

Die approbierte Originalversion dieser Diplom-/
Masterarbeit ist in der Hauptbibliothek der Tech-
nischen Universität Wien aufgestellt und zugänglich.

<http://www.ub.tuwien.ac.at>



The approved original version of this diploma or
master thesis is available at the main library of the
Vienna University of Technology.

<http://www.ub.tuwien.ac.at/eng>

TECHNISCHE
UNIVERSITÄT
WIEN

Vienna University of Technology

DIPLOMARBEIT

Structured Preconditioners for Matrix-free Iterative Methods

Ausgeführt am Institut für
Analysis und Scientific Computing
der Technischen Universität Wien

unter Anleitung von
Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Winfried Auzinger

durch
Dipl.-Ing. Gerhard Niederbrucker BSc BSc

Comeniusgasse 8/19
1170 Wien

Datum

Unterschrift

Zusammenfassung

Das Berechnen der sogenannten Quasispezies in Eigen's Quasispeziesmodell ist gleichbedeutend mit der Berechnung des dominierenden Eigenvektors von sehr großen, stark strukturierten Matrizen. Insbesondere wächst die Dimension dieser Matrizen exponentiell mit dem für die Praxis entscheidenden Modellparameter. Daher leiden Standardmethoden am sogenannten „Fluch der Dimensionalität“ und können somit keine Ergebnisse in praktisch relevanten Dimensionen liefern. Um dieses Problem zu umgehen, wurde gezeigt, dass die auftretenden Matrizen eine Repräsentation als Kronecker-Produkt erlauben, welche unmittelbar zu effizienten impliziten Matrix-Vektor Routinen führt. Darauf aufbauend erweist sich die Potenzmethode als einfaches Werkzeug zur Berechnung des dominanten Eigenvektors in praktisch relevanten Dimensionen. Der gravierende Nachteil der Potenzmethode ist ihre langsame Konvergenz im Falle einer schlechten Separation des dominierenden Eigenwerts. In dieser Arbeit zeigen wir, wie dieses Problem mittels so genannter *shift-and-invert* Methoden auch im Falle einer schlechten Separation, wie sie in der Praxis zu erwarten ist, effizient gelöst werden kann.

Shift-and-invert Methoden erfordern die Lösung eines linearen Gleichungssystems in jedem Iterationsschritt. Grundsätzlich ist die iterative Lösung von linearen Gleichungssystemen ein bestens erforschtes Gebiet und jede Menge effiziente Ansätze, wie zum Beispiel Krylow-Unterraum-Verfahren, sind bekannt. Das eigentliche Problem ist diesbezüglich die Bereitstellung eines Vorkonditionierers der eine Lösung in möglichst wenigen Iterationsschritten erlaubt. Standardverfahren zur Vorkonditionierung scheitern im Kontext dieser Arbeit auf Grund der Größe und Struktur der betrachteten Matrizen. Daher ist es das vorrangige Ziel dieser Arbeit effektive Vorkonditionierer für das betrachtete Problem zu entwickeln, die es schlussendlich ermöglichen auch Probleme mit schlechter Separation des dominanten Eigenwerts effizient zu behandeln.

Im Zuge dessen erarbeiten wir Themen aus verschiedenen Bereichen: Hinsichtlich wohl bekannter Resultate aus der Literatur erörtern wir Krylow-Unterraum-Verfahren sowie die Theorie und Anwendungen des Kronecker-Produkts, jeweils im Lichte des betrachteten Eigenwertproblems. Die zentralen neuartigen Resultate dieser Arbeit bestehen aus einer umfassenden Theorie Hamming-Distanz-basierter Matrizen. Letztere sind Matrizen deren Eintrag (i, j) einzig von der Hamming-Distanz zwischen den Indizes i und j (geeignet interpretiert als endliche Zeichenkette) abhängt. Diese neuartigen Resultate beinhalten dabei unter anderem algebraische und algorithmische Aspekte, so wie die Struktur dieser Klasse von Matrizen im Allgemeinen. Basierend auf den Resultaten zu Hamming-Distanz-basierten Matrizen leiten wir effiziente Vorkonditionierer für das betrachtete Problem her. Zusätzlich stellen wir Experimente der so erhaltenen numerischen Lösungsverfahren dar, welche Verbesserungen der Gesamtlaufzeit von zumindest einer Größenordnung aufzeigen.

Abstract

Computing the so-called quasispecies in Eigen’s quasispecies model is tantamount to the computation of the dominating eigenvector of a highly structured large scale matrix. In particular, all of the involved matrices are data-sparse matrices whose dimension grows exponentially with the practically important model parameter. As an immediate consequence, standard solvers suffer from the curse of dimensionality and are incapable of reaching practically relevant dimensions. In order to overcome this issue, it was shown that the required matrices allow for a Kronecker product representation which directly implies that highly efficient implicit matrix vector routines can be provided. Based on such efficient matrix vector products, the power iteration serves as simple tool to compute the dominating eigenvector also for problems at a relevant scale. The severe drawback of the power iteration is its inferior convergence speed in case of a bad separation of the dominating eigenvalue. In this thesis we show how to overcome this issue by employing more subtle shift-and-invert methods in order to provide efficient solvers also in case of a bad separation of the dominating eigenvalue—as it has to be expected in practice.

Shift-and-invert methods require the solution of a linear system in every iteration. The iterative solution of linear systems is in principle a well-developed subject and a myriad of solvers, e. g., Krylov subspace methods, exist. The actual problem in this context is to provide a good preconditioner such that only a considerably small number of iterations are required. In particular, standard preconditioning approaches fail in the context of the problems considered in this thesis due to the high dimensionality and structure of the respective matrices. Therefore, the high level aim of this thesis is to provide suitable preconditioners for the problem at hand, in order to allow for an efficient computation of the quasispecies also in cases of a bad separation of the dominating eigenvalue.

In the course of that, we elaborate on different fields: In terms of existing work we discuss Krylov subspace methods and the theory and application of the Kronecker product each with respect to the considered extreme scale eigenvalue problem. The central novel results presented in this thesis consist of a comprehensive theory of Hamming distance-based matrices, i. e., matrices whose (i, j) -th element solely depends on the Hamming distance between the indices i and j (appropriately interpreted as finite strings). These novel results cover among others algebraic and algorithmic aspects, as well as the structure of this family of matrices in general. Based on the theory of Hamming distance-based matrices, we eventually derive efficient preconditioners for the problem at hand. Moreover, we provide experimental data of the resulting solvers which depict overall performance gains by at least an order of magnitude.

Contents

Preface	ix
1. Introduction	1
1.1. The Curse of Dimensionality	2
1.2. Matrix-free Iterative Methods	2
1.3. Quasispecies Model	5
1.4. Outline and Contributions	11
2. Matrix-free Iterative Methods: Algorithms & Tools	13
2.1. Linear Systems	13
2.2. Preconditioning	16
2.3. Eigenvalue Problems	19
3. The Kronecker Product	25
3.1. Basic Properties	25
3.2. Kronecker Products and Fast Algorithms	31
3.3. Approximation with Kronecker Products	34
4. Matrices & Hamming Distance	39
4.1. Prerequisites	39
4.2. Binary Alphabets	42
4.3. Arbitrary Alphabets	58
5. Applications of Hamming Distance-based Matrices	71
5.1. Generalizations in the Quasispecies Model	71
5.2. Preconditioning and Shift-and-Invert Methods	73
5.3. Further Applications	79
6. Summary & Conclusions	81
A. Notation and Conventions	83
B. Perron-Frobenius Theory	85
Bibliography	89

Preface

First and foremost I want to thank Winfried Auzinger for supervising this thesis and the freedom he granted me for pursuing the research which led to the results presented in this thesis. His careful reading and valuable comments greatly helped to improve the overall quality and readability of the whole text.

Parts of this work and its entire motivation are based on prior research I carried out at the University of Vienna which would not have been possible without the kind support of the following people: First of all, it was Christoph W. Überhuber who initiated the foundations of this work and, unknowingly, also many other incidents reaching far beyond the realms of scientific research. Furthermore, Peter K. Schuster—at that time president of the Austrian Academy of Sciences—despite his full calender generously provided many insightful comments and background material which enabled a smooth start within the subject. Last but not least, I also want to thank Wilfried N. Gansterer with whom I thereafter pursued the research this thesis builds upon and who eventually became my PhD advisor and employer. Moreover, I am grateful for the funding I received during the composition of this thesis from the Austrian Science Fund (FWF) under project number S10608 (NFN SISE).

Needless to say, I want to wholeheartedly thank my (expanding) family for the invaluable support I received over the last years prior to writing these lines.

Wien, September 2013

Gerhard Niederbrucker

1. Introduction

Matrices consisting of a small number of nonzero elements only, i. e., less than $\mathcal{O}(n^2)$ nonzero elements, are commonly called *sparse matrices*. Such matrices naturally appear in many application contexts, e. g., as a result of discretizations of partial differential equations. In practice, sparse matrices can be in many ways efficiently represented by storing their nonzero elements only (together with certain supplementary information about their respective location). Based on such compact representations, it is a natural aim to perform matrix computations involving sparse matrices in a way such that the overall effort depends on the number of nonzero elements rather than on the matrix dimension—as it is the case for standard routines. More generally, matrices which can be parametrized by a subquadratic amount of data are called *data-sparse*. For instance, Toeplitz matrices [Golub and Van Loan, 1996, § 4.7] are dense $n \times n$ matrices which can be parametrized by $2n$ instead of n^2 scalars. Such well-structured (dense) matrices usually allow to exploit their data sparsity in terms of very efficient matrix operations. In the case of an $n \times n$ Toeplitz matrix, the matrix vector product with an arbitrary vector can be computed in $\mathcal{O}(n \log n)$ time by employing the fast Fourier transform [Golub and Van Loan, 1996, § 4.7.7].

This thesis is primarily motivated by a large scale eigenvalue problem involving data-sparse matrices. In particular, these matrices also allow for designing a fast matrix vector multiplication routine—even though they are not of a “standard type”. Hence, the problem is in principle well suited for matrix-free iterative algorithms which solely require a matrix vector multiplication routine to be available, instead of an explicit representation of the matrix. Simple vector iterations for solving the considered problem can easily be implemented. In contrast to that, things become more involved if more efficient shift-and-invert solvers are considered, which require the solution of a linear system in each iteration. Such solvers become a necessity in cases where the problem is ill-conditioned. While iteratively solving a linear system is in principle also a standard task, the computation of an appropriate preconditioner is an a priori difficult task, since: (i) the system matrices are not explicitly available, (ii) the problem dimensions are of an extreme scale, and (iii) most importantly, the computation and application of the preconditioner itself is a nontrivial task raising substantial costs—yet, it should provide benefits in terms of overall runtime. In order to design such preconditioners, we introduce and analyze a class of well-structured data-sparse matrices which enables the efficient computation and application of suitable preconditioners. Moreover, the results regarding this class of matrices are not only important with respect to the design of preconditioners, but also for substantial generalizations of the solvable instances of the considered problem.

The following three sections further introduce the aspects and problems this thesis addresses and moreover, [Section 1.4](#) provides an overview of the main contributions and outlines the remainder of the thesis.

1.1. The Curse of Dimensionality

Consider a finite set of letters \mathcal{A} . Then there exist $|\mathcal{A}|^d$ different strings of length d which can be composed by the letters in \mathcal{A} . This effect of an exponential growth of possibilities with respect to the considered dimensionality is commonly known as *combinatorial explosion*. Phenomena like the combinatorial explosion are what make the computational treatment of real-world problems, e. g., in biology, often very difficult or even infeasible. More generally, many fields refer to the *curse of dimensionality*—an expression which was coined by Bellman [1961]—whenever significant obstacles occur by raising the dimensionality of the considered problem. For instance, in statistics, the convergence of an estimator to the true value of a smooth function exponentially slows down with the dimension of the considered space, i. e., with increasing dimension, exponentially more samples are required in general.

It is a misbelief that high-end (super)computers alone are sufficient to overcome the curse of dimensionality. Today’s fastest supercomputers¹ are (theoretically) able to run about $\mathcal{O}(10^{18})$ floating-point operations per second, compared to about $\mathcal{O}(10^9)$ floating-point operations per second for standard personal computers. That means, for a problem whose complexity grows like $\mathcal{O}(2^d)$, one can expect *at most* a gain of a factor of two in terms of tractable problem dimension d —*no matter which computer or implementation is used!* Thus, a substantial gain of the tractable problem dimension of a problem with exponentially growing complexity cannot be achieved by simply employing larger and faster hardware. It are substantial theoretical improvements together with efficient algorithms and their respective implementation which let supercomputers eventually unfold their full potential and usefulness.

In order to scale applications to extreme dimensions it is a necessity to introduce structure into the problem such that the occurring objects are of a (data-)sparse nature. This can, for instance, be achieved by model assumptions or by structured approximations (see, e. g., Hackbusch and Khoromskij [2007] and references therein). It is obviously an application dependent property whether or not the assumption of additional problem structure still leads to meaningful results. Throughout this thesis we consider a particular problem stemming from biochemistry (see Section 1.3) which is known to have a data-sparse representation and we show, by novel theoretical insights, that even substantially more general problem formulations are efficiently realizable. Moreover, we show how subtle numerical solvers can be adapted to the extreme scale problem at hand.

1.2. Matrix-free Iterative Methods

Assume we want to solve a linear system $Ax = b$ with an invertible matrix $A \in \mathbb{R}^{n \times n}$. Traditional direct methods for solving such systems such as *Gaussian elimination* require to (repeatedly) access the elements of A in an order which depends on the particular algorithm. Moreover, a storage complexity of $\mathcal{O}(n^2)$ elements is inevitable. In case of (data-)sparse matrices either of these properties is inadmissible. First of all, it is costly to explicitly refer to particular matrix elements in (data-)sparse matrices.

