{\Sigma}}^{(3)} &= \mathcal{H}_{8,3}^{3,3}\bar{\boldsymbol{\Sigma}}^{(2)}.\end{aligned}$$

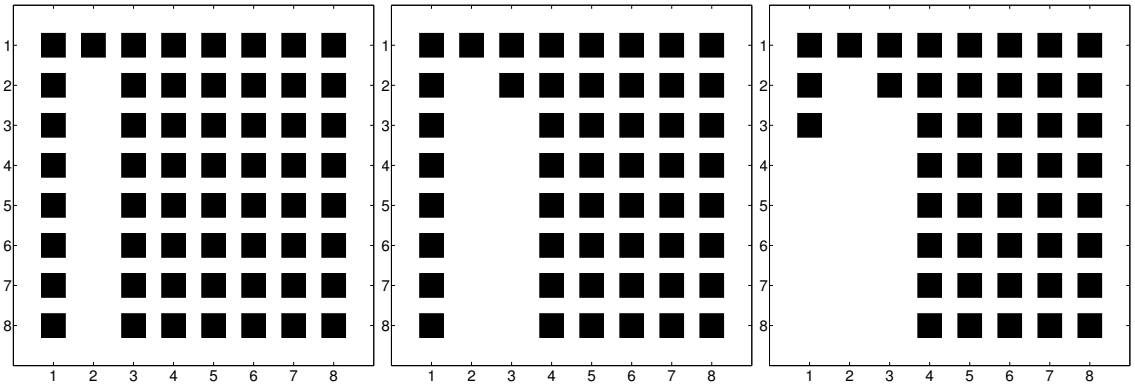
In these two example sequences, we are considering applications of Householder transformations such that always $i_1 = j$ for $\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \mathcal{H}_{8,j}^{i_1,j}\bar{\boldsymbol{\Sigma}}^{(\lambda-1)}$, $\lambda, i_1, j \in \{1, 2, 3\}$. Under this assumption, the preferred sequence of column annihilations is achieved when we choose $\lambda = i_1 = j$, i.e., when we start with the first column, proceed to the second, then the third, and so on. If we disregard this order as in the example shown in Figure 5.4, the process of column annihilations is reversed for all columns right of the one that is annihilated in the respective decomposition step (cf. Figure 5.4c).

Would it help us to allow $i_1 \neq j$ for $\bar{\boldsymbol{\Sigma}}^{(\lambda)} = \mathcal{H}_{i_2,j}^{i_1,j}\bar{\boldsymbol{\Sigma}}^{(\lambda-1)}$, $\lambda \in \{1, 2, \ldots, \Lambda\}$, $i_1 \in$



(a) $\bar{\mathbf{\Sigma}}^{(1)} = \mathcal{H}_{8,1}^{1,1}\bar{\mathbf{\Sigma}}^{(0)}$ (b) $\bar{\mathbf{\Sigma}}^{(2)} = \mathcal{H}_{8,2}^{2,2}\bar{\mathbf{\Sigma}}^{(1)}$ (c) $\bar{\mathbf{\Sigma}}^{(3)} = \mathcal{H}_{8,3}^{2,3}\bar{\mathbf{\Sigma}}^{(2)}$

Figure 5.6: $\bar{\mathbf{\Sigma}}^{(\lambda)}$, $\lambda \in \{1, 2, 3\}$, for a sequence of Householder transformations with a nondecreasing number of annihilated elements per column.



(a) $\bar{\mathbf{\Sigma}}^{(1)} = \mathcal{H}_{8,2}^{1,2}\bar{\mathbf{\Sigma}}^{(0)}$ (b) $\bar{\mathbf{\Sigma}}^{(2)} = \mathcal{H}_{8,3}^{2,3}\bar{\mathbf{\Sigma}}^{(1)}$ (c) $\bar{\mathbf{\Sigma}}^{(3)} = \mathcal{H}_{8,1}^{3,1}\bar{\mathbf{\Sigma}}^{(2)}$

Figure 5.7: $\bar{\mathbf{\Sigma}}^{(\lambda)}$, $\lambda \in \{1, 2, 3\}$, for a sequence of Householder transformations with a decreasing number of annihilated elements per column.

$\{1, 2, \ldots, KN-1\}$, $i_2 = KN$, $j \in \{1, 2, \ldots, KM\}$? If we wanted the $d \times d$ main diagonal blocks of the global interference alignment matrix $\mathbf{\Sigma} = \bar{\mathbf{\Sigma}}(1\!:\!Kd, 1\!:\!Kd)$ to consist only of strictly nonzero elements, it seemed reasonable to regard (extended) Householder matrices which annihilate the elements of $\mathbf{\Sigma}$ specified by Lemma 5.10 (and potentially other elements of $\bar{\mathbf{\Sigma}}$ where appropriate). Corresponding decomposition steps for $i_2 = KN$ and all $\kappa \in \{1, 2, \ldots, K-1\}$ are

$$\begin{aligned}
\bar{\mathbf{\Sigma}}^{((\kappa-1)d+1)} &= \mathcal{H}_{KN,(\kappa-1)d+1}^{\kappa d,(\kappa-1)d+1}\bar{\mathbf{\Sigma}}^{((\kappa-1)d)}, \\
\bar{\mathbf{\Sigma}}^{((\kappa-1)d+2)} &= \mathcal{H}_{KN,(\kappa-1)d+2}^{\kappa d,(\kappa-1)d+2}\bar{\mathbf{\Sigma}}^{((\kappa-1)d+1)}, \\
&\vdots \\
\bar{\mathbf{\Sigma}}^{((\kappa-1)d+d)} &= \mathcal{H}_{KN,(\kappa-1)d+d}^{\kappa d,(\kappa-1)d+d}\bar{\mathbf{\Sigma}}^{((\kappa-1)d+d-1)}.
\end{aligned}$$

However, by considering the sample sequence

$$\begin{aligned}\bar{\mathbf{\Sigma}}^{(1)} &= \mathcal{H}_{8,1}^{1,1}\bar{\mathbf{\Sigma}}^{(0)},\\ \bar{\mathbf{\Sigma}}^{(2)} &= \mathcal{H}_{8,2}^{2,2}\bar{\mathbf{\Sigma}}^{(1)}, \text{ and}\\ \bar{\mathbf{\Sigma}}^{(3)} &= \mathcal{H}_{8,3}^{2,3}\bar{\mathbf{\Sigma}}^{(2)},\end{aligned}$$

as shown in Figure 5.6, we notice that the application of Householder reflections is not sufficient for being able to achieve this. As soon as after $\mathcal{H}_{i_2,j}^{i_1,j}$ in a subsequent decomposition step $\mathcal{H}_{i_2,j'}^{i_1',j'}$ with $i_1 \geq i_1'$ and $j \neq j'$ is applied, a *fill-in* occurs in column j, i.e., previously annihilated elements become nonzero again. We can see this behavior in Figures 5.6b and 5.6c.

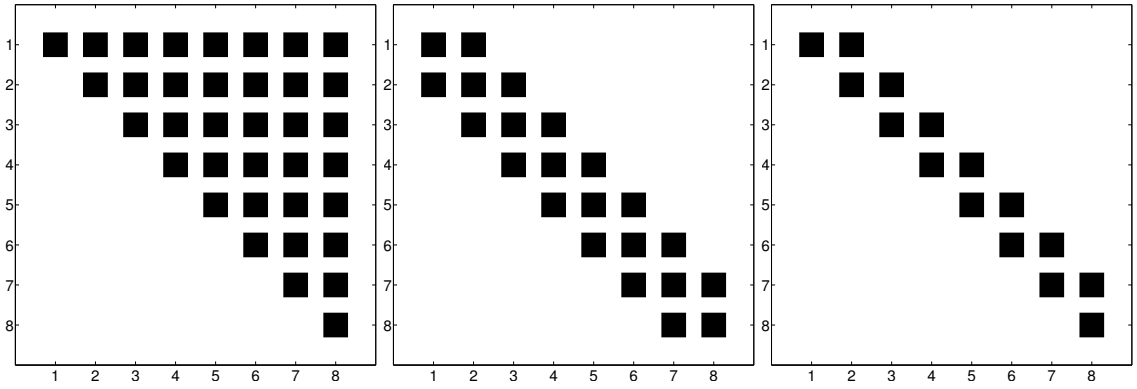
For further clarification, Figure 5.7 utilizes the sample sequence

$$\begin{aligned}\bar{\mathbf{\Sigma}}^{(1)} &= \mathcal{H}_{8,2}^{1,2}\bar{\mathbf{\Sigma}}^{(0)},\\ \bar{\mathbf{\Sigma}}^{(2)} &= \mathcal{H}_{8,3}^{2,3}\bar{\mathbf{\Sigma}}^{(1)}, \text{ and}\\ \bar{\mathbf{\Sigma}}^{(3)} &= \mathcal{H}_{8,1}^{3,1}\bar{\mathbf{\Sigma}}^{(2)}\end{aligned}$$

to illustrate that the significant aspect for preventing a fill-in is the increasing number of top elements that are not annihilated in a column and not the order of column annihilations. Nonetheless, since we want to bring the top-left $Kd \times Kd$ submatrix of $\bar{\mathbf{\Sigma}}^{(\lambda)}$, $\lambda \in \{0, 1, \ldots, \Lambda - 1\}$, to block-diagonal form, the annihilation of columns from the leftmost to the rightmost is beneficial to us.

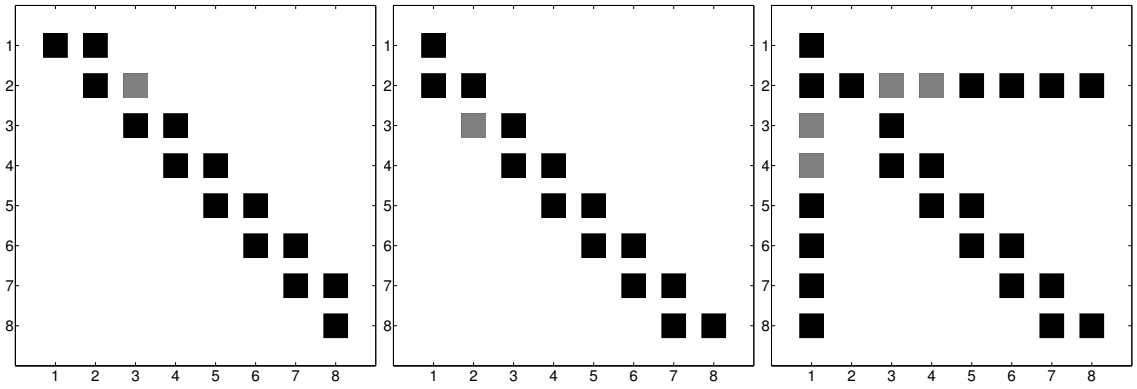
When we algorithmize the pattern indicated by the example of Figure 5.5, we obtain the *QR factorization* method based on Householder transformations [GV13, pp. 248–249] [TB97, p. 73]. As shown in Figure 5.8a, the result of the QR factorization method is an upper triangular matrix. A similar algorithmic pattern can be utilized to bring a matrix to tridiagonal and even bidiagonal form [GV13, pp. 284–285]. Instead of only premultiplying Householder matrices, they are, in both cases, alternately premultiplied and postmultiplied. The results are illustrated in Figures 5.8b and 5.8c.

Clearly, these resulting matrices and their submatrices are not block-diagonal matrices. But can we apply Householder reflections in a comparable manner to obtain $\mathbf{\Sigma}$ in block-diagonal form? Let us again assume that $K = 2$, $d = 2$, $M = 4$, and $N = 4$. Figure 5.9a shows the result after bidiagonalization we already have seen, but highlights the (only) nonzero element that has to be annihilated in this example. If we start the bidiagonalization process with postmultiplying a Householder matrix, the outcome is not anymore an upper but a lower bidiagonal matrix, as the illustration in Figure 5.9b shows. However, there still is one nonzero element that needs to be annihilated in our example. But if we try to achieve this with a Householder transformation, we face the problem we have discussed by means of Figure 5.6. The resulting matrix, which contains four nonzero elements to be annihilated, is depicted in Figure 5.9c.



(a) QR factorization (b) Tridiagonalization (c) Bidiagonalization

Figure 5.8: Results of the QR factorization, tridiagonalization, and bidiagonalization methods based on Householder transformations.



(a) (b) (c)

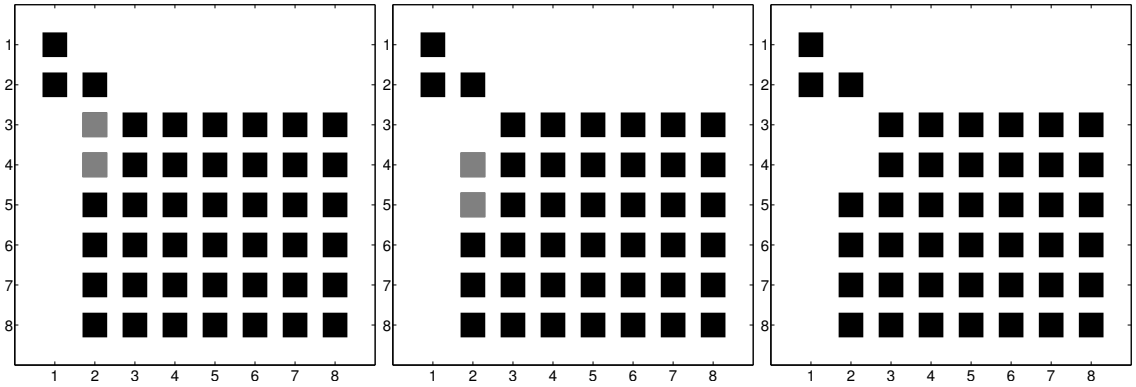
Figure 5.9: An attempt to utilize an approach similar to the bidiagonalization method based on Householder transformations for obtaining the block-diagonal submatrix $\mathbf{\Sigma}$ of $\bar{\mathbf{\Sigma}}$ for $K = 2$, $d = 2$, $M = 4$, and $N = 4$.

5.3.2 Application of Givens rotations

We have to conclude that the application of Householder transformations is not sufficient for overcoming this problem. For this reason, we want to analyze if Givens transformations (cf. Chapter 3) provide a feasible solution. Because of only two affected rows or columns per transformation, a major benefit of Givens rotations is that they allow elements to be annihilated more selectively. Despite this, the briefly presented QR decomposition [GV13, pp. 252–253], tridiagonalization, and bidiagonalization methods can also be based on Givens rotations.

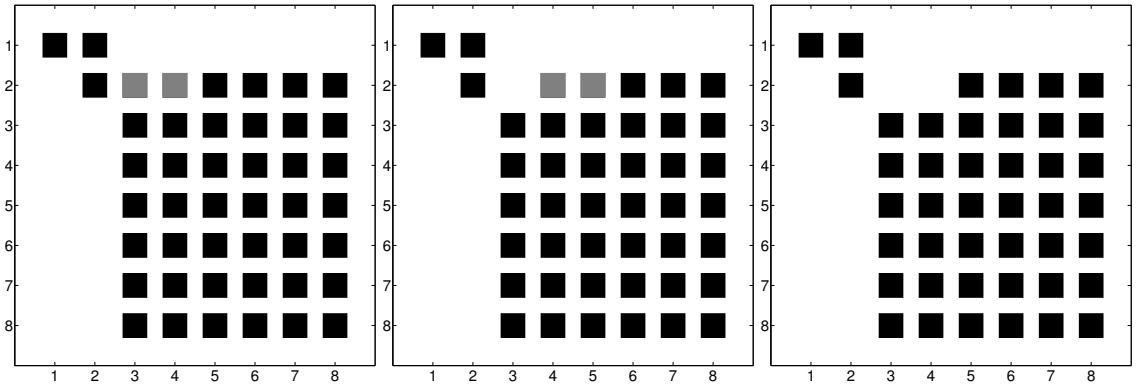
As our first example that takes advantage of Givens rotations, we consider a sequence of Householder and Givens transformations that starts like the Householder-based bidiagonalization method leading to a lower bidiagonal matrix, in particular

$$\bar{\mathbf{\Sigma}}^{(1)} = \bar{\mathbf{\Sigma}}^{(0)} \mathcal{H}_{1,8}^{1,1},$$



(a) $\bar{\boldsymbol{\Sigma}}^{(3)} = \bar{\boldsymbol{\Sigma}}^{(2)} \mathcal{H}_{2,8}^{2,2}$ (b) $\bar{\boldsymbol{\Sigma}}^{(4)} = \mathcal{G}_{3,2}^{4,2} \bar{\boldsymbol{\Sigma}}^{(3)}$ (c) $\bar{\boldsymbol{\Sigma}}^{(5)} = \mathcal{G}_{4,2}^{5,2} \bar{\boldsymbol{\Sigma}}^{(4)}$

Figure 5.10: Transformation of the global channel matrix $\boldsymbol{H} = \bar{\boldsymbol{\Sigma}}^{(0)}$ to $\bar{\boldsymbol{\Sigma}} = \bar{\boldsymbol{\Sigma}}^{(5)}$ by premultiplying Givens matrices after a series of Householder reflections.



(a) $\bar{\boldsymbol{\Sigma}}^{(3)} = \mathcal{H}_{8,2}^{2,2} \bar{\boldsymbol{\Sigma}}^{(2)}$ (b) $\bar{\boldsymbol{\Sigma}}^{(4)} = \bar{\boldsymbol{\Sigma}}^{(3)} \mathcal{G}_{2,3}^{2,4}$ (c) $\bar{\boldsymbol{\Sigma}}^{(5)} = \bar{\boldsymbol{\Sigma}}^{(4)} \mathcal{G}_{2,4}^{2,5}$

Figure 5.11: Transformation of the global channel matrix $\boldsymbol{H} = \bar{\boldsymbol{\Sigma}}^{(0)}$ to $\bar{\boldsymbol{\Sigma}} = \bar{\boldsymbol{\Sigma}}^{(5)}$ by postmultiplying Givens matrices after a series of Householder reflections.

$$
\begin{aligned}
\bar{\boldsymbol{\Sigma}}^{(2)} &= \mathcal{H}_{8,1}^{2,1} \bar{\boldsymbol{\Sigma}}^{(1)}, \\
\bar{\boldsymbol{\Sigma}}^{(3)} &= \bar{\boldsymbol{\Sigma}}^{(2)} \mathcal{H}_{2,8}^{2,2}, \\
\bar{\boldsymbol{\Sigma}}^{(4)} &= \mathcal{G}_{3,2}^{4,2} \bar{\boldsymbol{\Sigma}}^{(3)}, \text{ and} \\
\bar{\boldsymbol{\Sigma}}^{(5)} &= \mathcal{G}_{4,2}^{5,2} \bar{\boldsymbol{\Sigma}}^{(4)}.
\end{aligned}
$$

Figure 5.10 visualizes the last three decomposition steps of the above sequence of transformations. In Figures 5.10a and 5.10b, the elements $\bar{\sigma}_{3,2}^{(3)}$ and $\bar{\sigma}_{4,2}^{(3)}$ of $\bar{\boldsymbol{\Sigma}}^{(3)}$ and the elements $\bar{\sigma}_{4,2}^{(4)}$ and $\bar{\sigma}_{5,2}^{(4)}$ of $\bar{\boldsymbol{\Sigma}}^{(4)}$ are highlighted to indicate which elements are central to the following Givens rotation. For $K = 2$, $d = 2$, $M = 4$, and $N = 4$, the top-left $Kd \times Kd = 4 \times 4$ submatrix of the resulting $KN \times KM = 8 \times 8$ matrix $\bar{\boldsymbol{\Sigma}}^{(5)}$ is a block-diagonal matrix with blocks of size $d \times d = 2 \times 2$ (cf. Figure 5.10c). Therefore, in this particular example and in agreement with Lemma 5.10, it holds that $\bar{\boldsymbol{\Sigma}}^{(5)} = \bar{\boldsymbol{\Sigma}}$.

As another example, we want to look at the sequence

$$\begin{aligned}
\bar{\boldsymbol{\Sigma}}^{(1)} &= \mathcal{H}_{8,1}^{1,1}\bar{\boldsymbol{\Sigma}}^{(0)},\\
\bar{\boldsymbol{\Sigma}}^{(2)} &= \bar{\boldsymbol{\Sigma}}^{(1)}\mathcal{H}_{1,8}^{1,2},\\
\bar{\boldsymbol{\Sigma}}^{(3)} &= \mathcal{H}_{8,2}^{2,2}\bar{\boldsymbol{\Sigma}}^{(2)},\\
\bar{\boldsymbol{\Sigma}}^{(4)} &= \bar{\boldsymbol{\Sigma}}^{(3)}\mathcal{G}_{2,3}^{2,4}, \text{ and}\\
\bar{\boldsymbol{\Sigma}}^{(5)} &= \bar{\boldsymbol{\Sigma}}^{(4)}\mathcal{G}_{2,4}^{2,5},
\end{aligned}$$

which starts just as the bidiagonalization method based on Householder transformations that leads to an upper bidiagonal matrix. Figure 5.11 illustrates the last three decomposition steps of this sequence of applications of Householder reflections and Givens rotations. The elements central to the subsequent Givens rotation, $\bar{\sigma}_{2,3}^{(3)}$ and $\bar{\sigma}_{2,4}^{(3)}$ of $\bar{\boldsymbol{\Sigma}}^{(3)}$ as well as $\bar{\sigma}_{2,4}^{(4)}$ and $\bar{\sigma}_{2,5}^{(4)}$ of $\bar{\boldsymbol{\Sigma}}^{(4)}$, are highlighted again (cf. Figures 5.11a and 5.11b). We can easily see that, if $K = 2$, $d = 2$, $M = 4$, and $N = 4$, the equation $\bar{\boldsymbol{\Sigma}}^{(5)} = \bar{\boldsymbol{\Sigma}}$ also holds for this example's resulting matrix (cf. Figure 5.11c).

In this section, we have, by means of selected examples, examined strategies for solving the relaxed interference alignment decomposition problem. Eventually, we have seen that, at least for our examples, appropriate combinations of applications of Householder reflections and Givens rotations allow the factorization of the global channel matrix $\boldsymbol{H}$ in conformance with Equation 5.4 in a finite number of decomposition steps. As required by Definition 5.6, the resulting global interference alignment matrix $\boldsymbol{\Sigma} = \bar{\boldsymbol{\Sigma}}(1:Kd, 1:Kd)$ then is of block-diagonal form. In the following chapter, we will generalize and algorithmize these approaches and apply various combinations of Householder and Givens transformations with distinct properties.

Chapter 6

Direct solution to the relaxed matrix factorization problem

Based on the techniques developed in Chapter 5, we want to discuss an actual algorithm that applies Householder reflections and Givens rotations to find a direct solution to our problem of factorizing the global channel matrix $\boldsymbol{H}$ into a product of two unitary matrices and the extended global interference alignment matrix $\bar{\boldsymbol{\Sigma}}$. We will introduce and evaluate various variants of a constructive algorithm with different advantages and disadvantages for solving the relaxed interference alignment decomposition problem.

First, in Sections 6.1.1 to 6.1.3, we will look at algorithm variants that adapt the tridiagonalization method briefly outlined in Chapter 5 for the relaxed interference alignment decomposition problem. The next variant of the algorithm, as presented in Section 6.1.4, constructs $\bar{\boldsymbol{\Sigma}}$ such that the diagonal of its submatrix $\boldsymbol{\Sigma}$ consists of full blocks. In Section 6.1.5, a variant that produces bidiagonal blocks is explained. Finally, in Section 6.2, we will prove the constructive algorithm's correctness and examine some of its characteristics.

6.1 Algorithm description

In Chapter 5, we referred to the tridiagonalization method based on Householder transformations (cf. Figures 5.5 and 5.8). We also explained how to utilize Givens rotations after a sequence of Householder reflections in order to obtain a block-diagonal submatrix (cf. Figures 5.10 and 5.11).

In this section, we want to derive, illustrate, and analyze variants of a constructive algorithm that combine those two approaches. The result of these algorithm variants is the extended global interference alignment matrix $\bar{\boldsymbol{\Sigma}}$ with the blocks of its block-diagonal submatrix $\boldsymbol{\Sigma}$ being in tridiagonal form. Similar to Section 5.3, the special structures of the extended global postcoding and precoding matrices $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ (cf. Definitions 5.1 and 5.3) are not taken into consideration, i.e., we expect the products of the Householder and Givens

matrices used for transforming $\boldsymbol{H}$ to $\bar{\boldsymbol{\Sigma}}$ only to yield the unitary matrices $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ (cf. Lemma 5.8 and Corollary 5.9). Consequently, our algorithm variants are supposed to solve the relaxed interference alignment decomposition problem (cf. Definition 5.6).

6.1.1 Basic tridiagonal blocks variant

CRIAD$^{(\mathrm{t}'')}$, the algorithm variant we want to discuss first, closely follows the concepts explained in Chapter 5 ("CRIAD" is an acronym for "Constructive Relaxed Interference Alignment Decomposition", while "t" stands for a tridiagonal blocks variant). In each decomposition step, exactly one Householder or Givens transformation is performed (cf. Definition 5.7). Elements are annihilated column- and row-wise from top-left to bottom-right such that all but the last $d \times d$ main diagonal block of the resulting $Kd \times Kd$ matrix $\boldsymbol{\Sigma} = \bar{\boldsymbol{\Sigma}}(1\!:\!Kd, 1\!:\!Kd)$ are in tridiagonal form.

To achieve this, the following rough strategy is pursued for each of these blocks in the respective columns and rows of the current $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{0, 1, \ldots, \Lambda - 1\}$, where $\Lambda \in \mathbb{N}_0$ denotes the total number of decomposition steps (cf. Definition 5.7):

1. Annihilate all possible elements below the block's diagonal in the first (leftmost) column by premultiplying a proper Householder matrix without reversing previous element annihilations.

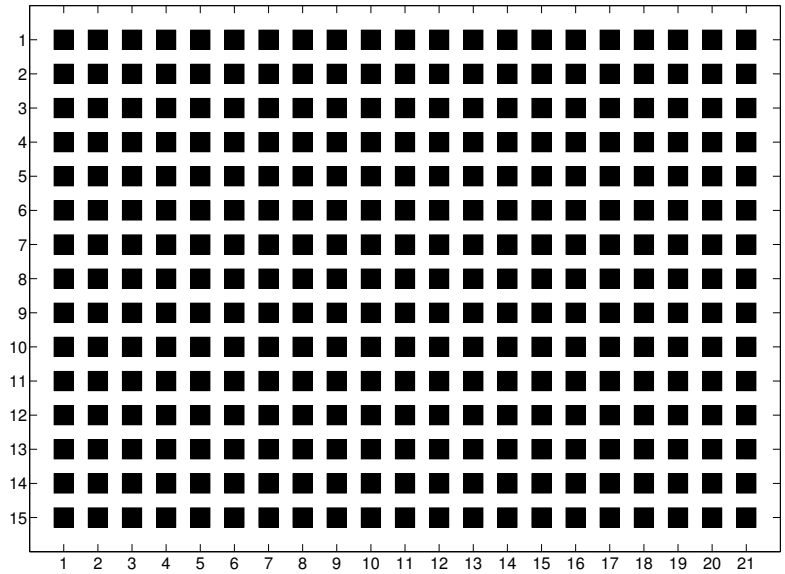
2. Annihilate all possible elements right of the block's diagonal in the first (topmost) row by postmultiplying an appropriate Householder matrix without reversing previous element annihilations.

3. Repeat the previous two decomposition steps for the second, third, …, $(d-1)^{\text{th}}$ column and row, respectively.

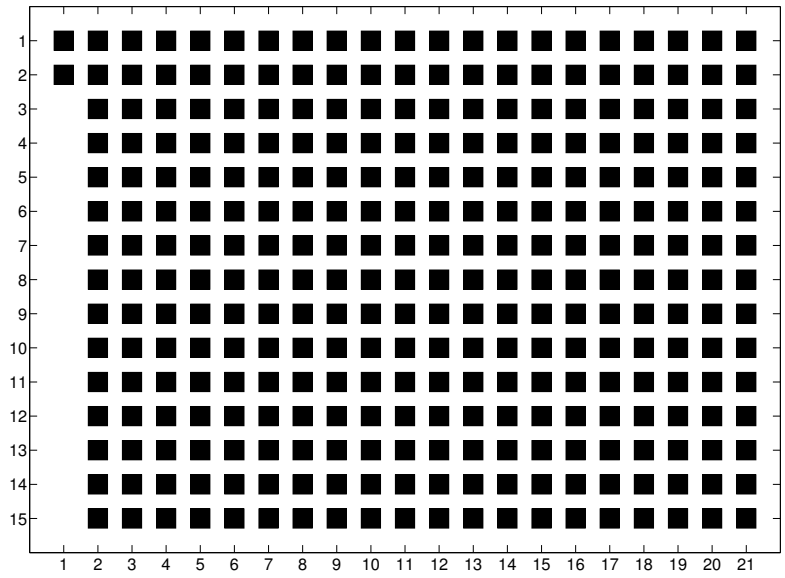
4. Annihilate all possible elements below the block in the last (rightmost) column by premultiplying a series of proper Givens matrices without reversing previous element annihilations.

5. Annihilate all possible elements right of the block in the last (lowermost) row by postmultiplying a series of appropriate Givens matrices without reversing previous element annihilations.

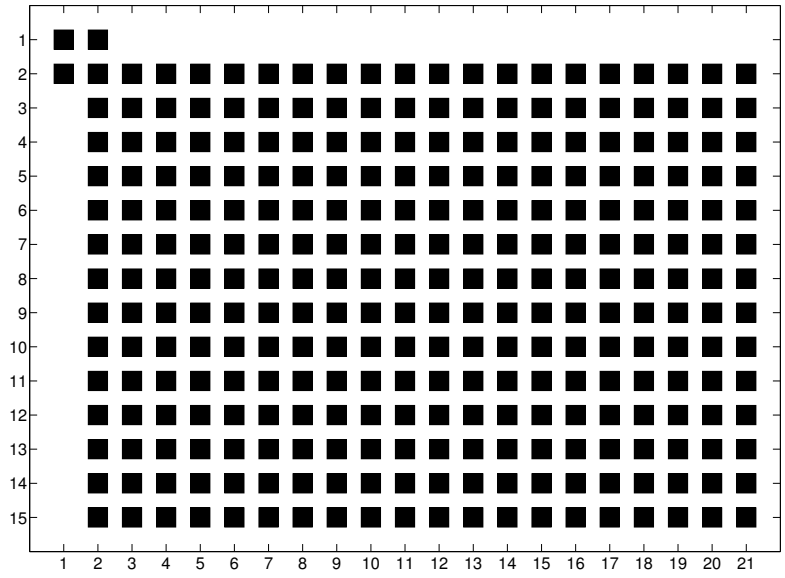
Figure 6.1 illustrates this strategy with the help of an example for $K = 3$, $d = 3$, $M = 7$, and $N = 5$. Similar to Chapter 5, each (potentially) nonzero matrix element is represented by a small black square. Strictly zero elements of the matrix appear as white space. The global channel matrix $\boldsymbol{H}$ and thus the initial matrix $\bar{\boldsymbol{\Sigma}}^{(0)}$ (cf. Definition 5.7) is shown in Figure 6.1a. The first three phases of the strategy outlined above are depicted in Figures 6.1b to 6.1e for the first (top-left) $d \times d$ block of our example matrix. The elements



(a) $\bar{\boldsymbol{\Sigma}}^{(0)} := \boldsymbol{H}$

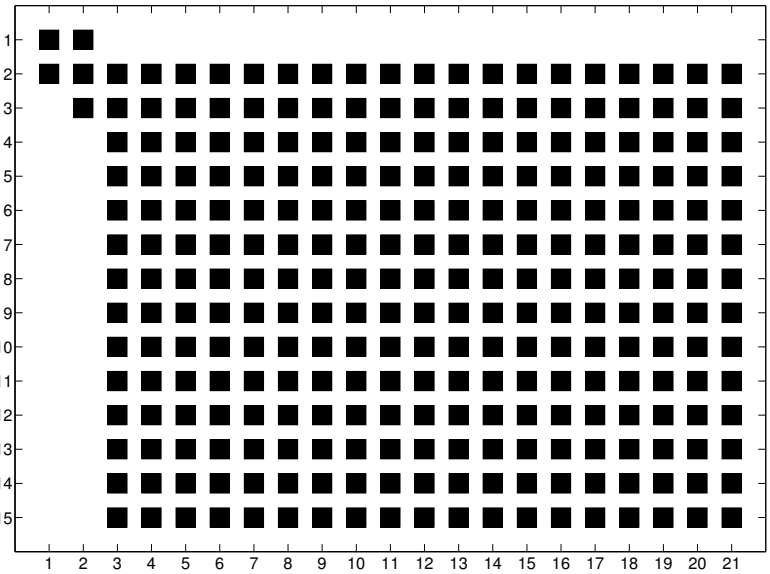


(b) $\bar{\boldsymbol{\Sigma}}^{(1)} = \mathscr{H}^{2,1}_{15,1}\bar{\boldsymbol{\Sigma}}^{(0)}$

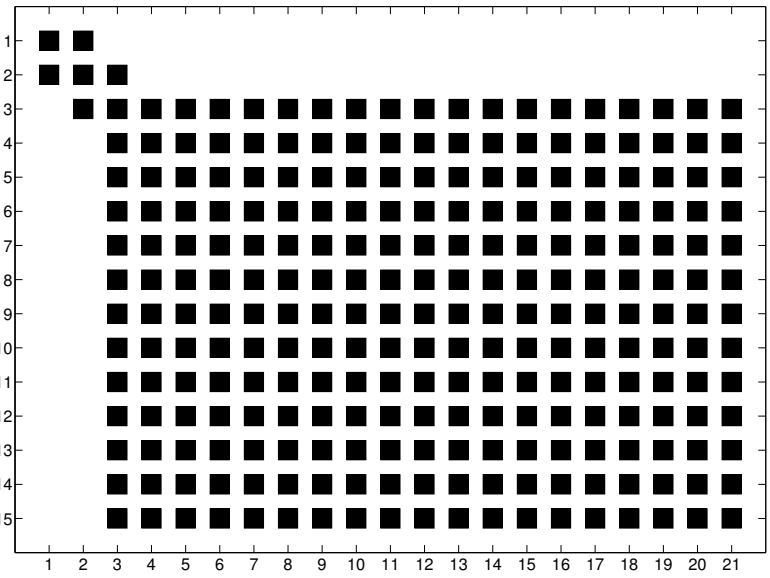


(c) $\bar{\boldsymbol{\Sigma}}^{(2)} = \bar{\boldsymbol{\Sigma}}^{(1)}\mathscr{H}^{1,2}_{1,21}$

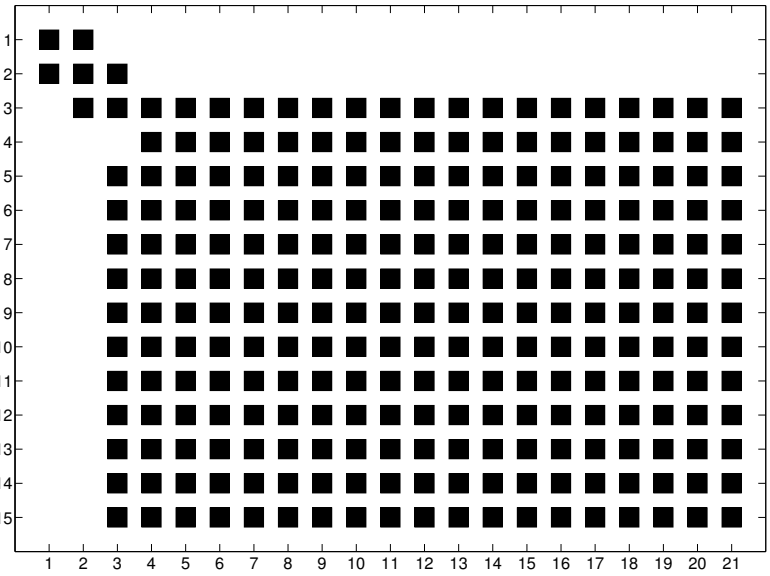
Figure 6.1: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 56\}$, of the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}'')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(d) $\bar{\boldsymbol{\Sigma}}^{(3)} = \mathcal{H}^{3,2}_{15,2}\bar{\boldsymbol{\Sigma}}^{(2)}$

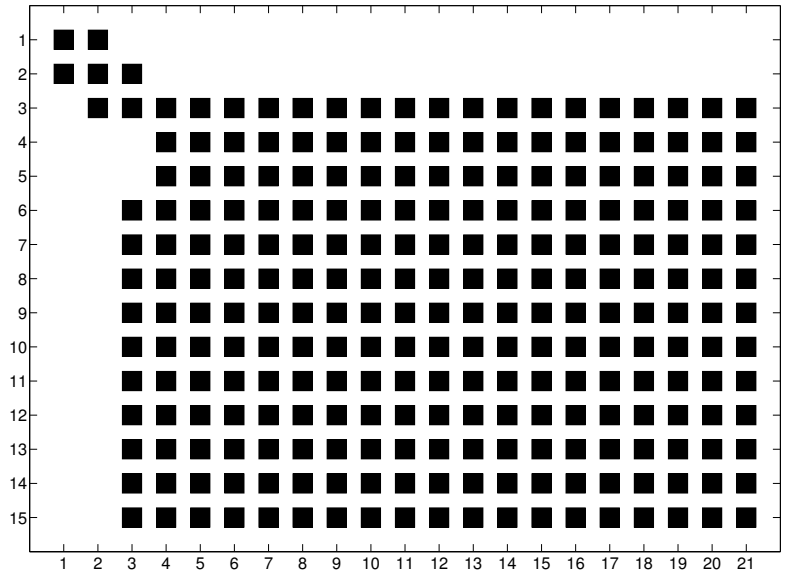


(e) $\bar{\boldsymbol{\Sigma}}^{(4)} = \bar{\boldsymbol{\Sigma}}^{(3)}\mathcal{H}^{2,3}_{2,21}$

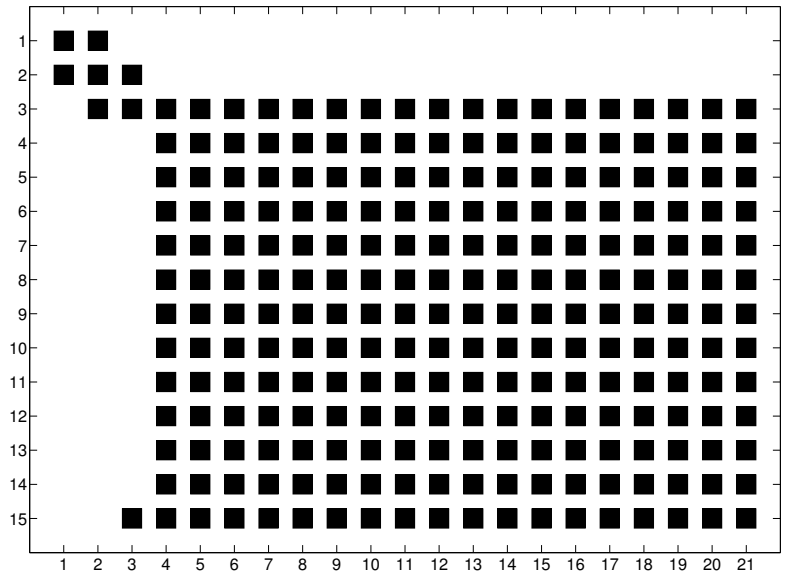


(f) $\bar{\boldsymbol{\Sigma}}^{(5)} = \mathcal{G}^{5,3}_{4,3}\bar{\boldsymbol{\Sigma}}^{(4)}$

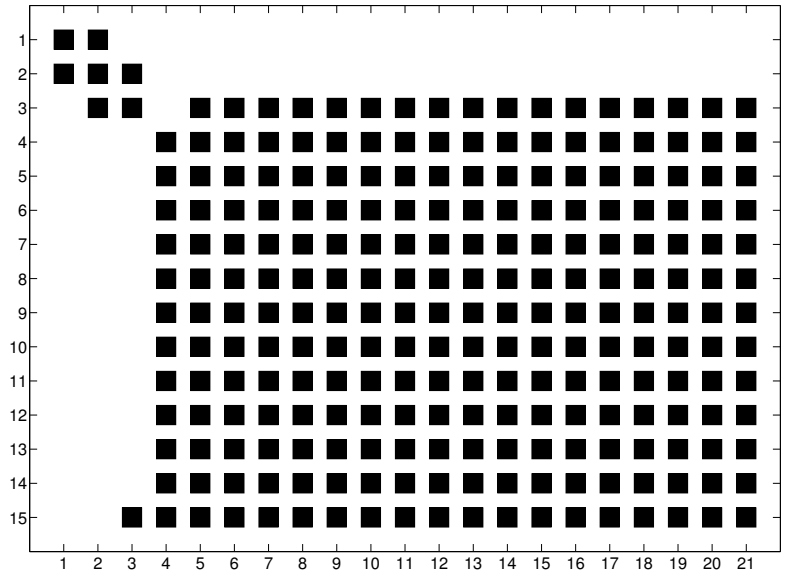
Figure 6.1: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 56\}$, of the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}'')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(g) $\bar{\mathbf{\Sigma}}^{(6)} = \mathcal{G}_{5,3}^{6,3}\bar{\mathbf{\Sigma}}^{(5)}$

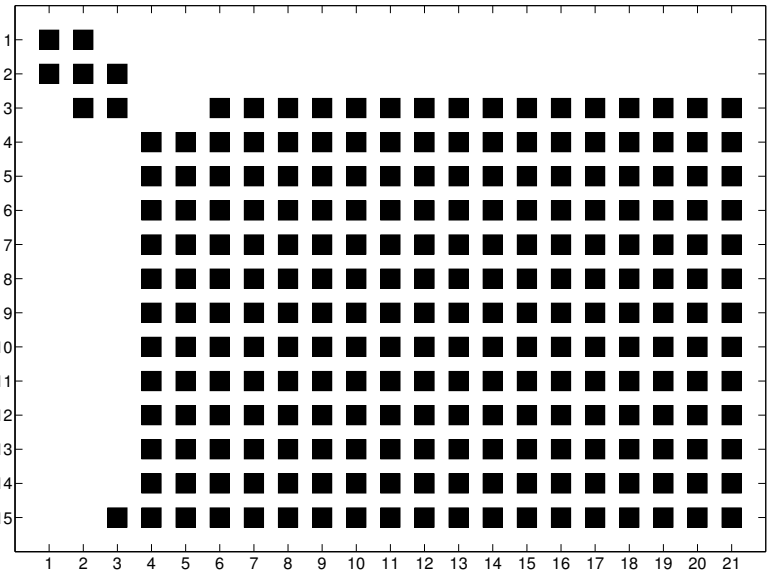


(h) $\bar{\mathbf{\Sigma}}^{(15)} = \mathcal{G}_{14,3}^{15,3}\bar{\mathbf{\Sigma}}^{(14)}$

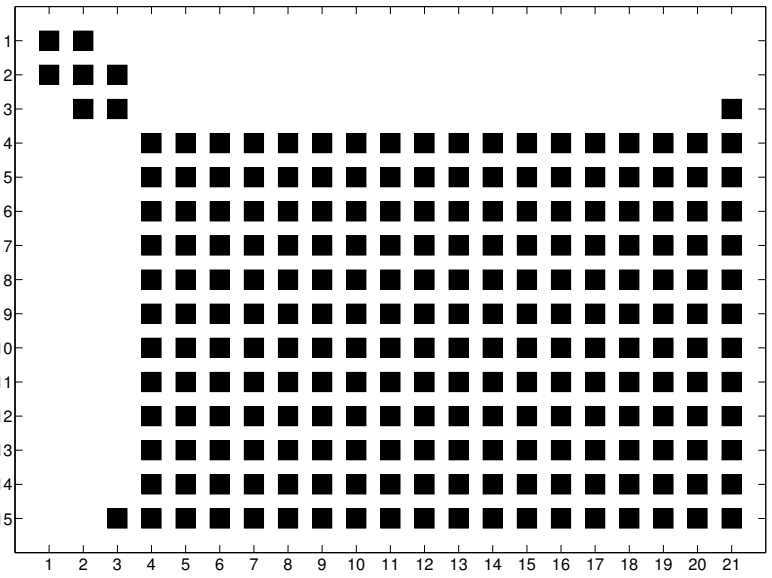


(i) $\bar{\mathbf{\Sigma}}^{(16)} = \bar{\mathbf{\Sigma}}^{(15)}\mathcal{G}_{3,4}^{3,5}$

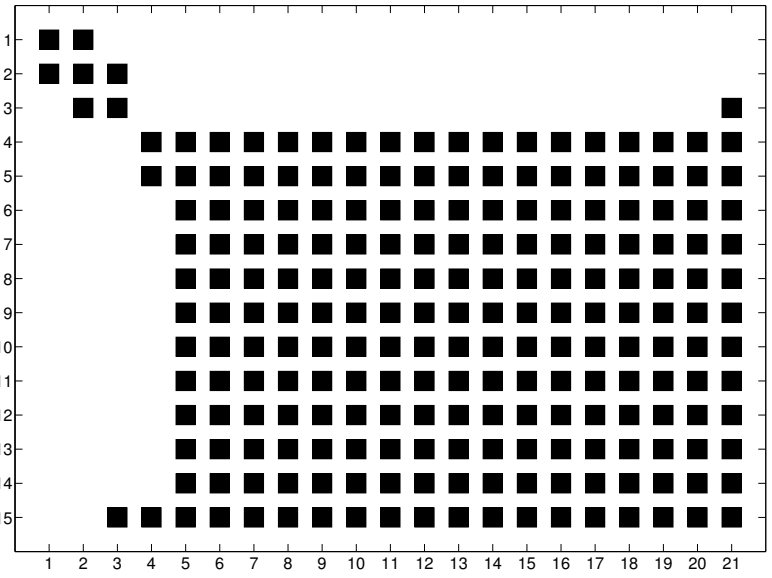
Figure 6.1: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 56\}$, of the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}'')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(j) $\bar{\boldsymbol{\Sigma}}^{(17)} = \bar{\boldsymbol{\Sigma}}^{(16)} \mathcal{G}_{3,5}^{3,6}$

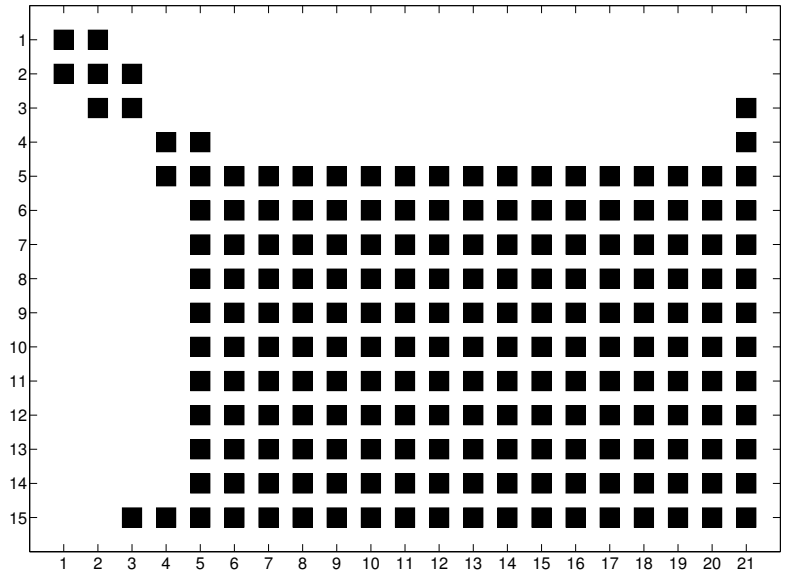


(k) $\bar{\boldsymbol{\Sigma}}^{(32)} = \bar{\boldsymbol{\Sigma}}^{(31)} \mathcal{G}_{3,20}^{3,21}$

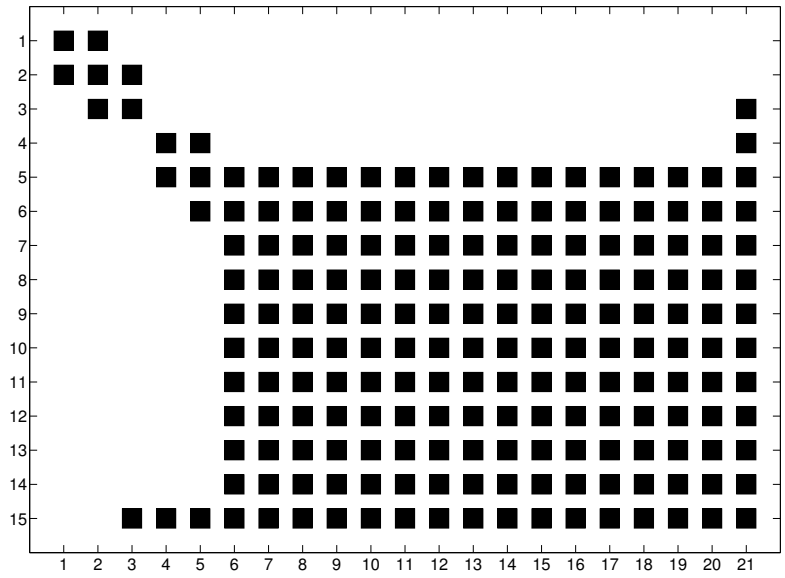


(l) $\bar{\boldsymbol{\Sigma}}^{(33)} = \mathcal{H}_{14,4}^{5,4} \bar{\boldsymbol{\Sigma}}^{(32)}$

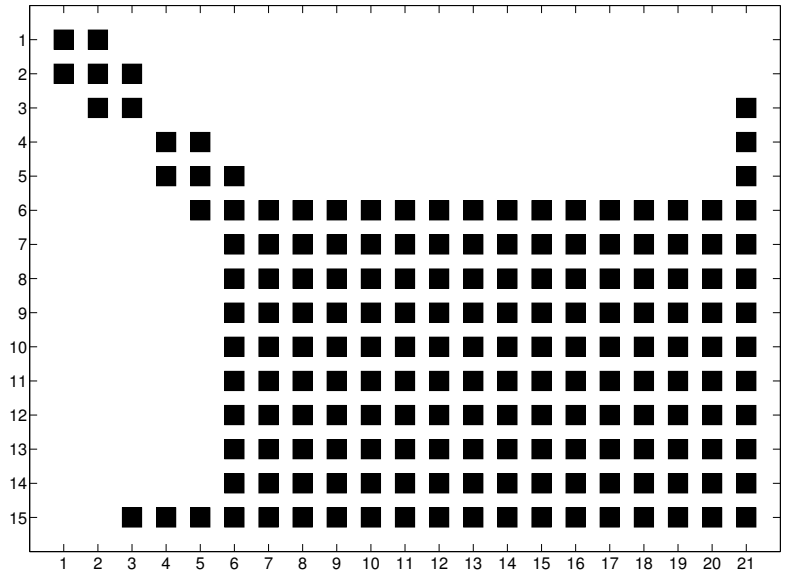
Figure 6.1: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 56\}$, of the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}'')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(m) $\bar{\mathbf{\Sigma}}^{(34)} = \bar{\mathbf{\Sigma}}^{(33)} \mathcal{H}^{4,5}_{4,20}$

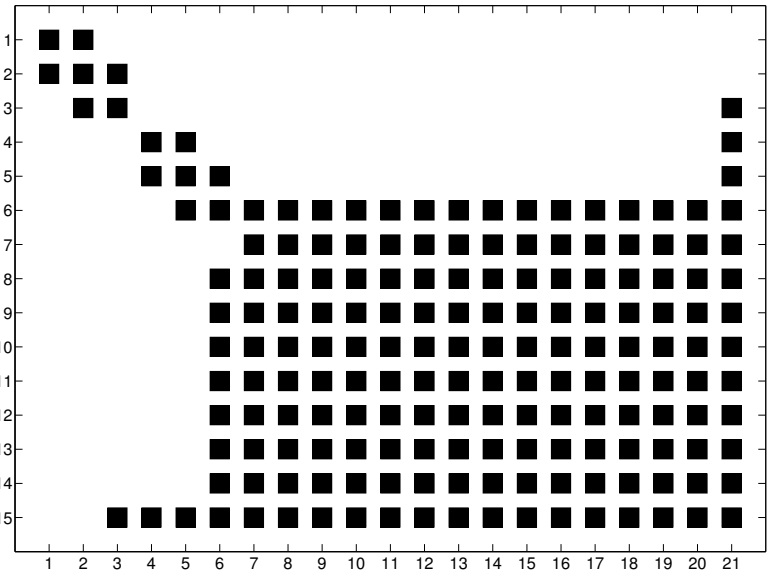


(n) $\bar{\mathbf{\Sigma}}^{(35)} = \mathcal{H}^{6,5}_{14,5} \bar{\mathbf{\Sigma}}^{(34)}$

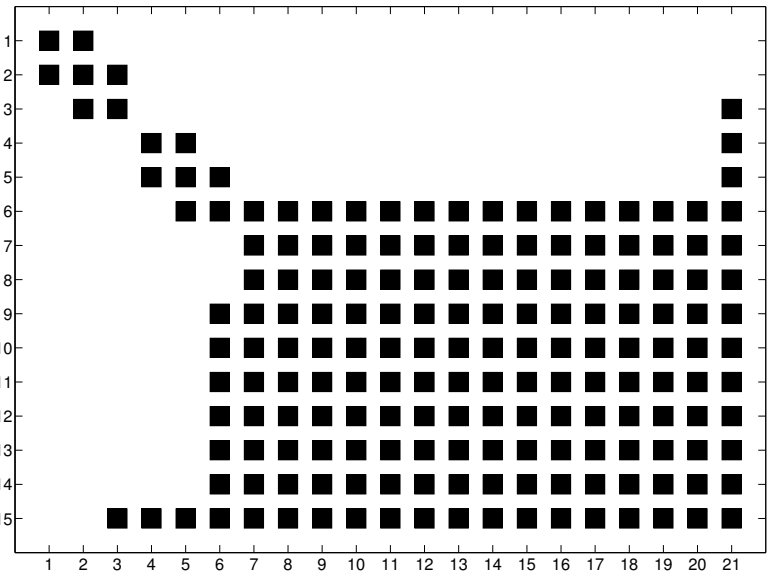


(o) $\bar{\mathbf{\Sigma}}^{(36)} = \bar{\mathbf{\Sigma}}^{(35)} \mathcal{H}^{5,6}_{5,20}$

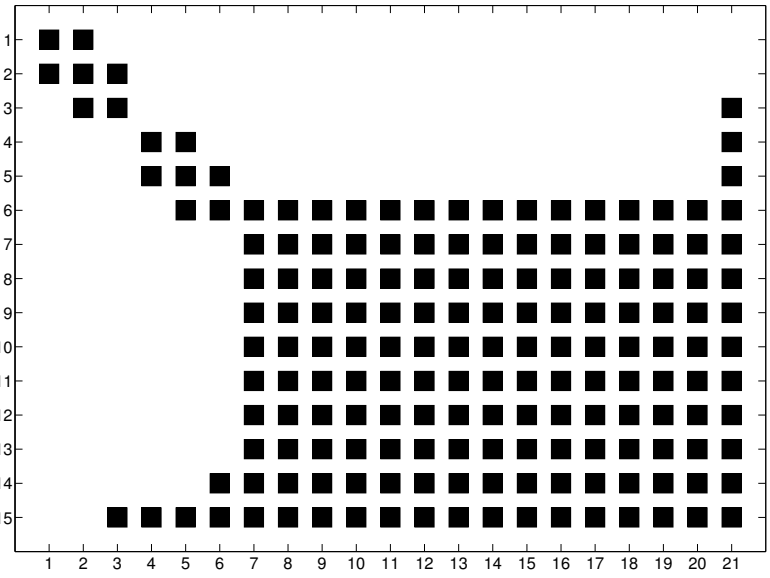
Figure 6.1: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 56\}$, of the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}'')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(p) $\bar{\boldsymbol{\Sigma}}^{(37)} = \mathcal{G}_{7,6}^{8,6}\bar{\boldsymbol{\Sigma}}^{(36)}$

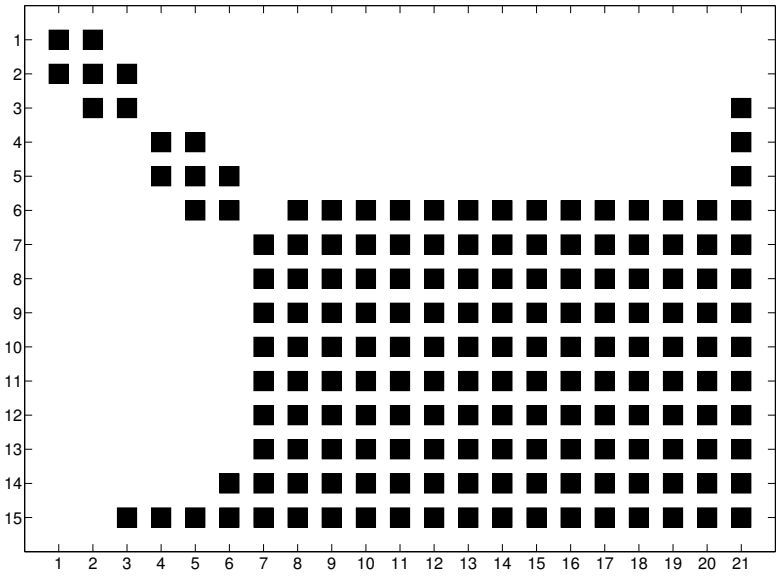


(q) $\bar{\boldsymbol{\Sigma}}^{(38)} = \mathcal{G}_{8,6}^{9,6}\bar{\boldsymbol{\Sigma}}^{(37)}$

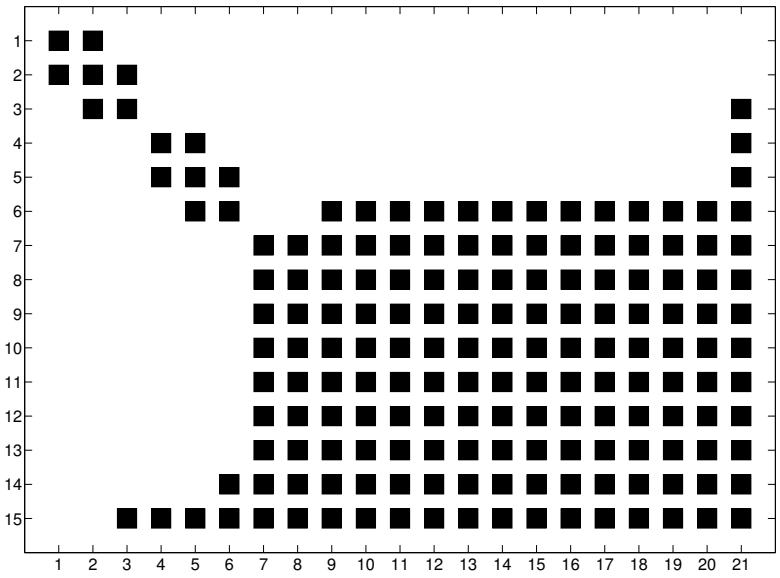


(r) $\bar{\boldsymbol{\Sigma}}^{(43)} = \mathcal{G}_{13,6}^{14,6}\bar{\boldsymbol{\Sigma}}^{(42)}$

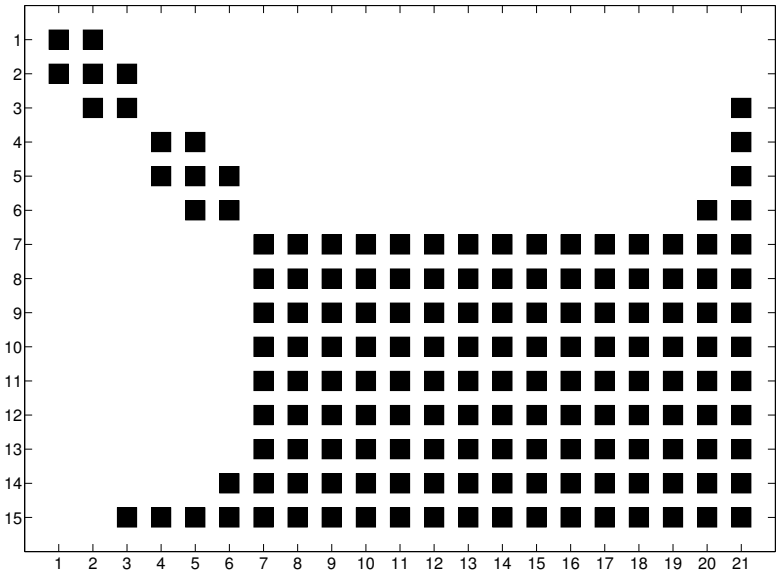
Figure 6.1: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 56\}$, of the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}'')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(s) $\bar{\mathbf{\Sigma}}^{(44)} = \bar{\mathbf{\Sigma}}^{(43)} \mathcal{G}_{6,7}^{6,8}$



(t) $\bar{\mathbf{\Sigma}}^{(45)} = \bar{\mathbf{\Sigma}}^{(46)} \mathcal{G}_{6,8}^{6,9}$



(u) $\bar{\mathbf{\Sigma}} = \bar{\mathbf{\Sigma}}^{(56)} = \bar{\mathbf{\Sigma}}^{(55)} \mathcal{G}_{6,19}^{6,20}$

Figure 6.1: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 56\}$, of the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}'')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.

in the first two columns and rows below and right of the block's diagonal are annihilated by the sequence

$$\begin{aligned}
\bar{\mathbf{\Sigma}}^{(1)} &= \mathcal{H}^{2,1}_{15,1}\bar{\mathbf{\Sigma}}^{(0)},\\
\bar{\mathbf{\Sigma}}^{(2)} &= \bar{\mathbf{\Sigma}}^{(1)}\mathcal{H}^{1,2}_{1,21},\\
\bar{\mathbf{\Sigma}}^{(3)} &= \mathcal{H}^{3,2}_{15,2}\bar{\mathbf{\Sigma}}^{(2)}, \text{ and}\\
\bar{\mathbf{\Sigma}}^{(4)} &= \bar{\mathbf{\Sigma}}^{(3)}\mathcal{H}^{2,3}_{2,21}
\end{aligned}$$

of Householder matrix applications.

Figures 6.1f to 6.1h demonstrate three decomposition steps that annihilate elements in the last column below the first $d \times d$ block by applying the sequence

$$\begin{aligned}
\bar{\mathbf{\Sigma}}^{(5)} &= \mathcal{G}^{5,3}_{4,3}\bar{\mathbf{\Sigma}}^{(4)},\\
\bar{\mathbf{\Sigma}}^{(6)} &= \mathcal{G}^{6,3}_{5,3}\bar{\mathbf{\Sigma}}^{(5)},\\
\bar{\mathbf{\Sigma}}^{(7)} &= \mathcal{G}^{7,3}_{6,3}\bar{\mathbf{\Sigma}}^{(6)},\\
&\vdots\\
\bar{\mathbf{\Sigma}}^{(14)} &= \mathcal{G}^{14,3}_{13,3}\bar{\mathbf{\Sigma}}^{(13)}, \text{ and}\\
\bar{\mathbf{\Sigma}}^{(15)} &= \mathcal{G}^{15,3}_{14,3}\bar{\mathbf{\Sigma}}^{(14)}
\end{aligned}$$

of Givens rotations, which corresponds to the fourth phase of our strategy. As we can see in Figures 6.1f to 6.1h, the nonzero element of the two elements relevant for each of these applications of Givens matrices is shifted down to element $\bar{\sigma}^{(15)}_{15,3}$ in the last row of matrix $\bar{\mathbf{\Sigma}}^{(15)}$, i.e., almost all elements below the block, with the exception of element $\bar{\sigma}^{(15)}_{15,3}$, are annihilated.

The fifth phase of our strategy is accomplished by applying the sequence

$$\begin{aligned}
\bar{\mathbf{\Sigma}}^{(16)} &= \bar{\mathbf{\Sigma}}^{(15)}\mathcal{G}^{3,5}_{3,4},\\
\bar{\mathbf{\Sigma}}^{(17)} &= \bar{\mathbf{\Sigma}}^{(16)}\mathcal{G}^{3,6}_{3,5},\\
\bar{\mathbf{\Sigma}}^{(18)} &= \bar{\mathbf{\Sigma}}^{(17)}\mathcal{G}^{3,7}_{3,6},\\
&\vdots\\
\bar{\mathbf{\Sigma}}^{(31)} &= \bar{\mathbf{\Sigma}}^{(30)}\mathcal{G}^{3,20}_{3,19}, \text{ and}\\
\bar{\mathbf{\Sigma}}^{(32)} &= \bar{\mathbf{\Sigma}}^{(31)}\mathcal{G}^{3,21}_{3,20}
\end{aligned}$$

of Givens transformations, which partly is illustrated in Figures 6.1i to 6.1k. Now, nearly all elements right of the top-left $d \times d$ block, except for element $\bar{\sigma}^{(32)}_{3,21}$ in the last column of matrix $\bar{\mathbf{\Sigma}}^{(32)}$, are annihilated (cf. Figure 6.1k). Therefore, the first block is finished and we can repeat our strategy for the second $d \times d$ block.

In a similar way to the initial row and column annihilations for the first block, we apply

Householder transformations to annihilate all possible elements in the first and second columns and rows below and right of the diagonal of the second block. The decomposition steps

$$\begin{aligned}
\bar{\mathbf{\Sigma}}^{(33)} &= \mathcal{H}_{14,4}^{5,4}\bar{\mathbf{\Sigma}}^{(32)},\\
\bar{\mathbf{\Sigma}}^{(34)} &= \bar{\mathbf{\Sigma}}^{(33)}\mathcal{H}_{4,20}^{4,5},\\
\bar{\mathbf{\Sigma}}^{(35)} &= \mathcal{H}_{14,5}^{6,5}\bar{\mathbf{\Sigma}}^{(34)}, \text{ and}\\
\bar{\mathbf{\Sigma}}^{(36)} &= \bar{\mathbf{\Sigma}}^{(35)}\mathcal{H}_{5,20}^{5,6}
\end{aligned}$$

are shown in Figures 6.1l to 6.1o. In contrast to the Householder-based element annihilations for the first block, the elements in the last row and last column of matrix $\bar{\mathbf{\Sigma}}^{(\lambda)}$, $\lambda \in \{33, 34, 35, 36\}$, cannot be annihilated to avoid fill-ins, i.e., to prevent zero elements from becoming nonzero again (cf. Chapter 5).

Subsequently, analogous to the first block, Givens rotations are applied for annihilating the elements in the last column and row below and right of the second block. While the sequence

$$\begin{aligned}
\bar{\mathbf{\Sigma}}^{(37)} &= \mathcal{G}_{7,6}^{8,6}\bar{\mathbf{\Sigma}}^{(36)},\\
\bar{\mathbf{\Sigma}}^{(38)} &= \mathcal{G}_{8,6}^{9,6}\bar{\mathbf{\Sigma}}^{(37)},\\
\bar{\mathbf{\Sigma}}^{(39)} &= \mathcal{G}_{9,6}^{10,6}\bar{\mathbf{\Sigma}}^{(38)},\\
&\vdots\\
\bar{\mathbf{\Sigma}}^{(42)} &= \mathcal{G}_{12,6}^{13,6}\bar{\mathbf{\Sigma}}^{(41)}, \text{ and}\\
\bar{\mathbf{\Sigma}}^{(43)} &= \mathcal{G}_{13,6}^{14,6}\bar{\mathbf{\Sigma}}^{(42)}
\end{aligned}$$

of Givens transformations, as in part shown in Figures 6.1p to 6.1r, performs element annihilations in the last column below the second block, the decomposition steps

$$\begin{aligned}
\bar{\mathbf{\Sigma}}^{(44)} &= \bar{\mathbf{\Sigma}}^{(43)}\mathcal{G}_{6,7}^{6,8},\\
\bar{\mathbf{\Sigma}}^{(45)} &= \bar{\mathbf{\Sigma}}^{(44)}\mathcal{G}_{6,8}^{6,9},\\
\bar{\mathbf{\Sigma}}^{(46)} &= \bar{\mathbf{\Sigma}}^{(45)}\mathcal{G}_{6,9}^{6,10},\\
&\vdots\\
\bar{\mathbf{\Sigma}}^{(55)} &= \bar{\mathbf{\Sigma}}^{(54)}\mathcal{G}_{6,18}^{6,19}, \text{ and}\\
\bar{\mathbf{\Sigma}}^{(56)} &= \bar{\mathbf{\Sigma}}^{(55)}\mathcal{G}_{6,19}^{6,20}
\end{aligned}$$

annihilate all possible elements in the last row right of the second $d \times d$ block. Three of these decomposition steps are visualized in Figures 6.1s to 6.1u.

Since the top-left $Kd \times Kd$ submatrix of $\bar{\mathbf{\Sigma}}^{(56)}$ has the form aimed at for solving the (relaxed) interference alignment decomposition problem (cf. Chapter 5), it holds that

$$\bar{\boldsymbol{\Sigma}} = \bar{\boldsymbol{\Sigma}}^{(56)}.$$

The third (last) $d \times d$ block is not yet in tridiagonal form. We could apply the same techniques we have used before for the first and second block again for the third block to bring it to tridiagonal form, but due to our already accomplished objective of solving the relaxed interference alignment decomposition problem, we will refrain from doing that.

The approach we just have studied by means of a specific example can be generalized for (almost) arbitrary values of K, d, M, and N (cf. Table 6.1). Algorithm 6.1 lists the CRIAD$^{(\mathrm{t}'')}$ tridiagonal blocks algorithm variant that utilizes this procedure for solving the relaxed interference alignment decomposition problem. While the variable $\bar{\boldsymbol{\Sigma}}$ always holds the most current $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{0, 1, \ldots, \Lambda\}$, during decomposition of the given global channel matrix $\boldsymbol{H}$, $\hat{\boldsymbol{U}}^{\mathrm{H}}$ is the product of all premultiplied Givens and Householder matrices, and $\hat{\boldsymbol{V}}$ is the product of all postmultiplied Givens and Householder matrices (cf. Lemma 5.8).

First, in Lines 2 to 4, the variable $\bar{\boldsymbol{\Sigma}}$ is initialized to $\boldsymbol{H} = \bar{\boldsymbol{\Sigma}}^{(0)}$, and $\hat{\boldsymbol{U}}^{\mathrm{H}}$ as well as $\hat{\boldsymbol{V}}$ are initialized to identity matrices of appropriate sizes. Then, in Lines 6 and 7, the thresholds τ_1 and τ_2 are computed, where τ_1 and τ_2 define the lowermost row and the rightmost column to be included in the element annihilation process, respectively. This algorithm variant considers the whole $KN \times KM$ matrix for element annihilations, but this will be different for other algorithm variants we will discuss later.

Next, in Lines 9 to 43, the algorithm considers one $d \times d$ block κ of the $Kd \times Kd$ submatrix $\boldsymbol{\Sigma}$ of $\bar{\boldsymbol{\Sigma}}$ at a time, i.e., in the body of the outer loop all required element annihilations for exactly one $d \times d$ block are performed. Let us examine this loop body in more detail. As we have seen in our example, we want to perform the appropriate Householder and Givens transformations in order to annihilate the elements below and right of the regarded $d \times d$ block.