¹See <http://www.top500.org/list/2013/06/>, Top 500 list of supercomputers, edition June 2013.

Second, due to the sheer dimension (data-)sparse considered in practice show, it is absolutely impossible to store a quadratic amount of data. Having said that, recall that matrix vector products with (data-)sparse matrices can be performed very efficiently without requiring an explicit representation of the respective matrix. Since (a reasonable amount of) vectors can easily be stored also for huge dimensions it appears to be natural to aim at designing algorithms which are based on the idea of simply keeping several vectors and utilizing a matrix vector routine without explicitly referring to the matrix. The following definition proved to be very useful in this context.

Definition 1.1. For a square matrix $A \in \mathbb{R}^{n \times n}$ a vector $r \in \mathbb{R}^n$ and an integer $d \in \mathbb{N}$ we call

$$\mathcal{K}_d(A, r) := \text{span}\{r, Ar, A^2r, \dots, A^{d-1}r\}$$

the *Krylov subspace* spanned by the matrix A and the vector r . In case there is no ambiguity, we also simply refer by \mathcal{K}_d to $\mathcal{K}_d(A, r)$. //

Obviously, Krylov subspaces are in compliance with the previously stated goals—yet, it is not clear how to solve, e. g., linear systems utilizing such spaces. As a first step recall the Cayley-Hamilton theorem [Havlicek, 2006, Theorem 8.6.5] which states that for a matrix $A \in \mathbb{R}^{n \times n}$ and its characteristic polynomial $\chi(z)$, it holds for the matrix polynomial $\chi(A)$ that $\chi(A) = \mathbf{0}$. As an immediate consequence of the Cayley-Hamilton it can be deduced that A^{-1} can be expressed as linear combination of powers (with exponent smaller than n) of A . Therefore, the solution of a linear system $Ax = b$ is an element of the space $\mathcal{K}_n(A, b)$. But how can it be found?

The major idea behind the class of so-called *Krylov subspace methods* is to iteratively construct an approximate solution $x_k \in x_0 + \mathcal{K}_k(A, b)$ of $Ax = b$ which satisfies the so-called *Galerkin condition* $b - Ax_k \perp \mathcal{L}_k$ where \mathcal{L}_k is a k -dimensional subspace dependent on the particular algorithm. Hence, Krylov subspace methods are traditionally *projection methods*. Two popular choices for the subspace \mathcal{L}_k are given by $\mathcal{L}_k = \mathcal{K}_k$ as well as by $\mathcal{L}_k = A\mathcal{K}_k$. The multitude of different Krylov subspaces arises in particular from different choices of \mathcal{L}_k .

The obvious basis of the Krylov subspace \mathcal{K}_d given by [Definition 1.1](#) is problematic from a practical point of view, since $A^k b$ converges towards the dominating eigenvector of A (cf. [Section 2.3](#)). Hence, the vectors given by this trivial choice for the basis of \mathcal{K}_d become almost linearly dependent for increasing dimension d . Consequently, a main ingredient of Krylov subspace methods is the construction of an orthogonal/orthonormal basis of the Krylov subspace \mathcal{K}_d . In the best case, such an orthogonal basis is constructed by the algorithm *on-the-fly*, i. e., without an explicit (re-)orthogonalization procedure such as the Gram-Schmidt process [Havlicek, 2006, Theorem 11.5.7]. Two fundamental procedures for the iterative computation of orthonormal Krylov subspace basis are the Arnoldi iteration and the Lanczos iteration (see [Algorithm 1.1](#)), respectively. While the latter is restricted to symmetric matrices, the Arnoldi iteration works for general matrices as well. In both cases the iteration aborts in case the basis vectors computed so far span an A -invariant subspace [Saad, 2003, Proposition 6.6]. The idea behind the Arnoldi iteration is to repeatedly multiply the lastly obtained basis vector v_j by the matrix A and then run the (modified) Gram-Schmidt orthogonalization procedure on all basis vectors obtained so far, in order to obtain an extended orthonormal

Algorithm 1.1 Arnoldi iteration vs. Lanczos iteration [Saad, 2003, Algorithm 6.2/6.15]

Input: Multiplication routine for computing $v \mapsto Av$ **Output:** Orthonormal basis $\{v_1, \dots, v_j\}$ of $\mathcal{K}_j(A, v_1)$

1: $v_1 \leftarrow$ normalized initial choice 2: for $j \leftarrow 1, 2, \dots$ do 3: $w_j \leftarrow Av_j$ 4: for $i \leftarrow 1, \dots, j$ do 5: $h_{i,j} \leftarrow (w_j, v_i)$ 6: $w_j \leftarrow w_j - h_{i,j}v_i$ 7: end for 8: $h_{j+1,j} \leftarrow \ w_j\ $ 9: if $h_{j+1,j} = 0$ then 10: Stop iteration 11: end if 12: $v_{j+1} = w_j/h_{j+1,j}$ 13: end for	1: $v_1 \leftarrow$ normalized initial choice 2: $\beta_1 \leftarrow 0$ 3: $v_0 \leftarrow \mathbf{0}$ 4: for $j \leftarrow 1, 2, \dots$ do 5: $w_j \leftarrow Av_j - \beta_j v_{j-1}$ 6: $\alpha_j \leftarrow (w_j, v_j)$ 7: $w_j \leftarrow w_j - \alpha_j v_j$ 8: $\beta_{j+1} \leftarrow \ w_j\ $ 9: if $\beta_{j+1} = 0$ then 10: Stop iteration 11: end if 12: $v_{j+1} \leftarrow w_j/\beta_{j+1}$ 13: end for
---	---

basis. Hence, the Arnoldi iteration is on the one hand very simple but on the other hand it has the drawback that the entire set of basis vectors has to be kept throughout the iteration. This property can raise severe storage problems in case many iterations are required and the vector dimension is large. Note that the scalars $h_{i,j}$ computed by the Arnoldi iteration define a matrix H_j of Hessenberg form which satisfies $H_j = V_j^T A V_j$ where $V_j = [v_1, \dots, v_j]$. By the assumption that A is symmetric it can easily be shown that the matrix H_j computed by the Arnoldi iteration is tridiagonal [Saad, 2003, Theorem 6.19]. This observation allows to implement the Arnoldi iteration based on a *short recurrence*, i. e., without an explicit (re-)orthogonalization step. The Lanczos iteration is the result obtained by utilizing this simplification (see [Algorithm 1.1](#)). While the Lanczos iteration is very elegant—theoretically, in exact arithmetic—its practical implementation introduces subtle problems due to the limitations of floating point arithmetic which introduce problems in the implicit orthogonalization of the computed basis vectors [Cullum and Willoughby, 2002]. It is noteworthy that algorithms with short recurrences are not only available for symmetric matrices. In particular, the more subtle *Lanczos biorthogonalization procedure* [Saad, 2003, § 7.1] allows to extend the beneficial properties of the Lanczos iteration to the asymmetric case. As the term “biorthogonalization” suggests this procedure is based on the idea of simultaneously computing an appropriate orthonormal basis of the Krylov subspaces with respect to A and A^T , respectively.

Based on basic orthogonalization procedures such as the Arnoldi and Lanczos iteration it is easy to derive complete algorithms for solving linear systems and eigenvalue problems, respectively [Saad, 2003, §§ 6, 7]. Later, in [Chapter 2](#), we discuss a selection of Krylov subspace methods for solving linear systems and eigenvalue problems more explicitly. More concretely, this selection consists only of Krylov subspaces methods with short recurrences, i. e., methods closely related to the family of Lanczos iterations.

1.3. Quasispecies Model

[Niederbrucker and Gansterer, 2011a, §1]

Solving Eigen’s quasispecies model [Eigen, 1971] for the evolution of virus populations involves the computation of the dominant eigenvector of a matrix whose dimension grows exponentially with the chain length of the virus to be modeled. Most biologically interesting chain lengths are well beyond the reach of general purpose solvers. Thus, more specific solvers which exploit the properties of the problem under consideration are required, in order to raise the tractable chain lengths to a biologically meaningful scale.

1.3.1. Overview

Under some assumptions on the environment the evolution and the long-term behavior of a virus population can be modeled by the *quasispecies model* [Eigen, 1971]. In this model, each virus is represented by an RNA molecule and each RNA molecule is represented as a string over a finite alphabet with a fixed length ν , the so-called *chain length*. Unless otherwise stated, we consider a binary alphabet. The probability of a single point mutation in an RNA sequence is modeled by the (uniform) *error rate* p satisfying $0 < p \leq 1/2$. Given a fixed chain length ν , all $N := 2^\nu$ possible RNA sequences have to be considered, since any RNA molecule can potentially mutate into any other one (although the overall probability for some of these mutations may be very low).

Given that initially a single species exists, the goal is to study how this initial population evolves over time by tracking the relative concentration of each species with respect to the entire population. In the limit, the population converges to an equilibrium which means that a stationary distribution of the relative concentrations is reached. The stationary distribution which is obtained by taking the limit over the time is the so-called *quasispecies* [Eigen and Schuster, 1977]. The name “quasispecies” is motivated by the fact that the quasispecies is not a particular species but an equilibrium distribution which is reached in the model with respect to the used model parameters. The overall goal is to numerically compute the relative concentrations of the quasispecies which describes a stationary distribution in the model. More specifically, the computation of these relative concentrations is tantamount to the numerical computation of the eigenvector corresponding to the largest eigenvalue of a nonnegative matrix $W \in \mathbb{R}^{N \times N}$, which basically describes the constitution and the mutation probabilities of the N different RNA molecules with chain length ν (see Section 1.3.2). This eigenvector contains the information about the relative concentrations within the quasispecies. Since the dimension N of this eigenvalue problem grows *exponentially* with the chain length ν of the RNA molecules to be modeled, numerical solutions of the model based on general purpose methods are restricted to very small chain lengths and most biologically interesting cases are out of scope of standard methods, even on high-end machines. Hence, more specific algorithmic approaches are required in order to utilize the problem structure such that the range of computationally tractable chain lengths can be substantially increased.

1.3.2. Model Definition and Biochemical Background

We briefly review the quasispecies model which leads to the large scale eigenvalue problem we are dealing with. For a more detailed discussion and further references we refer to survey articles in the literature [Schuster, 2006, 2008, 2011].

As stated above, each RNA molecule is formally represented by a string over a binary alphabet. In particular, by X_i with $1 \leq i \leq N$ we denote the RNA molecule and the corresponding species which is represented by the binary encoding (b_1, \dots, b_ν) of length ν of the integer $i - 1$. The error-free sequence X_1 associated with the integer 0 is called the *master sequence*. As a distance measure between species X_i and X_j we use the Hamming distance $d_H(X_i, X_j)$ (cf. Chapter 4) which represents the minimal number of elementwise mutations required to transform X_i into X_j .

Due to Eigen [1971] the non-linear system of ODEs

$$\begin{aligned} \frac{dx_i}{dt} &= \sum_{j=1}^N f_j \cdot q_{i,j} \cdot x_j(t) - x_i(t) \cdot \Phi(t), \\ \Phi(t) &= \sum_{j=1}^N f_j \cdot x_j(t), \quad \sum_{j=1}^N x_j(t) = 1, \end{aligned} \tag{1.1}$$

with $i = 1, \dots, N$, models the evolution of RNA molecules. In this model x_i denotes the relative concentration of the molecular species X_i . Initially, only the master sequence exists, i.e., $x_1(0) = 1$ and $x_i(0) = 0$ for $i \neq 1$. Moreover, the positive fitness value f_i describes the constitution of the molecular species X_i and the (i, j) -th entry $q_{i,j}$ of the *mutation matrix* Q represents the probability that sequence X_j mutates into sequence X_i . Throughout the literature the so-called uniform error model is used,

$$q_{i,j} := p^{d_H(X_i, X_j)} \cdot (1 - p)^{\nu - d_H(X_i, X_j)}. \tag{1.2}$$

By definition, the mutation probabilities $q_{i,j}$ depend only on $d_H(X_i, X_j)$ and therefore, the entire matrix Q contains only $\nu + 1$ different values. It should be pointed out that this standard model assumes that p is an *average* error rate over all possible mutations since it does not depend on the position or on the overall number of the mutations.

According to Thompson and McBride [1974], the ODE system (1.1) can be straightforwardly transformed into a linear system with constant coefficients $\dot{z} = Wz$ by a suitable change of variables. In particular, $W = QF$ where Q is defined by (1.2) and F , the so-called *fitness landscape*, is a diagonal matrix with the fitness values $f_i > 0$ along its diagonal. The search for the quasispecies then reduces to the computation of the eigenvector corresponding to the dominating eigenvalue of W [Schuster, 2008]. In fact, there are several mathematically equivalent formulations of the problem with slightly differing structure:

$$QF x_R = \lambda x_R, \quad F^{\frac{1}{2}} Q F^{\frac{1}{2}} x_S = \lambda x_S, \quad F Q x_L = \lambda x_L.$$

Since F is diagonal, their solutions can easily be transformed into each other:

$$x_R = F^{-\frac{1}{2}} x_S, \quad x_S = F^{-\frac{1}{2}} x_L, \quad x_R = F^{-1} x_L.$$

Note that in the special case where all values in F are equal, the problem reduces to the computation of the dominating eigenvector of a bistochastic matrix, which is trivial and leads to an eigenvector where all entries are equal. This is not at all surprising, since for equally fit sequences we clearly expect the uniform distribution as result.

Since the variables x_i in (1.1) represent relative concentrations, we are only interested in solutions where all components of the computed eigenvector are nonnegative (negative concentrations do not have a physically meaningful interpretation). As a positive matrix, W satisfies the conditions of the Perron-Frobenius theorem (cf. Appendix B) and thus, the aforementioned nonnegativity property is guaranteed. Based on the computed eigenvector with the relative concentrations for each sequence, one can compute cumulative concentrations of so-called *error classes*. The error class $\Gamma_{k,i}$ contains all sequences j , which have a certain Hamming distance k from the fixed sequence i :

$$\Gamma_{k,i} := \{j \mid 1 \leq j \leq N \wedge d_H(X_i, X_j) = k\} \quad (1.3)$$

Since the error classes with respect to the master sequence are particularly relevant, we define $\Gamma_k := \Gamma_{k,1}$. By definition the error class Γ_k contains $\binom{\nu}{k}$ sequences (see Lemma 4.1). In the style of the definition of the error classes Γ_k we denote the $\nu + 1$ different values of Q by Q_{Γ_k} , i. e., $Q_{\Gamma_k} := p^k \cdot (1 - p)^{\nu - k}$ for $0 \leq k \leq \nu$. When the meaning is clear from the context we also use just i instead of X_i for denoting the binary string corresponding to the integer i .

After the dominating eigenvector of W has been computed, the cumulative concentrations $[\Gamma_k] := \sum_{j \in \Gamma_k} x_j$ of the error classes Γ_k in the stationary distribution are obtained. Plotting these cumulative concentrations for different error rates p leads to curves as the ones shown in Figure 1.1. Such figures can be used to get a better understanding of how a certain fitness landscape influences the evolution of the virus population. They would be even more interesting at the level of granularity of single sequences but they are very rare in the literature, due to the limitations in chain lengths which can be handled computationally. Depending on the concrete fitness values, the *error threshold phenomenon* may occur or not. If it occurs, there is an ordered stationary distribution of the concentrations up to a critical value p_{\max} for the error rate p , where some sequences clearly dominate, whereas others do not appear at all or only in very low concentrations. For $p > p_{\max}$, the structure of the population changes suddenly into a uniform stationary distribution where all sequences occur in the same concentration, which is equivalent to random replication. Note that in Figure 1.1 *cumulative* concentrations are shown for the error classes. Although all sequences have the same concentration for $p > p_{\max}$, the cumulative concentrations of the error classes differ because their cardinality differs. Typical values for p_{\max} on certain fitness landscapes are in the range 0.01–0.1 [Schuster, 2008, 2011], depending on the concrete fitness values and the chain length. Such small values for p_{\max} are quite surprising since random replication as exact solution of the ODE system (1.1) is obtained only for $p = 0.5$ [Schuster, 2008]. This sudden change from an ordered distribution to random replication is of potential interest as a building block for new antiviral strategies [Eigen, 2002] because the error rates of RNA viruses are usually close to this critical value [Drake, 1993] and an increase of the error rate p is possible by the use of pharmaceutical drugs.

1. Introduction

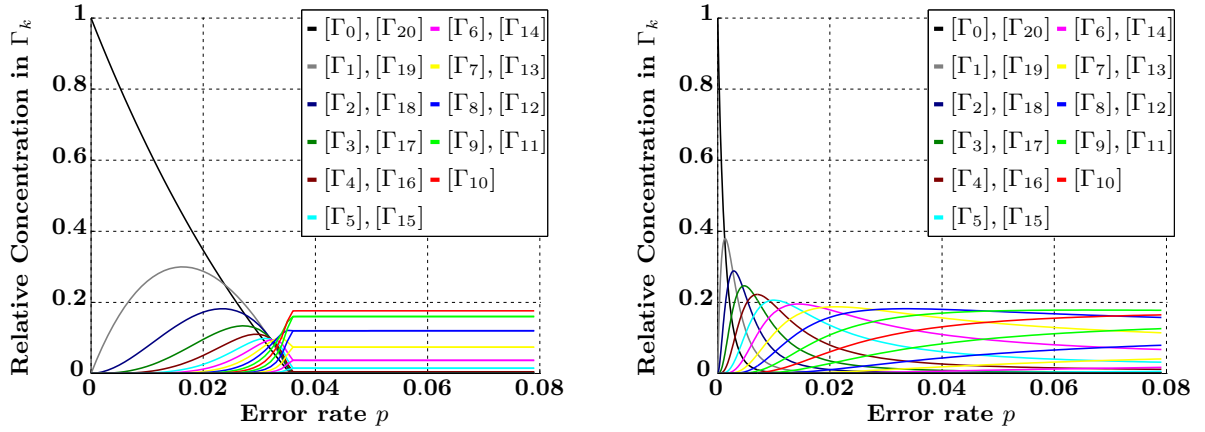


Figure 1.1.: A visualization of the error threshold phenomenon is shown on the left for $\nu = 20$ and the *single peak fitness landscape* with $f_1 = 2$, $f_i = 1$ for all $1 < i \leq N$. An ordered stationary distribution results up to $p_{\max} \approx 0.035$, and for $p > p_{\max}$ a sudden change to the uniform distribution of all sequences occurs. Error classes with the same number of elements (i. e., Γ_k and $\Gamma_{\nu-k}$) are depicted in the same color. Therefore, their curves meet when the uniform distribution is reached at the error threshold. On the right, the behavior for $\nu = 20$ and the so-called *linear landscape* defined as $f_i = f_1 - (f_1 - f_\nu) \cdot d_H(i, 1)/\nu$ for all $1 \leq i \leq N$ with $f_1 = 2$ and $f_\nu = 1$ are shown. For this landscape a smooth transition into the uniform distribution is observed and the error threshold phenomenon does not occur [Niederbrucker and Gansterer, 2011a, Figure 1].

1.3.3. Existing Approaches

Traditionally, computational investigations in the large body of literature about the quasispecies model were limited to the case where the fitness landscape F is defined via the Hamming distance $f_{i,i} := \varphi(d_H(i, 1))$ for some function $\varphi: \{0, \dots, \nu\} \rightarrow \mathbb{R}$ [Schuster, 2011]. Practically, this means that all sequences with the same distance to the master sequence are equally fit, which is often a rather unrealistic assumption. With this simplifying assumption one expects that the problem reduces from an $N \times N$ to a $(\nu + 1) \times (\nu + 1)$ problem, since all sequences in the same error class Γ_k are considered equivalent. For this special constellation efficient *approximate* schemes have been developed [Nowak and Schuster, 1989; Swetina and Schuster, 1982], and this is the way how the quasispecies model and the error threshold phenomenon have usually been studied.

In contrast to these approximation methods, the objective of the recent work of Niederbrucker and Gansterer [2011a,b] is to deal with the general form of the problem without any special assumptions, where prior to that, no fast solvers were available [Schuster, 2011]. For the general problem without assumptions on the structure of the fitness landscape Niederbrucker and Gansterer [2011a,b] employed a power iteration-based approach (cf. Section 2.3) and introduced several novel fast matrix vector multiplication routines for the occurring matrices. Moreover, all matrix vector routines they consider are of a matrix-free (implicit) nature in order to enable a substantial raise of the computationally tractable chain lengths ν .

A first naive approach towards a fast implicit matrix vector routine was achieved by introducing an implicit sparse matrix vector product (called XMVP) [Niederbrucker and Gansterer, 2011b] for the matrix $W = QF$ arising in the quasispecies model. The implicit sparsification is thereby based on the utilization of the binary bitwise XOR operator and sparsifies the mutation matrix Q by taking into account only sequences within a maximum Hamming distance of d_H^{\max} . Thus, $\text{XMVP}(d_H^{\max})$ is a parametrized implicit matrix vector product where the parameter d_H^{\max} controls the degree of sparsification in the matrix. In particular, the parameter d_H^{\max} controls runtime complexity (the smaller the less; see Figure 1.2) and accuracy (the larger the better). Niederbrucker

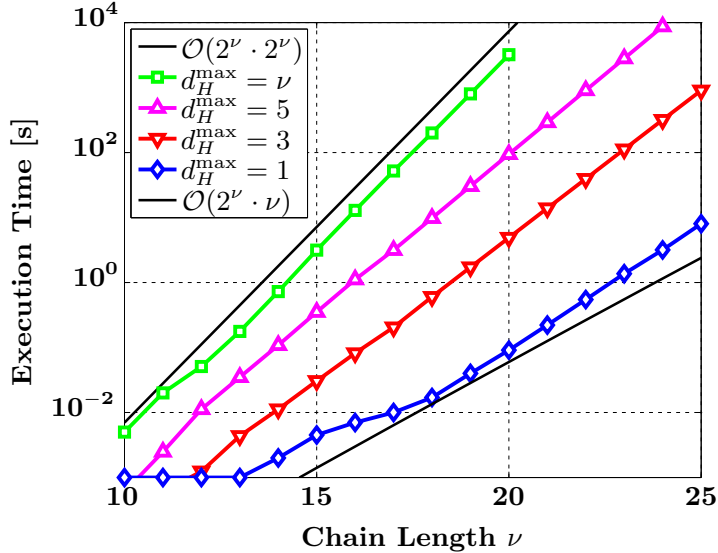


Figure 1.2.: Runtimes for executing $\text{XMVP}(d_H^{\max})$ for different choices of d_H^{\max} and increasing chain length ν .

and Gansterer [2011b] showed that $\text{XMVP}(\nu)$ is basically identical to the standard matrix vector product (henceforth called SMVP), which does not take into account the structure of W , and that $\text{XMVP}(d_H^{\max})$ reduces the space complexity to $\Theta(N)$ as well as the time complexity to $\Theta(N \cdot \sum_{k=0}^{d_H^{\max}} \binom{\nu}{k})$ from $\Theta(N^2)$ in both cases for SMVP. Figure 1.2 illustrates that based on the respective choice of d_H^{\max} , the matrix vector routine $\text{XMVP}(d_H^{\max})$ has a time complexity in the range between $\mathcal{O}(N^2)$ and $\mathcal{O}(N \log N)$. This illustrates that the sparsified XOR-based matrix vector product nicely reduces the memory requirements, but recall that the potential runtime reduction strongly depends on the level of tolerable inaccuracy in the computed concentrations of the quasispecies.

A closer inspection of the underlying mutation model reveals that the mutation matrix Q can be represented as a Kronecker product [Niederbrucker and Gansterer, 2011b]. This observation not only allows to understand the underlying structures much better but moreover, also leads to very efficient computations. In particular, by utilizing the Kronecker product representation of Q , the complexity of a matrix vector multiplication drops to $\Theta(N \log N)$, without any loss in accuracy. In the following, we refer by FMMP to this fully accurate fast matrix vector multiplication routine. The Kronecker product does not only provide structure which can be algorithmically utilized, but it also enables efficient implementations on parallel systems (see Section 3.2). Concretely, FMMP has

1. Introduction

been efficiently implemented on modern GPU hardware [Niederbrucker and Gansterer, 2011b]. Thus, besides the theoretical advances the Kronecker product representation of Q yields, it also allows for further practical improvements in terms of efficient highly parallelized implementations. Figure 1.3 concisely summarizes the speed-up the meth-

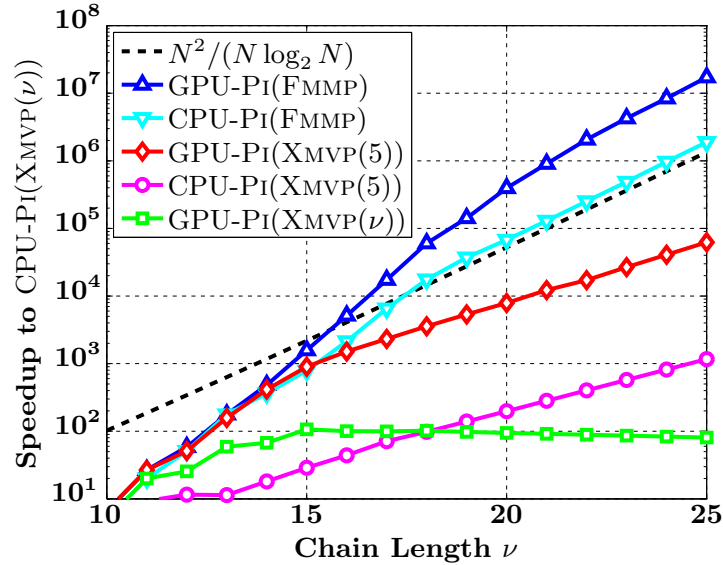


Figure 1.3.: Speedup factors for solving the quasispecies model based on various combinations of algorithms and hardware over a power iteration based on $\text{XMVP}(\nu)$ running on a single CPU core. For a given algorithm, different hardware platforms lead to (asymptotically) parallel speedup curves with the same slope, whereas the curves for different algorithms have different slopes [Niederbrucker and Gansterer, 2011a, Figure 4].

ods introduced by Niederbrucker and Gansterer [2011a,b] gain on a single-threaded CPU as well on a multi-threaded GPU platform.

1.3.4. Open Problems

The aforementioned power iteration-based approaches are very simple and efficient in cases the dominating eigenvalue of the matrix W is sufficiently well separated from the other eigenvalues, since it converges linearly of the order $\mathcal{O}(\lambda_2/\lambda_1)$ where λ_1, λ_2 denote the two largest (in magnitude) eigenvalues of W (see Section 2.3). On the other hand, in cases of a bad separation the power iteration suffers from a very bad convergence speed. In particular, the Perron-Frobenius theory (cf. Appendix B) solely guarantees that $\lambda_1 > \lambda_2$. Thus, in practice, a well-behaved spectrum (with respect to the needs of the power iteration) can not be guaranteed. Moreover, from an application point of view interesting fitness landscapes have to be expected to lead to a bad separation. With respect to this expectations we consider later in Chapter 5 also “synthetic” fitness landscapes with very bad spectral properties, in order to fathom the abilities and limitations of the employed algorithms.