To achieve this, we need to compute the correct indices for these operations. These indices are held by the variables φ, χ_1, χ_2, χ, and ψ. In Lines 10 and 11, the indices χ_1 and χ_2 are computed. χ_1 and χ_2 are the indices of the lowermost row and the rightmost column for column and row annihilations by Householder transformations. For the first (top-left) $d \times d$ block, χ_1 and χ_2 have their maximum values τ_1 and τ_2. For every other $d \times d$ block, the values of the variables χ_1 and χ_2 are decremented.

The loop in Lines 13 to 24 applies suitable Householder reflections to the columns and rows of $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$ which contain the current $d \times d$ block, one column and row at a time. In Line 14, the index ψ of the current column and the current row for element annihilations is computed. Subsequently, in Line 15, the index φ is calculated. φ identifies the topmost row for column annihilations and the leftmost column for row annihilations. It is the index of element z_1 of vector $\boldsymbol{z}$ in Definition 3.1. Hence, that element is not equal to zero after multiplying an appropriate Householder matrix. To obtain the $d \times d$ block in tridiagonal form, it holds that $\varphi = \psi + 1$.

While in Lines 17 and 18 the proper elements of the current column are annihilated

Algorithm 6.1 Constructive algorithm for solving the relaxed interference alignment decomposition problem (basic tridiagonal blocks variant).

function CRIAD$^{(\mathrm{t}'')}(\boldsymbol{H}, K, d, M, N)$
  $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \boldsymbol{I}_{KN}$
  $\bar{\boldsymbol{\Sigma}} \leftarrow \boldsymbol{H}$
  $\hat{\boldsymbol{V}} \leftarrow \boldsymbol{I}_{KM}$

  $\tau_1 \leftarrow KN$
  $\tau_2 \leftarrow KM$

  for all $\kappa \in \{1, 2, \ldots, K-1\}$ **do**
    $\chi_1 \leftarrow \tau_1 - \kappa + 1$
    $\chi_2 \leftarrow \tau_2 - \kappa + 1$

    for all $\delta \in \{1, 2, \ldots, d-1\}$ **do**
      $\psi \leftarrow (\kappa - 1)d + \delta$
      $\varphi \leftarrow \psi + 1$

      $\tilde{\boldsymbol{\mathcal{H}}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \varphi, \psi \rangle, \langle \chi_1, \psi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \tilde{\boldsymbol{\mathcal{H}}} \bar{\boldsymbol{\Sigma}}$
      $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \tilde{\boldsymbol{\mathcal{H}}} \hat{\boldsymbol{U}}^{\mathrm{H}}$

      $\tilde{\boldsymbol{\mathcal{H}}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \varphi \rangle, \langle \psi, \chi_2 \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \tilde{\boldsymbol{\mathcal{H}}}$
      $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \tilde{\boldsymbol{\mathcal{H}}}$
    end for

    $\psi \leftarrow \kappa d$

    for all $\chi \in \{\kappa d + 2, \kappa d + 3, \ldots, \chi_1\}$ **do**
      $\varphi \leftarrow \chi - 1$

      $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \chi, \psi \rangle, \langle \varphi, \psi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \mathcal{G} \bar{\boldsymbol{\Sigma}}$
      $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \mathcal{G} \hat{\boldsymbol{U}}^{\mathrm{H}}$
    end for

    for all $\chi \in \{\kappa d + 2, \kappa d + 3, \ldots, \chi_2\}$ **do**
      $\varphi \leftarrow \chi - 1$

      $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \chi \rangle, \langle \psi, \varphi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \mathcal{G}$
      $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \mathcal{G}$
    end for
  end for

  return $\langle \hat{\boldsymbol{U}}, \bar{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}} \rangle$
end function

by premultiplying a suitable Householder matrix, the corresponding row annihilation is performed by postmultiplying an applicable Householder matrix in Lines 21 and 22. The Householder matrices are obtained through the HOUSEHOLDER function listed in Algorithm 3.1. The lines

$$\begin{aligned}\tilde{\mathcal{H}} &\leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle\varphi, \psi\rangle, \langle\chi_1, \psi\rangle)\\ \bar{\boldsymbol{\Sigma}} &\leftarrow \tilde{\mathcal{H}}\bar{\boldsymbol{\Sigma}}\end{aligned}$$

are equivalent to our previous notation

$$\bar{\boldsymbol{\Sigma}}^{(\lambda+1)} = \mathcal{H}^{\varphi,\psi}_{\chi_1,\psi}\bar{\boldsymbol{\Sigma}}^{(\lambda)}.$$

Similarly, the lines

$$\begin{aligned}\tilde{\mathcal{H}} &\leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle\psi, \varphi\rangle, \langle\psi, \chi_2\rangle)\\ \bar{\boldsymbol{\Sigma}} &\leftarrow \bar{\boldsymbol{\Sigma}}\tilde{\mathcal{H}}\end{aligned}$$

have the same meaning as

$$\bar{\boldsymbol{\Sigma}}^{(\lambda+1)} = \bar{\boldsymbol{\Sigma}}^{(\lambda)}\mathcal{H}^{\psi,\varphi}_{\psi,\chi_2}.$$

In accordance with Lemma 5.8, $\hat{\boldsymbol{U}}^{\mathrm{H}}$ is premultiplied by the previously obtained Householder matrix $\tilde{\mathcal{H}}$ in Line 19 and $\hat{\boldsymbol{V}}$ is postmultiplied by the appropriate Householder matrix $\tilde{\mathcal{H}}$ in Line 23.

As we have seen in our example, a sequence of Givens rotations performs the final element annihilations for each $d \times d$ block after the Householder transformations have been applied. These Givens matrices are supposed to affect the column and row of $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$ equal to the last column and row of the $d \times d$ block. Therefore, in Line 26, the index ψ is set to the index κd of the last column and row of the current $d \times d$ block. In Lines 28 to 34, a sequence of Givens transformations is applied to elements of the respective column of $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$. Concretely, the elements with the row indices $\varphi = \kappa d + 1, \kappa d + 2, \ldots, \chi_1 - 1$ are annihilated. Likewise, in Lines 36 to 42, another sequence of Givens transformations is applied to elements of the respective row of $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$. Here, the elements with the column indices $\varphi = \kappa d + 1, \kappa d + 2, \ldots, \chi_2 - 1$ are annihilated.

The appropriate Givens matrices are obtained through the GIVENS function (cf. Algorithm 3.2). Analogous to our notation for Householder transformations, the lines

$$\begin{aligned}\mathcal{G} &\leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle\chi, \psi\rangle, \langle\varphi, \psi\rangle)\\ \bar{\boldsymbol{\Sigma}} &\leftarrow \mathcal{G}\bar{\boldsymbol{\Sigma}}\end{aligned}$$

correspond to

$$\bar{\boldsymbol{\Sigma}}^{(\lambda+1)} = \mathcal{G}^{\chi,\psi}_{\varphi,\psi}\bar{\boldsymbol{\Sigma}}^{(\lambda)},$$

and the lines

$$\begin{aligned}\boldsymbol{\mathcal{G}} &\leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle\psi, \chi\rangle, \langle\psi, \varphi\rangle)\\ \bar{\boldsymbol{\Sigma}} &\leftarrow \bar{\boldsymbol{\Sigma}}\boldsymbol{\mathcal{G}}\end{aligned}$$

have the same effect as

$$\bar{\boldsymbol{\Sigma}}^{(\lambda+1)} = \bar{\boldsymbol{\Sigma}}^{(\lambda)}\boldsymbol{\mathcal{G}}_{\psi,\varphi}^{\psi,\chi}.$$

Similar to Householder matrices, the applied Givens matrices as well are appropriately postmultiplied by $\hat{\boldsymbol{U}}^{\mathrm{H}}$ and premultiplied by $\hat{\boldsymbol{V}}$ to satisfy Lemma 5.8 (cf. Lines 33 and 41).

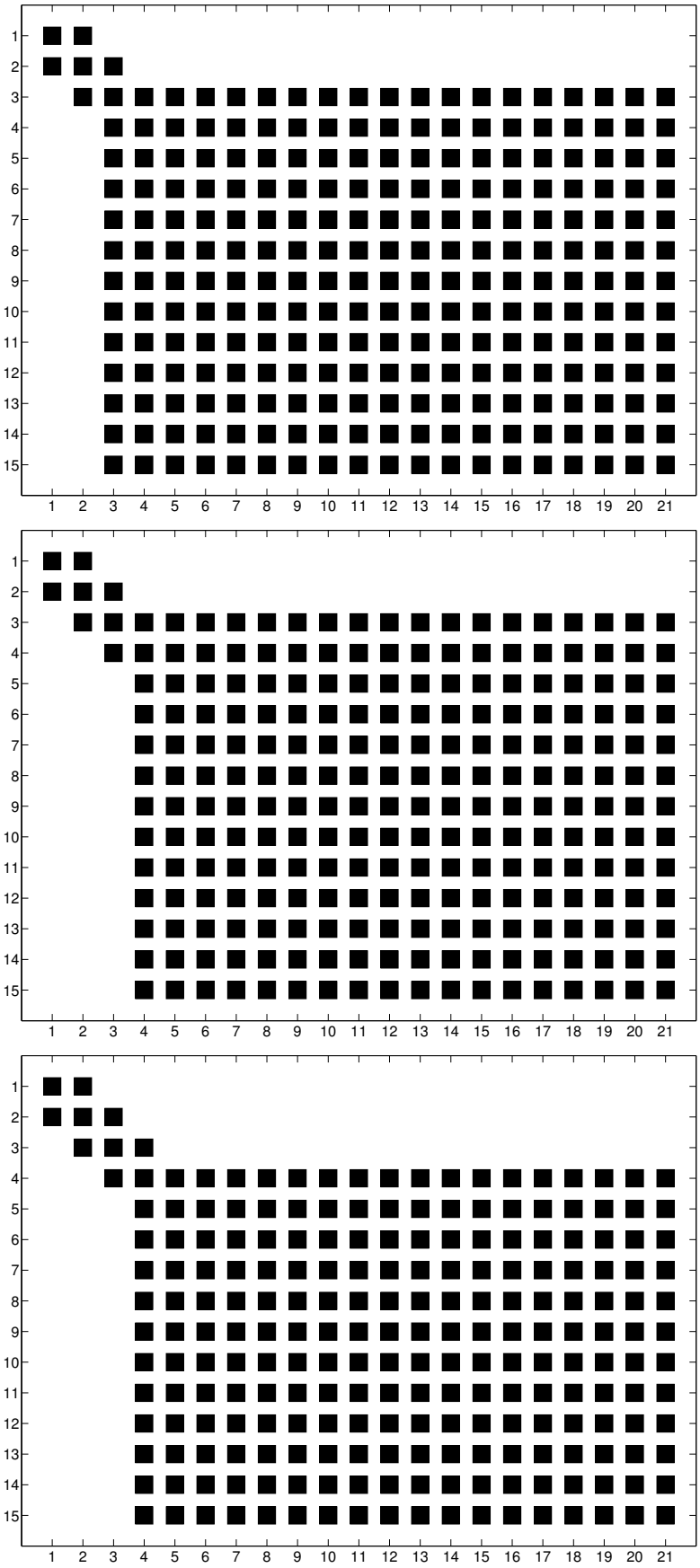
After the end of the outer loop in Line 43, $\bar{\boldsymbol{\Sigma}}$ has the form required by the (relaxed) interference alignment decomposition problem and all but the last (bottom-right) $d \times d$ block of the submatrix $\boldsymbol{\Sigma}$ of $\bar{\boldsymbol{\Sigma}}$ are tridiagonal matrices. To bring the last $d \times d$ block also to tridiagonal form, the meanwhile well-known techniques could be applied. But this is, as already previously stated, no requirement by the problem we want to solve. Finally, in Line 45, the triple $\langle\hat{\boldsymbol{U}}, \bar{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}}\rangle$ is returned.

6.1.2 Enhanced tridiagonal blocks variant

With the CRIAD$^{(\mathrm{t}'')}$ algorithm that we have discussed in the last section, we have already found a direct solution to the relaxed interference alignment decomposition problem. This first variant of our algorithm annihilates elements in the first $d-1$ columns and rows of $\bar{\boldsymbol{\Sigma}}$ below and right of each $d \times d$ block of the submatrix $\boldsymbol{\Sigma}$ of $\bar{\boldsymbol{\Sigma}}$ by utilizing Householder transformations. The element annihilations in the last column and row below and right of each of these $d \times d$ blocks are performed by applying Givens rotations. In general, this leads to significantly more multiplications with Givens matrices than with Householder matrices. Since Householder reflections in contrast to Givens rotations allow the annihilation of more than one matrix element in one transformation, we want to substitute as many Givens rotations as possible with Householder reflections in our next algorithm variant CRIAD$^{(\mathrm{t}')}$.

For the purpose of showing how we can achieve this, we are going to return to our example for $K = 3$, $d = 3$, $M = 7$, and $N = 5$ as illustrated in Figure 6.2. As we are already acquainted with the general mechanisms, we will concentrate on the differences between CRIAD$^{(\mathrm{t}'')}$ and CRIAD$^{(\mathrm{t}')}$. The first four decomposition steps are the same for both algorithm variants. Figure 6.2a visualizes the intermediary result after the fourth decomposition step. However, the next decomposition steps for annihilating the elements below the last column and right of the last row of the first (top-left) $d \times d$ block are different. Instead of the sequence

$$\begin{aligned}\bar{\boldsymbol{\Sigma}}^{(5)} &= \boldsymbol{\mathcal{G}}_{4,3}^{5,3}\bar{\boldsymbol{\Sigma}}^{(4)},\\ \bar{\boldsymbol{\Sigma}}^{(6)} &= \boldsymbol{\mathcal{G}}_{5,3}^{6,3}\bar{\boldsymbol{\Sigma}}^{(5)},\\ &\vdots\end{aligned}$$

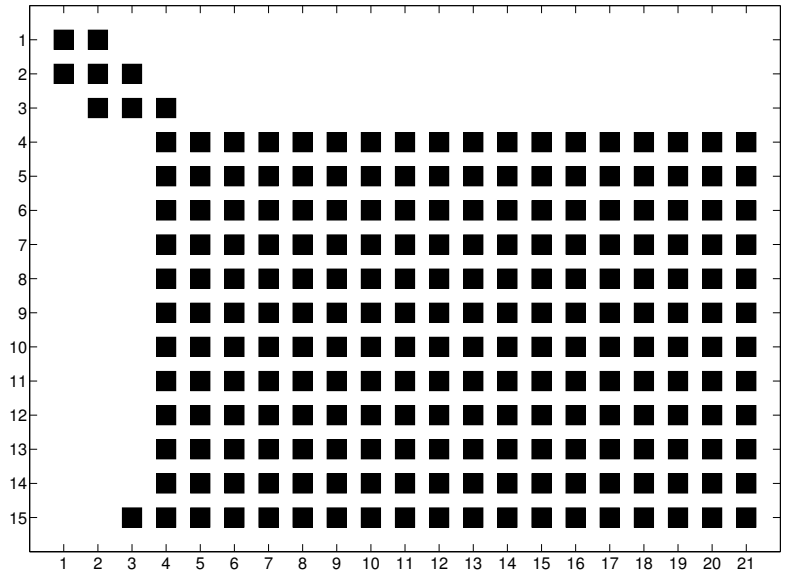


(a) $\bar{\boldsymbol{\Sigma}}^{(4)} = \bar{\boldsymbol{\Sigma}}^{(3)} \mathcal{H}^{2,3}_{2,21}$

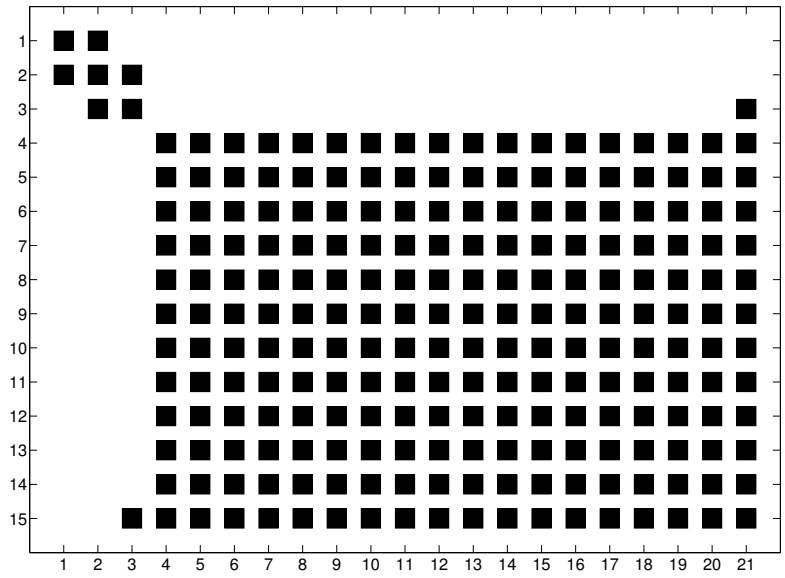
(b) $\bar{\boldsymbol{\Sigma}}^{(5)} = \mathcal{H}^{4,3}_{15,3} \bar{\boldsymbol{\Sigma}}^{(4)}$

(c) $\bar{\boldsymbol{\Sigma}}^{(6)} = \bar{\boldsymbol{\Sigma}}^{(5)} \mathcal{H}^{3,4}_{3,21}$

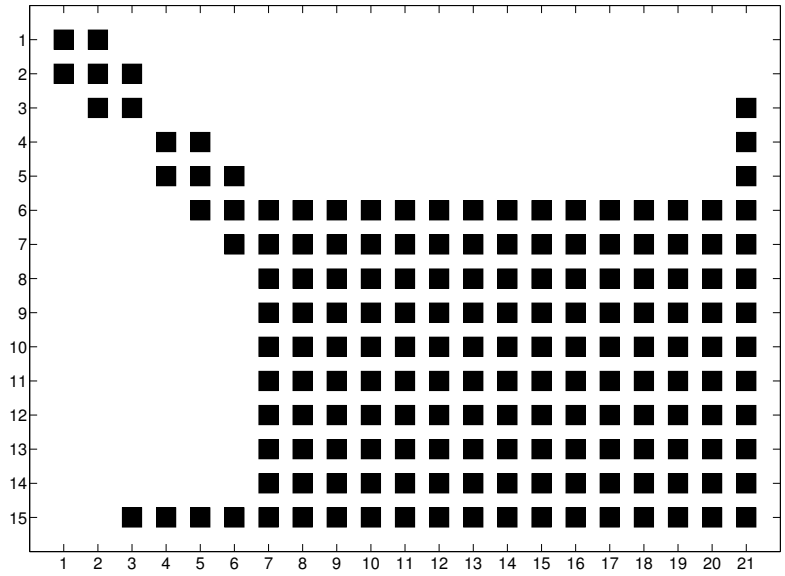
Figure 6.2: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 16\}$, of the enhanced tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(d) $\bar{\mathbf{\Sigma}}^{(7)} = \mathcal{G}_{4,3}^{15,3}\bar{\mathbf{\Sigma}}^{(6)}$

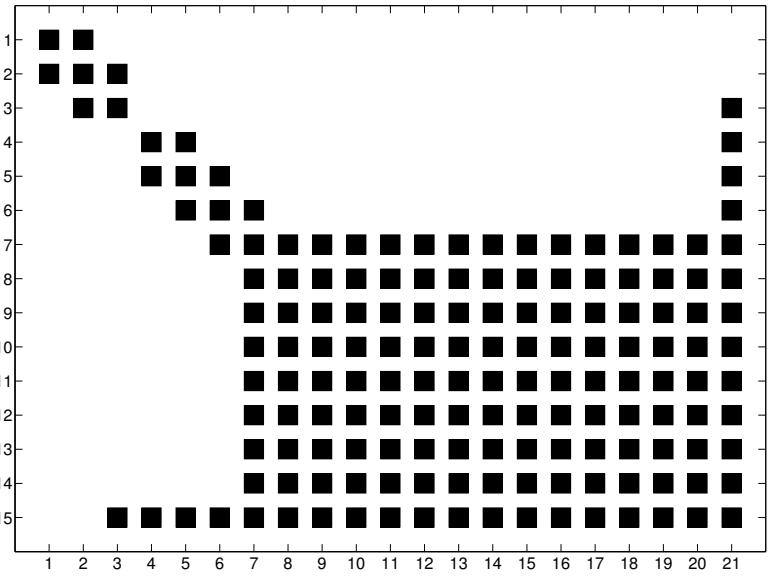


(e) $\bar{\mathbf{\Sigma}}^{(8)} = \bar{\mathbf{\Sigma}}^{(7)}\mathcal{G}_{3,4}^{3,21}$

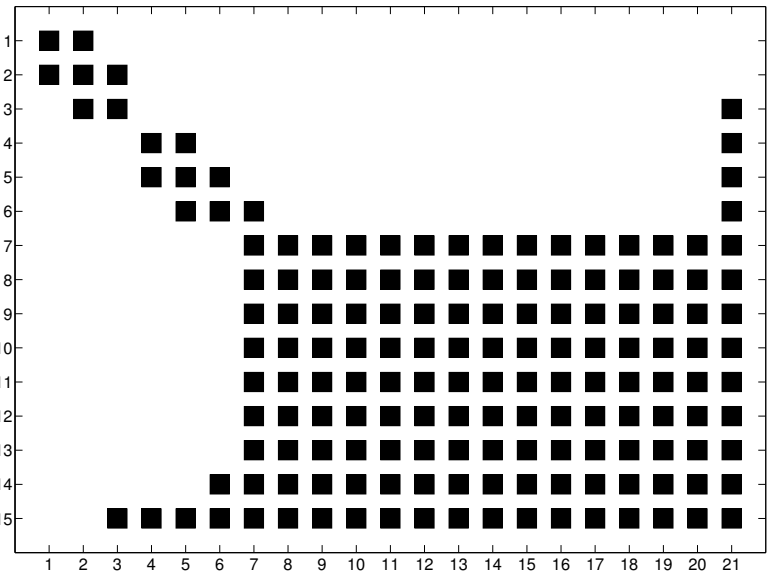


(f) $\bar{\mathbf{\Sigma}}^{(13)} = \mathcal{H}_{14,6}^{7,6}\bar{\mathbf{\Sigma}}^{(12)}$

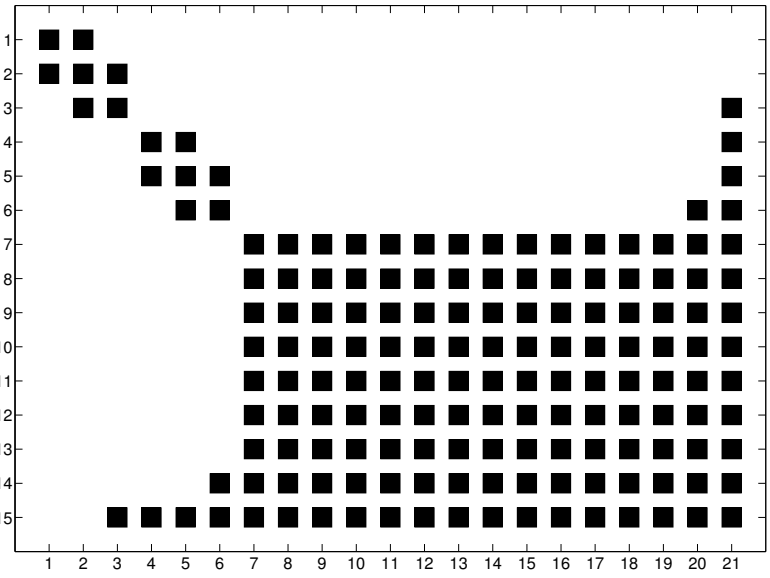
Figure 6.2: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 16\}$, of the enhanced tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(g) $\bar{\boldsymbol{\Sigma}}^{(14)} = \bar{\boldsymbol{\Sigma}}^{(13)} \mathscr{H}_{6,20}^{6,7}$



(h) $\bar{\boldsymbol{\Sigma}}^{(15)} = \mathcal{G}_{7,6}^{14,6} \bar{\boldsymbol{\Sigma}}^{(14)}$



(i) $\bar{\boldsymbol{\Sigma}} = \bar{\boldsymbol{\Sigma}}^{(16)} = \bar{\boldsymbol{\Sigma}}^{(15)} \mathcal{G}_{6,7}^{6,20}$

Figure 6.2: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 16\}$, of the enhanced tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t}')}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.

Algorithm 6.2 Constructive algorithm for solving the relaxed interference alignment decomposition problem (enhanced tridiagonal blocks variant).

function CRIAD$^{(\mathrm{t}')}(\boldsymbol{H}, K, d, M, N)$
  $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \boldsymbol{I}_{KN}$
  $\bar{\boldsymbol{\Sigma}} \leftarrow \boldsymbol{H}$
  $\hat{\boldsymbol{V}} \leftarrow \boldsymbol{I}_{KM}$

  $\tau_1 \leftarrow KN$
  $\tau_2 \leftarrow KM$

  for all $\kappa \in \{1, 2, \ldots, K-1\}$ **do**
    $\chi_1 \leftarrow \tau_1 - \kappa + 1$
    $\chi_2 \leftarrow \tau_2 - \kappa + 1$

    for all $\delta \in \{1, 2, \ldots, d\}$ **do**
      $\psi \leftarrow (\kappa - 1)d + \delta$
      $\varphi \leftarrow \psi + 1$

      $\tilde{\mathcal{H}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \varphi, \psi \rangle, \langle \chi_1, \psi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \tilde{\mathcal{H}} \bar{\boldsymbol{\Sigma}}$
      $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \tilde{\mathcal{H}} \hat{\boldsymbol{U}}^{\mathrm{H}}$

      $\tilde{\mathcal{H}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \varphi \rangle, \langle \psi, \chi_2 \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \tilde{\mathcal{H}}$
      $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \tilde{\mathcal{H}}$
    end for

    $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \chi_1, \psi \rangle, \langle \varphi, \psi \rangle)$
    $\bar{\boldsymbol{\Sigma}} \leftarrow \mathcal{G} \bar{\boldsymbol{\Sigma}}$
    $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \mathcal{G} \hat{\boldsymbol{U}}^{\mathrm{H}}$

    $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \chi_2 \rangle, \langle \psi, \varphi \rangle)$
    $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \mathcal{G}$
    $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \mathcal{G}$
  end for

  return $\langle \hat{\boldsymbol{U}}, \bar{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}} \rangle$
end function

$$
\begin{aligned}
\bar{\mathbf{\Sigma}}^{(15)} &= \mathcal{G}_{14,3}^{15,3}\bar{\mathbf{\Sigma}}^{(14)},\\
\bar{\mathbf{\Sigma}}^{(16)} &= \bar{\mathbf{\Sigma}}^{(15)}\mathcal{G}_{3,4}^{3,5},\\
\bar{\mathbf{\Sigma}}^{(17)} &= \bar{\mathbf{\Sigma}}^{(16)}\mathcal{G}_{3,5}^{3,6},\\
&\vdots\\
\bar{\mathbf{\Sigma}}^{(32)} &= \bar{\mathbf{\Sigma}}^{(31)}\mathcal{G}_{3,20}^{3,21},
\end{aligned}
$$

that is applied by CRIAD$^{(\mathrm{t}'')}$ (cf. Section 6.1.1), CRIAD$^{(\mathrm{t}')}$ utilizes the transformations

$$
\begin{aligned}
\bar{\mathbf{\Sigma}}^{(5)} &= \mathcal{H}_{15,3}^{4,3}\bar{\mathbf{\Sigma}}^{(4)},\\
\bar{\mathbf{\Sigma}}^{(6)} &= \bar{\mathbf{\Sigma}}^{(5)}\mathcal{H}_{3,21}^{3,4},\\
\bar{\mathbf{\Sigma}}^{(7)} &= \mathcal{G}_{4,3}^{15,3}\bar{\mathbf{\Sigma}}^{(6)}, \text{ and}\\
\bar{\mathbf{\Sigma}}^{(8)} &= \bar{\mathbf{\Sigma}}^{(7)}\mathcal{G}_{3,4}^{3,21},
\end{aligned}
$$

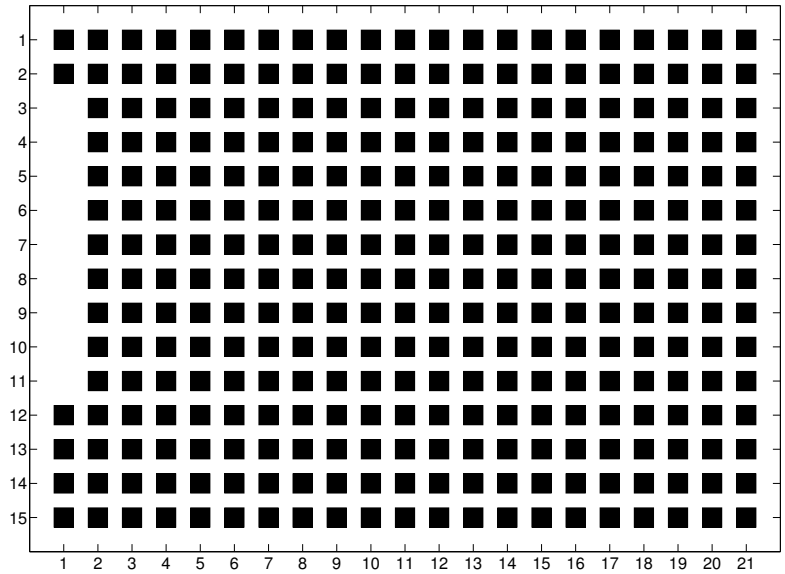
as shown in Figures 6.2b to 6.2e. We notice that, while CRIAD$^{(\mathrm{t}'')}$ needs 32 decomposition steps for annihilating the elements below and right of the first $d \times d$ block, CRIAD$^{(\mathrm{t}')}$ produces the same intermediary result after only eight decomposition steps. The element annihilations for the second $d \times d$ block follow the same scheme (cf. Figures 6.2f to 6.2i). Hence, the final result, which is visualized in Figure 6.2i, is obtained after only sixteen decomposition steps. CRIAD$^{(\mathrm{t}'')}$ needs in total 56 decomposition steps for the same result.

Algorithm 6.2 generalizes the approach we have outlined in our recent example. When we compare Algorithm 6.2 to Algorithm 6.1, we recognize many similarities between them. Only Line 13 and Lines 26 to 32 of Algorithm 6.2 are different from Algorithm 6.1. The inner loop in Lines 13 to 24 iterates over $\delta = 1, 2, \ldots, d$ instead of $\delta = 1, 2, \ldots, d-1$ as in Algorithm 6.1. Hence, elements below all columns and right of all rows of the current $d \times d$ block, which includes, in contrast to Algorithm 6.1, the last column and the last row, are annihilated by Householder transformations. After the sequence of $2d$ Householder matrix applications, only two Givens transformations are required per $d \times d$ block (cf. Lines 26 to 32), one to annihilate the element directly below the last column and one to annihilate the element directly right of the last row of the $d \times d$ block.

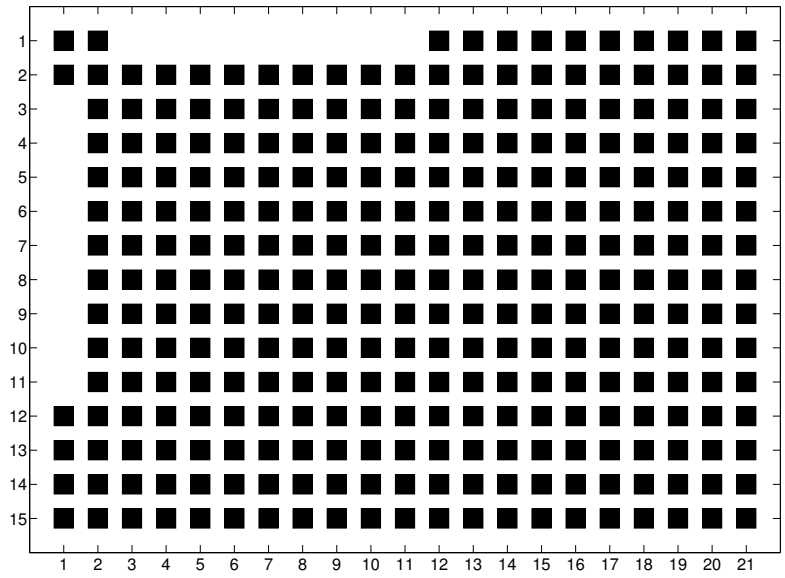
6.1.3 Further enhanced tridiagonal blocks variant

Both algorithm variants we have considered so far, CRIAD$^{(\mathrm{t}'')}$ and CRIAD$^{(\mathrm{t}')}$, annihilate more elements than necessary for solving the (relaxed) interference alignment decomposition problem. Since we only require the $Kd \times Kd$ submatrix $\mathbf{\Sigma}$ of the $KN \times KM$ matrix $\bar{\mathbf{\Sigma}}$ to be in block-diagonal form (cf. Definition 5.6), we may adjust the unitary transformations utilized by the CRIAD$^{(\mathrm{t}')}$ algorithm in order to minimize the number of elements that are annihilated.

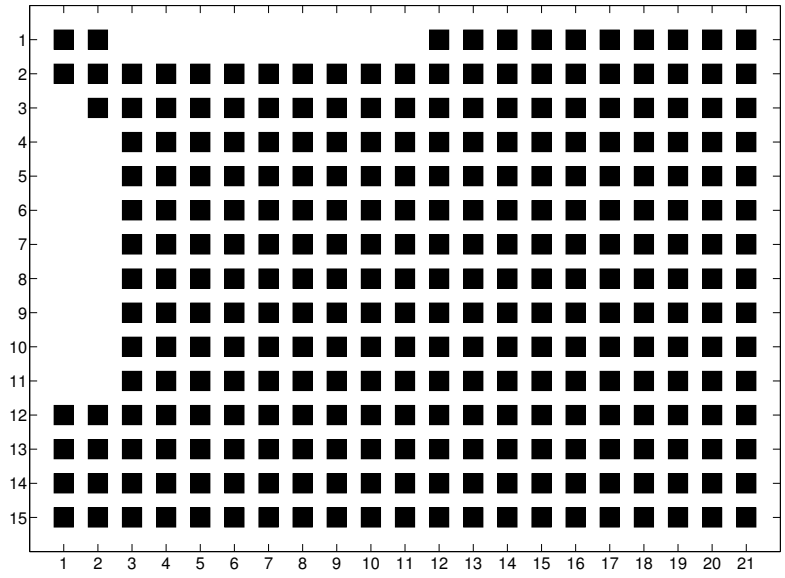
Algorithms 6.1 and 6.2 introduce the thresholds $\tau_1 \coloneqq KN$ and $\tau_2 \coloneqq KM$, which define the lowermost row and the rightmost column of $\bar{\mathbf{\Sigma}}$ for element annihilations. Thus, these



(a) $\bar{\mathbf{\Sigma}}^{(1)} = \mathcal{H}_{11,1}^{2,1} \bar{\mathbf{\Sigma}}^{(0)}$

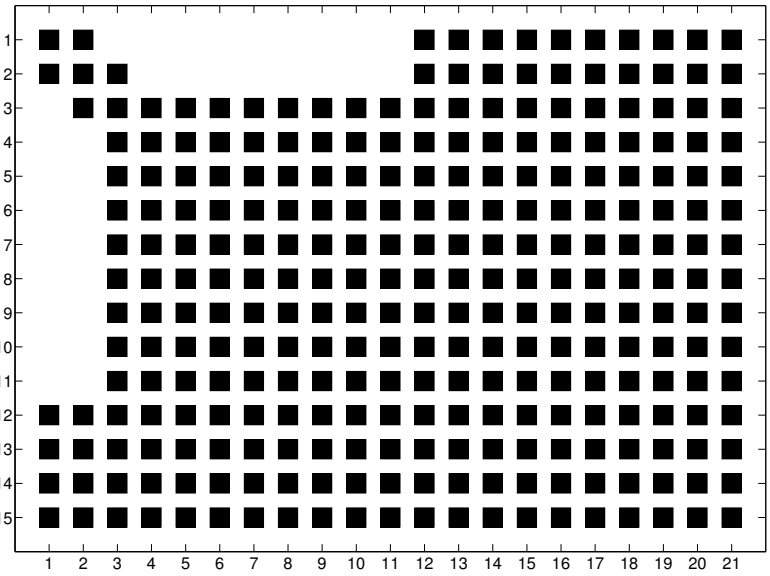


(b) $\bar{\mathbf{\Sigma}}^{(2)} = \bar{\mathbf{\Sigma}}^{(1)} \mathcal{H}_{1,11}^{1,2}$

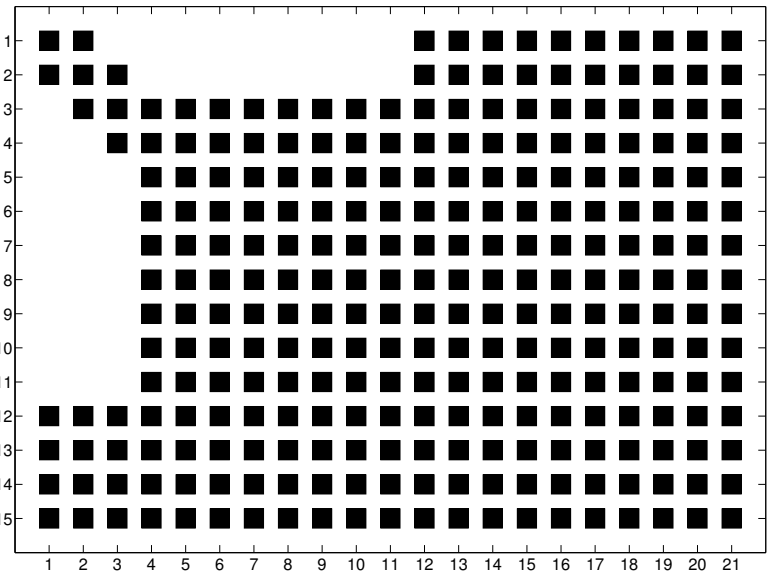


(c) $\bar{\mathbf{\Sigma}}^{(3)} = \mathcal{H}_{11,2}^{3,2} \bar{\mathbf{\Sigma}}^{(2)}$

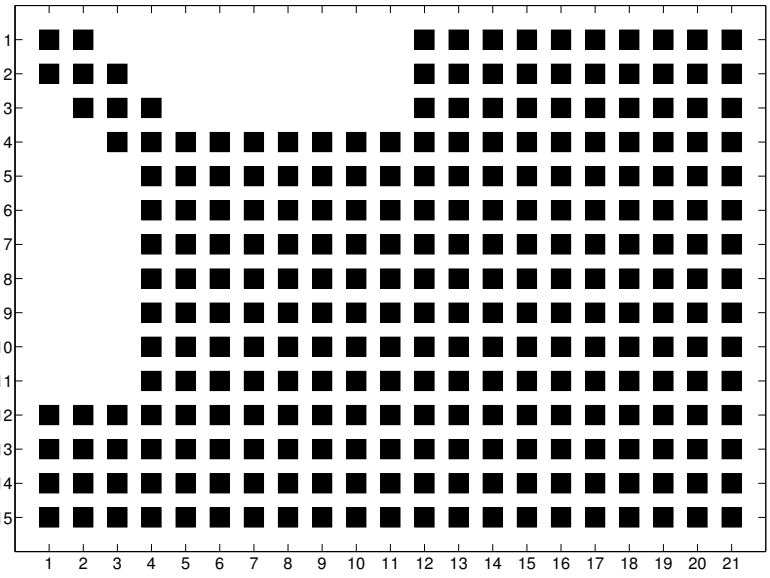
Figure 6.3: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 16\}$, of the further enhanced tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t})}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(d) $\bar{\mathbf{\Sigma}}^{(4)} = \bar{\mathbf{\Sigma}}^{(3)} \mathcal{H}^{2,3}_{2,11}$

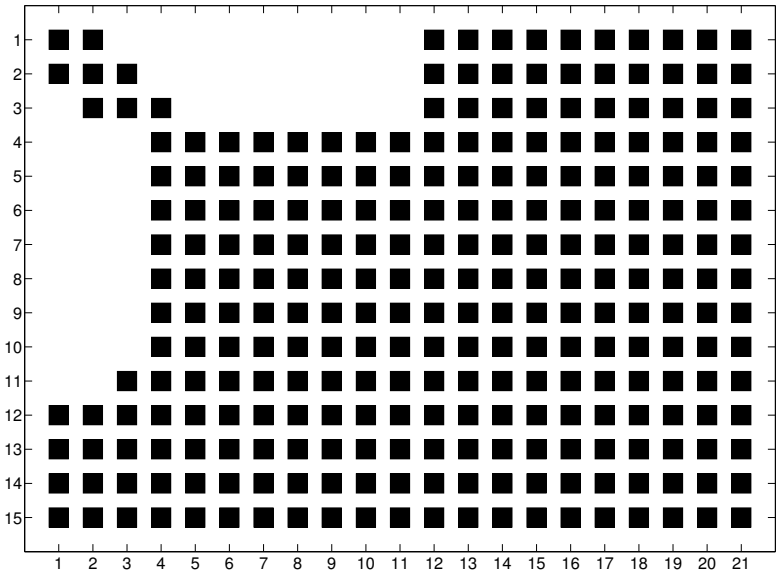


(e) $\bar{\mathbf{\Sigma}}^{(5)} = \mathcal{H}^{4,3}_{11,3} \bar{\mathbf{\Sigma}}^{(4)}$

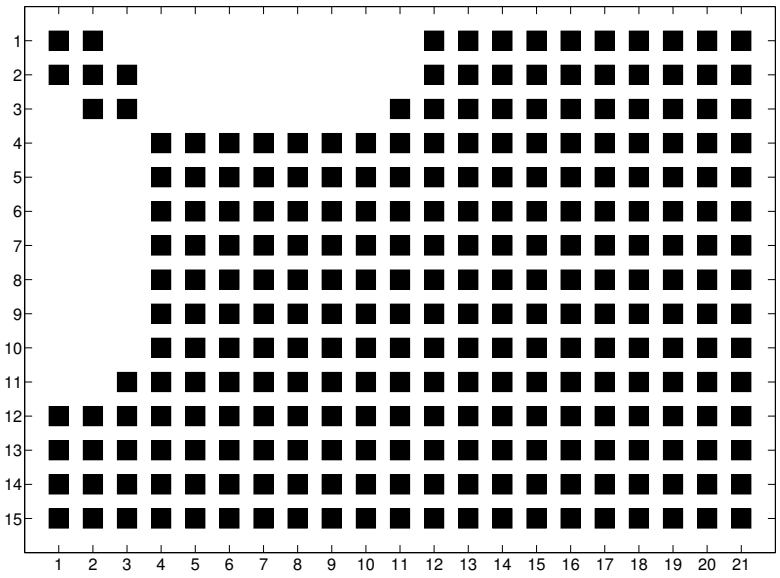


(f) $\bar{\mathbf{\Sigma}}^{(6)} = \bar{\mathbf{\Sigma}}^{(5)} \mathcal{H}^{3,4}_{3,11}$

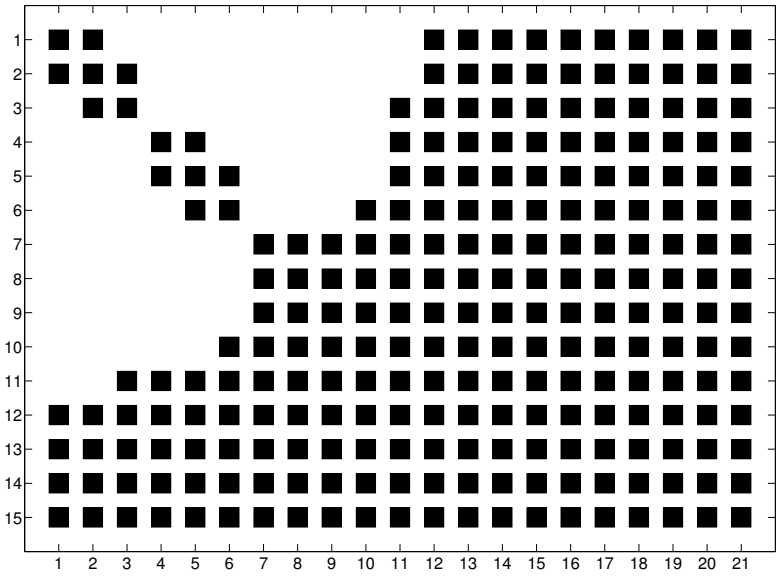
Figure 6.3: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 16\}$, of the further enhanced tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t})}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(g) $\bar{\mathbf{\Sigma}}^{(7)} = \mathcal{G}_{4,3}^{11,3}\bar{\mathbf{\Sigma}}^{(6)}$



(h) $\bar{\mathbf{\Sigma}}^{(8)} = \bar{\mathbf{\Sigma}}^{(7)}\mathcal{G}_{3,4}^{3,11}$



(i) $\bar{\mathbf{\Sigma}} = \bar{\mathbf{\Sigma}}^{(16)} = \bar{\mathbf{\Sigma}}^{(15)}\mathcal{G}_{6,7}^{6,10}$

Figure 6.3: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 16\}$, of the further enhanced tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t})}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.

variants of our algorithm set τ_1 to the index of the last row and τ_2 to the index of the last column of $\bar{\mathbf{\Sigma}}$. As we have observed for our strategy in Section 6.1.1, the indices of the lowermost (χ_1 in Algorithms 6.1 and 6.2) and the rightmost (χ_2 in Algorithms 6.1 and 6.2) elements that actually can be annihilated have to be decremented for every $d \times d$ block of $\mathbf{\Sigma}$ to prevent previously annihilated elements to become nonzero again. For this reason, exactly one additional row and column below and right of $\mathbf{\Sigma}$ per $d \times d$ block (except for the last one) is needed for element annihilations, i.e., in total $K - 1$ additional rows and columns. Considering this, we can define the threshold

$$\begin{aligned} \tau \coloneqq \tau_1 \coloneqq \tau_2 &\coloneqq Kd + K - 1 \\ &= K(d+1) - 1 \end{aligned} \tag{6.1}$$

for our further enhanced tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t})}$.

Before we discuss the general algorithm, we want to demonstrate the changes through our already familiar example for $K = 3$, $d = 3$, $M = 7$, and $N = 5$. Figures 6.3a to 6.3h show the eight decomposition steps required for annihilating the elements below and right of the first (top-left) $d \times d$ block. These decomposition steps are very similar to the decomposition steps applied by our previous algorithm variant CRIAD$^{(\mathrm{t}')}$. However, the indices used for the Householder and Givens transformations are different due to our new threshold τ. For instance, instead of

$$\bar{\mathbf{\Sigma}}^{(1)} = \mathcal{H}^{2,1}_{15,1}\bar{\mathbf{\Sigma}}^{(0)}$$

the first decomposition step of our new algorithm variant CRIAD$^{(\mathrm{t})}$ is

$$\bar{\mathbf{\Sigma}}^{(1)} = \mathcal{H}^{2,1}_{11,1}\bar{\mathbf{\Sigma}}^{(0)},$$

and instead of

$$\bar{\mathbf{\Sigma}}^{(8)} = \bar{\mathbf{\Sigma}}^{(7)}\mathcal{G}^{3,21}_{3,4}$$

the Givens transformation

$$\bar{\mathbf{\Sigma}}^{(8)} = \bar{\mathbf{\Sigma}}^{(7)}\mathcal{G}^{3,11}_{3,4}$$

is applied. Analogous to our previous algorithm variants, the same strategy can as well be executed for the second $d \times d$ block, which leads to the result illustrated in Figure 6.3i. As we have seen, the number of element annihilations performed by our algorithm is considerably reduced when we adjust the threshold τ appropriately.

The general procedure is listed in Algorithm 6.3, which introduces very few changes in comparison to Algorithm 6.2. For defining the threshold τ as in Equation 6.1, Lines 6 and 7 of Algorithm 6.2 are replaced by Line 6 of Algorithm 6.3. Since we now only have

Algorithm 6.3 Constructive algorithm for solving the relaxed interference alignment decomposition problem (further enhanced tridiagonal blocks variant).

function $\mathrm{CRIAD}^{(\mathrm{t})}(\boldsymbol{H}, K, d, M, N)$
  $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \boldsymbol{I}_{KN}$
  $\bar{\boldsymbol{\Sigma}} \leftarrow \boldsymbol{H}$
  $\hat{\boldsymbol{V}} \leftarrow \boldsymbol{I}_{KM}$

  $\tau \leftarrow K(d+1) - 1$

  for all $\kappa \in \{1, 2, \ldots, K-1\}$ **do**
    $\chi \leftarrow \tau - \kappa + 1$

    for all $\delta \in \{1, 2, \ldots, d\}$ **do**
      $\psi \leftarrow (\kappa - 1)d + \delta$
      $\varphi \leftarrow \psi + 1$

      $\tilde{\mathcal{H}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \varphi, \psi \rangle, \langle \chi, \psi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \tilde{\mathcal{H}} \bar{\boldsymbol{\Sigma}}$
      $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \tilde{\mathcal{H}} \hat{\boldsymbol{U}}^{\mathrm{H}}$

      $\tilde{\mathcal{H}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \varphi \rangle, \langle \psi, \chi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \tilde{\mathcal{H}}$
      $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \tilde{\mathcal{H}}$
    end for

    $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \chi, \psi \rangle, \langle \varphi, \psi \rangle)$
    $\bar{\boldsymbol{\Sigma}} \leftarrow \mathcal{G} \bar{\boldsymbol{\Sigma}}$
    $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \mathcal{G} \hat{\boldsymbol{U}}^{\mathrm{H}}$

    $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \chi \rangle, \langle \psi, \varphi \rangle)$
    $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \mathcal{G}$
    $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \mathcal{G}$
  end for

  return $\langle \hat{\boldsymbol{U}}, \bar{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}} \rangle$
end function

one threshold τ, the variable χ can be put in place of both χ_1 and χ_2 (cf. Line 9 of Algorithm 6.3).

6.1.4 Full blocks variant

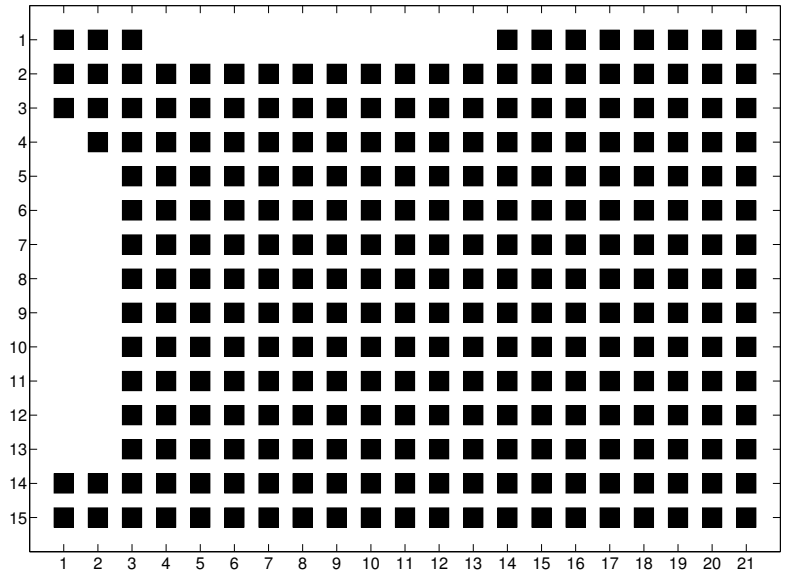
With the algorithm variants discussed in Sections 6.1.1 to 6.1.3, we have learned how to solve the relaxed interference alignment decomposition problem by reducing the $d \times d$ blocks of the global interference alignment matrix $\boldsymbol{\Sigma}$ to tridiagonal form. In this section, we want to attain the goal of transforming the global channel matrix $\boldsymbol{H}$ to the extended global interference alignment matrix $\bar{\boldsymbol{\Sigma}}$ with the $d \times d$ blocks of its submatrix $\boldsymbol{\Sigma}$ being full matrices. Although there is no such requirement imposed by the relaxed interference alignment decomposition problem, the solution in consideration of this additional constraint may offer deeper insight into the inner workings of our constructive algorithm. Particularly, we will better recognize the effects on the acceptable values for the parameters K, d, M, and N. This will lead us to the bidiagonal blocks variant of the constructive algorithm, which will be introduced in Section 6.1.5.

For understanding the necessary modifications to the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t})}$ (cf. Section 6.1.3) in pursuance of our objective to obtain a comparable full blocks algorithm variant CRIAD$^{(\mathrm{f})}$, we will once again turn back to our meanwhile well-known example for $K = 3$, $d = 3$, $M = 7$, and $N = 5$. As before, we will transform $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{0, 1, \ldots, \Lambda - 1\}$, with $\bar{\boldsymbol{\Sigma}}^{(0)} = \boldsymbol{H}$ and $\bar{\boldsymbol{\Sigma}}^{(\Lambda)} = \bar{\boldsymbol{\Sigma}}$, solely by pre- or postmultiplying a Householder or Givens matrix. In contrast to the tridiagonal blocks algorithm variants (and also the bidiagonal blocks algorithm variant, as we will see in the next section), we assume $d \geq 2$ instead of $d \geq 1$. This allows a slightly optimized CRIAD$^{(\mathrm{f})}$ algorithm with a better threshold τ and is no real limitation because the CRIAD$^{(\mathrm{t})}$ algorithm naturally produces full blocks, i.e., 1×1 "blocks", for $d = 1$.

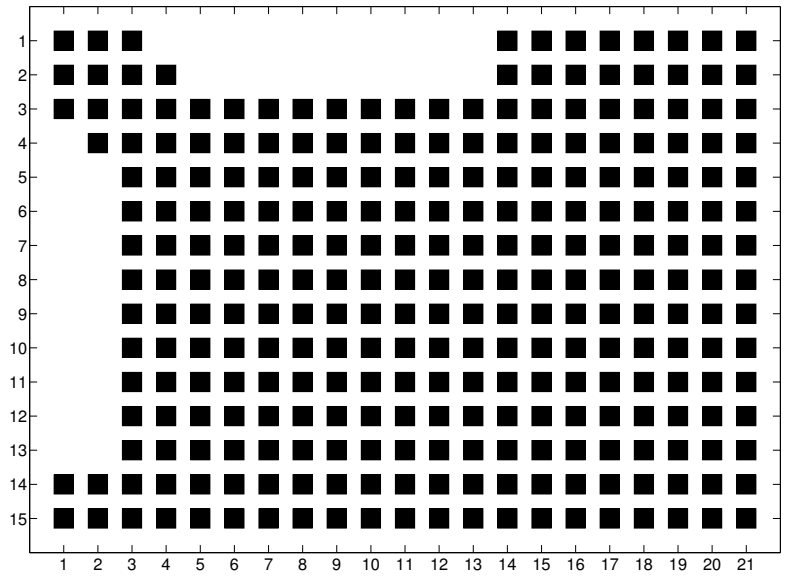
At first, we want to focus on the unitary transformations required for annihilating the elements below and right of the first (top-left) $d \times d$ block. Since we already know the relevant techniques (cf. Chapter 5 and Sections 6.1.1 to 6.1.3), we can readily identify the first four decomposition steps

$$\begin{aligned}
\bar{\boldsymbol{\Sigma}}^{(1)} &= \mathcal{H}_{13,1}^{3,1} \bar{\boldsymbol{\Sigma}}^{(0)}, \\
\bar{\boldsymbol{\Sigma}}^{(2)} &= \bar{\boldsymbol{\Sigma}}^{(1)} \mathcal{H}_{1,13}^{1,3}, \\
\bar{\boldsymbol{\Sigma}}^{(3)} &= \mathcal{H}_{13,2}^{4,2} \bar{\boldsymbol{\Sigma}}^{(2)}, \text{ and} \\
\bar{\boldsymbol{\Sigma}}^{(4)} &= \bar{\boldsymbol{\Sigma}}^{(3)} \mathcal{H}_{2,13}^{2,4},
\end{aligned}$$

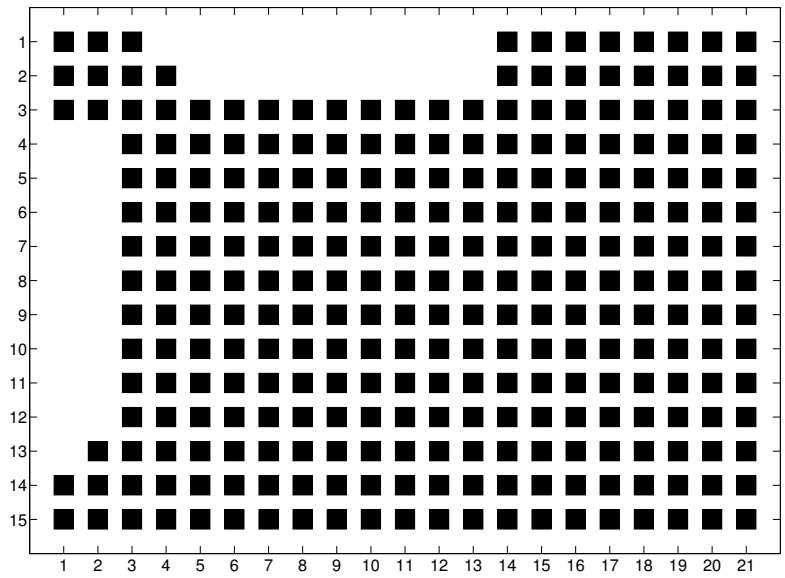
as partly illustrated in Figures 6.4a and 6.4b. To achieve the full $d \times d$ block form, the top and left indices for the Householder transformations here are greater than the indices that previously have been used by the tridiagonal blocks algorithm variants. Therefore, there already is an unwanted (potentially) nonzero element in both the second column and



(a) $\bar{\boldsymbol{\Sigma}}^{(3)} = \mathscr{H}^{4,2}_{13,2}\bar{\boldsymbol{\Sigma}}^{(2)}$

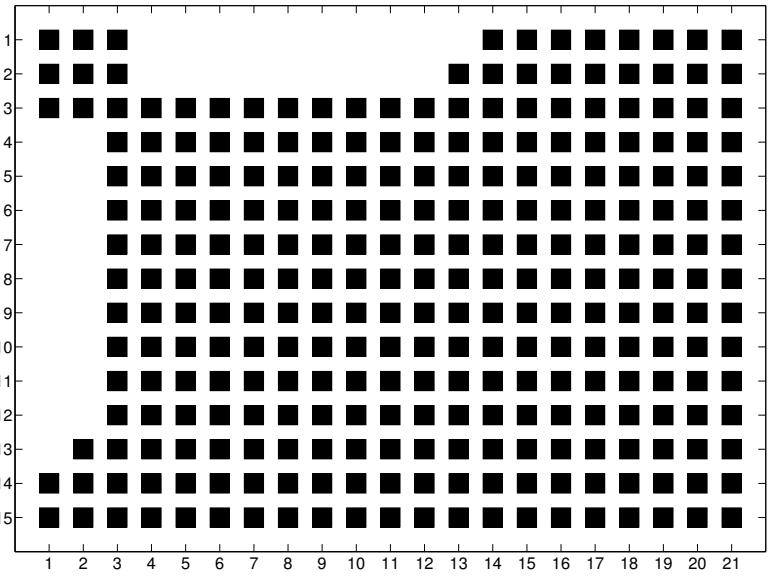


(b) $\bar{\boldsymbol{\Sigma}}^{(4)} = \bar{\boldsymbol{\Sigma}}^{(3)}\mathscr{H}^{2,4}_{2,13}$

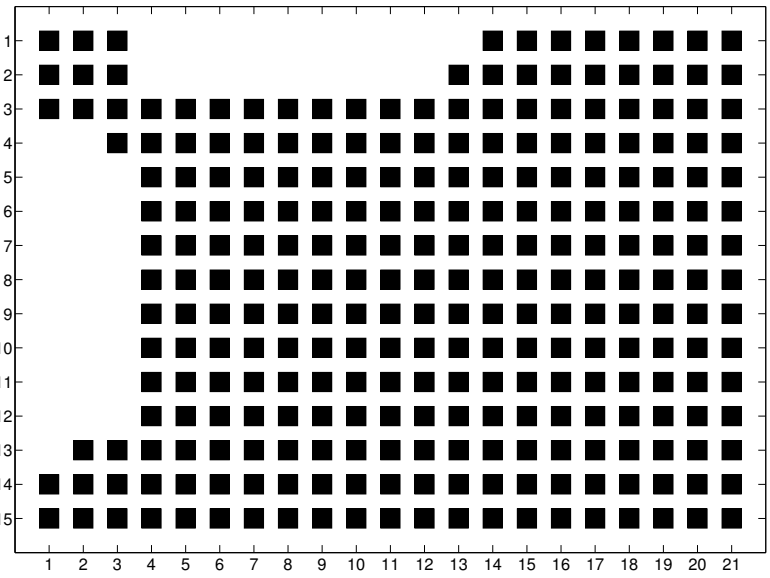


(c) $\bar{\boldsymbol{\Sigma}}^{(5)} = \mathscr{G}^{13,2}_{4,2}\bar{\boldsymbol{\Sigma}}^{(4)}$

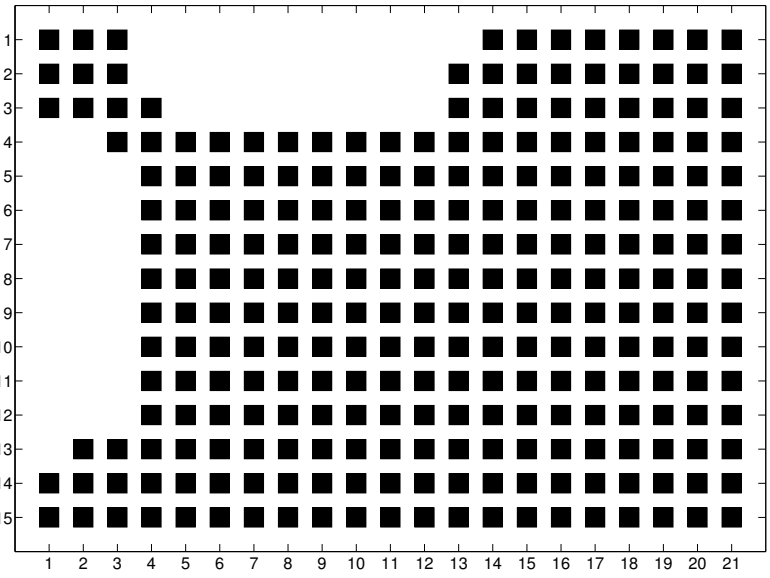
Figure 6.4: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 20\}$, of the full blocks algorithm variant CRIAD$^{(\mathrm{f})}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(d) $\bar{\boldsymbol{\Sigma}}^{(6)} = \bar{\boldsymbol{\Sigma}}^{(5)} \mathcal{G}_{2,4}^{2,13}$

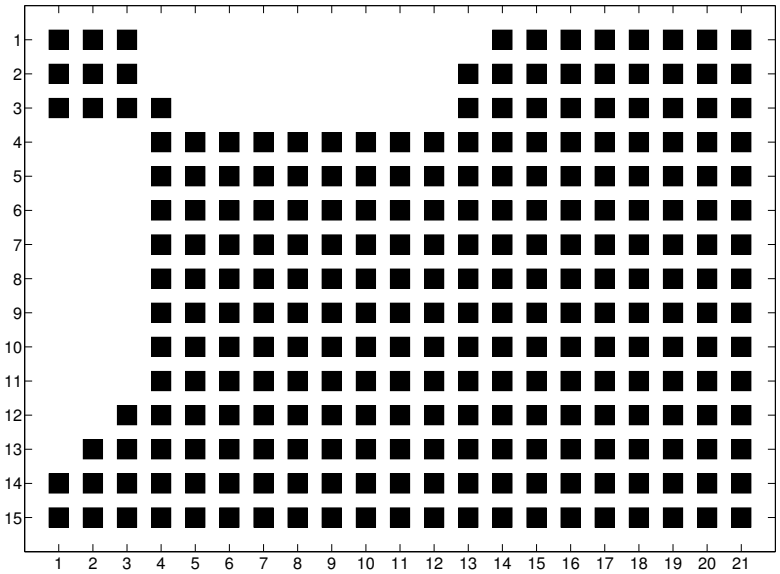


(e) $\bar{\boldsymbol{\Sigma}}^{(7)} = \mathcal{H}_{12,3}^{4,3} \bar{\boldsymbol{\Sigma}}^{(6)}$

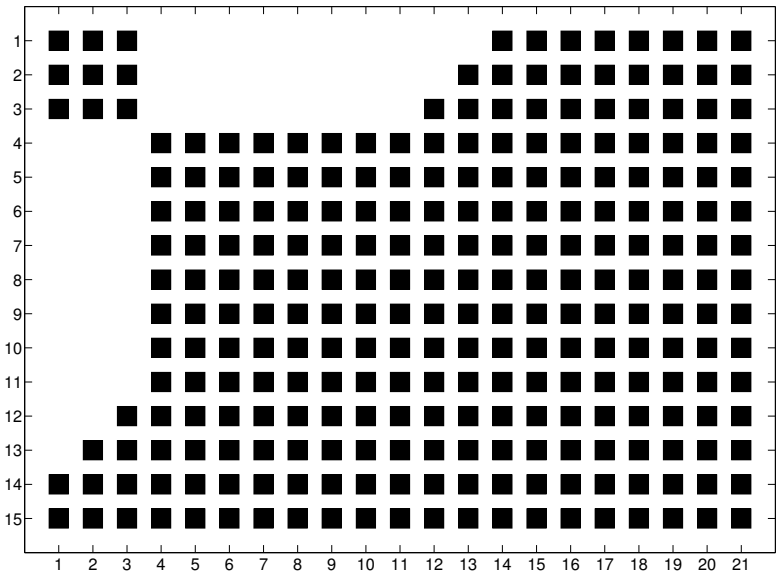


(f) $\bar{\boldsymbol{\Sigma}}^{(8)} = \bar{\boldsymbol{\Sigma}}^{(7)} \mathcal{H}_{3,12}^{3,4}$

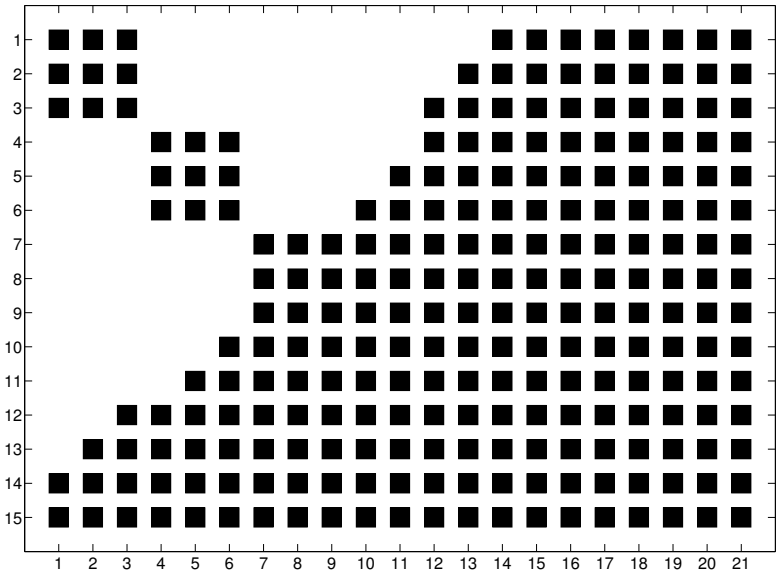
Figure 6.4: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 20\}$, of the full blocks algorithm variant CRIAD$^{(\mathrm{f})}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(g) $\bar{\mathbf{\Sigma}}^{(9)} = \mathcal{G}_{4,3}^{12,3}\bar{\mathbf{\Sigma}}^{(8)}$



(h) $\bar{\mathbf{\Sigma}}^{(10)} = \bar{\mathbf{\Sigma}}^{(9)}\mathcal{G}_{3,4}^{3,12}$



(i) $\bar{\mathbf{\Sigma}} = \bar{\mathbf{\Sigma}}^{(20)} = \bar{\mathbf{\Sigma}}^{(19)}\mathcal{G}_{6,7}^{6,10}$

Figure 6.4: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 20\}$, of the full blocks algorithm variant CRIAD$^{(\mathrm{f})}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.

second row after applying this sequence of four Householder matrix multiplications (cf. Figure 6.4b). By utilizing the Givens rotations

$$\bar{\boldsymbol{\Sigma}}^{(5)} = \mathcal{G}_{4,2}^{13,2}\bar{\boldsymbol{\Sigma}}^{(4)} \text{ and}$$
$$\bar{\boldsymbol{\Sigma}}^{(6)} = \bar{\boldsymbol{\Sigma}}^{(5)}\mathcal{G}_{2,4}^{2,13},$$

these nonzero elements can be shifted down and to the right, respectively (cf. Figures 6.4c and 6.4d).

Hence, we have applied two Householder and two Givens transformations for annihilating all elements below and right of the second column and row that have to be strictly zero. The same scheme can be put into use for annihilating the elements below and right of the third (and thus last) column and row of the first $d \times d$ block. These two more Householder and two more Givens matrix multiplications bring the top-left $d \times d$ block to the intended form, as shown in Figures 6.4e to 6.4h.

Altogether, six Householder reflections (one per column and one per row) and four Givens rotations (one per column and one per row except for the first column and row) have been required for the appropriate element annihilations below and right of the first $d \times d$ block. Because of the two premultiplied Givens matrices for shifting nonzero elements down, two more rows below the $Kd \times Kd$ submatrix $\boldsymbol{\Sigma}$ are needed for the first $d \times d$ block. Likewise, two more columns right of $\boldsymbol{\Sigma}$ are necessary for the first $d \times d$ block.

The same approach can be pursued for annihilating the elements below and right of the second $d \times d$ block. After these ten more decomposition steps, it holds that $\bar{\boldsymbol{\Sigma}}^{(20)} = \bar{\boldsymbol{\Sigma}}$, as the illustration in Figure 6.4i indicates. Similar to our observation for the first $d \times d$ block, there need to be two more rows below and two more columns right of $\boldsymbol{\Sigma}$ available for the second $d \times d$ block. In general, the $KN \times KM$ matrices $\boldsymbol{H}$ and $\bar{\boldsymbol{\Sigma}}$ must be large enough to provide at least $d-1$ rows below and $d-1$ columns right of the top-left $Kd \times Kd$ submatrix for each but the last $d \times d$ diagonal block of this submatrix, i.e., for in total $K-1$ blocks.

It follows immediately the threshold (cf. Section 6.1.3)

$$\tau = Kd + (K-1)(d-1) \tag{6.2}$$

for the full blocks algorithm variant. If we allowed $d = 1$, the method for annihilating the elements below and right of the first column and row of each $d \times d$ block would be the same as for the other columns and rows, i.e., two additional Givens rotations per $d \times d$ block would be necessary. This would lead to the threshold $\tau = Kd + d(K-1) = d(2K-1)$.