In principle, applying a shift to the power iteration can lead to better convergence properties. The problem of the simple power iteration is that the choice of this shift

is limited by the constraint that the prior to applying the shift dominating eigenvalue has to remain the dominating eigenvalue. Otherwise the dominating eigenvector—our objective we want to compute—changes. Hence, the choice of the shift is very limited. In contrast to that, *shift-and-invert* methods such as the *inverse iteration* or the *Rayleigh quotient iteration* (see [Section 2.3](#)) allow (conceptually) for an “arbitrary good” shift. This is due to the fact that such iterations run a power iteration on a shifted and inverted variant of the input matrix. That means, such algorithms proceed by solving linear systems of the form $(A - \mu\mathcal{I})x_{k+1} = x_k$ in each iteration. Thus, since the eigenvalues of the inverse of a matrix A are the reciprocals of the eigenvalues of A , the shift μ has to be chosen such that $\lambda_1 - \mu$ is the smallest eigenvalue of $A - \mu\mathcal{I}$, in order that $(\lambda_1 - \mu)^{-1}$ is the largest eigenvalue of $(A - \mu\mathcal{I})^{-1}$.

A high level objective of this thesis is to extend the work of Niederbrucker and Gansterer [[2011a,b](#)] to shift-and-invert methods. This aim raises the problem of solving the shifted system efficiently in an iterative fashion, e. g., by Krylov subspace methods which utilize the available efficient matrix vector routines. Since the efficiency of Krylov subspace methods for linear systems heavily relies on the application of an appropriate preconditioner, the design of efficient preconditioners is one of the central challenges in this thesis. Due to the high dimensionality of the problems we consider, general purpose preconditioning methods fail and we design—based on a rich theory about the occurring structured matrices—efficient special purpose preconditioners. Clearly, in cases where the power iteration converges due to good spectral properties fast, the more subtle shift-and-invert methods can not be expected to yield performance improvements (due to the costs the iterative solution of a linear system causes). Accordingly, for problems with more difficult spectral properties we show in [Chapter 5](#) that improvements of an order of magnitude can be achieved.

1.4. Outline and Contributions

The central goal of this thesis is to provide a highly efficient solver for the extreme scale eigenvalue problem occurring in the quasispecies model. Accordingly, the following chapters are either concerned with reviewing existing methods with respect to this kind of problem, or with providing novel results. It is noteworthy that the applicability of these novel results usually reaches beyond the narrow focus of the quasispecies model.

Following the motivation for Krylov subspace methods in [Section 1.2](#), we review in [Chapter 2](#) certain concrete methods for iteratively solving linear systems and eigenvalue problems, respectively. In particular, this selection consists of the methods we utilize in the experimental part of this thesis (see [Chapter 5](#)). Moreover, we give an introduction to the concept of preconditioning. As mentioned earlier, almost all fast transforms, i. e., dense matrix vector products with sub-quadratic complexity, are closely related to the Kronecker product [[Van Loan, 2000](#)]. In [Chapter 3](#), we shed light on the Kronecker product from several perspectives. Besides its basic properties, we particularly focus on the rich structure the Kronecker product provides and how it enables to solve many problems at the factor level instead of the level of the entire product. Moreover, we discuss the important relationship between the Kronecker product and fast matrix algorithms, as well as problems related to the approximation with Kronecker products.

1. Introduction

The novel contributions of this thesis consist of [Chapter 4](#) and [Chapter 5](#). In [Chapter 4](#), we provide an in-depth analysis of so-called Hamming distance-based matrices. This class of matrices is defined by the property that the (i, j) -th element of the matrix solely depends on the Hamming distance between the indices i and j (appropriately interpreted as finite strings). Conceptually, such matrices appear in the quasispecies model in terms of the mutation matrix Q . Yet, so far, the explicit treatment of this kind of matrices was bypassed by restricting the considered mutation models to certain special cases in which the mutation matrix Q can, e. g., be represented in terms of a Kronecker product. The treatment of Hamming distance-based matrices in [Chapter 4](#) is comprehensive in the sense that we consider the treatment of these matrices from a theoretical as well as from a computational (practical) point of view. Concretely, we give a complete characterization of the structure of these matrices and generically show, e. g., how the eigenvalues and the elements of Hamming distance-based matrices are related to each other. Moreover, we also provide results on the algebraic structure of the set of all Hamming distance-based matrices, show how to perform common matrix operations efficiently within this class of matrices and we analytically solve several approximation problems regarding Hamming distance-based matrices.

In [Chapter 5](#) we employ the results obtained and discussed in the chapters before in order to provide efficient solvers. In particular, we utilize at several points the results obtained in [Chapter 4](#): *(i)* to generalize the instances of the quasispecies model which can be computationally handled, *(ii)* to provide efficient matrix algorithms for being applied in the respective Krylov subspace methods, and *(iii)* to efficiently compute structured preconditioners which are additionally applied in order to improve the overall performance. Moreover, we indicate other potential applications for preconditioners based on Hamming distance-based matrices as well as the for the introduced theory of Hamming distance-based matrices as such.

Finally, [Chapter 6](#) summarizes and concludes the thesis. In order to complement the main text, [Appendix A](#) consists of a collection of frequently used definitions and notations. Moreover, [Appendix B](#) contains supplementary material to the Perron-Frobenius theory of nonnegative matrices, which is applied in the quasispecies model.

Summarizing, the contributions of this thesis are twofold and significantly differ in their nature. On the one hand, from a high level point of view, we consider the iterative solution of an extreme scale eigenvalue problem which provides a rich structure—a problem from numerical analysis. On the other hand, the main theoretical contributions consist of the comprehensive analysis of the structure and properties of Hamming distance-based matrices—certainly a piece of discrete mathematics. Altogether, we show how a tandem of fast special purpose algorithms and appropriate implementations can lead to tremendous improvements over standard approaches which do not target a specific problem.

2. Matrix-free Iterative Methods: Algorithms & Tools

Following the motivation and introduction of Krylov subspace methods in [Section 1.2](#), we give in the following a brief overview over particular Krylov subspace methods for the fundamental tasks of solving linear systems and eigenvalue problems, respectively. That means we consider matrix algorithms which solely rely on a routine for evaluating matrix vector products with the input matrix, instead of an explicit representation of the matrix itself. Details and further material can, e. g., be found in the textbooks of Trefethen and Bau [[1997](#)], Golub and Van Loan [[1996](#)], and Saad [[2003](#)].

2.1. Linear Systems

The solution of linear systems $Ax = b$ is one of the most fundamental operations in computational science. Depending on the shape of the system matrix A and the available resources different approaches are used in practice. In case the problem dimensions are reasonably small (with respect to the target system), direct solvers based on factorizations of A such as the LU decomposition or the QR decomposition can be employed. Among other reasons, such approaches suffer in case of a sparse large scale matrix A from the fact that decompositions of sparse matrices are in general dense. If direct methods fail, iterative approaches such as Krylov subspace methods have to be used. Needless to say, Krylov subspace methods are far from being the only kind of iterative methods for the solution of linear systems. For instance, there exists a widely considered type of methods which is based on stationary iterations [[Saad, 2003, §4](#)]. Such methods are based on a linear iteration of the form $x_{k+1} = Mx_k + Nb$, where the matrices M, N have to be carefully chosen with respect to A . Accordingly, the exact solution in this schemes is the stationary point of the iteration. Methods of this type are, e. g., the Jacobi and Gauss-Seidl method as well as SSOR [[Saad, 2003, §4.1](#)].

The introduction of Krylov subspace methods in [Section 1.2](#) revealed that orthogonality is a fundamental concept in the theory of Krylov subspace methods. At this point it is important to keep in mind that in floating-point arithmetic rigorous orthogonality is (besides exceptions) impossible to achieve. Nevertheless, the basic theory always assumes exact arithmetic. This includes especially all assertions involving assumptions or conclusions about orthogonality properties. The assumption of exact arithmetic is clearly handy for deriving theoretical results but one must not neglect the side effects floating-point arithmetic causes. In general, it is a fairly delicate task to reason about the effects of floating-point arithmetic on certain Krylov subspace methods. This is particularly true for Lanczos-style methods where the (re-)orthogonalization happens implicitly which makes it very sensitive to (small) perturbations with increasing number

of iterations. For an extensive treatment of the subtleties of the Lanczos process see Cullum and Willoughby [2002].

In iterative methods—especially with respect to floating-point arithmetic—it is often a priori not clear when to stop. A very simple and popular a posteriori measure for the quality of an iteratively computed approximate solution x_j of a linear system $Ax = b$ is the *residual* $r_j := b - Ax_j$. Intuitively it seems to be a good idea to aim in each iteration for a residual r_j whose norm $\|r_j\|$ is as small as possible. This is exactly the idea of the GMRES algorithm [Saad and Schultz, 1986]. More concretely, the GMRES algorithm is based on the Arnoldi iteration with the only difference that after each step in the iteration the vector $x_j \in x_0 + \mathcal{K}_j(A, r_0)$ is computed which minimizes the residual norm $\|r_j\|_2$. This is done by solving a certainly structured least squares problem. In case of GMRES $\mathcal{L}_j = A\mathcal{K}_j$ and it can be shown that the Galerkin condition with respect to this particular choice of \mathcal{L}_k is equivalent to the minimization of the residual norm [Saad, 2003, Proposition 5.3]. As an Arnoldi-based method GMRES clearly also has the practical disadvantage that the full Krylov subspace basis has to be kept in memory.

Under the assumption that A is symmetric and positive definite the less memory intensive *Conjugate Gradient* (CG) method [Hestenes and Stiefel, 1952] can be employed. As Algorithm 2.1 depicts, CG is based on a short recurrence and—not surprisingly—it can be derived from the Lanczos iteration [Saad, 2003, § 6.7.1]. The assumption that A

Algorithm 2.1 Conjugate Gradient method [Trefethen and Bau, 1997, Algorithm 38.1]

Input: symmetric positive definite $A \in \mathbb{R}^{n \times n}$ (i. e., a routine $v \mapsto Av$), $b \in \mathbb{R}^n$

Output: approximate solution $x_j \in \mathbb{R}^n$ of $Ax = b$

```

1:  $x_0 \leftarrow \mathbf{0}$ ,  $r_0 \leftarrow b$ 
2:  $p_0 \leftarrow r_0$ 
3: for  $j \leftarrow 1, 2, \dots$  do
4:    $\alpha_j \leftarrow (r_j, r_j) / (Ap_j, p_j)$ 
5:    $x_{j+1} \leftarrow x_j + \alpha_j p_j$ 
6:    $r_{j+1} \leftarrow r_j - \alpha_j Ap_j$ 
7:    $\beta_j \leftarrow (r_{j+1}, r_{j+1}) / (r_j, r_j)$ 
8:    $p_j \leftarrow r_{j+1} + \beta_j p_j$ 
9: end for

```

is positive definite allows to define the A -norm $\|x\|_A := \sqrt{(Ax, x)}$ (also called *energy-norm*). Based on this norm, the aim of the CG algorithm is to minimize the A -norm of the *error* $e_k = x_k - x_*$ in each iteration, where x_* denotes the correct solution. From a global point of view, the CG algorithm aims at minimizing the quadratic form $E(x) = (Ax, x)/2 - (b, x)$. In contrast to the simple steepest descent method which goes in every iteration along the direction of the steepest descent (here $\nabla E(x_k) = -r_k$), the CG algorithm uses more cleverly chosen search directions (p_j in Algorithm 2.1). In particular, the search direction p_j are pairwise A -conjugate (orthogonal with respect to $(\cdot, \cdot)_A$) whereas the residuals r_j are pairwise orthogonal, i. e., $(p_j, p_i)_A = (Ap_j, p_i) = 0$ and $(r_j, r_i) = 0$ for all $i < j$ [Trefethen and Bau, 1997, Theorem 38.1]. Analogous to GMRES where the residual norms converge monotonously, in case of CG it holds that the errors e_j converge monotonously, i. e., $\|e_j\|_A \leq \|e_{j-1}\|_A$ [Trefethen and Bau, 1997,

Theorem 38.2]. Before we elaborate some more details on the convergence (speed) of the CG algorithm we introduce its generalization beyond the case of symmetric positive matrices.

The requirement of a symmetric positive definite matrix in CG is obviously a substantial restriction. Due to the connection of the CG method to the Lanczos iteration it does not surprise that also a more general CG variant called BiCG exists, which utilizes the idea of biorthogonalization. Besides that BiCG works for general matrices A , it also allows for transpose-free implementations, which solely require the matrix vector product with A (see Algorithm 2.2). Hence, BiCG has in theory very attractive properties but in practice it often turns out that it suffers from a potentially irregular convergence and consequently, also from inaccuracies introduced by floating-point arithmetic. Concerning this issues, a practically more robust variant of BiCG called BiCGSTAB [van der Vorst, 1992]¹ was introduced.

Algorithm 2.2 BiCG Stabilized

 [Saad, 2003, Algorithm 7.7]

Input: $A \in \mathbb{R}^{n \times n}$ (i. e., a routine $v \mapsto Av$), $b \in \mathbb{R}^n$

Output: approximate solution $x_j \in \mathbb{R}^n$ of $Ax = b$

```

1:  $x_0 \leftarrow$  initial guess,  $r_0 \leftarrow b - Ax_0$ 
2:  $r_0^* \leftarrow$  initial guess
3:  $p_0 \leftarrow r_0$ 
4: for  $j \leftarrow 0, 1, \dots$  do
5:    $\alpha_j \leftarrow (r_j, r_0^*) / (Ap_j, r_0^*)$ 
6:    $s_j \leftarrow r_j - \alpha_j Ap_j$ 
7:    $\omega_j \leftarrow (As_j, s_j) / (As_j, As_j)$ 
8:    $x_{j+1} \leftarrow x_j + \alpha_j p_j + \omega_j s_j$ 
9:    $r_{j+1} \leftarrow s_j - \omega_j As_j$ 
10:   $\beta_j \leftarrow (r_{j+1}, r_0^*) / (r_j, r_0^*) \cdot \alpha_j / \omega_j$ 
11:   $p_{j+1} \leftarrow r_{j+1} + \beta_j (p_j - \omega_j Ap_j)$ 
12: end for

```

In general, the convergence behavior of Krylov subspace methods is tightly coupled to spectral properties of the respective input matrices. An important aspect one should not neglect is that methods like CG can theoretically—in exact arithmetic—be viewed as direct methods. In particular, for a symmetric positive definite $n \times n$ matrix CG converges in exact arithmetic after at most n steps to the correct solution. Hence, under the assumption of a matrix vector product with quadratic complexity CG is tantamount to an $\mathcal{O}(n^3)$ effort, in full analogy to factorization-based direct solvers. More generally, the following assertion about the convergence of CG holds.