When comparing Equation 6.1 to Equation 6.2, we deduce that τ for the further enhanced tridiagonal and the full blocks algorithm variants is equal if and only if $d = 2$. Because of $d \geq 2$, the threshold τ for the full blocks algorithm variant always is greater than or equal the threshold τ for the further enhanced tridiagonal blocks algorithm variant.

Algorithm 6.4 Constructive algorithm for solving the relaxed interference alignment decomposition problem (full blocks variant).

function CRIAD$^{(\mathrm{f})}(\boldsymbol{H}, K, d, M, N)$
  $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \boldsymbol{I}_{KN}$
  $\bar{\boldsymbol{\Sigma}} \leftarrow \boldsymbol{H}$
  $\hat{\boldsymbol{V}} \leftarrow \boldsymbol{I}_{KM}$

  $\tau \leftarrow Kd + (K-1)(d-1)$

  for all $\kappa \in \{1, 2, \ldots, K-1\}$ **do**
    $\varphi \leftarrow \kappa d$
    $\chi \leftarrow \tau - (\kappa - 1)(d-1)$
    $\psi \leftarrow (\kappa - 1)d + 1$

    $\tilde{\mathcal{H}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \varphi, \psi \rangle, \langle \chi, \psi \rangle)$
    $\bar{\boldsymbol{\Sigma}} \leftarrow \tilde{\mathcal{H}} \bar{\boldsymbol{\Sigma}}$
    $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \tilde{\mathcal{H}} \hat{\boldsymbol{U}}^{\mathrm{H}}$

    $\tilde{\mathcal{H}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \varphi \rangle, \langle \psi, \chi \rangle)$
    $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \tilde{\mathcal{H}}$
    $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \tilde{\mathcal{H}}$

    $\varphi \leftarrow \kappa d + 1$

    for all $\delta \in \{2, 3, \ldots, d\}$ **do**
      $\chi \leftarrow \tau - (\kappa - 1)(d-1) - \delta + 2$
      $\psi \leftarrow (\kappa - 1)d + \delta$

      $\tilde{\mathcal{H}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \varphi, \psi \rangle, \langle \chi, \psi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \tilde{\mathcal{H}} \bar{\boldsymbol{\Sigma}}$
      $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \tilde{\mathcal{H}} \hat{\boldsymbol{U}}^{\mathrm{H}}$

      $\tilde{\mathcal{H}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \varphi \rangle, \langle \psi, \chi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \tilde{\mathcal{H}}$
      $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \tilde{\mathcal{H}}$

      $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \chi, \psi \rangle, \langle \varphi, \psi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \mathcal{G} \bar{\boldsymbol{\Sigma}}$
      $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \mathcal{G} \hat{\boldsymbol{U}}^{\mathrm{H}}$

      $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \chi \rangle, \langle \psi, \varphi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \mathcal{G}$
      $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \mathcal{G}$
    end for
  end for

  return $\langle \hat{\boldsymbol{U}}, \bar{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}} \rangle$
end function

Moreover, since τ has to be a valid column and row index of the $KN \times KM$ matrices $\boldsymbol{H}$ and $\bar{\boldsymbol{\Sigma}}$, i.e., $KM \geq \tau$ and $KN \geq \tau$, the further enhanced tridiagonal algorithm variant CRIAD$^{(\mathrm{t})}$ may be able to handle a larger number of combinations of the parameters K, d, M, and N than the full blocks algorithm variant CRIAD$^{(\mathrm{f})}$. The two different thresholds τ can easily be seen for our example when contrasting Figures 6.3i and 6.4i.

Algorithm 6.4 lists the generalized full blocks variant CRIAD$^{(\mathrm{f})}$ of the constructive algorithm for solving the relaxed interference alignment decomposition problem. Both the general structure and the occurring variables are similar to the algorithm variants CRIAD$^{(\mathrm{t}'')}$, CRIAD$^{(\mathrm{t}')}$, and CRIAD$^{(\mathrm{t})}$ (cf. Algorithms 6.1 to 6.3). In Line 6 of Algorithm 6.4, the threshold τ is initialized according to Equation 6.2. Within the body of the outer loop in Lines 8 to 43, the elements below and right of precisely one $d \times d$ block are annihilated. Analogous to our previous algorithm variants, ψ defines the current column and row for element annihilations, while φ and χ denote both the indices of the first and last elements affected by Householder transformations and the indices of the elements altered by Givens rotations in each of these current columns and rows (cf. Lines 9 to 11 and Lines 21, 24, and 25).

Like in our example, the elements below the first column of the $d \times d$ block are annihilated by a single Householder reflection (cf. Lines 13 to 15). In Lines 17 to 19, the elements right of the first row of the $d \times d$ block are annihilated by another appropriate Householder transformation. The proper element annihilations below and right of the second to last columns and rows of the $d \times d$ block are achieved by the unitary transformations within the body of the inner loop in Lines 23 to 42. First, in Lines 27 to 33, two Householder matrix multiplications annihilate elements in the respective column and row. Then, in Lines 35 to 41, two Givens rotations shift the position of two unwanted nonzero elements down and to the right, respectively.

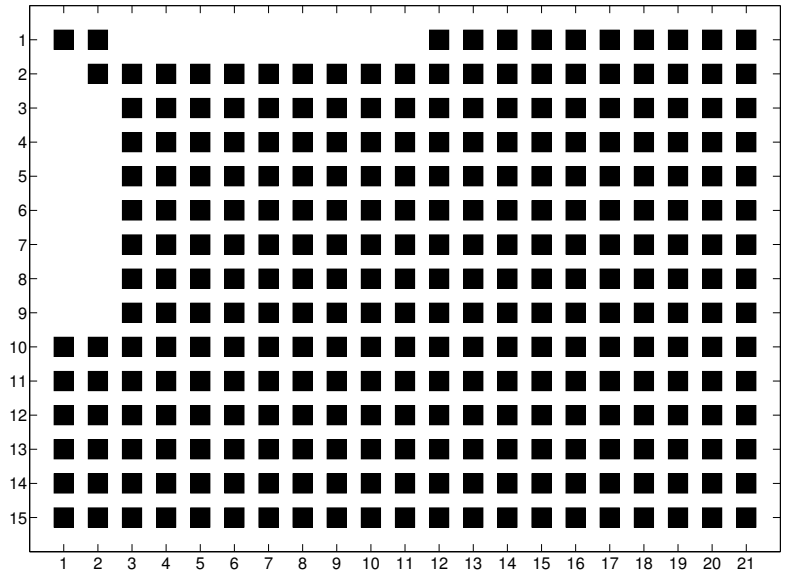
6.1.5 Bidiagonal blocks variant

In Section 6.1.4, we have noticed that, due to a smaller threshold τ, the tridiagonal blocks algorithm variant CRIAD$^{(\mathrm{t})}$ may find solutions for more combinations of K, d, M, and N than the full blocks algorithm variant CRIAD$^{(\mathrm{f})}$. According to Corollary 2.7, which states necessary but not sufficient conditions for the existence of a solution to the interference alignment problem on the symmetric K-user time-varying MIMO interference channel, either

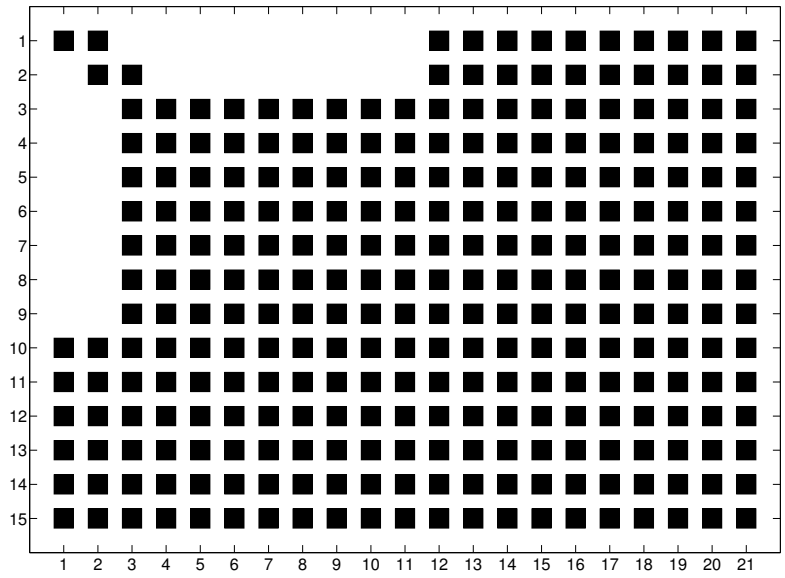
$$(M = d \ \wedge \ N > d) \text{ or} \tag{6.3}$$

$$(M > d \ \wedge \ N = d) \tag{6.4}$$

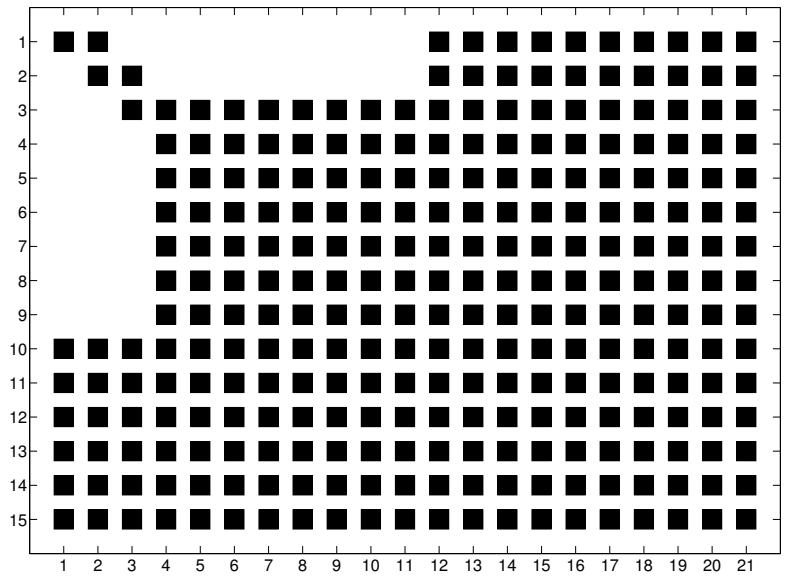
may be allowed. Although the relaxed interference alignment decomposition problem is a different problem, we still want to review our constructive algorithm variants for support of the input parameters d, M, and N in agreement with Equations 6.3 and 6.4. From



(a) $\bar{\boldsymbol{\Sigma}}^{(3)} = \mathcal{H}^{2,2}_{9,2}\bar{\boldsymbol{\Sigma}}^{(2)}$

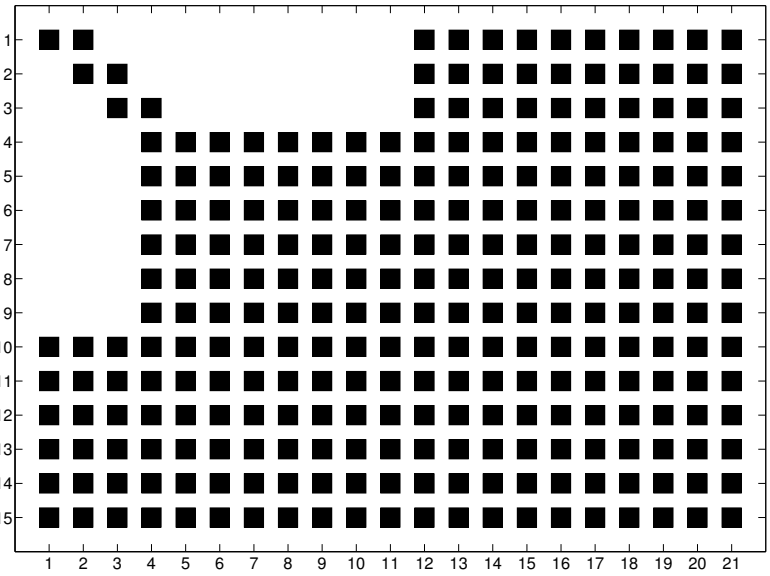


(b) $\bar{\boldsymbol{\Sigma}}^{(4)} = \bar{\boldsymbol{\Sigma}}^{(3)}\mathcal{H}^{2,3}_{2,11}$

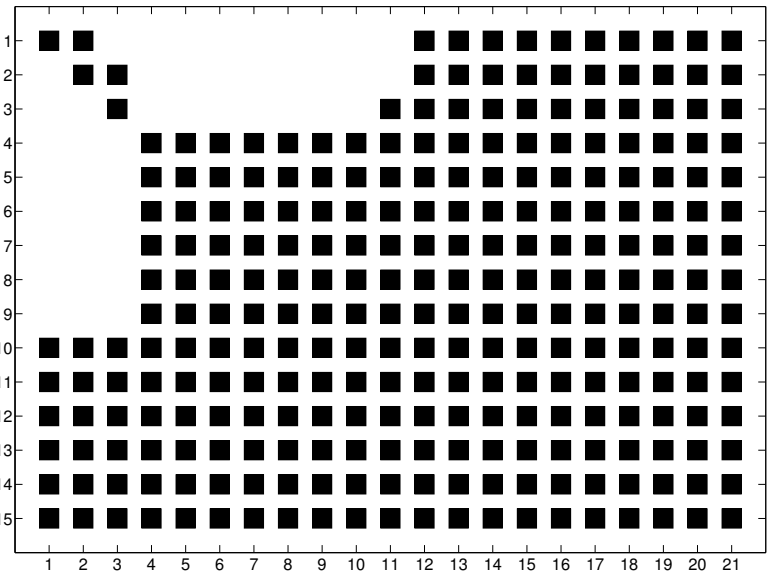


(c) $\bar{\boldsymbol{\Sigma}}^{(5)} = \mathcal{H}^{3,3}_{9,3}\bar{\boldsymbol{\Sigma}}^{(4)}$

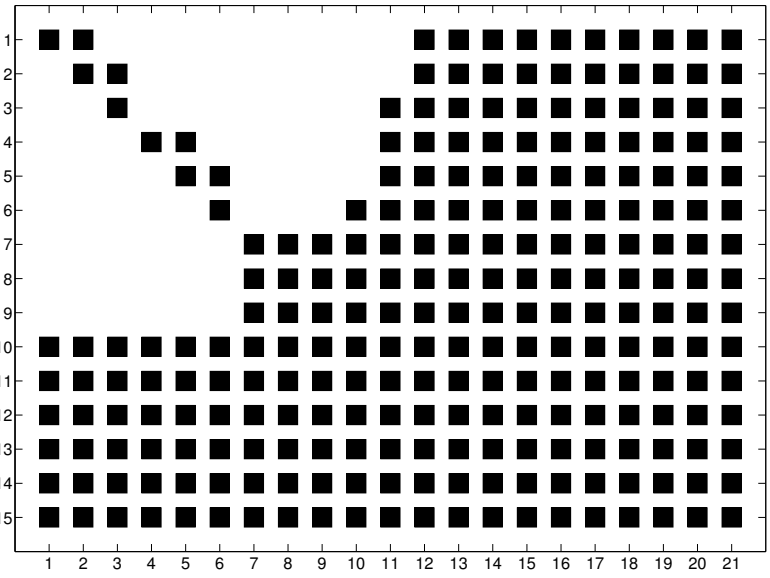
Figure 6.5: Selected intermediary results $\bar{\boldsymbol{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 14\}$, of the bidiagonal blocks algorithm variant CRIAD$^{(\mathrm{b})}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.



(d) $\bar{\mathbf{\Sigma}}^{(6)} = \bar{\mathbf{\Sigma}}^{(5)} \mathcal{H}_{3,11}^{3,4}$



(e) $\bar{\mathbf{\Sigma}}^{(7)} = \bar{\mathbf{\Sigma}}^{(6)} \mathcal{G}_{3,4}^{3,11}$



(f) $\bar{\mathbf{\Sigma}} = \bar{\mathbf{\Sigma}}^{(14)} = \bar{\mathbf{\Sigma}}^{(13)} \mathcal{G}_{6,7}^{6,10}$

Figure 6.5: Selected intermediary results $\bar{\mathbf{\Sigma}}^{(i)}$, $i \in \{0, 1, \ldots, 14\}$, of the bidiagonal blocks algorithm variant CRIAD$^{(\mathrm{b})}$ for $K = 3$, $d = 3$, $M = 7$, and $N = 5$.

Equations 6.1 and 6.2, it follows for both the tridiagonal and full blocks algorithm variants that $Kd < \tau \leq KM$ and $Kd < \tau \leq KN$. Thus, the input parameters have to fulfill $M > d$ and $N > d$. We have to conclude that our previous algorithm variants cannot find a solution for d, M, and N that comply with Equation 6.3 or Equation 6.4.

Is there a way to improve our algorithm in order to support either M or N being equal to d? Since the tridiagonal blocks algorithm variant in this regard provides better results than the full blocks algorithm variant, we will attempt to push this approach further by bidiagonalizing the $d \times d$ blocks of the submatrix $\mathbf{\Sigma}$ of $\bar{\mathbf{\Sigma}}$. For this purpose, let us return a last time to our example for $K = 3$, $d = 3$, $M = 7$, and $N = 5$. When we remember Chapter 5, in particular Figures 5.9a and 5.9b, we know that in general there are two alternatives for bidiagonalizing a matrix. We may annihilate all elements below the main diagonal and keep one element right of each main diagonal element, or vice versa. For our example, we select the former alternative. As partly illustrated in Figures 6.5a to 6.5d, the first decomposition steps thus are the Householder transformations

$$\begin{aligned}
\bar{\mathbf{\Sigma}}^{(1)} &= \mathcal{H}_{9,1}^{1,1}\bar{\mathbf{\Sigma}}^{(0)},\\
\bar{\mathbf{\Sigma}}^{(2)} &= \bar{\mathbf{\Sigma}}^{(1)}\mathcal{H}_{1,11}^{1,2},\\
\bar{\mathbf{\Sigma}}^{(3)} &= \mathcal{H}_{9,2}^{2,2}\bar{\mathbf{\Sigma}}^{(2)},\\
\bar{\mathbf{\Sigma}}^{(4)} &= \bar{\mathbf{\Sigma}}^{(3)}\mathcal{H}_{2,11}^{2,3},\\
\bar{\mathbf{\Sigma}}^{(5)} &= \mathcal{H}_{9,3}^{3,3}\bar{\mathbf{\Sigma}}^{(4)}, \text{ and}\\
\bar{\mathbf{\Sigma}}^{(6)} &= \bar{\mathbf{\Sigma}}^{(5)}\mathcal{H}_{3,11}^{3,4}.
\end{aligned}$$

For the moment, we are ignoring how the appropriate thresholds τ_1 and τ_2 have been chosen. We will get back to that shortly.

In Figure 6.5d, we can immediately identify element $\bar{\sigma}_{3,4}^{(6)}$ as the only element that remains to be annihilated for the first (top-left) $d \times d$ block. Similar to the tridiagonal and full blocks algorithm variants, a Givens rotation can shift the position of the nonzero element to the right. After the decomposition step

$$\bar{\mathbf{\Sigma}}^{(7)} = \bar{\mathbf{\Sigma}}^{(6)}\mathcal{G}_{3,4}^{3,11},$$

the matrix has the shape depicted in Figure 6.5e. Hence, only six Householder and one Givens transformation have been necessary for annihilating the elements below and right of the first $d \times d$ block, which now is in bidiagonal form. The same pattern can be applied to the second $d \times d$ block. Following another seven decomposition steps, we obtain the final result shown in Figure 6.5f.

As we already have for the tridiagonal blocks algorithm variants, we abstain from reducing the last (bottom-right) $d \times d$ block. Although the bidiagonalization of the last $d \times d$ block can easily be accomplished by three more Householder reflections or four more Givens rotations, there is no benefit in applying these transformations for solving the

relaxed interference alignment decomposition problem.

For bidiagonalizing the $d \times d$ blocks, we can start by either premultiplying or postmultiplying an appropriate Householder matrix. The only difference is whether the resulting $d \times d$ blocks are upper or lower bidiagonal matrices (cf. Figures 5.9a and 5.9b). Furthermore, this determines if the Givens matrices have to be premultiplied or postmultiplied to annihilate the last nonzero element below or right of every $d \times d$ block. If the first Householder matrix is premultiplied, the Givens matrices have to be postmultiplied, and vice versa. Hence, in one execution of the algorithm, Givens matrices are either premultiplied or postmultiplied.

The general bidiagonal blocks algorithm variant CRIAD$^{(\mathrm{b})}$ is listed in Algorithm 6.5. In Lines 6 and 7, the thresholds τ_1 and τ_2 are initialized such that

$$\tau_1 = Kd \text{ and} \tag{6.5}$$

$$\tau_2 = Kd + K - 1. \tag{6.6}$$

Accordingly, threshold τ_1 is equal to the index of the last row or column of the $Kd \times Kd$ submatrix $\mathbf{\Sigma}$, while threshold τ_2 adds $K - 1$ rows or columns, i.e., one row or column per $d \times d$ block, except for the last (bottom-right) one. As we have seen in Sections 6.1.1 to 6.1.4 as well as in our recent example that demonstrates the approach followed by the CRIAD$^{(\mathrm{b})}$ algorithm, an additional row or column outside of the submatrix $\mathbf{\Sigma}$ is requisite for every Givens transformation. Since, as noted above, the Givens matrices are either premultiplied or postmultiplied, it is sufficient that either the row or column threshold is greater than Kd, which justifies Equations 6.5 and 6.6.

We have not yet specified which of the thresholds τ_1 and τ_2 is the row threshold and which is the column threshold. Considering that our algorithm is supposed to be able to find solutions for the largest possible set of combinations of the input parameters K, d, M, and N, including Equations 6.3 and 6.4, it is advisable to decide based on the given input parameters whether τ_1 is the row or column threshold. Concretely, τ_1 shall be the row threshold if and only if the given global channel matrix $\boldsymbol{H}$ has more columns than rows, i.e., if $KN < KM$, or, equivalently, if $N < M$. Hence, when τ_1 is the row threshold, the Givens matrices have to be postmultiplied, and thus the first Householder matrix has to be premultiplied. In Algorithm 6.5, Lines 19 to 25 and Lines 38 to 40 are appropriate when τ_1 is the row threshold, while Lines 27 to 33 and Lines 42 to 44 are proper for τ_1 as column threshold.

The other parts of Algorithm 6.5 follow the scheme familiar from Algorithms 6.1 to 6.4. In each iteration of the outer loop in Lines 10 to 46, the elements below and right of exactly one $d \times d$ block are annihilated. For this reason, the elements below and right of all but the last (bottom-right) $d \times d$ block are annihilated after the $(K-1)^{\text{th}}$ iteration. In the body of the inner loop in Lines 13 to 35, a suitable Householder transformation is applied for each of the d columns and d rows of the current $d \times d$ block. Subsequent to the Householder

Algorithm 6.5 Constructive algorithm for solving the relaxed interference alignment decomposition problem (bidiagonal blocks variant).

function CRIAD$^{(\mathrm{b})}(\boldsymbol{H}, K, d, M, N)$
  $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \boldsymbol{I}_{KN}$
  $\bar{\boldsymbol{\Sigma}} \leftarrow \boldsymbol{H}$
  $\hat{\boldsymbol{V}} \leftarrow \boldsymbol{I}_{KM}$

  $\tau_1 \leftarrow Kd$
  $\tau_2 \leftarrow Kd + K - 1$
  $\chi_1 \leftarrow \tau_1$

  for all $\kappa \in \{1, 2, \ldots, K-1\}$ **do**
    $\chi_2 \leftarrow \tau_2 - \kappa + 1$

    for all $\delta \in \{1, 2, \ldots, d\}$ **do**
      $\psi \leftarrow (\kappa - 1)d + \delta$
      $\varphi_1 \leftarrow \psi$
      $\varphi_2 \leftarrow \psi + 1$

      if $N < M$ **then**
        $\tilde{\boldsymbol{\mathcal{H}}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \varphi_1, \psi \rangle, \langle \chi_1, \psi \rangle)$
        $\bar{\boldsymbol{\Sigma}} \leftarrow \tilde{\boldsymbol{\mathcal{H}}} \bar{\boldsymbol{\Sigma}}$
        $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \tilde{\boldsymbol{\mathcal{H}}} \hat{\boldsymbol{U}}^{\mathrm{H}}$

        $\tilde{\boldsymbol{\mathcal{H}}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \varphi_2 \rangle, \langle \psi, \chi_2 \rangle)$
        $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \tilde{\boldsymbol{\mathcal{H}}}$
        $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \tilde{\boldsymbol{\mathcal{H}}}$
      else
        $\tilde{\boldsymbol{\mathcal{H}}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \varphi_1 \rangle, \langle \psi, \chi_1 \rangle)$
        $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \tilde{\boldsymbol{\mathcal{H}}}$
        $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \tilde{\boldsymbol{\mathcal{H}}}$

        $\tilde{\boldsymbol{\mathcal{H}}} \leftarrow \textsc{Householder}(\bar{\boldsymbol{\Sigma}}, \langle \varphi_2, \psi \rangle, \langle \chi_2, \psi \rangle)$
        $\bar{\boldsymbol{\Sigma}} \leftarrow \tilde{\boldsymbol{\mathcal{H}}} \bar{\boldsymbol{\Sigma}}$
        $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \tilde{\boldsymbol{\mathcal{H}}} \hat{\boldsymbol{U}}^{\mathrm{H}}$
      end if
    end for

    if $N < M$ **then**
      $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \psi, \chi_2 \rangle, \langle \psi, \varphi_2 \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \mathcal{G}$
      $\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \mathcal{G}$
    else
      $\mathcal{G} \leftarrow \textsc{Givens}(\bar{\boldsymbol{\Sigma}}, \langle \chi_2, \psi \rangle, \langle \varphi_2, \psi \rangle)$
      $\bar{\boldsymbol{\Sigma}} \leftarrow \mathcal{G} \bar{\boldsymbol{\Sigma}}$
      $\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \mathcal{G} \hat{\boldsymbol{U}}^{\mathrm{H}}$
    end if
  end for

  return $\langle \hat{\boldsymbol{U}}, \bar{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}} \rangle$
end function

matrix multiplications, a proper Givens rotation is put into use (cf. Lines 37 to 45). The variables φ_1, φ_2, χ_1 (which depends on the threshold τ_1), χ_2 (which depends on τ_2), and ψ have similar functions as in our previous algorithm variants (cf. Sections 6.1.1 to 6.1.4) and are computed appropriately (cf. Lines 8, 11, and 14 to 16).

6.2 Algorithm correctness and characteristics

In the last section, we have described several variants of a constructive algorithm for solving the relaxed interference alignment decomposition problem. Although we have argued why these algorithm variants actually solve the problem, we have not yet proved that. In this section, we want to prove the correctness of the most advanced of the variants we have discussed, the CRIAD$^{(\mathrm{b})}$ algorithm (cf. Section 6.1.5). Apart from that, we will summarize the different preconditions for our five constructive algorithm variants. Finally, we will analyze how many decomposition steps the algorithm variants need until they have constructed a solution.

Theorem 6.1 (Correctness of the CRIAD$^{(\mathrm{b})}$ algorithm)**.** Let $K \in \mathbb{N}$, $d \in \mathbb{N}$, $M \in \mathbb{N}$, $N \in \mathbb{N}$, and $\boldsymbol{H} \in \mathbb{C}^{KN \times KM}$ be given as algorithm input such that $K \geq 2$, $1 \leq d \leq N$, $1 \leq d \leq M$, $NM > d^2$, and $N + M \geq (K+1)\,d$. Then, the CRIAD$^{(\mathrm{b})}$ algorithm (cf. Algorithm 6.5) solves the relaxed interference alignment decomposition problem (cf. Definition 5.6) and terminates.

Proof. The relaxed interference alignment decomposition problem is solved if and only if the following three conditions are satisfied for $\bar{\boldsymbol{\Sigma}} = (\bar{\sigma}_{ij}) \in \mathbb{C}^{KN \times KM}$, $\hat{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$, and $\hat{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ (cf. Definition 5.6):

1. $\hat{\boldsymbol{U}}^{\mathrm{H}} \boldsymbol{H} \hat{\boldsymbol{V}} = \bar{\boldsymbol{\Sigma}}$,
2. $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ are unitary matrices, and
3. $\boldsymbol{\Sigma} = \bar{\boldsymbol{\Sigma}}(1\!:\!Kd, 1\!:\!Kd)$ is a block-diagonal matrix with main diagonal blocks $\boldsymbol{\Sigma}_k \in \mathbb{C}^{d \times d}$, $k \in \{1, 2, \ldots, K\}$.

The second condition is satisfied if $\hat{\boldsymbol{U}}^{\mathrm{H}}$ and $\hat{\boldsymbol{V}}$ are products of (extended) Householder and Givens matrices (cf. Lemma 5.8 and Corollary 5.9). The third condition is satisfied if at least the elements of $\bar{\boldsymbol{\Sigma}}$ specified by Lemma 5.10 are equal to zero.

We are going to show that four loop invariants, which are derived from the above conditions, are maintained at iteration $\delta \in \{1, 2, \ldots, d-1\}$ of the inner loop in Lines 13 to 35 and at iteration $\kappa \in \{1, 2, \ldots, K-1\}$ of the outer loop in Lines 10 to 46 of Algorithm 6.5. The first two loop invariants are exactly the first two of the above conditions. The third and fourth loop invariants are based on Lemma 5.10. For the inner loop, the third loop

invariant is

$$\forall \kappa' \in \{1, 2, \ldots, \kappa\}\colon\ \forall \delta' \in \{1, 2, \ldots, \delta\}\colon$$
$$\bar{\sigma}_{\kappa' d+1,(\kappa'-1)d+\delta'} = \bar{\sigma}_{\kappa' d+2,(\kappa'-1)d+\delta'} = \cdots = \bar{\sigma}_{Kd,(\kappa'-1)d+\delta'} = 0,$$

and the fourth loop invariant is

$$\forall \kappa' \in \{1, 2, \ldots, \kappa\}\colon\ \forall \delta' \in \{1, 2, \ldots, \delta\}\colon$$
$$\bar{\sigma}_{(\kappa'-1)d+\delta',\kappa' d+1} = \bar{\sigma}_{(\kappa'-1)d+\delta',\kappa' d+2} = \cdots = \bar{\sigma}_{(\kappa'-1)d+\delta',Kd} = 0.$$

For the outer loop, the third loop invariant is

$$\forall \kappa' \in \{1, 2, \ldots, \kappa\}\colon\ \forall \delta \in \{1, 2, \ldots, d\}\colon$$
$$\bar{\sigma}_{\kappa' d+1,(\kappa'-1)d+\delta} = \bar{\sigma}_{\kappa' d+2,(\kappa'-1)d+\delta} = \cdots = \bar{\sigma}_{Kd,(\kappa'-1)d+\delta} = 0,$$

and the fourth loop invariant is

$$\forall \kappa' \in \{1, 2, \ldots, \kappa\}\colon\ \forall \delta \in \{1, 2, \ldots, d\}\colon$$
$$\bar{\sigma}_{(\kappa'-1)d+\delta,\kappa' d+1} = \bar{\sigma}_{(\kappa'-1)d+\delta,\kappa' d+2} = \cdots = \bar{\sigma}_{(\kappa'-1)d+\delta,Kd} = 0.$$

After the initialization of $\hat{\boldsymbol{U}}^{\mathrm{H}}$, $\bar{\boldsymbol{\Sigma}}$, and $\hat{\boldsymbol{V}}$ in Lines 2 to 4 of Algorithm 6.5, it holds that

$$\hat{\boldsymbol{U}}^{\mathrm{H}} \boldsymbol{H} \hat{\boldsymbol{V}} = \boldsymbol{I}_{KN} \boldsymbol{H} \boldsymbol{I}_{KM} = \boldsymbol{H} = \bar{\boldsymbol{\Sigma}}.$$

$\hat{\boldsymbol{U}}^{\mathrm{H}} = \hat{\boldsymbol{U}} = \boldsymbol{I}_{KN}$ as well as $\hat{\boldsymbol{V}} = \boldsymbol{I}_{KM}$ are trivially unitary and no elements of $\bar{\boldsymbol{\Sigma}} = \boldsymbol{H}$ are required to be equal to zero.

Let $\kappa \in \{1, 2, \ldots, K-1\}$ be fixed but arbitrary. Assume that the four loop invariants for the inner loop are satisfied at iteration $\delta - 1$ of the inner loop. Then, the conditions

$$\bar{\sigma}_{\kappa d+1,(\kappa-1)d+\delta} = \bar{\sigma}_{\kappa d+2,(\kappa-1)d+\delta} = \cdots = \bar{\sigma}_{Kd,(\kappa-1)d+\delta} = 0 \tag{6.7}$$

and

$$\bar{\sigma}_{(\kappa-1)d+\delta,\kappa d+1} = \bar{\sigma}_{(\kappa-1)d+\delta,\kappa d+2} = \cdots = \bar{\sigma}_{(\kappa-1)d+\delta,Kd} = 0 \tag{6.8}$$

have to be true for satisfying the third and fourth loop invariant at iteration $\delta \in \{1, 2, \ldots, d-1\}$ of the inner loop.