Theorem 2.1 (Trefethen and Bau [1997], Theorem 38.4). *If a symmetric positive definite matrix A has k distinct eigenvalues, then CG converges in at most k steps.*

The assertion of Theorem 2.1 nicely illustrates how Krylov subspace methods potentially benefit from “good” spectral properties. While this is again a statement whose

¹This paper was awarded to be the most cited paper in mathematics in the 90s [van der Vorst, 2004].

ultimate validity is only guaranteed in exact arithmetic, it can also be observed in floating-point arithmetic that clusters of similar eigenvalues greatly enhance the convergence speed of CG. As it can be expected, it is not only the structure of the spectrum what matters, but also the range of eigenvalues, e. g., in terms of the condition number with respect to the Euclidean norm $\kappa_2 = \lambda_{\max}/\lambda_{\min}$.

Theorem 2.2 (Trefethen and Bau [1997], Theorem 38.5). *In the CG algorithm, applied to a symmetric positive definite matrix A , the errors e_k satisfy*

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2 \left(\frac{\sqrt{\kappa_2} - 1}{\sqrt{\kappa_2} + 1} \right)^k.$$

If the goal is to solve a linear system $Ax = b$, it is usually not important if indeed exactly the particular system given by A was solved—what matters is the computation of the correct solution. Hence, any linear system equivalent to $Ax = b$ might be considered. In particular, with respect to [Theorem 2.1](#) and [Theorem 2.2](#), it is clear how to judge different equivalent systems good or bad. The aim to find an equivalent system with particularly good (spectral) properties leads to the topic of preconditioning, which we introduce next.

2.2. Preconditioning

Consider again a linear system $Ax = b$. As motivated before, we are interested in the computation of a matrix P , the so-called *preconditioner* which allows to improve, e. g., the condition number of a particular linear system. This can be done, among others, in the following two ways:

(i). *Left preconditioning:* $Ax = b \Leftrightarrow PP^{-1}Ax = b \Leftrightarrow P^{-1}Ax = P^{-1}b$

(ii). *Right preconditioning:* $Ax = b \Leftrightarrow AP^{-1}Px = b \Leftrightarrow AP^{-1}y = b, Px = y$

Moreover, left and right preconditioning can obviously also be combined. Needless to say, the choice of P is in most cases a trade off: On the one hand, P^{-1} should be a good approximate of the inverse of A in order that $P^{-1}A$ is well-conditioned, i. e., $P^{-1}A \approx \mathcal{I}$. On the other hand, the computation as well as the application of P^{-1} should be as cheap as possible to keep the additional costs raised by the preconditioning low. These goals are particularly hard to achieve in the usual context of a (data-)sparse matrix A , since the exact inverse of A will in general be a (less structured) dense matrix. Therefore, it is challenging to find a good tradeoff between computational effort and approximation quality, which is essential such that the properties of the preconditioned problem are indeed (substantially) better than the properties of the original problem.

We saw already that Krylov subspace methods commonly only require a matrix vector multiplication routine to be available. Therefore, such methods are of a very generic nature and specific properties of particular problems will usually not immediately lead to good convergence properties. Hence, it can be seen as one of the aims of a good preconditioner to utilize the available specific problem structure. Moreover, it is commonly believed that the choice of an appropriate preconditioner is more important than the

choice of the particular Krylov subspace method. Besides that preconditioners can be used to incorporate knowledge about the problem, their application is usually motivated by two aims: First and foremost, the major aim is to reduce the condition number to a reasonably small value (cf. [Theorem 2.2](#)). Moreover, certain methods also strongly benefit from a clustering of eigenvalues (cf. [Theorem 2.1](#)).

Out of the many kinds, we consider in the following two fairly generic types of preconditioners. More specific methods, for instance those used for linear systems stemming from the discretization of partial differential equations, are not included in this brief overview (see, e. g., Saad [[2003](#), §§ 9, 10]).

2.2.1. Incomplete Factorizations

Recall that common direct solvers for the solution of a linear system $Ax = b$ are based on computing factorizations of A , e. g., $A = LU$ with an upper triangular matrix L and a lower triangular matrix U . In particular, the major (i. e., cubic) effort of a direct solver lies in the computation of the matrix factorization since the solution of a linear system based on an available factorization can cheaply be computed in a quadratic amount of time. Hence, it appears to be a reasonable way to define a preconditioner in terms of a matrix factorization. In particular, the task is to find an efficiently computable approximate factorization, e. g., $LU \approx A$ which is a good approximate of A in order to obtain a well conditioned preconditioned system.

In case a sparse matrix A is given, its incomplete factorization has to be sparse as well—yet, it should be a good approximation. This is a nontrivial problem, since factorizations of sparse matrices are in most cases dense. A natural approach is to define a sparsity pattern \mathcal{S} (the set of elements which are guaranteed to be zero) and accordingly, to compute an approximate factorization with this sparsity pattern. It turns out that only for certain sparsity patterns according factorizations can be computed, i. e., in general there does not exist a factorization showing a particular pre-defined sparsity pattern \mathcal{S} . Nevertheless, certain classes of matrices permit sparse decompositions.

Definition 2.1 (*M*-matrix). A square matrix $A \in \mathbb{R}^{n \times n}$ which satisfies

- (i). $a_{i,i} > 0$ for all $1 \leq i \leq n$
- (ii). $a_{i,j} \leq 0$ for all $i \neq j$ with $1 \leq i, j \leq n$
- (iii). A is regular, i. e., A is invertible
- (iv). A^{-1} is a non-negative matrix (cf. [Definition B.1](#))

is said to be an *M*-matrix. //

Concretely, let A be an *M*-matrix and \mathcal{S} be a given sparsity pattern which does not include diagonal elements. Then there exists an incomplete factorization $A \approx LU$ with the sparsity pattern \mathcal{S} [Saad, [2003](#), Theorem 10.2]. The assumption of an *M*-matrix is obviously fairly restrictive but on the other hand, it covers, e. g., several important kinds of matrices stemming from the discretization of partial differential equations.

For more general scenarios basically two branches of directions are known in the area of incomplete factorizations. First, based on the previously introduced idea of computing a factorization based on a given sparsity pattern people considered generalizations of this approach, where a certain level of additional nonzero entries (*fill-in*) is allowed [Saad, 2003, §10.3.3] in order to obtain a sparse factorization. Needless to say, it is a matter of the particular problem if the fill-in required for computing a factorization is reasonably small or not. A counterpart to the aforementioned sparsity pattern driven approaches are methods which in principle compute complete factorizations but drop out values according to some strategy in order to eventually obtain a sparse factorization. A straightforward approach is to simply specify a numerical threshold τ below which elements are not kept in the factorization [Saad, 2003, §10.4]. While such simple strategies can prove very successful in certain cases it is again in general not clear whether a particular threshold τ leads to a sufficient sparsification or not and second, if the obtained factorization is still reasonably accurate with respect to the aim for a good preconditioner.

Altogether, the idea behind preconditioners based on incomplete factorizations is very simple and intuitive. Nevertheless, its practical implementation can raise subtle problems depending on the respective matrices which are considered.

2.2.2. (Data-)Sparse Approximate Inverse

In the previous section we considered an indirect (factorization-based) approach towards the approximate inversion of a matrix. It is even more natural to aim at computing an approximate inverse directly. Once again, the underlying problem is that the inverse A^{-1} of a sparse matrix A is in general a dense matrix.

As counterpart to the incomplete factorizations discussed in Section 2.2.1 one can consider searching an approximate inverse \tilde{A}^{-1} of A with a particular predefined sparsity pattern. Grote and Huckle [1997] generically showed how to compute a so-called *sparse approximate inverse* of a matrix, i. e., an approximate inverses with a predefined sparsity pattern. More concretely, they suggest to search the matrix M which minimizes $\|AM - \mathcal{I}\|_F$, i. e., which approximates the inverse of A best with respect to the Frobenius norm. This can be done efficiently by observing for matrices $A, M \in \mathbb{R}^{m \times m}$ the relationship

$$\|AM - \mathcal{I}\|_F^2 = \sum_{i=1}^m \|(AM - \mathcal{I})e_i\|_2^2,$$

where e_i denotes the i -th canonical basis vector, i. e., e_i is zero except of the i -th entry, which is one. This characterization shows that minimizing $\|AM - \mathcal{I}\|_F$ can be done by columnwise minimizing $\|AM(:, i) - e_i\|_2$ for all i , independently. Note that this approach does not suffer from the problem of fill-in, as it is the case for incomplete factorizations based on a predefined sparsity pattern. On the other hand, it is still the size and structure of the provided sparsity pattern which determines the quality of the approximation. So while Grote and Huckle [1997] provide a generic framework for computing an (with respect to the Frobenius norm optimal) approximate inverse, it is a tricky task to find for concrete matrices a (small) sparsity pattern which leads to a good approximation.

Of course the computation of an approximate inverse does not only make sense in the context of sparse matrices but especially also in the case of data-sparse matrices. Due to the rich structure of data-sparse matrices it is often possible to analytically compute the best approximate inverse, i. e., the matrix M which minimizes $\|AM - \mathcal{I}\|_F$. In [Section 3.3](#) we will for example see how to compute the best approximating Kronecker product. Moreover, for highly structured matrices, such as Toeplitz matrices, a rich preconditioning theory is available [[Benedetto *et al.*, 1993](#); [Chan and Ng, 1993](#)].

As elaborated in [Section 1.3](#), we are confronted with data-sparse matrices in the quasispecies model. Therefore, the idea of preconditioners based on a data-sparse approximate inverse fits very well in this context. In particular, our analytical treatment of Hamming distance-based matrices also includes approximation results. Later, in [Chapter 5](#), we utilize these results in order to derive efficient preconditioners.

2.3. Eigenvalue Problems

So far we exclusively dealt with the iterative solution of linear systems. Not surprisingly, one can in an analogous spirit also derive solvers for eigenvalue problems. For instance, the Lanczos process (cf. [Section 1.2](#)) can also be turned into an efficient eigenvalue solver [[Cullum and Willoughby, 2002](#)], with analogous pros and cons as the ones discussed before in the case of linear systems. In this section we discuss methods which are suited for the matrix-free computation of the extremal eigenpair of a given matrix. Moreover, the related problem of computing the extremal singular value and its corresponding singular vectors is discussed.

2.3.1. Extremal Eigenpairs

The quasispecies model motivates the investigation of the problem of computing the eigenvector corresponding to the largest eigenvalue in magnitude of a particular matrix (cf. [Section 1.3](#)). Therefore, we focus in this brief overview of iterative methods for eigenvalue problems on methods which have a natural focus on this particular task. Moreover, we make throughout this section the following assumptions in order to compute the dominating eigenpair of a matrix A : We are given a matrix vector multiplication routine for computing $v \mapsto Av$. Moreover, $A \in \mathbb{R}^{n \times n}$ is a square real matrix and by λ_i with $1 \leq i \leq n$, we denote its eigenvalues ordered by absolute value. Additionally, we assume that $|\lambda_1| > |\lambda_i|$ for all $i > 1$. Thus, the problem of finding the dominating eigenpair of a matrix A is always well defined. In case of the quasispecies model the gap $|\lambda_1| > |\lambda_i|$ is guaranteed by the Perron-Frobenius theory (see [Appendix B](#)).

In [Section 1.2](#) we motivated the need for the computation of an orthogonal Krylov subspace basis by the fact that $A^k r$ “converges” with increasing k . Now we can turn this “problem” into something positive, by deriving a very simple, yet in certain cases very efficient method for computing the dominating eigenpair of a matrix. [Algorithm 2.3](#) depicts the *power iteration* which simply repeatedly applies the input matrix A . Additionally, in every iteration the newly obtained vector is normalized in order to enable convergence and avoid numerical problems. It can be shown that the convergence of the power iteration is closely related to the ratio λ_2/λ_1 . Concretely, the eigenvector

Algorithm 2.3 Power Iteration [Trefethen and Bau, 1997, Algorithm 27.1]

Input: $A \in \mathbb{R}^{n \times n}$ (i. e., a routine $v \mapsto Av$)**Output:** approximation of dominating eigenpair λ_j, b_j 1: $b_0 \leftarrow$ normalized initial guess2: **for** $j \leftarrow 1, 2, \dots$ **do**3: $b_j \leftarrow Ab_{j-1}$ 4: $b_j \leftarrow b_j / \|b_j\|$ 5: $\lambda_j \leftarrow (b_j, b_{j-1})$ $\triangleright \lambda_j$ is the Rayleigh quotient $\frac{(Ab_{j-1}, b_{j-1})}{(b_{j-1}, b_{j-1})}$ 6: **end for**

approximation converges linearly with rate λ_2/λ_1 whereas the eigenvalue approximation converges quadratically with rate λ_2/λ_1 [Trefethen and Bau, 1997, Theorem 27.1]. Therefore, in case the ratio λ_2/λ_1 is close to one, the convergence of the power iteration is clearly inferior. Since the shifted matrix $A - \mu\mathcal{I}$ has the same eigenvectors as A and the eigenvalues of $A - \mu\mathcal{I}$ are given by $\lambda_i - \mu$, a well chosen shift can lead to an improved ratio of the two dominating eigenvalues $(\lambda_k - \mu)/(\lambda_1 - \mu)$. The problem is of course that the choice of potential shifts is limited by the constraint that $\lambda_1 - \mu$ has to remain the largest eigenvalue in magnitude. Thus, in this simple form shifts are of limited success.