First, assume $N < M$. Lines 19 and 20 of Algorithm 6.5 are equivalent to

$$\bar{\boldsymbol{\Sigma}} \leftarrow \mathscr{H}_{\chi_1,\psi}^{\varphi_1,\psi} \bar{\boldsymbol{\Sigma}} \tag{6.9}$$

(cf. Algorithm 3.1 and Definition 5.11), where

$$\mathscr{H}_{\chi_1,\psi}^{\varphi_1,\psi} = \mathscr{H}_{\tau_1,\psi}^{\psi,\psi} = \mathscr{H}_{Kd,(\kappa-1)d+\delta}^{(\kappa-1)d+\delta,(\kappa-1)d+\delta} \in \mathbb{C}^{KN \times KN}.$$

It follows that

$$\bar{\sigma}_{(\kappa-1)d+\delta+1,(\kappa-1)d+\delta} = \bar{\sigma}_{(\kappa-1)d+\delta+2,(\kappa-1)d+\delta} = \cdots = \bar{\sigma}_{Kd,(\kappa-1)d+\delta} = 0 \tag{6.10}$$

(cf. Definition 5.11) after iteration δ of the inner loop. Because of

$$(\kappa-1)d+\delta+1 \leq (\kappa-1)d+(d-1)+1 = (\kappa-1)d+d = \kappa d < \kappa d+1,$$

Equation 6.7 is satisfied. Lines 23 and 24 are equivalent to

$$\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}} \mathscr{H}_{\psi,\chi_2}^{\psi,\varphi_2} \tag{6.11}$$

(cf. Algorithm 3.1 and Definition 5.11), where

$$\mathscr{H}_{\psi,\chi_2}^{\psi,\varphi_2} = \mathscr{H}_{\psi,\tau_2-\kappa+1}^{\psi,\psi+1} = \mathscr{H}_{(\kappa-1)d+\delta,Kd+K-1-\kappa+1}^{(\kappa-1)d+\delta,(\kappa-1)d+\delta+1} = \mathscr{H}_{(\kappa-1)d+\delta,Kd+K-\kappa}^{(\kappa-1)d+\delta,(\kappa-1)d+\delta+1} \in \mathbb{C}^{KM\times KM}.$$

It follows that

$$\bar{\sigma}_{(\kappa-1)d+\delta,(\kappa-1)d+\delta+2} = \bar{\sigma}_{(\kappa-1)d+\delta,(\kappa-1)d+\delta+3} = \cdots = \bar{\sigma}_{(\kappa-1)d+\delta,Kd+K-\kappa} = 0 \tag{6.12}$$

(cf. Definition 5.11) after iteration δ of the inner loop. Because of

$$(\kappa-1)d+\delta+2 \leq (\kappa-1)d+(d-1)+2 = (\kappa-1)d+d+1 = \kappa d+1$$

and

$$Kd+K-\kappa \geq Kd+K-(K-1) = Kd+1 > Kd,$$

Equation 6.8 is satisfied. Due to the precondition

$$1 \leq d \leq N \ \wedge \ 1 \leq d \leq M \ \wedge \ NM > d^2,$$

which implicates

$$(M \geq d \ \wedge \ N > d) \ \vee \ (M > d \ \wedge \ N \geq d),$$

and the assumption that $N < M$, it holds that $M \geq d+1$ and thus

$$Kd+K-\kappa \leq Kd+K-1 = K(d+1)-1 \leq KM-1 < KM, \tag{6.13}$$

i.e., the column indices are within valid bounds. Since the first and second loop invariants hold at iteration $\delta-1$ of the inner loop, and thus

$$\bar{\boldsymbol{\Sigma}}(\varphi_1:\chi_1, 1:\psi-1) = \bar{\boldsymbol{\Sigma}}((\kappa-1)d+\delta:Kd, 1:(\kappa-1)d+\delta-1) = \mathbf{0} \tag{6.14}$$

and

$$\bar{\boldsymbol{\Sigma}}(1:\psi-1,\varphi_2:\chi_2)=\bar{\boldsymbol{\Sigma}}(1:(\kappa-1)d+\delta-1,(\kappa-1)d+\delta+1:Kd+K-\kappa)=\mathbf{0}, \quad (6.15)$$

no fill-ins occur due to the Householder transformations in Equations 6.9 and 6.11 (cf. Sections 3.2 and 5.3). Hence, if $N < M$, the third and fourth loop invariants are satisfied at iteration δ of the inner loop. Equations 6.9 and 6.11 combined are equivalent to

$$\bar{\boldsymbol{\Sigma}} \leftarrow \mathcal{H}^{\varphi_1,\psi}_{\chi_1,\psi}\bar{\boldsymbol{\Sigma}}\mathcal{H}^{\psi,\varphi_2}_{\psi,\chi_2}. \quad (6.16)$$

Together with

$$\hat{\boldsymbol{U}}^{\mathrm{H}} \leftarrow \mathcal{H}^{\varphi_1,\psi}_{\chi_1,\psi}\hat{\boldsymbol{U}}^{\mathrm{H}}, \quad (6.17)$$

$$\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}}\mathcal{H}^{\psi,\varphi_2}_{\psi,\chi_2} \quad (6.18)$$

(cf. Lines 21 and 25), and the validity of $\hat{\boldsymbol{U}}^{\mathrm{H}}\boldsymbol{H}\hat{\boldsymbol{V}} = \bar{\boldsymbol{\Sigma}}$ at the beginning of iteration δ, it follows that, if $N < M$, the first loop invariant is satisfied at iteration δ of the inner loop. Because of Corollary 5.9 and Lines 21 and 25, the second loop invariant is satisfied as well at iteration δ of the inner loop, if $N < M$. For $N \geq M$ and Lines 27 to 33 of Algorithm 6.5, an analogous argumentation applies. Therefore, all four loop invariants are satisfied at iteration δ of the inner loop.

Assume now that the four loop invariants are satisfied at iteration $\kappa - 1$ of the outer loop and at iteration $d - 1$ of the inner loop in iteration κ of the outer loop. Then, the conditions

$$\bar{\sigma}_{\kappa d+1,(\kappa-1)d+d} = \bar{\sigma}_{\kappa d+2,(\kappa-1)d+d} = \cdots = \bar{\sigma}_{Kd,(\kappa-1)d+d} = 0 \quad (6.19)$$

and

$$\bar{\sigma}_{(\kappa-1)d+d,\kappa d+1} = \bar{\sigma}_{(\kappa-1)d+d,\kappa d+2} = \cdots = \bar{\sigma}_{(\kappa-1)d+d,Kd} = 0 \quad (6.20)$$

have to be true for satisfying the third and fourth loop invariant at iteration $\kappa \in \{1, 2, \ldots, K-1\}$ of the outer loop. Assume again $N < M$. Then, it follows from Equation 6.10 and

$$(\kappa-1)d+\delta+1 = (\kappa-1)d+d+1 = \kappa d+1 \quad (6.21)$$

that Equation 6.19 is satisfied after iteration d of the inner loop in iteration κ of the outer loop. From Equation 6.12 and

$$(\kappa-1)d+\delta+2 = (\kappa-1)d+d+2 = \kappa d+2,$$

we deduce that $\bar{\sigma}_{(\kappa-1)d+d,\kappa d+1}$ (cf. Equation 6.20) not necessarily is equal to zero after iteration d of the inner loop in iteration κ of the outer loop. Lines 38 and 39 of Algorithm 6.5 are equivalent to

$$\bar{\boldsymbol{\Sigma}} \leftarrow \bar{\boldsymbol{\Sigma}}\mathcal{G}^{\psi,\chi_2}_{\psi,\varphi_2} \quad (6.22)$$

(cf. Algorithm 3.2 and Definition 5.11), where

$$\boldsymbol{\mathcal{G}}_{\psi,\varphi_2}^{\psi,\chi_2} = \boldsymbol{\mathcal{G}}_{\psi,\psi+1}^{\psi,\tau_2-\kappa+1} = \boldsymbol{\mathcal{G}}_{(\kappa-1)d+\delta,(\kappa-1)d+\delta+1}^{(\kappa-1)d+\delta,Kd+K-1-\kappa+1} = \boldsymbol{\mathcal{G}}_{(\kappa-1)d+d,(\kappa-1)d+d+1}^{(\kappa-1)d+d,Kd+K-\kappa} \in \mathbb{C}^{KM\times KM}.$$

Because of Equation 6.21 and Definition 5.11, it follows that $\bar{\sigma}_{(\kappa-1)d+d,\kappa d+1} = 0$. Thus, Equation 6.20 is satisfied after iteration κ of the outer loop. Due to Equation 6.13, the column indices are within valid bounds again. Because of Equations 6.14 and 6.15, the Householder and Givens transformations in Equations 6.9, 6.11, and 6.22 do not cause fill-ins (cf. Sections 3.2 and 5.3). Therefore, if $N < M$, the third and fourth loop invariants are satisfied at iteration κ of the outer loop. Equations 6.16 and 6.22 combined are equivalent to

$$\bar{\boldsymbol{\Sigma}} \leftarrow \boldsymbol{\mathcal{H}}_{\chi_1,\psi}^{\varphi_1,\psi} \bar{\boldsymbol{\Sigma}} \boldsymbol{\mathcal{H}}_{\psi,\chi_2}^{\psi,\varphi_2} \boldsymbol{\mathcal{G}}_{\psi,\varphi_2}^{\psi,\chi_2}.$$

Together with Equations 6.17 and 6.18 as well as

$$\hat{\boldsymbol{V}} \leftarrow \hat{\boldsymbol{V}} \boldsymbol{\mathcal{G}}_{\psi,\varphi_2}^{\psi,\chi_2}$$

(cf. Line 40), and the validity of $\hat{\boldsymbol{U}}^{\mathrm{H}} \boldsymbol{H} \hat{\boldsymbol{V}} = \bar{\boldsymbol{\Sigma}}$ after iteration $d-1$ of the inner loop in iteration κ of the outer loop, it follows that, if $N < M$, the first loop invariant is satisfied at iteration κ of the outer loop. Due to Corollary 5.9 and Line 40, the second loop invariant too is satisfied at iteration κ of the outer loop, if $N < M$. For $N \geq M$ and Lines 42 to 44 of Algorithm 6.5, an analogous argumentation applies again. Therefore, all four loop invariants are satisfied at iteration κ of the outer loop.

It follows that after iteration $K-1$, i.e., after the last iteration, of the outer loop, the loop invariants

$$\begin{aligned}&\forall \kappa \in \{1,2,\ldots,K-1\}\colon\ \forall \delta \in \{1,2,\ldots,d\}\colon\\ &\bar{\sigma}_{\kappa d+1,(\kappa-1)d+\delta} = \bar{\sigma}_{\kappa d+2,(\kappa-1)d+\delta} = \cdots = \bar{\sigma}_{Kd,(\kappa-1)d+\delta} = 0\end{aligned}$$

and

$$\begin{aligned}&\forall \kappa \in \{1,2,\ldots,K-1\}\colon\ \forall \delta \in \{1,2,\ldots,d\}\colon\\ &\bar{\sigma}_{(\kappa-1)d+\delta,\kappa d+1} = \bar{\sigma}_{(\kappa-1)d+\delta,\kappa d+2} = \cdots = \bar{\sigma}_{(\kappa-1)d+\delta,Kd} = 0\end{aligned}$$

are satisfied. These are exactly the conditions required by Lemma 5.10. Hence, the three conditions of Definition 5.6 are fulfilled and the CRIAD$^{(\mathrm{b})}$ algorithm solves the relaxed interference alignment decomposition problem. Since K and d are finite numbers, the algorithm always terminates. □

As we have shown in Theorem 6.1, the CRIAD$^{(\mathrm{b})}$ algorithm supports all combinations of the input parameters d, M, and N in conformance with Equations 6.3 and 6.4. Since this is not true for CRIAD$^{(\mathrm{f})}$ and CRIAD$^{(\mathrm{t})}$, CRIAD$^{(\mathrm{b})}$ is the only algorithm variant that

Table 6.1: Minimal values of K, d, M, and N (cf. Table 4.1) as input parameters for the constructive algorithm variants CRIAD$^{(\mathrm{f})}$, CRIAD$^{(\mathrm{t})}$, and CRIAD$^{(\mathrm{b})}$.

	CRIAD$^{(\mathrm{f})}$ (cf. Algorithm 6.4)	CRIAD$^{(\mathrm{t})}$ (cf. Algorithm 6.3)	CRIAD$^{(\mathrm{b})}$ (cf. Algorithm 6.5)
K	≥ 2	≥ 2	≥ 2
d	≥ 2	≥ 1	≥ 1
M	$\geq 2d - 1 - \frac{d-1}{K}$	$\geq d + 1$	$\geq \begin{cases} d, & \text{if } N \geq d+1 \\ d+1, & \text{otherwise} \end{cases}$
N	$\geq 2d - 1 - \frac{d-1}{K}$	$\geq d + 1$	$\geq \begin{cases} d, & \text{if } M \geq d+1 \\ d+1, & \text{otherwise} \end{cases}$

supports all parameter combinations that are valid for the relaxed interference alignment decomposition problem (cf. Definition 5.6). Table 6.1 summarizes the minimal values of K, d, M, and N as input parameters for the algorithm variants CRIAD$^{(\mathrm{f})}$, CRIAD$^{(\mathrm{t})}$, and CRIAD$^{(\mathrm{b})}$.

After having solved our example problem for $K = 3$, $d = 3$, $M = 7$, and $N = 5$ with five different constructive algorithm variants in Section 6.1, we shall compare the number of decomposition steps required for factorizing $\boldsymbol{H}$ according to Definition 5.6. Because of

$$\bar{\boldsymbol{\Sigma}}^{(14)} = \bar{\boldsymbol{\Sigma}}$$

(cf. Figure 6.5f), the bidiagonal blocks algorithm variant CRIAD$^{(\mathrm{b})}$ needs fourteen decomposition steps. In contrast, the tridiagonal and full blocks algorithm variants CRIAD$^{(\mathrm{t}'')}$, CRIAD$^{(\mathrm{t}')}$, CRIAD$^{(\mathrm{t})}$, and CRIAD$^{(\mathrm{f})}$ necessitate 56 (cf. Figure 6.1u), sixteen (cf. Figure 6.2i), again sixteen (cf. Figure 6.3i), and twenty (cf. Figure 6.4i) decomposition steps, respectively. Therefore, the bidiagonal blocks algorithm variant requires the least number of decomposition steps.

In general, the bidiagonal blocks algorithm variant CRIAD$^{(\mathrm{b})}$ applies one Householder reflection per row and another one per column of each $d \times d$ block. Moreover, one Givens rotation per $d \times d$ block is utilized.

Lemma 6.2 (Decomposition steps of the CRIAD$^{(\mathrm{b})}$ algorithm)**.** The CRIAD$^{(\mathrm{b})}$ algorithm (cf. Algorithm 6.5) needs $(K-1)(2d+1)$ decomposition steps (cf. Definition 5.7) for solving the relaxed interference alignment decomposition problem (cf. Definition 5.6).

Proof. In each of the d iterations of the inner loop in Lines 13 to 35 of Algorithm 6.5,

exactly two Householder reflections are applied (either in Lines 19 to 25 or in Lines 27 to 33). Hence, there are $2d$ Householder reflections per iteration of the outer loop in Lines 10 to 46. Additionally, exactly one Givens rotation is applied in each of the $K-1$ iterations of the outer loop (either in Lines 38 to 40 or in Lines 42 to 44). In total, $2d\,(K-1)$ Householder reflections and $K-1$ Givens rotations are utilized. Thus, the CRIAD$^{(\mathrm{b})}$ algorithm needs $2d\,(K-1)+K-1=(K-1)\,(2d+1)$ decomposition steps. □

Analogous considerations result in $(K-1)\,(2d+2)$ and $(K-1)\,(4d-2)$ decomposition steps for the further enhanced tridiagonal algorithm variant CRIAD$^{(\mathrm{t})}$ and the full blocks algorithm variant CRIAD$^{(\mathrm{f})}$, respectively.

Chapter 7

Direct solution to the general matrix factorization problem

In Chapter 4, we have formulated the interference alignment problem based on the general problem introduction in Chapter 2 and have analyzed an iterative solution to this problem. Next, in Chapter 5, we have reformulated the interference alignment problem as the equivalent interference alignment decomposition problem and have defined the related but simpler relaxed interference alignment decomposition problem. Then, in Chapter 6, we have proposed a direct solution to the latter problem by applying Householder and Givens transformations.

The remaining question to be discussed in this chapter is whether there exists a direct solution to the interference alignment decomposition problem on the basis of Householder reflections and Givens rotations. In Section 7.1, we will argue that products of Givens and Householder matrices in general do not have the form required by the interference alignment decomposition problem, and thus the application of solely these unitary transformations cannot lead to a direct solution to this problem. Subsequently, in Section 7.2, we will briefly sketch some approaches that may be worth investigating in the future.

7.1 Solution based on Householder and Givens transformations

For finding a solution to the interference alignment decomposition problem, a given global channel matrix $\boldsymbol{H} \in \mathbb{C}^{KN \times KM}$ (cf. Definition 5.1) is to be factorized such that

$$\boldsymbol{H} = \bar{\boldsymbol{U}} \bar{\boldsymbol{\Sigma}} \bar{\boldsymbol{V}}^{\mathrm{H}} \tag{7.1}$$

(cf. Equation 5.3), where $\bar{\boldsymbol{\Sigma}} \in \mathbb{C}^{KN \times KM}$ is the extended global interference alignment matrix (cf. Definition 5.4), and $\bar{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$ and $\bar{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ are the extended global postcoding and precoding matrices, respectively (cf. Definition 5.3). Both $\boldsymbol{U} =$

$\bar{\boldsymbol{U}}(:, 1:Kd)$ and $\boldsymbol{V} = \bar{\boldsymbol{V}}(:, 1:Kd)$ have to be sparse block matrices (cf. Definition 5.1), while $\boldsymbol{\Sigma} = \bar{\boldsymbol{\Sigma}}(1:Kd, 1:Kd)$ has to be a block-diagonal matrix (cf. Definition 5.5). In Chapter 6, we have developed an algorithm for obtaining the factorization

$$\boldsymbol{H} = \hat{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^{\mathrm{H}}, \tag{7.2}$$

where the submatrix $\boldsymbol{\Sigma}$ of $\bar{\boldsymbol{\Sigma}}$ is of block-diagonal form. However, $\hat{\boldsymbol{U}}^{\mathrm{H}} \in \mathbb{C}^{KN \times KN}$ and $\hat{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ are products of (extended) Householder and Givens matrices (cf. Definitions 3.2 and 3.3) and therefore general unitary matrices. In particular, we have not yet attempted to bring $\hat{\boldsymbol{U}}(:, 1:Kd)$ and $\hat{\boldsymbol{V}}(:, 1:Kd)$ to the sparse block form postulated for $\boldsymbol{U}$ and $\boldsymbol{V}$. It follows that $\hat{\boldsymbol{U}} \neq \bar{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}} \neq \bar{\boldsymbol{V}}$.

But is it possible to apply Householder reflections and Givens rotations such that $\boldsymbol{H}$ is decomposed as in Equation 7.1 with $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{\Sigma}$ being matrices of the required sparse block forms? In this section, we are going to argue that, in general, this is impossible. Since we understand how Householder and Givens matrices are constructed and how matrices are multiplied, our intuition tells us that products of several (extended) Householder or Givens matrices are supposed to be dense matrices, no matter which strategy is chosen for introducing zeros into the given matrix. This is particularly clear for Householder matrices, as they generally are dense matrices themselves.

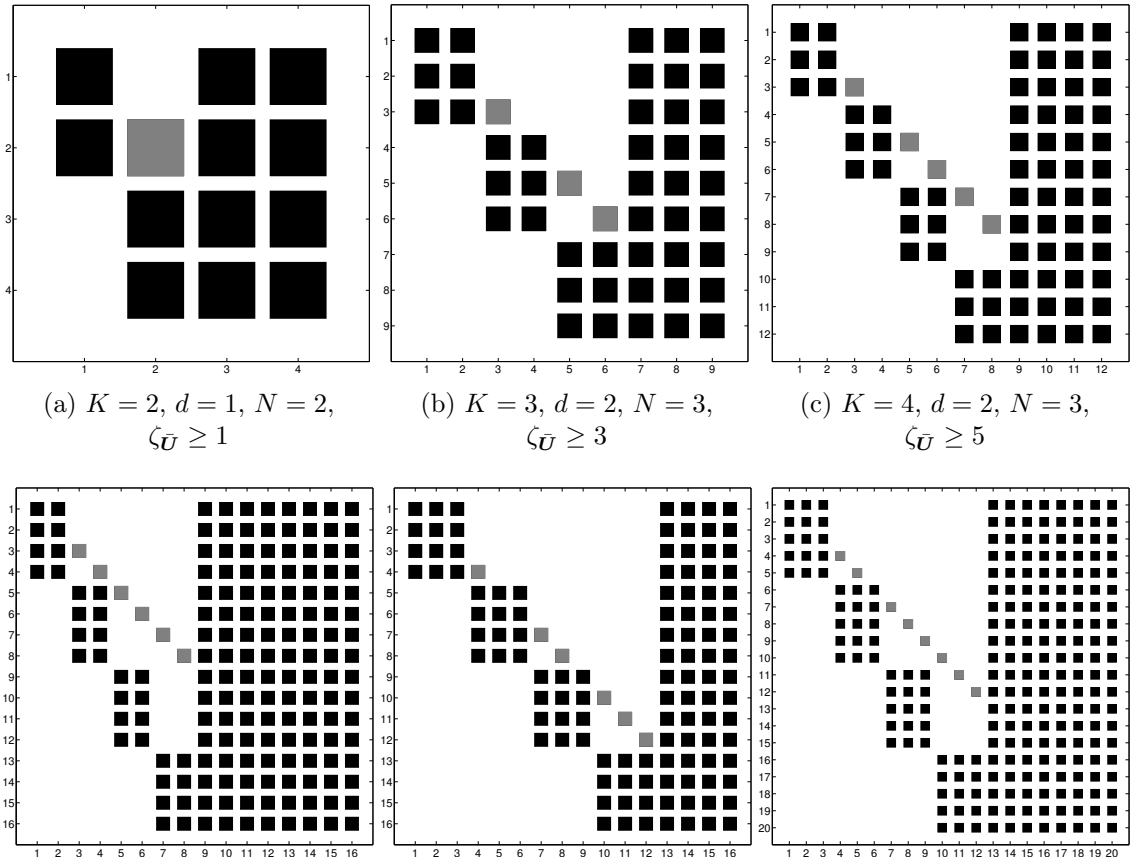
For the purpose of providing a mathematically sound and thus more convincing argument, we want to focus on the strictly zero main diagonal elements of $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ in comparison to the main diagonal elements of (extended) Householder and Givens matrices. The following definition will be expedient for our considerations.

Definition 7.1 (Zero main diagonal elements of a matrix)**.** Let $\boldsymbol{A}$ be a matrix. Then, $\zeta_{\boldsymbol{A}}$ denotes the number of main diagonal elements of $\boldsymbol{A}$ that are equal to zero.

The submatrix $\boldsymbol{U}$ of $\bar{\boldsymbol{U}}$ is a matrix of size $KN \times Kd$ with blocks $\boldsymbol{U}_j$, $j \in \{1, 2, \ldots, K\}$, of size $N \times d$ such that

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{U}_1 & & & \boldsymbol{0} \\ & \boldsymbol{U}_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \boldsymbol{U}_K \end{pmatrix} \tag{7.3}$$

(cf. Definition 5.1). We are regarding $\bar{\boldsymbol{U}}$ for $K \geq 2$ and $N > d \geq 1$ (cf. Table 4.1). Figure 7.1 shows a few examples of $\bar{\boldsymbol{U}}$, in which (potentially) nonzero elements are illustrated as black squares and strictly zero elements as white space. Strictly zero main diagonal elements are highlighted in gray. The smallest possible matrix $\bar{\boldsymbol{U}}$ for our parameters K, d, and N is visualized in Figure 7.1a. It has one strictly zero main diagonal element. Figures 7.1a to 7.1f indicate that the number of strictly zero main diagonal elements is, at least approximately, directly proportional to the size of $\bar{\boldsymbol{U}}$. The following lemma states a



(a) $K = 2$, $d = 1$, $N = 2$, $\zeta_{\bar{\boldsymbol{U}}} \geq 1$

(b) $K = 3$, $d = 2$, $N = 3$, $\zeta_{\bar{\boldsymbol{U}}} \geq 3$

(c) $K = 4$, $d = 2$, $N = 3$, $\zeta_{\bar{\boldsymbol{U}}} \geq 5$

(d) $K = 4$, $d = 2$, $N = 4$, $\zeta_{\bar{\boldsymbol{U}}} \geq 6$

(e) $K = 4$, $d = 3$, $N = 4$, $\zeta_{\bar{\boldsymbol{U}}} \geq 6$

(f) $K = 4$, $d = 3$, $N = 5$, $\zeta_{\bar{\boldsymbol{U}}} \geq 8$

Figure 7.1: Main diagonal elements of the extended global postcoding matrix $\bar{\boldsymbol{U}}$ that are equal to zero.

lower bound for $\zeta_{\bar{\boldsymbol{U}}}$, which depends on K, d, and N.

Lemma 7.2 (Zero main diagonal elements of $\bar{\boldsymbol{U}}$)**.** Let $K \geq 2$ and $N > d \geq 1$ (cf. Table 4.1). Then, it holds for the extended global postcoding matrix $\bar{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$ (cf. Definition 5.3) that

$$\zeta_{\bar{\boldsymbol{U}}} \geq \sum_{k=2}^{K} d - \max(0,\, kd - (k-1)N) > 0$$

(cf. Definition 7.1).

Proof. According to Definitions 5.1 and 5.3, all elements of the submatrix $\boldsymbol{U} \in \mathbb{C}^{KN \times Kd}$ of $\bar{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$ except for the blocks $\boldsymbol{U}_k \in \mathbb{C}^{N \times d}$, $k \in \{1, 2, \ldots, K\}$, are equal to zero. All (potentially) nonzero elements in one row or column of $\boldsymbol{U}$ are elements of exactly one $\boldsymbol{U}_k$. Since we assume $N > d$, each main diagonal element $\bar{u}_{ii}$ of $\bar{\boldsymbol{U}}$, $i \in \{1, 2, \ldots, KN\}$, is either a (potentially) nonzero element of or a strictly zero element above the $\boldsymbol{U}_k$ in column i of $\bar{\boldsymbol{U}}$.

As each $\boldsymbol{U}_k$ has d columns, there are at most d main diagonal elements of $\bar{\boldsymbol{U}}$ above each $\boldsymbol{U}_k$. Again because of $N > d$, there is at least one main diagonal element of $\bar{\boldsymbol{U}}$ above each $\boldsymbol{U}_k$, with the exception of the top-left block $\boldsymbol{U}_1$, which always contains the first d main diagonal elements of $\bar{\boldsymbol{U}}$. If $N \geq 2d$, the d main diagonal elements $\bar{u}_{d+1,d+1}, \bar{u}_{d+2,d+2}, \ldots, \bar{u}_{2d,2d}$ are strictly zero elements above $\boldsymbol{U}_2$. Put differently, if $d < N < 2d$, exactly $2d - N$ main diagonal elements of $\bar{\boldsymbol{U}}$ are elements of $\boldsymbol{U}_2$. Hence, there are $d - \max(0,\, 2d - N)$ strictly zero main diagonal elements of $\bar{\boldsymbol{U}}$ above $\boldsymbol{U}_2$.

A similar argument holds for $\boldsymbol{U}_3$. If $2N \geq 3d$, the d main diagonal elements $\bar{u}_{2d+1,2d+1}, \bar{u}_{2d+2,2d+2}, \ldots, \bar{u}_{3d,3d}$ are strictly zero elements above $\boldsymbol{U}_3$. Thus, there are $d - \max(0,\, 3d - 2N)$ strictly zero main diagonal elements of $\bar{\boldsymbol{U}}$ above $\boldsymbol{U}_3$. Therefore, by induction, there are $d - \max(0,\, kd - (k-1)N)$ strictly zero main diagonal elements of $\bar{\boldsymbol{U}}$ above $\boldsymbol{U}_k$, $k \in \{2, 3, \ldots, K\}$, and in total

$$\sum_{k=2}^{K} d - \max(0,\, kd - (k-1)N)$$

strictly zero main diagonal elements of $\bar{\boldsymbol{U}}$. Since other main diagonal elements of $\bar{\boldsymbol{U}}$ may also be equal to zero, it holds that

$$\zeta_{\bar{\boldsymbol{U}}} \geq \sum_{k=2}^{K} d - \max(0,\, kd - (k-1)N).$$

Because of $N > d$, $k \geq 2$, and

$$\begin{aligned} kd - (k-1)N &= kd - kN + N \\ &= kd - kN + N + 2d - 2d + N - N \\ &= 2d - N - (kN - kd - 2N + 2d) \\ &= 2d - N - (k-2)(N-d), \end{aligned}$$

it follows that

$$kd - (k-1)N \leq 2d - N < 2d - d = d$$

and thus

$$d - \max(0,\, kd - (k-1)N) > 0.$$

Hence, the assertion of the lemma holds. □

For decomposing $\boldsymbol{H}$ as in Equation 7.1 solely through applications of Householder and Givens transformations, $\boldsymbol{H}$ is premultiplied by a product of (extended) Householder and Givens matrices that must be equal to $\bar{\boldsymbol{U}}^{\mathrm{H}}$ (cf. Lemma 5.8, which analogically shows that $\hat{\boldsymbol{U}}^{\mathrm{H}}$ is a product of Householder and Givens matrices). Hence, we are actually interested in the number of strictly zero main diagonal elements of $\bar{\boldsymbol{U}}^{\mathrm{H}}$. The following corollary

transfers the result of Lemma 7.2 to $\zeta_{\bar{\boldsymbol{U}}^{\mathrm{H}}}$.

Corollary 7.3 (Zero main diagonal elements of $\bar{\boldsymbol{U}}^{\mathrm{H}}$)**.** Let $K \geq 2$ and $N > d \geq 1$ (cf. Table 4.1). Then, it holds for the conjugate transpose $\bar{\boldsymbol{U}}^{\mathrm{H}}$ of the extended global postcoding matrix $\bar{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$ (cf. Definition 5.3) that

$$\zeta_{\bar{\boldsymbol{U}}^{\mathrm{H}}} \geq \sum_{k=2}^{K} d - \max(0,\, kd - (k-1)N) > 0$$

(cf. Definition 7.1).

Proof. Since the main diagonal elements of $\bar{\boldsymbol{U}}^{\mathrm{H}}$ are the complex conjugates of the main diagonal elements of the square matrix $\bar{\boldsymbol{U}}$, and the complex conjugate of zero is zero, the assertion readily follows from Lemma 7.2. □

Because the submatrix $\boldsymbol{V}$ of $\bar{\boldsymbol{V}}$ is a matrix of size $KM \times Kd$ with blocks $\boldsymbol{V}_j$, $j \in \{1, 2, \ldots, K\}$, of size $M \times d$ such that

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{V}_1 & & & \boldsymbol{0} \\ & \boldsymbol{V}_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \boldsymbol{V}_K \end{pmatrix} \tag{7.4}$$

(cf. Definition 5.1), and we are considering $\bar{\boldsymbol{V}}$ for $K \geq 2$ and $M > d \geq 1$ (cf. Table 4.1), the extended global postcoding and precoding matrices $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ have similar characteristics. For this reason, Figure 7.1 and Lemma 7.2 analogously apply to $\bar{\boldsymbol{V}}$.

Corollary 7.4 (Zero main diagonal elements of $\bar{\boldsymbol{V}}$)**.** Let $K \geq 2$ and $M > d \geq 1$ (cf. Table 4.1). Then, it holds for the extended global precoding matrix $\bar{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ (cf. Definition 5.3) that

$$\zeta_{\bar{\boldsymbol{V}}} \geq \sum_{k=2}^{K} d - \max(0,\, kd - (k-1)M) > 0$$

(cf. Definition 7.1).

Proof. As stated in Definitions 5.1 and 5.3, the extended global precoding matrix $\bar{\boldsymbol{V}} \in \mathbb{C}^{KM \times KM}$ is defined similarly to the extended global postcoding matrix $\bar{\boldsymbol{U}} \in \mathbb{C}^{KN \times KN}$, the only difference is the replacement of N by M. Thus, the assertion readily follows from Lemma 7.2. □

Due to Corollaries 7.3 and 7.4, we are familiar with the lower bounds of $\zeta_{\bar{\boldsymbol{U}}^{\mathrm{H}}}$ and $\zeta_{\bar{\boldsymbol{V}}}$. In particular, we understand that $\bar{\boldsymbol{U}}^{\mathrm{H}}$ and $\bar{\boldsymbol{V}}$ always contain at least one strictly zero main diagonal element for $K \geq 2$, $M > d \geq 1$, and $N > d \geq 1$. Let us now turn to the

main diagonal elements of a Givens matrix $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{G}}(\theta, \varphi, j_1, j_2) \in \mathbb{C}^{n \times n}$, where θ and φ are angles, and j_1 and j_2 are the indices of the elements affected by the Givens rotation (cf. Definition 3.3).

Lemma 7.5 (Zero main diagonal elements of $\boldsymbol{\mathcal{G}}$)**.** Let the rotation angle $\theta \neq \frac{2a+1}{2}\pi$, $a \in \mathbb{Z}$, be given. Moreover, let $1 \leq j_1 \neq j_2 \leq n$. Then, it holds for the Givens matrix $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{G}}(\theta, \varphi, j_1, j_2) \in \mathbb{C}^{n \times n}$ (cf. Definition 3.3) that

$$\zeta_{\boldsymbol{\mathcal{G}}} = 0$$

(cf. Definition 7.1).

Proof. It follows directly from Definition 3.3 for each main diagonal element g_{jj} of $\boldsymbol{\mathcal{G}}$, $j \in \{1, 2, \ldots, n\}$, that

$$g_{jj} = 1 \ \vee \ g_{jj} = \cos(\theta).$$

Because of

$$\forall a \in \mathbb{Z}\colon \ \forall \theta \neq \frac{2a+1}{2}\pi\colon \ \cos(\theta) \neq 0,$$

the assertion of the lemma holds. □

According to Lemma 7.5, $\zeta_{\boldsymbol{\mathcal{G}}} = 0$ if the rotation angle $\theta \neq \frac{2a+1}{2}\pi$ for $a \in \mathbb{Z}$. However, the Givens transformation $\boldsymbol{\mathcal{G}}^{\mathrm{H}}\boldsymbol{A} = \boldsymbol{B}$ for matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{G}}(\theta, \varphi, j_1, j_2)$ with $\theta = \frac{2a+1}{2}\pi$, $a \in \mathbb{Z}$, i.e., $\cos(\theta) = 0$ and $\sin(\theta) = 1$, implies that $\boldsymbol{B}(j_1 : j_1, :) = -\boldsymbol{A}(j_2 : j_2, :)e^{i\varphi}$ and $\boldsymbol{B}(j_2 : j_2, :) = \boldsymbol{A}(j_1 : j_1, :)e^{-i\varphi}$, while the other rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ are pairwise equal to each other. Likewise, $\boldsymbol{A}\boldsymbol{\mathcal{G}} = \boldsymbol{B}$ means that $\boldsymbol{B}(:, j_1 : j_1) = -\boldsymbol{A}(:, j_2 : j_2)e^{-i\varphi}$ and $\boldsymbol{B}(:, j_2 : j_2) = \boldsymbol{A}(:, j_1 : j_1)e^{i\varphi}$ for $\theta = \frac{2a+1}{2}\pi$, $a \in \mathbb{Z}$, while the other columns of $\boldsymbol{A}$ and $\boldsymbol{B}$ are pairwise equal to each other. Therefore, and since $ze^{i\varphi} \neq 0$ and $ze^{-i\varphi} \neq 0$ for all $z \neq 0$, no zeros are introduced for the rotation angle $\theta = \frac{2a+1}{2}\pi$, $a \in \mathbb{Z}$. It follows that such Givens matrices most likely serve no purpose for solving the interference alignment decomposition problem.

The subsequent lemma concerns the main diagonal elements of an extended Householder matrix $\tilde{\boldsymbol{\mathcal{H}}} = \tilde{\boldsymbol{\mathcal{H}}}(\boldsymbol{z}, j_1, j_2) \in \mathbb{C}^{n \times n}$, where $\boldsymbol{z}$ is the vector for element annihilations, and j_1 and j_2 are the indices of the first and last elements involved in the Householder transformation (cf. Definition 3.2).

Lemma 7.6 (Zero main diagonal elements of $\tilde{\boldsymbol{\mathcal{H}}}$)**.** Let $\boldsymbol{z} = (z_1 \ z_2 \ \ldots \ z_m)^{\mathrm{T}} \in \mathbb{C}^m$ with $z_1 = re^{i\varphi}$ and $r, \varphi \in \mathbb{R}$ be given such that

$$\frac{|z_1 \pm \|\boldsymbol{z}\|_2 e^{i\varphi}|}{\|\boldsymbol{z} \pm \|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1\|_2} \neq \frac{1}{4} \text{ and } \forall j \in \{2, 3, \ldots, m\}\colon \ \frac{|z_j|}{\|\boldsymbol{z} \pm \|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1\|_2} \neq \frac{1}{4}.$$

Furthermore, let $m \leq n$, $1 \leq j_1 < j_2 \leq n$, and $j_2 - j_1 + 1 = m$. Then, it holds for the

extended Householder matrix $\tilde{\boldsymbol{\mathcal{H}}} = \tilde{\boldsymbol{\mathcal{H}}}(\boldsymbol{z}, j_1, j_2) \in \mathbb{C}^{n\times n}$ (cf. Definition 3.2) that

$$\zeta_{\tilde{\boldsymbol{\mathcal{H}}}} = 0$$

(cf. Definition 7.1).