Recall that the eigenvalues of the inverse of a matrix A are given by the reciprocals of the eigenvalues of A . Hence, if μ is a reasonably good approximation of the extremal eigenvalue λ_1 of A , then $\lambda_1 - \mu$ will be the smallest eigenvalue of the matrix $A - \mu\mathcal{I}$ and consequently, the dominating eigenvalue of $(A - \mu\mathcal{I})^{-1}$. This observation immediately leads to the *inverse iteration* as depicted in Algorithm 2.4. The analysis of the inverse

Algorithm 2.4 Inverse Iteration [Trefethen and Bau, 1997, Algorithm 27.2]

Input: $A \in \mathbb{R}^{n \times n}$ (i. e., a routine $v \mapsto Av$), $\mu \in \mathbb{R}$ **Output:** approximation of dominating eigenpair λ_j, b_j 1: $b_0 \leftarrow$ normalized initial guess for the dominating eigenvector2: **for** $k \leftarrow 1, 2, \dots$ **do**3: Solve $(A - \mu\mathcal{I})b_j = b_{j-1}$ 4: $b_j \leftarrow b_j / \|b_j\|$ 5: $\lambda_j \leftarrow (b_j, b_{j-1})$ $\triangleright \lambda_j$ is the Rayleigh quotient $\frac{(Ab_{j-1}, b_{j-1})}{(b_{j-1}, b_{j-1})}$ 6: **end for**

iteration can be done along the lines of the analysis of the power iteration. More precisely, the convergence of the inverse iteration relies on the quantity $(\lambda_1 - \mu)/(\lambda_k - \mu)$ where k is the index which maximizes $\lambda_l - \mu$ over all $l \neq 1$. Analogously to the power iteration the eigenvector approximation converges linearly with rate $(\lambda_1 - \mu)/(\lambda_k - \mu)$ and the eigenvalue approximation converges quadratically with rate $(\lambda_1 - \mu)/(\lambda_k - \mu)$ [Trefethen and Bau, 1997, Theorem 27.2]. Therefore, the better the shift μ is chosen the faster the inverse iteration converges. From a practical point of view, it is not immediately clear how to choose the shift (well). This problem can be solved, e. g., by a priori knowledge about the spectrum or by running a few steps of the power iteration in order to get a first approximation of the dominating eigenvalue.

A seemingly big obstacle of shift-and-invert methods as the inverse iteration is given by the fact that with a more accurate shift the shifted system becomes (almost) singular. Contrasting with this expectation, it turns out that this pitfall is not a problem [Trefethen and Bau, 1997, Exercise 27.5]. The open question in the inverse iteration is if and how a “good” shift can improve the convergence speed. It turns out that a clever (adaptive) choice of the shift indeed leads to a substantial improvement. A variant of the inverse iteration with adaptive shifts is given by the Rayleigh quotient iteration (see Algorithm 2.5). It turns out that the simple tweak of choosing the current shift

Algorithm 2.5 Rayleigh Quotient Iteration [Trefethen and Bau, 1997, Algorithm 27.3]

Input: $A \in \mathbb{R}^{n \times n}$ (i. e., a routine $v \mapsto Av$)

Output: approximation of dominating eigenpair λ_j, b_j

- 1: $b_0 \leftarrow$ normalized initial guess for the dominating eigenvector
 - 2: $\mu_0 \leftarrow$ initial guess for the dominating eigenvalue
 - 3: **for** $j \leftarrow 1, 2, \dots$ **do**
 - 4: Solve $(A - \mu_j \mathcal{I}_n)b_j = b_{j-1}$
 - 5: $b_j \leftarrow b_j / \|b_j\|$
 - 6: $\lambda_j \leftarrow (b_j, b_{j-1})$ $\triangleright \lambda_j$ is the Rayleigh quotient $\frac{(Ab_{j-1}, b_{j-1})}{(b_{j-1}, b_{j-1})}$
 - 7: $\mu_j \leftarrow \lambda_j$
 - 8: **end for**
-

as the lastly obtained eigenvalue approximation leads to substantial improvements. In particular, it can be shown that for a symmetric matrix the eigenvector as well as the eigenvalue approximation converge cubically, whenever the Rayleigh quotient iteration converges [Trefethen and Bau, 1997, Theorem 27.3]. Note that this impressive gain in the provable convergence speed is what requires the symmetry of A —the Rayleigh as such works also for asymmetric matrices.

2.3.2. Singular Value Decomposition

One of the most influential tools in the theory and application of (numerical) linear algebra is the *singular value decomposition* (SVD), which is in the case of real matrices characterized as follows.

Theorem 2.3 (Golub and Van Loan [1996, Theorem 2.5.2]). *Let $A \in \mathbb{R}^{m \times n}$ be a real matrix. Then there exist orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ as well as a diagonal matrix $\Sigma \in \mathbb{R}^{m \times n}$ such that $A = U\Sigma V^T$. Moreover, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m,n)} \geq 0$ holds for the diagonal elements of Σ .*

Let $A = U\Sigma V^T$ be the SVD of a matrix A . Then the diagonal elements of Σ are called *singular values* of A . Likewise, we call the columns of U and V , the left and right *singular vectors* of A , respectively. The nonzero singular values of a matrix A are closely related to the eigenvalues of A by the fact that they are the square roots of the eigenvalues of the matrices AA^T and $A^T A$, respectively [Trefethen and Bau, 1997, Theorem 5.4]. Hence, in case of a symmetric matrix its singular values are the

absolute values of its eigenvalues. Accordingly, it can be found that the left singular vectors are the eigenvectors of AA^T , whereas the right singular vectors are given by the eigenvectors of $A^T A$. One of the most illustrative applications of the SVD is the low-rank approximation of a matrix.

Example 2.1 (Rank- k approximation). Any matrix A can be expressed as sum of rank-1 matrices in terms of the SVD of A . Let $A = U\Sigma V^T$ be the SVD of A and r its rank, then $A = \sum_{i=1}^r \sigma_i U(:, i) V(:, i)^T$. Moreover, recall the assumption that the singular values are ordered, i. e., $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0)$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$. Then $\sum_{i=1}^k \sigma_i U(:, i) V(i, :)^T$ is the best rank- k approximation of A with respect to $\|\cdot\|_2$ and $\|\cdot\|_F$ [Golub and Van Loan, 1996, Theorem 2.5.3]. //

We will later use this result about optimal rank- k approximation in the course of the derivation and computation of preconditioners. For the multitude of further important applications of the SVD see, e. g., the textbooks of Golub and Van Loan [1996] and Trefethen and Bau [1997] where they appear over and over. As one might expect, also the direct computation of the SVD is tantamount to a cubic effort [Trefethen and Bau, 1997, § 31], in full analogy to other dense matrix decompositions.

Consider the problem of computing the best rank-1 approximation of matrix A . By [Example 2.1](#) we know that this problem can easily be solved once we have the dominating singular value with its corresponding vectors at hand. It turns out that once more the Lanczos process can be applied. [Algorithm 2.6](#) depicts the so-called SVD Lanczos process [Golub and Van Loan, 1996] which allows to efficiently compute the dominating singular value together with its associated singular vectors. The simple idea behind the Lanczos SVD process is to derive—via the Lanczos process—a very small matrix B (i. e., a square matrix whose dimensionality is given by the number of executed Lanczos steps) of which it can be shown that its dominating singular value(s) and vector(s) accurately approximate those of A . Based on the SVD of the (very small) matrix B the dominating singular value and its associated singular vectors are eventually obtained.

Algorithm 2.6 SVD Lanczos

[Golub and Van Loan, 1996, pp. 495–496]

Input: $A \in \mathbb{R}^{m \times n}$ (i. e., routines $v \mapsto Av$, $v \mapsto A^T v$), maximal iterations j_{\max} **Output:** approximation of the largest singular value/vectors $\sigma \in \mathbb{R}$, $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$

- 1: $v_1 \leftarrow$ initial guess
- 2: $p_0 \leftarrow v_1$, $\beta_1 \leftarrow 0$, $j \leftarrow 0$, $u_0 = \mathbf{0}$
- 3: **while** $\beta_j \neq 0$ and $j \leq j_{\max}$ **do**
- 4: $v_{j+1} \leftarrow p_j / \beta_j$
- 5: $j \leftarrow j + 1$
- 6: $r_j \leftarrow Av_j - \beta_{j-1}u_{j-1}$
- 7: $\alpha_j \leftarrow \|r_j\|_2$
- 8: $u_j \leftarrow r_j / \alpha_j$
- 9: $p_j \leftarrow A^T u_j - \alpha_j v_j$
- 10: $\beta_j \leftarrow \|p_j\|_2$
- 11: **end while**

$$12: B \leftarrow \begin{pmatrix} \alpha_1 & \beta_1 & 0 & 0 \\ 0 & \alpha_2 & \ddots & 0 \\ 0 & 0 & \ddots & \beta_{j-1} \\ 0 & 0 & 0 & \alpha_j \end{pmatrix} \in \mathbb{R}^{j \times j}$$

- 13: Compute the dominating singular value σ_B and corresponding vectors u_B, v_B of B
- 14: $\sigma \leftarrow \sigma_B$
- 15: $u \leftarrow (u_1, u_2, \dots, u_j)u_B$ $\triangleright (u_1, u_2, \dots, u_j)$ is a $m \times j$ matrix
- 16: $v \leftarrow (v_1, v_2, \dots, v_j)v_B$ $\triangleright (v_1, v_2, \dots, v_j)$ is a $n \times j$ matrix

3. The Kronecker Product

Definition 3.1. For matrices $A \in \mathbb{R}^{m_1 \times n_1}$ and $B \in \mathbb{R}^{m_2 \times n_2}$ we define the *Kronecker product* (or *tensor product*) of A and B as the $m_1 m_2 \times n_1 n_2$ block matrix

$$A \otimes B := \begin{pmatrix} a_{1,1}B & \cdots & a_{1,n_1}B \\ \vdots & \ddots & \vdots \\ a_{m_1,1}B & \cdots & a_{m_1,n_1}B \end{pmatrix}.$$

//

In this chapter we review the most important properties of the Kronecker product and show how to utilize it as a “language” for describing and manipulating efficient algorithms on an abstract level. Moreover, we discuss how to approximate given matrices by Kronecker products of matrices and show up relations to low rank approximation problems and the singular value decomposition. Whenever (full) proofs are omitted these can be found in the literature (see, e. g., Steeb [1997]).

3.1. Basic Properties

Proposition 3.1. Let $A \in \mathbb{R}^{m_1 \times n_1}$, $B \in \mathbb{R}^{m_2 \times n_2}$ and $C \in \mathbb{R}^{m_3 \times n_3}$ be arbitrary real matrices and $\alpha \in \mathbb{R}$ a real constant. Then the following properties hold.

- (i). $\mathcal{I}_m \otimes \mathcal{I}_n = \mathcal{I}_{mn}$
- (ii). $(\alpha A) \otimes B = \alpha(A \otimes B) = A \otimes (\alpha B)$
- (iii). $(A \otimes B)^T = A^T \otimes B^T$
- (iv). $A \otimes (B \otimes C) = (A \otimes B) \otimes C$

Proof. The validity of (i)–(iii) follows immediately from Definition 3.1. For the associativity of \otimes consider the following block representation of $(A \otimes B) \otimes C$.

$$\begin{aligned} & (A \otimes B) \otimes C = \\ & = \left(\begin{array}{ccc|ccc} a_{1,1}b_{1,1}C & \cdots & a_{1,1}b_{1,n_2}C & & & \\ \vdots & \ddots & \vdots & & & \\ a_{1,1}b_{m_2,1}C & \cdots & a_{1,1}b_{m_2,n_2}C & (a_{1,2}B) \otimes C & \cdots & (a_{1,n_1}B) \otimes C \\ \hline & & & (a_{2,2}B) \otimes C & \cdots & (a_{2,n_1}B) \otimes C \\ & & & \vdots & \ddots & \vdots \\ & & & (a_{m_1,2}B) \otimes C & \cdots & (a_{m_1,n_1}B) \otimes C \end{array} \right) \end{aligned}$$

3. The Kronecker Product

$$\begin{aligned}
 &= \begin{pmatrix} (a_{1,1}B) \otimes C & \cdots & (a_{1,n_1}B) \otimes C \\ \vdots & \ddots & \vdots \\ (a_{m_1,1}B) \otimes C & \cdots & (a_{m_1,n_1}B) \otimes C \end{pmatrix} \\
 &\stackrel{(ii)}{=} \begin{pmatrix} a_{1,1}(B \otimes C) & \cdots & a_{1,n_1}(B \otimes C) \\ \vdots & \ddots & \vdots \\ a_{m_1,1}(B \otimes C) & \cdots & a_{m_1,n_1}(B \otimes C) \end{pmatrix} = A \otimes (B \otimes C).
 \end{aligned}$$

□

Since the Kronecker product is associative the expression $A_1 \otimes A_2 \otimes \dots \otimes A_n$ is unambiguous. Thus, we will henceforth use $\bigotimes_{i=1}^n A_i$ as abbreviation for $A_1 \otimes A_2 \otimes \dots \otimes A_n$. Due to its inherently asymmetric definition, it is not surprising that the Kronecker product is not commutative as the following example confirms.