Proof. In agreement with Definition 3.1, $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}(\boldsymbol{z}) \in \mathbb{C}^{m\times m}$ is defined as

$$\boldsymbol{\mathcal{H}} \coloneqq \boldsymbol{I}_m - \beta \boldsymbol{\hbar}\boldsymbol{\hbar}^{\mathrm{H}},$$

where $\beta \coloneqq \frac{2}{\boldsymbol{\hbar}^{\mathrm{H}}\boldsymbol{\hbar}}$ and $\boldsymbol{\hbar} = (\hbar_1\ \hbar_2\ \ldots\ \hbar_m)^{\mathrm{T}} \coloneqq \boldsymbol{z} \pm \|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1$. It follows for each main diagonal element $\hbar_{jj}$ of $\boldsymbol{\mathcal{H}}$, $j \in \{1, 2, \ldots, m\}$, that

$$\hbar_{jj} = 1 - \beta\hbar_j\overline{\hbar_j} = 1 - \frac{2}{\boldsymbol{\hbar}^{\mathrm{H}}\boldsymbol{\hbar}}|\hbar_j|^2.$$

Hence, $\hbar_{jj} \neq 0$ if and only if

$$\begin{aligned} & 1 - \frac{2}{\boldsymbol{\hbar}^{\mathrm{H}}\boldsymbol{\hbar}}|\hbar_j|^2 \neq 0 \\ \Leftrightarrow \quad & \frac{|\hbar_j|^2}{\boldsymbol{\hbar}^{\mathrm{H}}\boldsymbol{\hbar}} \neq \frac{1}{2} \\ \Leftrightarrow \quad & \frac{|\hbar_j|}{\|\boldsymbol{\hbar}\|_2} \neq \frac{1}{4}. \end{aligned}$$

Because of $|\hbar_1| = |z_1 \pm \|\boldsymbol{z}\|_2 e^{i\varphi}|$, $|\hbar_j| = |z_j|$ for $j \in \{2, 3, \ldots, m\}$, and $\|\boldsymbol{\hbar}\|_2 = \|\boldsymbol{z} \pm \|\boldsymbol{z}\|_2 e^{i\varphi}\boldsymbol{e}_1\|_2$, it holds that

$$\forall j \in \{1, 2, \ldots, m\}\colon\ \hbar_{jj} \neq 0.$$

Since the main diagonal elements of $\tilde{\boldsymbol{\mathcal{H}}}$ that are not main diagonal elements of $\boldsymbol{\mathcal{H}}$ are equal to one (cf. Definition 3.2), the assertion of the lemma is true. □

All elements of the given global channel matrix $\boldsymbol{H} \in \mathbb{C}^{KN\times KM}$ are completely arbitrary. For this reason, all elements of each vector $\boldsymbol{z} = \boldsymbol{H}(j_1 : j_2, k : k)$, $1 \leq j_1 < j_2 \leq KN$, $1 \leq k \leq KM$, or $\boldsymbol{z} = \boldsymbol{H}(k : k, j_1 : j_2)^{\mathrm{H}}$, $1 \leq j_1 < j_2 \leq KM$, $1 \leq k \leq KN$, are completely arbitrary as well. Therefore, in almost all cases, the conditions of Lemma 3.1 for $\boldsymbol{z}$ are met, and the extended Householder matrix $\tilde{\boldsymbol{\mathcal{H}}} = \tilde{\boldsymbol{\mathcal{H}}}(\boldsymbol{z}, j_1, j_2) \in \mathbb{C}^{KN\times KN}$ or $\tilde{\boldsymbol{\mathcal{H}}} = \tilde{\boldsymbol{\mathcal{H}}}(\boldsymbol{z}, j_1, j_2) \in \mathbb{C}^{KM\times KM}$ that is applied first does not contain any main diagonal elements that are equal to zero. Since the elements of $\boldsymbol{H}$ are arbitrary, the elements of $\bar{\boldsymbol{\Sigma}}^{(\lambda)}$, $\lambda \in \{1, 2, \ldots, \Lambda\}$ (cf. Definition 5.7), can also be considered arbitrary (except for the elements that have been annihilated in decomposition steps $1, 2, \ldots, \lambda - 1$, which most likely are not affected by any further Householder transformation). Thus, in almost all cases, all extended Householder matrices utilized for factorizing a given $\boldsymbol{H}$ do not contain any main diagonal elements that are equal to zero.

As previously noted, $\bar{\boldsymbol{U}}^{\mathrm{H}}$, and for the same reason $\bar{\boldsymbol{V}}$, have to be products of (extended)

Householder and Givens matrices when these are the only allowed transformations for factorizing $\boldsymbol{H}$ according to Equation 7.1. For a product $\boldsymbol{AB} = \boldsymbol{C}$, where $\boldsymbol{A} = (a_{ij}) \in \mathbb{C}^{n\times n}$, $\boldsymbol{B} = (b_{ij}) \in \mathbb{C}^{n\times n}$, and $\boldsymbol{C} = (c_{ij}) \in \mathbb{C}^{n\times n}$ are matrices with $\zeta_{\boldsymbol{A}} = \zeta_{\boldsymbol{B}} = 0$, it holds that $c_{ii} = \sum_{j=1}^{n} a_{ij}b_{ji}$ and $a_{ii}b_{ii} \neq 0$. It follows that $c_{ii} = 0$ if and only if

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}b_{ji} = -a_{ii}b_{ii}, \tag{7.5}$$

where $-a_{ii}b_{ii} \neq 0$. If $\boldsymbol{A}$ and $\boldsymbol{B}$ themselves are products of (extended) Householder and Givens matrices, it is very unlikely that Equation 7.5 is fulfilled for any $i \in \{1, 2, \ldots, n\}$. For example, let $\boldsymbol{A} = \mathcal{G}^{\mathrm{H}} = \mathcal{G}(\theta, \varphi, j_1, j_2)^{\mathrm{H}}$ (cf. Definition 3.3). Then, we get

$$c_{ii} = \begin{cases} b_{j_1,j_1}\cos(\theta) - b_{j_2,j_1}\sin(\theta)e^{i\varphi}, & \text{if } i = j_1 \\ b_{j_2,j_2}\cos(\theta) + b_{j_1,j_2}\sin(\theta)e^{-i\varphi}, & \text{if } i = j_2 \\ b_{ii} \neq 0, & \text{otherwise.} \end{cases} \tag{7.6}$$

Based on these considerations, it is plausible to assume that in almost every case relevant for solving the interference alignment decomposition problem, products of (extended) Householder and Givens matrices, which are highly dependent on the completely arbitrary input matrix $\boldsymbol{H}$, do not contain a single main diagonal element that is equal to zero.

Because of Corollaries 7.3 and 7.4, which state that $\bar{\boldsymbol{U}}^{\mathrm{H}}$ and $\bar{\boldsymbol{V}}$ contain at least one, and in most cases even more, strictly zero main diagonal elements, we deduce that in general $\bar{\boldsymbol{U}}^{\mathrm{H}}$ and $\bar{\boldsymbol{V}}$ cannot be products of (extended) Householder and Givens matrices. For this reason, there is no general direct solution to the interference alignment decomposition problem (cf. Definition 5.5) solely based on Householder and Givens transformations.

7.2 Possible alternatives

Alternatively, or in addition to applying Householder reflections and Givens rotations, other strategies may be pursued that could lead to a direct solution to the interference alignment decomposition problem. These include, but are not limited to, the following:

- apply unitary transformations other than Householder and Givens transformations,
- if the condition number can be estimated, apply non-unitary bijective linear transformations, and
- combine these approaches.

Another (part of a) strategy could be to apply Householder reflections and Givens rotations in two stages. At least for certain values of K, d, M, and N, the results $\hat{\boldsymbol{U}}$,

$\bar{\boldsymbol{\Sigma}}$, and $\hat{\boldsymbol{V}}$ of Algorithms 6.1 to 6.5 could be input parameters for another CRIAD$^{(\mathrm{b})}$-like algorithm that factorizes $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}^{\mathrm{H}}$ such that $\hat{\boldsymbol{U}} = \boldsymbol{A}\bar{\boldsymbol{U}}\boldsymbol{B}$ and $\hat{\boldsymbol{V}}^{\mathrm{H}} = \boldsymbol{C}\bar{\boldsymbol{V}}^{\mathrm{H}}\boldsymbol{D}$, and thus

$$\boldsymbol{H} = \boldsymbol{A}\bar{\boldsymbol{U}}\boldsymbol{B}\bar{\boldsymbol{\Sigma}}\boldsymbol{C}\bar{\boldsymbol{V}}^{\mathrm{H}}\boldsymbol{D}, \tag{7.7}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are unitary matrices, and $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$ are unitary matrices in accordance with Definition 5.3. However, by itself, this result is not advantageous, but it could be combined with other approaches, for example in a hybrid strategy that integrates a direct with an iterative algorithm in order to reduce the number of iterations until the iterative algorithm converges.

Chapter 8

Implementation and experiments

This chapter describes prototype Matlab/Octave implementations of the previously discussed iterative and constructive algorithms for solving the interference alignment problem and the relaxed interference alignment decomposition problem (cf. Chapters 4 and 6). Despite the fact that these algorithms only solve a related and not the same problem, the operation counts of their implementations—based on numerical experiments for the iterative algorithm—are compared in order to show the relevance and potential of a direct solution to the interference alignment decomposition problem.

In Sections 8.1 and 8.2, these iterative and direct implementations, the source code of which is listed in Appendix A, as well as some critical implementation aspects are explained. Subsequently, in Section 8.3, the results of two series of numerical experiments with the iterative implementation are illustrated. Eventually, in Section 8.4, the iterative implementation is contrasted with the direct implementation in respect of operation counts for particular input parameters.

8.1 Iterative implementation

In Chapter 4, we have studied an algorithm that provides an iterative solution to the interference alignment problem. The corresponding iterative prototype implementation consists of two Matlab/Octave source code files:

- *iterative.m* (cf. Listing A.1) requests user input for the parameters K, d, M, and N, specifies the maximum remaining leakage interference and the maximum number of iterations, generates random complex channel matrices, calls the function which implements the iterative algorithm, and visualizes the results (cf. Listing A.13), while

- *iia.m* (cf. Listing A.2) is the actual implementation of the symmetric variant of the distributed iterative optimization algorithm for solving the interference alignment problem (cf. Algorithm 4.2).

Convergence is achieved if and when the remaining leakage interference

$$\sum_{k=1}^{K} |I_k|, \tag{8.1}$$

where I_k is the total leakage interference at receiver R_k (cf. Equation 4.1), is less than or equal the specified maximum remaining leakage interference. As a second stopping criterion, a maximum number of iterations is utilized. Hence, the iteration stops in any case, even if the leakage interference remains too high after the defined maximum number of iterations.

8.2 Direct implementation

Another Matlab/Octave prototype has been implemented on the basis of the constructive algorithm for solving the relaxed interference alignment decomposition problem (cf. Chapter 6). This direct implementation encompasses the following Matlab/Octave source code files:

- *direct.m* (cf. Listing A.3) lets the user decide which algorithm variant and input parameters K, d, M, and N to use, generates a random complex global channel matrix, calls the function which implements the appropriate constructive algorithm, and visualizes the results (cf. Listing A.13),
- *criadt__.m* (cf. Listing A.4) is the implementation of the basic tridiagonal blocks constructive algorithm variant CRIAD$^{(\mathrm{t}'')}$ (cf. Algorithm 6.1),
- *criadt_.m* (cf. Listing A.5) is the implementation of the enhanced tridiagonal blocks constructive algorithm variant CRIAD$^{(\mathrm{t}')}$ (cf. Algorithm 6.2),
- *criadt.m* (cf. Listing A.6) is the implementation of the further enhanced tridiagonal blocks constructive algorithm variant CRIAD$^{(\mathrm{t})}$ (cf. Algorithm 6.3),
- *criadf.m* (cf. Listing A.7) is the implementation of the full blocks constructive algorithm variant CRIAD$^{(\mathrm{f})}$ (cf. Algorithm 6.4),
- *criadb.m* (cf. Listing A.8) is the implementation of the bidiagonal blocks constructive algorithm variant CRIAD$^{(\mathrm{b})}$ (cf. Algorithm 6.5),
- *prehouse.m* (cf. Listing A.9) premultiplies a matrix by an (extended) Householder matrix,
- *posthouse.m* (cf. Listing A.10) postmultiplies a matrix by an (extended) Householder matrix,
- *pregivens.m* (cf. Listing A.11) premultiplies a matrix by a Givens matrix, and

- *postgivens.m* (cf. Listing A.12) postmultiplies a matrix by a Givens matrix.

For applying a Householder reflection, the Householder vector $\boldsymbol{\hbar}$ and the factor β (cf. Definition 3.1) are obtained through a Matlab/Octave library function, which performs a numerically stable computation that differs from Algorithm 3.1. Furthermore, in conformance with [GV13, p. 236], the structure of Householder matrices is exploited in order to reduce the number of operations required for a Householder matrix premultiplication or postmultiplication.

Likewise, for applying a Givens rotation, the matrix coefficients c, s, and $-\overline{s}$ (cf. Definition 3.3) are computed by a Matlab/Octave library function. By exploiting the simple structure of a Givens matrix, the elements of only two rows or two columns are multiplied by these Givens matrix coefficients [GV13, p. 241].

8.3 Numerical experiments

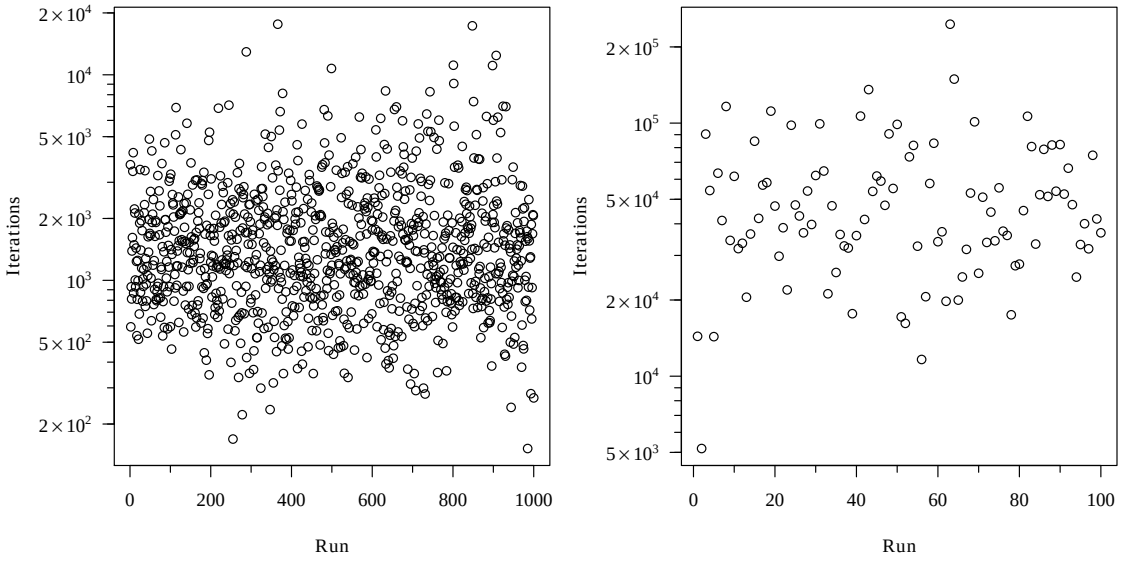
After briefly describing the iterative implementation in Section 8.1, two series of numerical experiments with this implementation and their results shall be presented in this section. The parameter values for these numerical experiments are listed in Table 8.1. In all experimental runs, the elements of the channel matrices were uniformly distributed random complex numbers z with $\Re(z) \in (-10, 10)$ and $\Im(z) \in (-10, 10)$. As stated in Table 8.1, the maximum remaining leakage interference was set to 10^{-5}, i.e., the iteration did not stop until

$$\sum_{k=1}^{K} |I_k| \leq 10^{-5} \tag{8.2}$$

(cf. Equation 8.1). The maximum number of iterations was set to a value large enough so that Equation 8.2 was always fulfilled before the maximum number of iterations was reached.

Table 8.1: The parameter values for the numerical experiments with the iterative implementation.

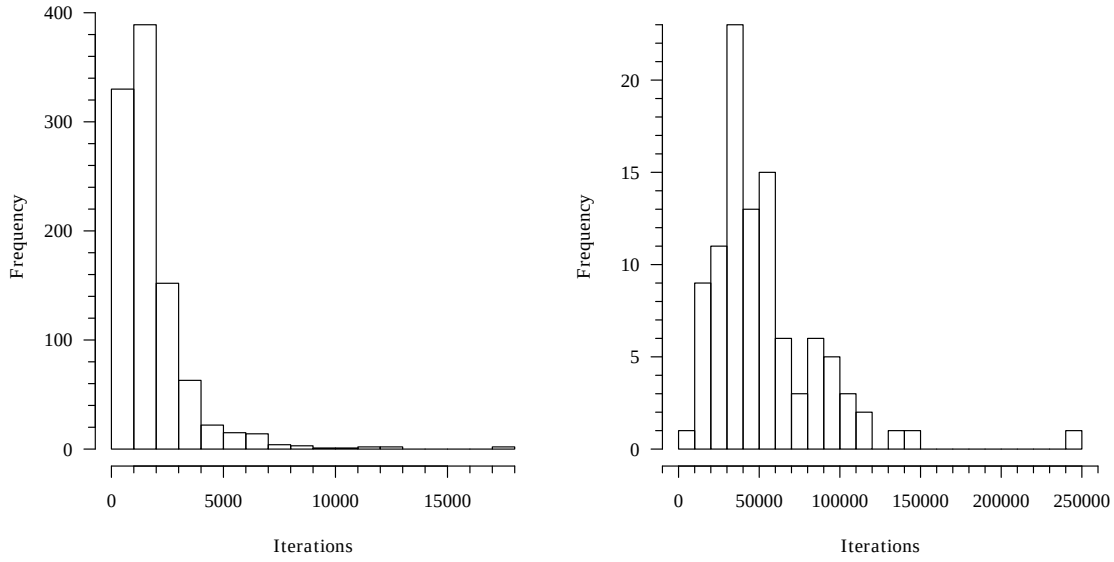
	First series	Second series
K	4	6
d	2	4
M	4	12
N	6	16
Maximum remaining leakage interference	10^{-5}	10^{-5}
Maximum number of iterations	50000	500000
Experimental runs	1000	100



(a) First series (b) Second series

Figure 8.1: The number of iterations required to reach remaining leakage interference of less than or equal 10^{-5} for each experimental run.

The purpose of the numerical experiments has been to find out how many iterations are required until Equation 8.2 is satisfied. Let $\boldsymbol{\iota}_1 \in \mathbb{N}^{1000}$ and $\boldsymbol{\iota}_2 \in \mathbb{N}^{100}$ denote the numbers of iterations measured in the first and second series of experimental runs. Figure 8.1 illustrates these measured values per experimental run. Histograms with bin widths of 1000 and 10000 numbers of iterations for the first and second experimental series are



(a) First series (b) Second series

Figure 8.2: Histograms of the number of iterations required to reach remaining leakage interference of less than or equal 10^{-5}.

Table 8.2: Summary statistics of the number of iterations required to reach remaining leakage interference of less than or equal 10^{-5}.

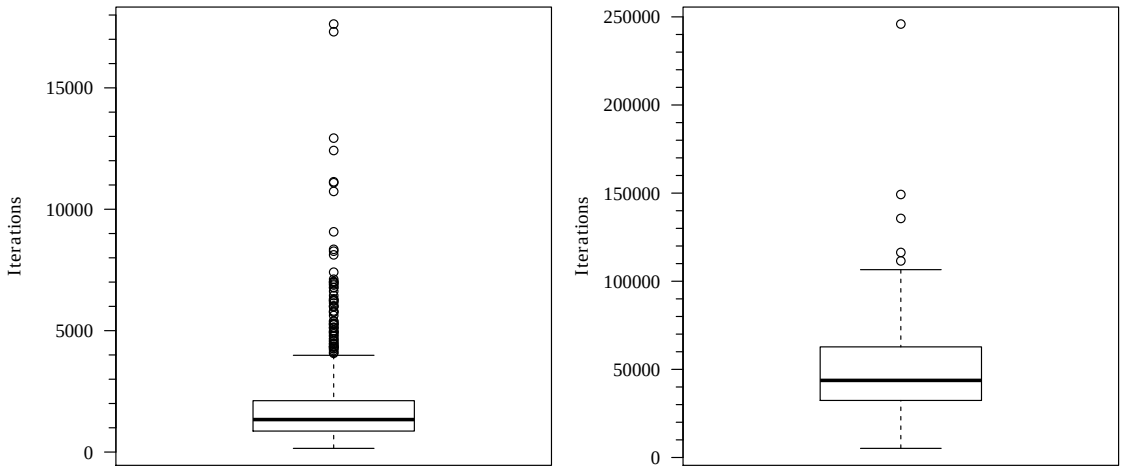
	First series	Second series
Minimum	$\min(\boldsymbol{\iota}_1) = 152$	$\min(\boldsymbol{\iota}_1) = 5179$
Maximum	$\max(\boldsymbol{\iota}_1) = 17629$	$\max(\boldsymbol{\iota}_2) = 245935$
Arithmetic mean	$\bar{\boldsymbol{\iota}}_1 \approx 1798$	$\bar{\boldsymbol{\iota}}_2 \approx 52699$
Lower quartile	$Q_1(\boldsymbol{\iota}_1) = 864$	$Q_1(\boldsymbol{\iota}_2) \approx 32367$
Median	$Q_2(\boldsymbol{\iota}_1) = 1338$	$Q_2(\boldsymbol{\iota}_2) = 43740$
Upper quartile	$Q_3(\boldsymbol{\iota}_1) = 2116$	$Q_3(\boldsymbol{\iota}_2) \approx 62758$
Interquartile range	$\mathrm{IQR}(\boldsymbol{\iota}_1) = 1252$	$\mathrm{IQR}(\boldsymbol{\iota}_2) = 30391$
Variance	$\sigma^2(\boldsymbol{\iota}_1) \approx 2679534$	$\sigma^2(\boldsymbol{\iota}_2) \approx 1187985596$
Standard deviation	$\sigma(\boldsymbol{\iota}_1) \approx 1637$	$\sigma(\boldsymbol{\iota}_2) \approx 34467$

shown in Figure 8.2. The histograms reveal a skewed right and presumably unimodal data distribution. Table 8.2 lists summary statistics for $\boldsymbol{\iota}_1$ and $\boldsymbol{\iota}_2$. It follows that, on average,

$$\iota_1 = 1798 \pm 1637 \tag{8.3}$$

numbers of iterations are required for $K = 4$, $d = 2$, $M = 4$, and $N = 6$. Similarly, we can see that, again on average,

$$\iota_2 = 52699 \pm 34467 \tag{8.4}$$



(a) First series (b) Second series

Figure 8.3: Box plots of the number of iterations required to reach remaining leakage interference of less than or equal 10^{-5}.

numbers of iterations are necessary for $K = 6$, $d = 4$, $M = 12$, and $N = 16$. Hence, about thirty times the number of iterations are needed for increasing K from 4 to 6, d from 2 to 4, M from 4 to 12, and N from 6 to 16. The lower quartiles, medians, upper quartiles, and interquartile ranges—as listed in Table 8.2—are visualized in the box plots in Figure 8.3. The lower and upper whiskers in these box plots represent the minima and $Q_3(\boldsymbol{\iota}_j) + 1.5\,\mathrm{IQR}(\boldsymbol{\iota}_j)$, $j \in \{1, 2\}$, respectively. The points above the upper whiskers can be considered outliers.

8.4 Operation counts

In this section, we are going to determine the computational cost of the iterative implementation (cf. Section 8.1) and the direct implementation (cf. Section 8.2) by counting complex operations (cf. Section 3.3). Based on the experimental results for the iterative implementation (cf. Section 8.3), we will then calculate and compare the particular operation counts for the parameter values $K = 4$, $d = 2$, $M = 4$, and $N = 6$ as well as $K = 6$, $d = 4$, $M = 12$, and $N = 16$.

First, we are going to count the operations of the iterative implementation (cf. Listing A.2). In general, the matrix operation $\boldsymbol{A} + \boldsymbol{B}$, where $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{m \times n}$, requires mn complex additions, while the matrix operation $\boldsymbol{A}\boldsymbol{B}$, where $\boldsymbol{A} \in \mathbb{C}^{m \times r}$ and $\boldsymbol{B} \in \mathbb{C}^{r \times n}$, necessitates $mn(r - 1)$ complex additions and mnr complex multiplications. Table 8.3 lists operation counts for single lines of the iterative implementation. It follows that the initialization in Lines 54 to 56 of Listing A.2 requires

$$\omega_i \coloneqq 2Kd^2M \tag{8.5}$$

operations. The first computation of $\boldsymbol{Q}_k$ in Lines 58 to 63 needs

$$\alpha_i \coloneqq K(K-1)(2dMN - dN + MN^2 - MN) \tag{8.6}$$

additions and

$$\mu_i \coloneqq K(K-1)(2dMN + MN^2) \tag{8.7}$$

multiplications. Furthermore, the optimization of the original network in Lines 70 to 79 necessitates

$$\omega_o \coloneqq 25KN^3 \tag{8.8}$$

operations per iteration. The optimization of the reciprocal network in Lines 82 to 94 requires

$$\alpha_r \coloneqq K(K-1)(2dMN - dM + M^2N - MN) \tag{8.9}$$

Table 8.3: Operation counts for single lines of the iterative implementation (cf. Listing A.2).

Line of Listing A.2	Operation count
55	$2d^2M$ [GV13, p. 255]
61, 101	$NN + Nd(M-1) + NM(d-1) + NN(M-1)$
	$= 2dMN - dN + MN^2 - MN$ additions and
	$NdM + NMd + NNM$
	$= 2dMN + MN^2$ multiplications
73	$25N^3$ [GV13, p. 391]
85	$MM + Md(N-1) + MN(d-1) + MM(N-1)$
	$= 2dMN - dM + M^2N - MN$ additions and
	$MdN + MNd + MMN$
	$= 2dMN + M^2N$ multiplications
88	$25M^3$ [GV13, p. 391]
104	$d - 1 + dN(N-1) + dd(N-1)$
	$= d^2N - d^2 + dN^2 - dN + d - 1$ additions and
	$dNN + ddN$
	$= d^2N + dN^2$ multiplications
105	1 addition

additions,

$$\mu_r \coloneqq K(K-1)(2dMN + M^2N) \tag{8.10}$$

multiplications, and

$$\omega_r \coloneqq 25KM^3 \tag{8.11}$$

further operations, i.e., further additions and multiplications, per iteration. Finally, the computation of the remaining leakage interference in Lines 98 to 106 needs

$$\alpha_l \coloneqq K\left((K-1)(2dMN - dN + MN^2 - MN) + d^2N - d^2 + dN^2 - dN + d\right) \tag{8.12}$$

additions and

$$\mu_l \coloneqq K\left((K-1)(2dMN + MN^2) + d^2N + dN^2\right) \tag{8.13}$$

multiplications per iteration. Hence, we know how many complex additions and multiplications both the initialization and each iteration require. For ι iterations, summing up the

Table 8.4: Operation counts for single lines of the direct implementation (bidiagonal blocks variant, cf. Listing A.8).

Line of Listing A.8	Operation count
30	1 multiplication
31, 38	2 additions and 1 multiplication
35	2 additions
40	1 addition
43	$3\frac{Kd+d+1}{2} + 2 \cdot 4\frac{Kd+d+1}{2}KM$ (average)
	$= 4K^2dM + 4KdM + \frac{3}{2}Kd + 4KM + \frac{3}{2}d + \frac{3}{2}$
44	$3\frac{Kd+K+d-1}{2} + 2 \cdot 4KN\frac{Kd+K+d-1}{2}$ (average)
	$= 4K^2dN + 4K^2N + 4KdN + \frac{3}{2}Kd - 4KN + \frac{3}{2}K + \frac{3}{2}d - \frac{3}{2}$
46	$3\frac{Kd+d+1}{2} + 2 \cdot 4KN\frac{Kd+d+1}{2}$ (average)
	$= 4K^2dN + 4KdN + \frac{3}{2}Kd + 4KN + \frac{3}{2}d + \frac{3}{2}$
47	$3\frac{Kd+K+d-1}{2} + 2 \cdot 4\frac{Kd+K+d-1}{2}KM$ (average)
	$= 4K^2dM + 4K^2M + 4KdM + \frac{3}{2}Kd - 4KM + \frac{3}{2}K + \frac{3}{2}d - \frac{3}{2}$
52	$20 + 2 \cdot 6KN$
	$= 12KN + 20$
54	$20 + 2 \cdot 6KM$
	$= 12KM + 20$

above yields

$$\begin{aligned}\alpha_i + \iota\alpha_r + \iota\alpha_l = K\big((K-1)\big((2\iota+1)(2d-1)MN + (\iota+1)(MN^2 - dN) + \\ \iota(M^2N - dM)\big) + \iota d(dN - d + N^2 - N + 1)\big)\end{aligned} \tag{8.14}$$

additions,

$$\begin{aligned}\mu_i + \iota\mu_r + \iota\mu_l = K\big((K-1)\big((4\iota+2)dMN + (\iota+1)MN^2 + \iota M^2N\big) + \\ \iota d(dN + N^2)\big)\end{aligned} \tag{8.15}$$

multiplications, and

$$\omega_i + \iota\omega_o + \iota\omega_r = K\left(2d^2M + 25\iota(N^3 + M^3)\right) \tag{8.16}$$

further operations for the iterative implementation.

After having determined the operation count for the iterative implementation, we are going to count the complex operations of the direct implementation, in particular the implementation of the bidiagonal blocks algorithm variant CRIAD$^{(b)}$. Generally, computing

Table 8.5: Operation counts for the iterative implementation (cf. Listing A.2) and the direct implementation (bidiagonal blocks variant, cf. Listing A.8) for two examples: $K = 4$, $d = 2$, $M = 4$, $N = 6$, $\iota = 1798$ (first example) and $K = 6$, $d = 4$, $M = 12$, $N = 16$, $\iota = 52699$ (second example).

	Operation count	
	First example	Second example
Iterative implementation	$68804128 \approx 6.9 \cdot 10^7$	$72751451400 \approx 7.3 \cdot 10^{10}$
Direct implementation	$12214 \approx 1.2 \cdot 10^4$	$419174 \approx 4.2 \cdot 10^5$

the Householder vector $\boldsymbol{\hbar}$ and the factor β (cf. Definition 3.1) for $\boldsymbol{z} \in \mathbb{C}^m$ requires about $3m$ complex operations, and applying a Householder matrix $\mathscr{H}$ to a matrix $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ needs $4mn$ complex operations [GV13, p. 236]. The computation of the coefficients of a Givens matrix $\mathcal{G}$ requires about twenty complex operations (if we define a square root as four operations). Premultiplying or postmultiplying a matrix $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ by $\mathcal{G}$ necessitates $6n$ or $6m$ complex operations [GV13, p. 241].

Table 8.4 lists (average) operation counts for single lines of Listing A.8 based on these general operation counts. The maximum length of the Householder vector $\boldsymbol{\hbar}$ in Lines 43 and 46 is Kd (if $\kappa = 1$ and $\delta = 1$), the minimum length is $d + 1$ (if $\kappa = K - 1$ and $\delta = d$). Hence, and since all lengths $d + 1, d + 2, \ldots, Kd$ are assumed exactly once, the average length is $\frac{Kd+d+1}{2}$. Similarly, the maximum length of $\boldsymbol{\hbar}$ in Lines 44 and 47 is $Kd + K - 2$ (if $\kappa = 1$ and $\delta = 1$), and the minimum length again is $d + 1$ (if $\kappa = K - 1$ and $\delta = d$). The (approximate) average length thus is $\frac{Kd+K+d-1}{2}$. Therefore, we count

$$(K-1)\Bigg(d\Bigg(4K^2dM + 4K^2dN + 4K^2N + 4KdM + 4KdN + 3Kd + 4KM - 4KN + \frac{3}{2}K + 3d + 4\Bigg) + 12KN + 22\Bigg) + 4 \tag{8.17}$$

(if $N < M$) or

$$(K-1)\Bigg(d\Bigg(4K^2dM + 4K^2dN + 4K^2M + 4KdM + 4KdN + 3Kd - 4KM + 4KN + \frac{3}{2}K + 3d + 4\Bigg) + 12KM + 22\Bigg) + 4 \tag{8.18}$$

(if $N \geq M$) operations for the bidiagonal blocks direct implementation.

On the basis of the experimentally determined numbers of iterations required by the iterative implementation (cf. Equations 8.3 and 8.4), we are now able to compare the operation counts for the iterative and direct implementations for selected examples. Table 8.5 contrasts the operation counts of these two implementations for $K = 4$, $d = 2$, $M = 4$,

$N = 6$, and $\iota = 1798$, as well as $K = 6$, $d = 4$, $M = 12$, $N = 16$, and $\iota = 52699$. The results reveal that the iterative implementation needs orders of magnitude more operations than the direct implementation. We want to emphasize again that the iterative and the direct implementation solve a related but distinct problem. Hence, the comparison of the operation counts can only provide a hint towards the potential of a direct solution to the interference alignment decomposition problem.

Chapter 9

Conclusion

After we have introduced the method of interference alignment on the K-user interference channel, we have defined the interference alignment problem based on the condition for interference alignment from the literature. Next, we have reviewed an iterative optimization algorithm by Gomadam et al. for solving the interference alignment problem. Since there is no proof that this algorithm converges to a global optimum, we subsequently have pursued the objective of finding a direct and optimal solution to the interference alignment problem.

For this purpose, we have adopted a reformulation of the interference alignment problem as an equivalent global matrix factorization problem. As a first step towards a general direct solution, we have discussed several variants of a direct (constructive) algorithm that solve the related but simplified relaxed interference alignment decomposition problem. These algorithm variants apply Householder reflections and Givens rotations in order to construct the global interference alignment matrix with bidiagonal, tridiagonal, and full main diagonal blocks. In contrast to the other algorithm variants, the bidiagonal blocks variant, which we have proved to solve the relaxed matrix factorization problem, is able to accept all input parameters values that conform to the feasibility criteria for interference alignment.

Afterwards, we have argued that, because of the required sparsity patterns, the extended global precoding and postcoding matrices cannot be products of Householder and Givens matrices, and hence, there is no direct solution to the interference alignment decomposition problem solely based on Householder reflections and Givens rotations. Eventually, we have determined the operation counts for prototype Matlab/Octave implementations of the iterative and direct algorithms. Based on numerical experiments with the iterative implementation, we have discovered that the direct implementation needs considerably less operations than the iterative implementation, which is another motivation for finding a direct solution to not only the relaxed but also the general matrix factorization problem.

In Section 7.2 of Chapter 7, we have outlined a few possible approaches for finding such a direct solution in future work. Apart from that, other interesting future tasks include the

development of a more efficient iterative algorithm for solving the interference alignment problem and the parallelization of the algorithms discussed in this thesis. It may also be worthwhile to provide and benchmark efficient implementations of these algorithms.

Appendix A

Source code

This appendix lists the source code of prototype Matlab/Octave implementations of the algorithms discussed in the previous chapters. The implementations are described in Chapter 8.