Example 3.1.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \mathcal{I}_3 = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \\ 3 & 0 & 0 & 4 & 0 & 0 \\ 0 & 3 & 0 & 0 & 4 & 0 \\ 0 & 0 & 3 & 0 & 0 & 4 \end{pmatrix} \neq \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 3 & 4 \end{pmatrix} = \mathcal{I}_3 \otimes \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Nevertheless, this counterexample gives already some evidence that $A \otimes B$ and $B \otimes A$ are closely related. Indeed, we will see that $A \otimes B = P(B \otimes A)Q$ for certain permutation matrices P and Q . //

Definition 3.2 (Perfect Shuffle). For positive integers p, q and $n = pq$, we define the *perfect shuffle* matrix $\mathcal{S}_{p,q} \in \mathbb{R}^{n \times n}$ as block matrix¹

$$\mathcal{S}_{p,q} := \begin{pmatrix} \mathcal{I}_n(1:q:n, :) \\ \mathcal{I}_n(2:q:n, :) \\ \vdots \\ \mathcal{I}_n(q:q:n, :) \end{pmatrix},$$

consisting of q blocks $\mathcal{I}_n(i:q:n, :) \in \mathbb{R}^{p \times n}$. //

Since $\mathcal{S}_{p,q}$ contains exactly one entry 1 in each row and column, perfect shuffles are permutation matrices. Therefore, all matrices $\mathcal{S}_{p,q}$ are orthogonal, i. e., $\mathcal{S}_{p,q} \mathcal{S}_{p,q}^T = \mathcal{I}_n$. Moreover, the transpose of a perfect shuffle is a perfect shuffle as well.

Example 3.2. Let $p = 3$ and $q = 2$, then $\mathcal{S}_{p,q}$ and $\mathcal{S}_{q,p}$ are each others transpose.

$$\mathcal{S}_{3,2} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T = \mathcal{S}_{2,3}^T$$

//

¹We use colon notation for submatrix specification (cf. [Notation A.4](#))

Lemma 3.2. *Let $p, q \in \mathbb{N}$ be arbitrary integers. Then the identity $\mathcal{S}_{p,q} = \mathcal{S}_{q,p}^T$ holds.*

Proof. The claim directly follows from the column-based equivalent of the row-based definition of $\mathcal{S}_{p,q}$ in [Definition 3.2](#), i. e.,

$$\mathcal{S}_{p,q} = \begin{pmatrix} \mathcal{I}_n(1:q:n, :) \\ \mathcal{I}_n(2:q:n, :) \\ \vdots \\ \mathcal{I}_n(q:q:n, :) \end{pmatrix} = (\mathcal{I}_n(:, 1:p:n), \mathcal{I}_n(:, 2:p:n), \dots, \mathcal{I}_n(:, p:p:n)) = \mathcal{S}_{q,p}^T.$$

□

The name perfect shuffle is motivated by the operation of $\mathcal{S}_{p,q}$ on vectors. Suppose we are given a deck of $n = pq$ cards represented by an n -dimensional vector x , then $y = \mathcal{S}_{p,q}x$ can be imagined as follows. The deck of n cards is first split up into p piles of height q and then we successively take one card from each pile to obtain y . The matrix vector multiplication $y^T = x^T \mathcal{S}_{p,q}$ can be interpreted analogously.

Example 3.3. Consider the perfect shuffle $\mathcal{S}_{3,2}$ and an arbitrary vector $x \in \mathbb{R}^6$. Then,

$$y = \mathcal{S}_{3,2}x = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \xrightarrow{\text{"split"}} \begin{array}{|c|c|c|} \hline x_1 & x_3 & x_5 \\ \hline x_2 & x_4 & x_6 \\ \hline \end{array} \xrightarrow{\text{"take"}} y = \begin{pmatrix} x_1 \\ x_3 \\ x_5 \\ x_2 \\ x_4 \\ x_6 \end{pmatrix}.$$

//

Perfect shuffle permutations allow to relate the columnwise vectorization $\text{vec}(A)$ of a matrix A with the vectorization of its transpose as well as they relate $A \otimes B$ to $B \otimes A$.

Proposition 3.3. *Let $A \in \mathbb{R}^{m_1 \times n_1}$ and $B \in \mathbb{R}^{m_2 \times n_2}$*

(i). $\text{vec}(A) = \mathcal{S}_{m_1, n_1} \cdot \text{vec}(A^T)$

(ii). $A \otimes B = \mathcal{S}_{m_1, m_2}^T \cdot (B \otimes A) \cdot \mathcal{S}_{n_1, n_2}$

Example 3.4. Let A be an arbitrary 3×2 matrix. Then one can directly confirm that

$$\text{vec}(A) = \begin{pmatrix} a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ a_{1,2} \\ a_{2,2} \\ a_{3,2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{1,1} \\ a_{1,2} \\ a_{2,1} \\ a_{2,2} \\ a_{3,1} \\ a_{3,2} \end{pmatrix} = \mathcal{S}_{3,2} \cdot \text{vec}(A^T).$$

Moreover, recall the counterexample for the commutativity of the Kronecker product given in [Example 3.1](#). According to [Proposition 3.3](#) these two matrices are related in the form

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \mathcal{I}_3 = \mathcal{S}_{3,2} \cdot \left[\mathcal{I}_3 \otimes \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \right] \cdot \mathcal{S}_{2,3}.$$

3. The Kronecker Product

This claim is verified by the following step-by-step computation (cf. [Notation A.2](#)).

$$\begin{array}{c|c|c} \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 3 & 4 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\ \hline \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 4 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \\ 3 & 0 & 0 & 4 & 0 & 0 \\ 0 & 3 & 0 & 0 & 4 & 0 \\ 0 & 0 & 3 & 0 & 0 & 4 \end{pmatrix} \end{array}$$

In the following we give a proof for the general assertions given in [Proposition 3.3](#). //

Proof of Proposition 3.3. Claim (i) follows directly from the interpretation of the matrix vector multiplication with a perfect shuffle matrix since

$$\mathcal{S}_{m_1, n_1} \cdot \text{vec}(A^T) = \mathcal{S}_{m_1, n_1} \cdot \begin{pmatrix} A(1, :)^T \\ \vdots \\ A(m_1, :)^T \end{pmatrix} = \begin{pmatrix} A(:, 1) \\ \vdots \\ A(:, n_1) \end{pmatrix} = \text{vec}(A)$$

For claim (ii), first note how $\mathcal{S}_{m_1, m_2}^T = \mathcal{S}_{m_2, m_1}$ operates on the columns of $B \otimes A$:

$$\mathcal{S}_{m_2, m_1} \cdot (B \otimes A) = \mathcal{S}_{m_2, m_1} \cdot \begin{pmatrix} b_{1,1}A & \dots & b_{1,n_2}A \\ \vdots & \ddots & \vdots \\ b_{m_2,1}A & \dots & b_{m_2,n_2}A \end{pmatrix} = \begin{pmatrix} B \otimes A(1, :) \\ \vdots \\ B \otimes A(m_1, :) \end{pmatrix}$$

Moreover, the blocks $B \otimes A(i, :)$ explicitly read as

$$B \otimes A(i, :) = \begin{pmatrix} b_{1,1}A(i, :) & \dots & b_{1,n_2}A(i, :) \\ \vdots & \ddots & \vdots \\ b_{m_2,1}A(i, :) & \dots & b_{m_2,n_2}A(i, :) \end{pmatrix}$$

Therefore, further multiplying by \mathcal{S}_{n_1, n_2} leads to

$$\begin{aligned} \begin{pmatrix} B \otimes A(1, :) \\ \vdots \\ B \otimes A(m_1, :) \end{pmatrix} \cdot \mathcal{S}_{n_1, n_2} &= \begin{pmatrix} Ba_{1,1} & \dots & Ba_{1,n_1} \\ \vdots & \ddots & \vdots \\ Ba_{m_1,1} & \dots & Ba_{m_1,n_1} \end{pmatrix} \\ &= \begin{pmatrix} a_{1,1}B & \dots & a_{1,n_1}B \\ \vdots & \ddots & \vdots \\ a_{m_1,1}B & \dots & a_{m_1,n_1}B \end{pmatrix} = A \otimes B. \end{aligned}$$

Thus, we proved the claim $A \otimes B = \mathcal{S}_{m_1, m_2}^T \cdot (B \otimes A) \cdot \mathcal{S}_{n_1, n_2}$ □

A key property of the Kronecker product is that the product $A \otimes B$ inherits most of the structure of its factors A and B . Concretely, if A and B have the property X then usually also $A \otimes B$ has the property X . In particular, if A and B are *non-negative, symmetric, triangular, orthogonal, stochastic* or *diagonal matrices*, their product $A \otimes B$ inherits the respective properties.

Theorem 3.4 (Mixed Product Property). *Let A, B, C and D be arbitrary matrices such that AC and BD are defined, then*

$$(A \otimes B)(C \otimes D) = AC \otimes BD. \quad (3.1)$$

Proof. Assume that $A \in \mathbb{R}^{m_1 \times p}$, $C \in \mathbb{R}^{p \times n_1}$ as well as $B \in \mathbb{R}^{m_2 \times q}$, $D \in \mathbb{R}^{q \times n_2}$, i. e., the products AC and BD are defined. Then the property is straightforwardly verified.

$$\begin{aligned} (A \otimes B)(C \otimes D) &= \begin{pmatrix} a_{1,1}B & \dots & a_{1,p}B \\ \vdots & \ddots & \vdots \\ a_{m_1,1}B & \dots & a_{m_1,p}B \end{pmatrix} \begin{pmatrix} c_{1,1}D & \dots & c_{1,n_1}D \\ \vdots & \ddots & \vdots \\ c_{p,1}D & \dots & c_{p,n_1}D \end{pmatrix} \\ &= \begin{pmatrix} (\sum_{k=1}^p a_{1,k}c_{k,1})BD & \dots & (\sum_{k=1}^p a_{1,k}c_{k,n_1})BD \\ \vdots & \ddots & \vdots \\ (\sum_{k=1}^p a_{m_1,k}c_{k,1})BD & \dots & (\sum_{k=1}^p a_{m_1,k}c_{k,n_1})BD \end{pmatrix} \\ &= \begin{pmatrix} [AC]_{1,1}BD & \dots & [AC]_{1,n_1}BD \\ \vdots & \ddots & \vdots \\ [AC]_{m_1,1}BD & \dots & [AC]_{m_1,n_1}BD \end{pmatrix} = AC \otimes BD \end{aligned}$$

□

In fact, the mixed product property states that for many computations it suffices to perform the respective computations on the individual factors instead of the entire Kronecker product. The following corollary states several useful properties of this kind.

Corollary 3.5. *Computations on Kronecker products usually require only independent computations on its factors. For example the following properties hold.*

- (i). *Let A and B be regular matrices, then $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.*
- (ii). *Let $A = Q_A R_A$ and $B = Q_B R_B$ be the QR decompositions of A and B , respectively. Then $A \otimes B = (Q_A \otimes Q_B)(R_A \otimes R_B)$ is the QR decomposition of $A \otimes B$.*
- (iii). *Let $A = L_A U_A$ and $B = L_B U_B$ be the LU decompositions of A and B , respectively. Then $A \otimes B = (L_A \otimes L_B)(U_A \otimes U_B)$ is the LU decomposition of $A \otimes B$.*
- (iv). *Let $A = U_A \Sigma_A V_A^T$ and $B = U_B \Sigma_B V_B^T$ be the SVDs of A and B , respectively. Then $A \otimes B = (U_A \otimes U_B)(\Sigma_A \otimes \Sigma_B)(V_A^T \otimes V_B^T)$ is the SVD of $A \otimes B$.*

Proof. By the mixed product property and the fact that the Kronecker product of orthogonal/triangular/diagonal matrices is orthogonal/triangular/diagonal. □

3. The Kronecker Product

Example 3.5. Given matrices B , C and X such that CXB^T is defined. Then the matrix equation $Y = CXB^T$ is equivalent to the linear system $\text{vec}(Y) = (B \otimes C)\text{vec}(X)$. Consider the case $B \in \mathbb{R}^{2 \times 2}$, $C \in \mathbb{R}^{2 \times 3}$ and $X \in \mathbb{R}^{3 \times 2}$. Moreover, let x_1 and x_2 denote the first and second column of X , respectively, i.e., $x_1 = X(:, 1)$, $x_2 = X(:, 2)$ and $X = (x_1, x_2)$. For this case, the stated equivalence is verified as follows.

$$\begin{aligned} Y &= CXB^T = C \cdot (x_1, x_2) \cdot B^T = (Cx_1, Cx_2) \cdot \begin{pmatrix} b_{1,1} & b_{2,1} \\ b_{1,2} & b_{2,2} \end{pmatrix} \\ &= (C(b_{1,1}x_1 + b_{1,2}x_2), C(b_{2,1}x_1 + b_{2,2}x_2)) \end{aligned}$$

$$(B \otimes C)\text{vec}(X) = \begin{pmatrix} b_{1,1}C & b_{1,2}C \\ b_{2,1}C & b_{2,2}C \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} C(b_{1,1}x_1 + b_{1,2}x_2) \\ C(b_{2,1}x_1 + b_{2,2}x_2) \end{pmatrix} = \text{vec}(Y)$$

Assume that B and C are square matrices of dimension n . In case we ignore the structure of $B \otimes C \in \mathbb{R}^{n^2 \times n^2}$, solving the n^2 -dimensional linear system $\text{vec}(Y) = (B \otimes C)\text{vec}(X)$ is tantamount to $\mathcal{O}(n^6)$ arithmetic operations. Contrary to that, by computing the LU decompositions of B and C independently to obtain a LU decomposition of $B \otimes C$ (cf. [Corollary 3.5](#)), this system can be solved by $\mathcal{O}(n^3)$ arithmetic operations only. //

Corollary 3.6. *The mixed property allows for the following two natural generalizations.*