Listing A.1: iterative.m

```
% The implementation is based on [Gui12, pp. 1–2].

K = input('K := ');
d = input('d := ');
M = input('M := ');
N = input('N := ');
fprintf('\n');

% Maximum remaining leakage interference
max_leak = 1e-10;

% Maximum number of iterations
max_iter = 1e5;

% Interval (a, b) for random numbers
a = -1e1;
b = 1e1;

% Random channel matrices
H = cell(K);
for k = 1 : K
  for j = 1 : K
    Re_H_kj = a + (b - a) * rand(N, M);
    Im_H_kj = a + (b - a) * rand(N, M);
    H{k, j} = Re_H_kj + 1i * Im_H_kj;
  end
end
```

```
28
29 % Iterative interference alignment
30 [U, V, leak, iter] = iia(H, K, d, M, N, max_leak, max_iter);
31
32 % Global matrix formulation
33 U_ = zeros(K * N, K * d);
34 H_ = cell2mat(H);
35 V_ = zeros(K * M, K * d);
36
37 for k = 1 : K
38   U_((k - 1) * N + 1 : k * N, (k - 1) * d + 1 : k * d) = U{k};
39   V_((k - 1) * M + 1 : k * M, (k - 1) * d + 1 : k * d) = V{k};
40 end
41
42 U_bar = [U_ null(U_')];
43 V_bar = [V_ null(V_')];
44
45 Sigma_bar = U_bar' * H_ * V_bar;
46
47 % Visualization of result
48 visualize(U_bar, 1e-13, 'U_bar');
49 visualize(Sigma_bar, 1e-5, 'Sigma_bar');
50 visualize(V_bar, 1e-13, 'V_bar');
51
52 if iter >= max_iter
53   disp('maximum number of iterations');
54 else
55   disp([num2str(iter) ' iterations']);
56 end
57
58 disp(['remaining leakage interference: ' num2str(leak)]);
```

Listing A.2: iia.m (cf. Algorithm 4.2)

```
1  function [U, V, leak, iter] = iia(H, K, d, M, N, max_leak, max_iter)
2  % IIA Distributed iterative optimization algorithm for solving the interference
3  % alignment problem (symmetric variant).
4  %
5  %   [U, V, leak, iter] = IIA(H, K, d, M, N, max_leak, max_iter) returns the
6  %   K x 1 cells U and V of N x d postcoding and M x d precoding matrices such
7  %   that U{k}' * H{k, j} * V{j} is approximately equal to 0 for j ~= k. H is a
8  %   K x K cell of N x M channel matrices, d is the degrees of freedom, M is the
9  %   number of antennas at each transmitter, N is the number of antennas at each
10 %   receiver, max_leak is the maximum remaining leakage interference, and
11 %   max_iter is the maximum number of iterations.
12 %
```

```
13 %   The implementation is based on [GCJ11, alg. 1].
14
15 if nargin < 5
16   error('required: H, K, d, M, N');
17 elseif nargin < 7
18   max_iter = 1e4;
19
20   if nargin < 6
21     max_leak = 1e-4;
22   end
23 end
24
25 if K < 2 || d < 1
26   error('required: K >= 2, d >= 1');
27 elseif M < d || N < d
28   error('required: M >= d, N >= d');
29 elseif N * M <= d^2
30   error('required: N * M > d^2');
31 elseif N + M < (K + 1) * d
32   error('required: N + M >= (K + 1) * d');
33 elseif ~iscell(H)
34   error('required: iscell(H)');
35 elseif size(H, 1) ~= K || size(H, 2) ~= K
36   error('required: size(H, 1) == K, size(H, 2) == K');
37 end
38
39 for k = 1 : K
40   for j = 1 : K
41     if size(H{k, j}, 1) ~= N || size(H{k, j}, 2) ~= M
42       error('required: size(H{k, j}, 1) == N, size(H{k, j}, 2) == M');
43     end
44   end
45 end
46
47 Q = cell(K, 1);
48 U = cell(K, 1);
49 V = cell(K, 1);
50 U_rec = cell(K, 1);
51 V_rec = cell(K, 1);
52
53 % Initialization
54 for j = 1 : K
55   V{j} = orth(rand(M, d) + 1i * rand(M, d));
56 end
57
```

```
for k = 1 : K
  Q{k} = 0;
  for j = setdiff(1 : K, k)
    Q{k} = Q{k} + H{k, j} * V{j} * V{j}' * H{k, j}';
  end
end

leak = realmax;
iter = 0;

while leak > max_leak && iter < max_iter
  % Optimization original network
  for k = 1 : K
    % Q{k} already computed

    [eig_vec, eig_val] = eig(Q{k});
    [~, ind] = sort(diag(eig_val));
    eig_vec = eig_vec(:, ind);
    U{k} = eig_vec(:, 1 : d);

    V_rec{k} = U{k};
  end

  % Optimization reciprocal network
  for j = 1 : K
    Q_rec_j = 0;
    for k = setdiff(1 : K, j)
      Q_rec_j = Q_rec_j + H{k, j}' * V_rec{k} * V_rec{k}' * H{k, j};
    end

    [eig_vec, eig_val] = eig(Q_rec_j);
    [~, ind] = sort(diag(eig_val));
    eig_vec = eig_vec(:, ind);
    U_rec{j} = eig_vec(:, 1 : d);

    V{j} = U_rec{j};
  end

  % Remaining leakage interference
  leak = 0;
  for k = 1 : K
    Q{k} = 0;
    for j = setdiff(1 : K, k)
      Q{k} = Q{k} + H{k, j} * V{j} * V{j}' * H{k, j}';
    end
```

```
103
104     I_k = trace(U{k}' * Q{k} * U{k});
105     leak = leak + abs(I_k);
106   end
107
108   iter = iter + 1;
109 end
110
111 end
```

Listing A.3: direct.m

```
1  variant = input('CRIAD [f, t'''', t'', t, b]: ', 's');
2  K = input('K := ');
3  d = input('d := ');
4  M = input('M := ');
5  N = input('N := ');
6  fprintf('\n');
7
8  % Zero if absolute value smaller than epsilon
9  epsilon = 1e-13;
10
11 % Interval (a, b) for random numbers
12 a = -1e1;
13 b = 1e1;
14
15 % Random global channel matrix
16 Re_H = a + (b - a) * rand(K * N, K * M);
17 Im_H = a + (b - a) * rand(K * N, K * M);
18 H = Re_H + 1i * Im_H;
19
20 % Constructive relaxed interference alignment decomposition
21 switch variant
22   case 'f'
23     [U_hat, Sigma_bar, V_hat] = criadf(H, K, d, M, N);
24   case 't'''''
25     [U_hat, Sigma_bar, V_hat] = criadt__(H, K, d, M, N);
26   case 't'''
27     [U_hat, Sigma_bar, V_hat] = criadt_(H, K, d, M, N);
28   case 't'
29     [U_hat, Sigma_bar, V_hat] = criadt(H, K, d, M, N);
30   case 'b'
31     [U_hat, Sigma_bar, V_hat] = criadb(H, K, d, M, N);
32   otherwise
33     error(['unknown: ' variant]);
34 end
```

```
35
36 % Visualization of result
37 visualize(U_hat, epsilon, 'U_hat');
38 visualize(Sigma_bar, epsilon, 'Sigma_bar');
39 visualize(V_hat, epsilon, 'V_hat');
40
41 % Correctness of result
42 if norm(U_hat * Sigma_bar * V_hat' - H) / norm(H) < epsilon
43   disp('result correct');
44 else
45   disp('result not correct');
46 end
```

Listing A.4: criadt__.m (cf. Algorithm 6.1)

```
1  function [U_hat, Sigma_bar, V_hat] = criadt__(H, K, d, M, N)
2  % CRIADT__ Constructive algorithm for solving the relaxed interference alignment
3  % decomposition problem (basic tridiagonal blocks variant).
4  %
5  %   [U_hat, Sigma_bar, V_hat] = CRIADT__(H, K, d, M, N), where H is the KN x KM
6  %   global channel matrix, d is the degrees of freedom, M is the number of
7  %   antennas at each transmitter, and N is the number of antennas at each
8  %   receiver.
9  %
10 %   See also PREHOUSE, POSTHOUSE, PREGIVENS, POSTGIVENS.
11
12 if nargin < 5
13   error('required: H, K, d, M, N');
14 end
15
16 [KN, KM] = size(H);
17
18 if K < 2 || d < 1
19   error('required: K >= 2, d >= 1');
20 elseif M <= d || N <= d
21   error('required: M > d, N > d');
22 elseif K * N ~= KN || K * M ~= KM
23   error('required: size(H, 1) == K * N, size(H, 2) == K * M');
24 end
25
26 U_hat_H = eye(KN);
27 Sigma_bar = H;
28 V_hat = eye(KM);
29
30 tau_1 = KN;
31 tau_2 = KM;
```

```
32
33 for kappa = 1 : K - 1
34   chi_1 = tau_1 - kappa + 1;
35   chi_2 = tau_2 - kappa + 1;
36
37   for delta = 1 : d - 1
38     psi = (kappa - 1) * d + delta;
39     phi = psi + 1;
40
41     [Sigma_bar, U_hat_H] = prehouse(Sigma_bar, phi, chi_1, psi, U_hat_H);
42     [Sigma_bar, V_hat] = posthouse(Sigma_bar, psi, phi, chi_2, V_hat);
43   end
44
45   psi = kappa * d;
46
47   for chi = kappa * d + 2 : chi_1
48     phi = chi - 1;
49
50     [Sigma_bar, U_hat_H] = pregivens(Sigma_bar, chi, phi, psi, U_hat_H);
51   end
52
53   for chi = kappa * d + 2 : chi_2
54     phi = chi - 1;
55
56     [Sigma_bar, V_hat] = postgivens(Sigma_bar, psi, chi, phi, V_hat);
57   end
58 end
59
60 U_hat = U_hat_H';
61
62 end
```

Listing A.5: criadt_.m (cf. Algorithm 6.2)

```
1  function [U_hat, Sigma_bar, V_hat] = criadt_(H, K, d, M, N)
2  % CRIADT_ Constructive algorithm for solving the relaxed interference alignment
3  % decomposition problem (enhanced tridiagonal blocks variant).
4  %
5  %   [U_hat, Sigma_bar, V_hat] = CRIADT_(H, K, d, M, N), where H is the KN x KM
6  %   global channel matrix, d is the degrees of freedom, M is the number of
7  %   antennas at each transmitter, and N is the number of antennas at each
8  %   receiver.
9  %
10 %   See also PREHOUSE, POSTHOUSE, PREGIVENS, POSTGIVENS.
11
12 if nargin < 5
```

```
  error('required: H, K, d, M, N');
end

[KN, KM] = size(H);

if K < 2 || d < 1
  error('required: K >= 2, d >= 1');
elseif M <= d || N <= d
  error('required: M > d, N > d');
elseif K * N ~= KN || K * M ~= KM
  error('required: size(H, 1) == K * N, size(H, 2) == K * M');
end

U_hat_H = eye(KN);
Sigma_bar = H;
V_hat = eye(KM);

tau_1 = KN;
tau_2 = KM;

for kappa = 1 : K - 1
  chi_1 = tau_1 - kappa + 1;
  chi_2 = tau_2 - kappa + 1;

  for delta = 1 : d
    psi = (kappa - 1) * d + delta;
    phi = psi + 1;

    [Sigma_bar, U_hat_H] = prehouse(Sigma_bar, phi, chi_1, psi, U_hat_H);
    [Sigma_bar, V_hat] = posthouse(Sigma_bar, psi, phi, chi_2, V_hat);
  end

  [Sigma_bar, U_hat_H] = pregivens(Sigma_bar, chi_1, phi, psi, U_hat_H);
  [Sigma_bar, V_hat] = postgivens(Sigma_bar, psi, chi_2, phi, V_hat);
end

U_hat = U_hat_H';

end
```

Listing A.6: criadt.m (cf. Algorithm 6.3)

```
function [U_hat, Sigma_bar, V_hat] = criadt(H, K, d, M, N)
% CRIADT Constructive algorithm for solving the relaxed interference alignment
% decomposition problem (further enhanced tridiagonal blocks variant).
%
```

```
5  %   [U_hat, Sigma_bar, V_hat] = CRIADT(H, K, d, M, N), where H is the KN x KM
6  %   global channel matrix, d is the degrees of freedom, M is the number of
7  %   antennas at each transmitter, and N is the number of antennas at each
8  %   receiver.
9  %
10 %   See also PREHOUSE, POSTHOUSE, PREGIVENS, POSTGIVENS.
11
12 if nargin < 5
13   error('required: H, K, d, M, N');
14 end
15
16 [KN, KM] = size(H);
17
18 if K < 2 || d < 1
19   error('required: K >= 2, d >= 1');
20 elseif M <= d || N <= d
21   error('required: M > d, N > d');
22 elseif K * N ~= KN || K * M ~= KM
23   error('required: size(H, 1) == K * N, size(H, 2) == K * M');
24 end
25
26 U_hat_H = eye(KN);
27 Sigma_bar = H;
28 V_hat = eye(KM);
29
30 tau = K * (d + 1) - 1;
31
32 for kappa = 1 : K - 1
33   chi = tau - kappa + 1;
34
35   for delta = 1 : d
36     psi = (kappa - 1) * d + delta;
37     phi = psi + 1;
38
39     [Sigma_bar, U_hat_H] = prehouse(Sigma_bar, phi, chi, psi, U_hat_H);
40     [Sigma_bar, V_hat] = posthouse(Sigma_bar, psi, phi, chi, V_hat);
41   end
42
43   [Sigma_bar, U_hat_H] = pregivens(Sigma_bar, chi, phi, psi, U_hat_H);
44   [Sigma_bar, V_hat] = postgivens(Sigma_bar, psi, chi, phi, V_hat);
45 end
46
47 U_hat = U_hat_H';
48
49 end
```

Listing A.7: criadf.m (cf. Algorithm 6.4)

```
function [U_hat, Sigma_bar, V_hat] = criadf(H, K, d, M, N)
% CRIADF Constructive algorithm for solving the relaxed interference alignment
% decomposition problem (full blocks variant).
%
%   [U_hat, Sigma_bar, V_hat] = CRIADF(H, K, d, M, N), where H is the KN x KM
%   global channel matrix, d is the degrees of freedom, M is the number of
%   antennas at each transmitter, and N is the number of antennas at each
%   receiver.
%
%   See also PREHOUSE, POSTHOUSE, PREGIVENS, POSTGIVENS.

if nargin < 5
  error('required: H, K, d, M, N');
end

[KN, KM] = size(H);

if K < 2 || d < 2
  error('required: K >= 2, d >= 2');
elseif M < 2 * d - 1 - (d - 1) / K || N < 2 * d - 1 - (d - 1) / K
  error('required: M >= 2 * d - 1 - (d - 1) / K, N >= 2 * d - 1 - (d - 1) / K');
elseif K * N ~= KN || K * M ~= KM
  error('required: size(H, 1) == K * N, size(H, 2) == K * M');
end

U_hat_H = eye(KN);
Sigma_bar = H;
V_hat = eye(KM);

tau = K * d + (K - 1) * (d - 1);

for kappa = 1 : K - 1
  phi = kappa * d;
  chi = tau - (kappa - 1) * (d - 1);
  psi = (kappa - 1) * d + 1;

  [Sigma_bar, U_hat_H] = prehouse(Sigma_bar, phi, chi, psi, U_hat_H);
  [Sigma_bar, V_hat] = posthouse(Sigma_bar, psi, phi, chi, V_hat);

  phi = kappa * d + 1;

  for delta = 2 : d
    chi = tau - (kappa - 1) * (d - 1) - delta + 2;
    psi = (kappa - 1) * d + delta;
```

```
45
46      [Sigma_bar, U_hat_H] = prehouse(Sigma_bar, phi, chi, psi, U_hat_H);
47      [Sigma_bar, V_hat] = posthouse(Sigma_bar, psi, phi, chi, V_hat);
48      [Sigma_bar, U_hat_H] = pregivens(Sigma_bar, chi, phi, psi, U_hat_H);
49      [Sigma_bar, V_hat] = postgivens(Sigma_bar, psi, chi, phi, V_hat);
50    end
51  end
52
53  U_hat = U_hat_H';
54
55  end
```

Listing A.8: criadb.m (cf. Algorithm 6.5)

```
1   function [U_hat, Sigma_bar, V_hat] = criadb(H, K, d, M, N)
2   % CRIADB Constructive algorithm for solving the relaxed interference alignment
3   % decomposition problem (bidiagonal blocks variant).
4   %
5   %   [U_hat, Sigma_bar, V_hat] = CRIADB(H, K, d, M, N), where H is the KN x KM
6   %   global channel matrix, d is the degrees of freedom, M is the number of
7   %   antennas at each transmitter, and N is the number of antennas at each
8   %   receiver.
9   %
10  %   See also PREHOUSE, POSTHOUSE, PREGIVENS, POSTGIVENS.
11
12  if nargin < 5
13    error('required: H, K, d, M, N');
14  end
15
16  [KN, KM] = size(H);
17
18  if K < 2 || d < 1
19    error('required: K >= 2, d >= 1');
20  elseif M < d || N < d || (M == d && N == d)
21    error('required: M >= d, N > d or M > d, N >= d');
22  elseif K * N ~= KN || K * M ~= KM
23    error('required: size(H, 1) == K * N, size(H, 2) == K * M');
24  end
25
26  U_hat_H = eye(KN);
27  Sigma_bar = H;
28  V_hat = eye(KM);
29
30  tau_1 = K * d;
31  tau_2 = K * (d + 1) - 1;
32  chi_1 = tau_1;
```

```
33
34 for kappa = 1 : K - 1
35   chi_2 = tau_2 - kappa + 1;
36
37   for delta = 1 : d
38     psi = (kappa - 1) * d + delta;
39     phi_1 = psi;
40     phi_2 = psi + 1;
41
42     if N < M
43       [Sigma_bar, U_hat_H] = prehouse(Sigma_bar, phi_1, chi_1, psi, U_hat_H);
44       [Sigma_bar, V_hat] = posthouse(Sigma_bar, psi, phi_2, chi_2, V_hat);
45     else
46       [Sigma_bar, V_hat] = posthouse(Sigma_bar, psi, phi_1, chi_1, V_hat);
47       [Sigma_bar, U_hat_H] = prehouse(Sigma_bar, phi_2, chi_2, psi, U_hat_H);
48     end
49   end
50
51   if N < M
52     [Sigma_bar, V_hat] = postgivens(Sigma_bar, psi, chi_2, phi_2, V_hat);
53   else
54     [Sigma_bar, U_hat_H] = pregivens(Sigma_bar, chi_2, phi_2, psi, U_hat_H);
55   end
56 end
57
58 U_hat = U_hat_H';
59
60 end
```

Listing A.9: prehouse.m

```
1  function [A, L] = prehouse(A, i_1, i_2, j, L)
2  % PREHOUSE Premultiply a matrix by an (extended) Householder matrix.
3  %
4  %   A = PREHOUSE(A, i_1, i_2, j) returns A = H * A, where H is the (extended)
5  %   Householder matrix computed by gallery('house', A(i_1 : i_2, j)).
6  %
7  %   [A, L] = PREHOUSE(A, i_1, i_2, j, L) returns A = H * A and L = H * L, where
8  %   H is the (extended) Householder matrix computed by
9  %   gallery('house', A(i_1 : i_2, j)).
10 %
11 %   The implementation is based on [GV13, p. 236].
12 %
13 %   See also GALLERY, POSTHOUSE, PREGIVENS.
14
15 if nargin < 4
```

```
  error('required: A, i_1, i_2, j');
end

[m, n] = size(A);

if m < 2 || n < 2
  error('required: size(A, 1) >= 2, size(A, 2) >= 2');
elseif i_1 >= i_2 || i_1 < 1 || i_1 > m || i_2 < 1 || i_2 > m
  error('required: i_1 < i_2, 1 <= i_1 <= size(A, 1), 1 <= i_2 <= size(A, 1)');
elseif j < 1 || j > n
  error('required: 1 <= j <= size(A, 2)');
end

[h, beta] = gallery('house', A(i_1 : i_2, j));

A(i_1 : i_2, :) = A(i_1 : i_2, :) - (beta * h) * (h' * A(i_1 : i_2, :));

if nargin >= 5
  if size(L, 1) ~= m || size(L, 2) ~= m
    error('required: size(L, 1) == size(L, 2) == size(A, 1)');
  end

  L(i_1 : i_2, :) = L(i_1 : i_2, :) - (beta * h) * (h' * L(i_1 : i_2, :));
end

end
```

Listing A.10: posthouse.m

```
function [A, R] = posthouse(A, i, j_1, j_2, R)
% POSTHOUSE Postmultiply a matrix by an (extended) Householder matrix.
%
%   A = POSTHOUSE(A, i, j_1, j_2) returns A = A * H, where H is the (extended)
%   Householder matrix computed by gallery('house', A(i, j_1 : j_2)').
%
%   [A, R] = POSTHOUSE(A, i, j_1, j_2, R) returns A = A * H and R = R * H, where
%   H is the (extended) Householder matrix computed by
%   gallery('house', A(i, j_1 : j_2)').
%
%   The implementation is based on [GV13, p. 236].
%
%   See also GALLERY, PREHOUSE, POSTGIVENS.

if nargin < 4
  error('required: A, i, j_1, j_2');
end
```

```
18
19 [m, n] = size(A);
20
21 if m < 2 || n < 2
22   error('required: size(A, 1) >= 2, size(A, 2) >= 2');
23 elseif i < 1 || i > m
24   error('required: 1 <= i <= size(A, 1)');
25 elseif j_1 >= j_2 || j_1 < 1 || j_1 > n || j_2 < 1 || j_2 > n
26   error('required: j_1 < j_2, 1 <= j_1 <= size(A, 2), 1 <= j_2 <= size(A, 2)');
27 end
28
29 [h, beta] = gallery('house', A(i, j_1 : j_2)');
30
31 A(:, j_1 : j_2) = A(:, j_1 : j_2) - (A(:, j_1 : j_2) * h) * (beta * h)';
32
33 if nargin >= 5
34   if size(R, 1) ~= n || size(R, 2) ~= n
35     error('required: size(R, 1) == size(R, 2) == size(A, 2)');
36   end
37
38   R(:, j_1 : j_2) = R(:, j_1 : j_2) - (R(:, j_1 : j_2) * h) * (beta * h)';
39 end
40
41 end
```

Listing A.11: pregivens.m

```
1  function [A, L] = pregivens(A, i_1, i_2, j, L)
2  % PREGIVENS Premultiply a matrix by a Givens matrix.
3  %
4  %   A = PREGIVENS(A, i_1, i_2, j) returns A = G * A, where G is the Givens
5  %   matrix computed by givens(A(i_1, j), A(i_2, j)).
6  %
7  %   [A, L] = PREGIVENS(A, i_1, i_2, j, L) returns A = G * A and L = G * L, where
8  %   G is the Givens matrix computed by givens(A(i_1, j), A(i_2, j)).
9  %
10 %   The implementation is based on [GV13, p. 241].
11 %
12 %   See also GIVENS, POSTGIVENS, PREHOUSE.
13
14 if nargin < 4
15   error('required: A, i_1, i_2, j');
16 end
17
18 [m, n] = size(A);
19
```

```
20 if m < 2 || n < 2
21   error('required: size(A, 1) >= 2, size(A, 2) >= 2');
22 elseif i_1 == i_2 || i_1 < 1 || i_1 > m || i_2 < 1 || i_2 > m
23   error('required: i_1 ~= i_2, 1 <= i_1 <= size(A, 1), 1 <= i_2 <= size(A, 1)');
24 elseif j < 1 || j > n
25   error('required: 1 <= j <= size(A, 2)');
26 end
27
28 G = givens(A(i_1, j), A(i_2, j));
29
30 for j_ = 1 : n
31   a_1 = A(i_1, j_);
32   a_2 = A(i_2, j_);
33   A(i_1, j_) = G(1, 1) * a_1 + G(1, 2) * a_2;
34   A(i_2, j_) = G(2, 1) * a_1 + G(2, 2) * a_2;
35 end
36
37 if nargin >= 5
38   if size(L, 1) ~= m || size(L, 2) ~= m
39     error('required: size(L, 1) == size(L, 2) == size(A, 1)');
40   end
41
42   for j_ = 1 : m
43     l_1 = L(i_1, j_);
44     l_2 = L(i_2, j_);
45     L(i_1, j_) = G(1, 1) * l_1 + G(1, 2) * l_2;
46     L(i_2, j_) = G(2, 1) * l_1 + G(2, 2) * l_2;
47   end
48 end
49
50 end
```

Listing A.12: postgivens.m

```
1  function [A, R] = postgivens(A, i, j_1, j_2, R)
2  % POSTGIVENS Postmultiply a matrix by a Givens matrix.
3  %
4  %   A = POSTGIVENS(A, i, j_1, j_2) returns A = A * G, where G is the Givens
5  %   matrix computed by givens(conj(A(i, j_1)), conj(A(i, j_2)))'.
6  %
7  %   [A, R] = POSTGIVENS(A, i, j_1, j_2, R) returns A = A * G and R = R * G,
8  %   where G is the Givens matrix computed by givens(conj(A(i, j_1)),
9  %   conj(A(i, j_2)))'.
10 %
11 %   The implementation is based on [GV13, p. 241].
12 %
```

```
%   See also GIVENS, PREGIVENS, POSTHOUSE.

if nargin < 4
  error('required: A, i, j_1, j_2');
end

[m, n] = size(A);

if m < 2 || n < 2
  error('required: size(A, 1) >= 2, size(A, 2) >= 2');
elseif i < 1 || i > m
  error('required: 1 <= i <= size(A, 1)');
elseif j_1 == j_2 || j_1 < 1 || j_1 > n || j_2 < 1 || j_2 > n
  error('required: j_1 ~= j_2, 1 <= j_1 <= size(A, 2), 1 <= j_2 <= size(A, 2)');
end

G = givens(conj(A(i, j_1)), conj(A(i, j_2)))';

for i_ = 1 : m
  a_1 = A(i_, j_1);
  a_2 = A(i_, j_2);
  A(i_, j_1) = a_1 * G(1, 1) + a_2 * G(2, 1);
  A(i_, j_2) = a_1 * G(1, 2) + a_2 * G(2, 2);
end

if nargin >= 5
  if size(R, 1) ~= n || size(R, 2) ~= n
    error('required: size(R, 1) == size(R, 2) == size(A, 2)');
  end

  for i_ = 1 : n
    r_1 = R(i_, j_1);
    r_2 = R(i_, j_2);
    R(i_, j_1) = r_1 * G(1, 1) + r_2 * G(2, 1);
    R(i_, j_2) = r_1 * G(1, 2) + r_2 * G(2, 2);
  end
end

end
```

Listing A.13: visualize.m

```
function visualize(A, epsilon, name)
% VISUALIZE Visualize the sparsity pattern of a matrix.
%
%   VISUALIZE(A, epsilon, name) prints nonzero elements of matrix A as 'X' and
```

```
5  %   zero elements, i.e., elements with absolute values smaller than epsilon, as
6  %   '0'.
7
8  if nargin < 3
9    error('required: A, epsilon, name');
10 end
11
12 B = abs(real(A)) > epsilon | abs(imag(A)) > epsilon;
13 B = num2str(B);
14 B(B == '1') = 'X';
15
16 fprintf('%s =\n\n', name);
17 disp(B);
18 fprintf('\n');
19
20 end
```

Bibliography

[AGK13] Mohammad Javad Abdoli, Akbar Ghasemi, and Amir Keyvan Khandani. "On the Degrees of Freedom of K-User SISO Interference and X Channels With Delayed CSIT". In: *IEEE Transactions on Information Theory* 59.10 (Oct. 2013), pp. 6542–6561.

[Ash65] Robert B. Ash. *Information Theory*. Interscience, 1965. ISBN: 0-470-03445-9.

[AV09] V. Sreekanth Annapureddy and Venugopal V. Veeravalli. "Gaussian Interference Networks: Sum Capacity in the Low-Interference Regime and New Outer Bounds on the Capacity Region". In: *IEEE Transactions on Information Theory* 55.7 (July 2009), pp. 3032–3050.

[AV11] V. Sreekanth Annapureddy and Venugopal V. Veeravalli. "Sum Capacity of MIMO Interference Channels in the Low Interference Regime". In: *IEEE Transactions on Information Theory* 57.5 (May 2011), pp. 2565–2581.

[BCT11] Guy Bresler, Dustin Cartwright, and David Tse. "Settling the feasibility of interference alignment for the MIMO interference channel: the symmetric square case". Preprint. Apr. 2011. URL: `http://arxiv.org/abs/1104.0888`.

[Car75] Aydano B. Carleial. "A Case Where Interference Does Not Reduce Capacity". In: *IEEE Transactions on Information Theory* 21.5 (Sept. 1975), pp. 569–570.

[Car78] Aydano B. Carleial. "Interference Channels". In: *IEEE Transactions on Information Theory* 24.1 (Jan. 1978), pp. 60–70.

[Cio09] John M. Cioffi. *Chapter 12: Multi-User Fundamentals*. Apr. 2009. URL: `http://www.stanford.edu/group/cioffi/doc/book/chap12.pdf`.

[CJ08] Viveck R. Cadambe and Syed A. Jafar. "Interference Alignment and Degrees of Freedom of the K-User Interference Channel". In: *IEEE Transactions on Information Theory* 54.8 (Aug. 2008), pp. 3425–3441.

[EC80] Abbas El Gamal and Thomas M. Cover. "Multiple User Information Theory". In: *Proceedings of the IEEE* 68.12 (Dec. 1980), pp. 1466–1483.

[ETW08] Raul H. Etkin, David N. C. Tse, and Hua Wang. "Gaussian Interference Channel Capacity to Within One Bit". In: *IEEE Transactions on Information Theory* 54.12 (Dec. 2008), pp. 5534–5562.

[GBS14] Óscar González, Carlos Beltrán, and Ignacio Santamaría. "A Feasibility Test for Linear Interference Alignment in MIMO Channels with Constant Coefficients". In: *IEEE Transactions on Information Theory* 60.3 (Mar. 2014), pp. 1840–1856.

[GCJ11] Krishna Gomadam, Viveck R. Cadambe, and Syed A. Jafar. "A Distributed Numerical Approach to Interference Alignment and Applications to Wireless Interference Networks". In: *IEEE Transactions on Information Theory* 57.6 (June 2011), pp. 3309–3322.

[Giv54] J. Wallace Givens. *Numerical computation of the characteristic values of a real symmetric matrix*. ORNL 1574. Oak Ridge National Laboratory, Feb. 1954.

[Giv58] J. Wallace Givens. "Computation of Plane Unitary Rotations Transforming a General Matrix to Triangular Form". In: *Journal of the Society for Industrial and Applied Mathematics* 6.1 (Mar. 1958), pp. 26–50.

[GJ10] Tiangao Gou and Syed A. Jafar. "Degrees of Freedom of the K User $M \times N$ MIMO Interference Channel". In: *IEEE Transactions on Information Theory* 56.12 (Dec. 2010), pp. 6040–6057.

[Gui12] Maxime Guillaud. "Note on an SVD-like decomposition for IA". Preprint. Feb. 2012.

[GV13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Fourth edition. The Johns Hopkins University Press, 2013. ISBN: 1-4214-0794-9.

[HCJ12] Chiachi Huang, Viveck R. Cadambe, and Syed A. Jafar. "Interference Alignment and the Generalized Degrees of Freedom of the X Channel". In: *IEEE Transactions on Information Theory* 58.8 (Aug. 2012), pp. 5130–5150.

[HK81] Te Sun Han and Kingo Kobayashi. "A New Achievable Rate Region for the Interference Channel". In: *IEEE Transactions on Information Theory* 27.1 (Jan. 1981), pp. 49–60.

[HN05] Anders Høst-Madsen and Aria Nosratinia. "The Multiplexing Gain of Wireless Networks". In: *Proceedings of the 2005 IEEE International Symposium on Information Theory*. Sept. 2005, pp. 2065–2069.

[Hou58a] Alston S. Householder. "A Class of Methods for Inverting Matrices". In: *Journal of the Society for Industrial and Applied Mathematics* 6.2 (June 1958), pp. 189–195.

[Hou58b] Alston S. Householder. "Unitary Triangularization of a Nonsymmetric Matrix". In: *Journal of the ACM* 5.4 (Oct. 1958), pp. 339–342.

[JR82] Lee W. Johnson and R. Dean Riess. *Numerical Analysis*. Second edition. Addison-Wesley Publishing Company, 1982. ISBN: 0-201-10392-3.

[JS08] Syed A. Jafar and Shlomo Shamai. "Degrees of Freedom Region of the MIMO X Channel". In: *IEEE Transactions on Information Theory* 54.1 (Jan. 2008), pp. 151–170.

[LAS14] Sina Lashgari, Amir Salman Avestimehr, and Changho Suh. "Linear Degrees of Freedom of the X-Channel With Delayed CSIT". In: *IEEE Transactions on Information Theory* 60.4 (Apr. 2014), pp. 2180–2189.

[MK09] Abolfazl Seyed Motahari and Amir Keyvan Khandani. "Capacity Bounds for the Gaussian Interference Channel". In: *IEEE Transactions on Information Theory* 55.2 (Feb. 2009), pp. 620–643.

[MMK06a] Mohammad Ali Maddah-Ali, Abolfazl S. Motahari, and Amir K. Khandani. *Communication over X Channel: Signalling and Multiplexing Gain*. UW-E&CE 2006-12. University of Waterloo, July 2006.

[MMK06b] Mohammad Ali Maddah-Ali, Abolfazl S. Motahari, and Amir K. Khandani. "Signaling over MIMO Multi-Base Systems: Combination of Multi-Access and Broadcast Schemes". In: *Proceedings of the 2006 IEEE International Symposium on Information Theory*. July 2006, pp. 2104–2108.

[MMK08] Mohammad Ali Maddah-Ali, Abolfazl S. Motahari, and Amir K. Khandani. "Communication Over MIMO X Channels: Interference Alignment, Decomposition, and Performance Analysis". In: *IEEE Transactions on Information Theory* 54.8 (Aug. 2008), pp. 3457–3470.

[RLL12] Meisam Razaviyayn, Gennady Lyubeznik, and Zhi-Quan Luo. "On the Degrees of Freedom Achievable Through Interference Alignment in a MIMO Interference Channel". In: *IEEE Transactions on Signal Processing* 60.2 (Feb. 2012), pp. 812–821.

[Sat81] Hiroshi Sato. "The Capacity of the Gaussian Interference Channel Under Strong Interference". In: *IEEE Transactions on Information Theory* 27.6 (Nov. 1981), pp. 786–788.

[SKC09] Xiaohu Shang, Gerhard Kramer, and Biao Chen. "A New Outer Bound and the Noisy-Interference Sum–Rate Capacity for Gaussian Interference Channels". In: *IEEE Transactions on Information Theory* 55.2 (Feb. 2009), pp. 689–699.

[SW49] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949. ISBN: 0-252-72548-4.

[SY78] Ferenc Szidarovszky and Sidney Yakowitz. *Principles and Procedures of Numerical Analysis*. Plenum Press, 1978. ISBN: 0-306-40087-1.

[TAV14] Marc Torrellas, Adrian Agustin, and Josep Vidal. "On the degrees of freedom of the K-user MISO interference channel with imperfect delayed CSIT". Preprint. May 2014. URL: `http://arxiv.org/abs/1403.7012`.

[TB97] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997. ISBN: 0-89871-361-7.

[Tyr97] Eugene E. Tyrtyshnikov. *A Brief Introduction to Numerical Analysis*. Birkhäuser, 1997. ISBN: 0-8176-3916-0.

Markus Levonyak

Curriculum vitae

Education

2014–present	Second bachelor's degree **Astronomy** (Astronomie) *University of Vienna* (Universität Wien) Vienna, Austria
2014	Exchange semester *Catholic University of Leuven* (Katholieke Universiteit Leuven) Leuven, Belgium
2012–present	Master's degree **Scientific Computing** *University of Vienna* (Universität Wien) Vienna, Austria
2008–2012	Bachelor's degree **Computer Science** (Informatik) *University of Hagen* (FernUniversität in Hagen) Hagen, Germany

Awards and scholarships

2015	Merit scholarship *University of Vienna* (Universität Wien) Vienna, Austria
2014	Erasmus scholarship *University of Vienna* (Universität Wien) Vienna, Austria
2014	Merit scholarship *University of Vienna* (Universität Wien) Vienna, Austria
2013	Award for outstanding bachelor's thesis *University of Hagen* (FernUniversität in Hagen) Hagen, Germany