(i). *Let A_i, B_i be matrices such that the products $A_i \cdot B_i$ are defined, then*

$$\left(\bigotimes_{i=1}^n A_i \right) \cdot \left(\bigotimes_{i=1}^n B_i \right) = \bigotimes_{i=1}^n (A_i \cdot B_i)$$

(ii). *Let A_i, B_i be matrices such that the products $\prod_{i=1}^n A_i$ and $\prod_{i=1}^n B_i$ are defined, then*

$$\prod_{i=1}^n (A_i \otimes B_i) = \left(\prod_{i=1}^n A_i \right) \otimes \left(\prod_{i=1}^n B_i \right)$$

Proof. By induction and the mixed product property. □

Example 3.6. In order to illustrate the two generalizations of the mixed product property from [Corollary 3.6](#), consider matrices A_i, B_i with $1 \leq i \leq 3$. First, assume that all products $A_i B_i$ exist. Then, by the mixed product property and associativity of \otimes ,

$$\begin{aligned} (A_1 \otimes A_2 \otimes A_3)(B_1 \otimes B_2 \otimes B_3) &= ((A_1 \otimes A_2) \otimes A_3)((B_1 \otimes B_2) \otimes B_3) \\ &= ((A_1 \otimes A_2)(B_1 \otimes B_2)) \otimes (A_3 B_3) \\ &= (A_1 B_1) \otimes (A_2 B_2) \otimes (A_3 B_3) \end{aligned}$$

Similarly, by assuming that all products $\prod_{i=1}^k A_i$ and $\prod_{i=1}^k B_i$ exist, we see that

$$(A_1 \otimes B_1) \otimes (A_2 \otimes B_2) \otimes (A_3 \otimes B_3) = (A_1 A_2 \otimes B_1 B_2)(A_3 B_3) = (A_1 A_2 A_3) \otimes (B_1 B_2 B_3).$$

//

The definition of the Kronecker product clearly also covers vectors. Assume we are given matrices A_i and vectors x_i such that $A_i x_i$ is defined. Then we get by [Corollary 3.6](#) that $y = (\bigotimes_{i=1}^n A_i)(\bigotimes_{i=1}^n x_i) = \bigotimes_{i=1}^n (A_i x_i)$. Thus, the following theorem does not surprise.

Theorem 3.7. *Let A and B be square matrices and let (λ_i, v_i) and (μ_j, w_j) be the eigenpairs of A and B , respectively. Then the eigenpairs of $A \otimes B$ are given by $(\lambda_i \mu_j, v_i \otimes w_j)$ for all i and j . Analogous results also hold for the singular values and vectors of $A \otimes B$. Moreover, all results extend accordingly to products of the form $\bigotimes_{i=1}^n A_i$.*

Corollary 3.8. *Let $A_i \in \mathbb{R}^{n_i \times n_i}$ be square matrices and B_i arbitrary matrices, then*

$$(i). \operatorname{tr}(\bigotimes_{i=1}^k A_i) = \prod_{i=1}^k \operatorname{tr}(A_i)$$

$$(ii). \det(\bigotimes_{i=1}^k A_i) = \prod_{i=1}^k \det(A_i)^{n_i}$$

$$(iii). \operatorname{rank}(\bigotimes_{i=1}^k B_i) = \prod_{i=1}^k \operatorname{rank}(B_i)$$

3.2. Kronecker Products and Fast Algorithms

In general, computing a matrix vector product $y = Mx$ for a dense square matrix $M \in \mathbb{R}^{n \times n}$ requires $\Theta(n^2)$ floating point operations. However, for certain specially structured matrices it is possible to break this barrier and to provide, e. g., algorithms requiring only $\Theta(n \log n)$ floating point operations. Matrix vector products with a subquadratic complexity are often called *fast transforms*. For instance, Cooley and Tukey [[1965](#)] showed in their seminal paper that the computation of the discrete Fourier transformation of a signal of length $n = 2^k$ requires only $\Theta(n \log_2 n)$ instead of $\Theta(n^2)$ arithmetic operations. Today, we are aware of many fast transforms and almost all of them are tightly related to the Kronecker product [[Van Loan, 2000](#)].

Modern computing architecture became highly complex and diverse, since every single processor provides several levels of parallelism. While it gets more and more difficult to produce hand-written code which fully utilizes the theoretical potential of the computing hardware, automated program generators like FFTW [[Frigo and Johnson, 2005](#)] or SPIRAL [[Püschel et al., 2005](#)] proved to be successful in producing highly efficient code for computing fast transforms on given target platforms. A main ingredient in the automated search for efficient codes in systems like SPIRAL is the Kronecker product formalism. The appropriate representation of algorithms in terms of Kronecker products allows to discover many theoretically equivalent variants of an algorithm, in order to search for the most efficient implementation on a certain platform.

Traditional concepts for speeding up computations are *vectorization* and *threading*. By vectorization we mean that a single instruction is performed on a vector of operands instead of a single operand. Contrary, by threading we mean that several instructions are performed independently in parallel. Both of these concepts can be found in the Kronecker formalism as well as the following definitions show.

3. The Kronecker Product

Definition 3.3 (Parallel Kronecker Product). Let $A \in \mathbb{R}^{m \times n}$ be an arbitrary matrix. Then,

$$\mathcal{I}_k \otimes A = \text{diag}(\underbrace{A, A, \dots, A}_{k \text{ times}}),$$

is called the *parallel Kronecker product*. //

Definition 3.4 (Vector Kronecker Product). Let $A \in \mathbb{R}^{m \times n}$ be an arbitrary matrix. Then,

$$A \otimes \mathcal{I}_k = \begin{pmatrix} a_{1,1}\mathcal{I}_k & \cdots & a_{1,n}\mathcal{I}_k \\ \vdots & \ddots & \vdots \\ a_{m,1}\mathcal{I}_k & \cdots & a_{m,n}\mathcal{I}_k \end{pmatrix}$$

is called the *vector Kronecker product*. //

These names can be made clear by investigating the matrix vector product with the parallel and vector Kronecker product, respectively. In case of a parallel Kronecker Product $y = (\mathcal{I}_k \otimes A)x$, the input vector x is split up in k parts where each of those parts gets independently (“in parallel”), multiplied. In case of the vector Kronecker product, the computation of $y = (A \otimes \mathcal{I}_k)x$ is tantamount to computing the usual matrix vector product with A but with k -dimensional variables instead of scalars. Moreover, the parallel and the vector Kronecker product can also be interpreted as “hidden” matrix-matrix multiplications. Consider a matrix $A \in \mathbb{R}^{n \times n}$ with $n = rc$ and a vector $x \in \mathbb{R}^n$. Then we observe the equivalences

$$\begin{aligned} y &= (\mathcal{I}_c \otimes A)x \equiv y_{r \times c} = Ax_{r \times c} \\ y &= (A \otimes \mathcal{I}_r)x \equiv y_{r \times c} = x_{r \times c}A^T \end{aligned}$$

The following example illustrates these representations as matrix products more concretely.

Example 3.7. Let the matrix $A \in \mathbb{R}^{2 \times 2}$ and the vector $x \in \mathbb{R}^6$ be arbitrary. Then,

$$y = (\mathcal{I}_3 \otimes A)x = \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & A \end{pmatrix} \begin{pmatrix} x(1:2) \\ x(3:4) \\ x(5:6) \end{pmatrix} = \begin{pmatrix} Ax(1:2) \\ Ax(3:4) \\ Ax(5:6) \end{pmatrix},$$

and moreover, as a matrix matrix product we get

$$y_{2 \times 3} = (Ax(1:2), Ax(3:4), Ax(5:6)) = A(x(1:2), x(3:4), x(5:6)) = Ax_{2 \times 3}.$$

Analogously, we observe for

$$y = (A \otimes \mathcal{I}_3)x = \begin{pmatrix} a_{1,1}\mathcal{I}_3 & a_{1,2}\mathcal{I}_3 \\ a_{2,1}\mathcal{I}_3 & a_{2,2}\mathcal{I}_3 \end{pmatrix} \begin{pmatrix} x(1:3) \\ x(4:6) \end{pmatrix} = \begin{pmatrix} a_{1,1}x(1:3) + a_{1,2}x(4:6) \\ a_{2,1}x(1:3) + a_{2,2}x(4:6) \end{pmatrix}$$

the matrix matrix product

$$\begin{aligned} y_{3 \times 2} &= (a_{1,1}x(1:3) + a_{1,2}x(4:6), a_{2,1}x(1:3) + a_{2,2}x(4:6)) \\ &= (x(1:3), x(4:6)) \begin{pmatrix} a_{1,1} & a_{2,1} \\ a_{1,2} & a_{2,2} \end{pmatrix} = x_{3 \times 2}A^T \end{aligned}$$

//

In order to see the expresivness of the Kronecker product formalism we will investigate in the following a simply structured linear transform.

Definition 3.5. Consider the following recursively defined matrices H_{2^k} ,

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_{2^k} = \begin{pmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{pmatrix} = H_2 \otimes H_{2^{k-1}} = \bigotimes_{i=1}^k H_2,$$

The matrices H_{2^k} are called *Walsh matrices* and are a special form of *Hadamard matrices*. In general, a matrix $H_n \in \{1, -1\}^{n \times n}$ is said to be a Hadamard matrix if its columns are pairwise orthogonal. //

Example 3.8. The linear transform $y = H_{2^k}x$ based on the Walsh matrices is called *Fast Walsh-Hadamard Transform* (FWHT). The FWHT can be efficiently computed in subquadratic time, e. g., in the following two ways:

(i). Recursive computation of $H_{2^k}x$

$$H_{2^k} \cdot x = \begin{pmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \bar{x}_1 + \bar{x}_2 \\ \bar{x}_1 - \bar{x}_2 \end{pmatrix}, \quad \begin{aligned} \bar{x}_1 &= H_{2^{k-1}} \cdot x_1 \\ \bar{x}_2 &= H_{2^{k-1}} \cdot x_2 \end{aligned} \quad (3.2)$$

(ii). Iterative computation of $H_{2^k}x$

$$H_{2^k} \cdot x = \begin{pmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} H_{2^{k-1}}(x_1 + x_2) \\ H_{2^{k-1}}(x_1 - x_2) \end{pmatrix} \quad (3.3)$$

Note that here “recursive computation” and “iterative computation” are supposed to be interpreted from a programmers point of view, i. e., the algorithm defined in (3.2) allows for a natural recursive implementation whereas the algorithm defined by (3.3) allows for a natural iterative implementation using two nested loops.

As an immediate consequence of the Master theorem [Cormen *et al.*, 2001, p. 73], both of these computations involve only $\Theta(k2^k)$ arithmetic operations. //

While Example 3.8 outlines how the matrix vector product with a Kronecker product of matrices can be computed efficiently, it does not yet reveal the full potential of the Kronecker product formalism as a “language” for describing algorithms. Observe, that by the mixed product property the matrix H_{2^k} can also be factorized as

$$H_{2^k} = (H_2 \otimes \mathcal{I}_{2^{k-1}})(\mathcal{I}_2 \otimes H_{2^{k-1}}). \quad (3.4)$$

Thus, by the interpretation of the parallel and vector Kronecker product the computation of $(H_2 \otimes \mathcal{I}_{2^{k-1}})(\mathcal{I}_2 \otimes H_{2^{k-1}})x$ reads as follows. First, we compute the parallel Kronecker product $(\bar{x}_1, \bar{x}_2)^\top = (\mathcal{I}_2 \otimes H_{2^{k-1}})(x_1, x_2)^\top$, i. e., $\bar{x}_1 = H_{2^{k-1}}x_1$ and $\bar{x}_2 = H_{2^{k-1}}x_2$ are computed independently in parallel. Second, we compute the vector Kronecker product $(H_2 \otimes \mathcal{I}_{2^{k-1}})(\bar{x}_1, \bar{x}_2)^\top$, i. e., we compute the matrix vector product with H_2 with 2^{k-1} -dimensional vectors instead of scalars. Therefore, we observe that (3.4) is a “description” of the recursion (3.2) in the Kronecker product formalism. Moreover, by the generalized mixed product property, we obtain the factorization

$$H_{2^k} = \prod_{i=1}^k (\mathcal{I}_{2^{i-1}} \otimes H_2 \otimes \mathcal{I}_{2^{k-i}}). \quad (3.5)$$

3. The Kronecker Product

The validity of this factorization can be immediately verified by noting that

$$\mathcal{I}_{2^{i-1}} \otimes H_2 \otimes \mathcal{I}_{2^{k-i}} = \underbrace{\mathcal{I}_2 \otimes \mathcal{I}_2 \otimes \cdots \otimes \mathcal{I}_2}_{i-1 \text{ times}} \otimes H_2 \otimes \underbrace{\mathcal{I}_2 \otimes \mathcal{I}_2 \otimes \cdots \otimes \mathcal{I}_2}_{k-i \text{ times}}$$

As it is easily verified, (3.5) “describes” the iterative algorithm (3.3): The first part, $H_2 \otimes \mathcal{I}_{2^{k-i}}$ means that H_2 is applied to vectors of length 2^{k-i} and overall, this vector operation gets applied to 2^{i-1} vectors in parallel. In general, for $k = k_1 + k_2 + \cdots + k_p$ we obtain the factorization

$$H_{2^k} = \prod_{i=1}^p (\mathcal{I}_{2^{k_1+k_2+\dots+k_{i-1}}} \otimes H_{2^{k_i}} \otimes \mathcal{I}_{2^{k_{i+1}+\dots+k_p}}). \quad (3.6)$$

which is analogously to the previous factorization verified by representing the respective identity matrices as Kronecker products of smaller identity matrices. The factorization (3.6) clearly subsumes the previous two special cases. By considering $p = 2$, $k_1 = 1$ and $k_2 = k-1$ we get (3.4) and by $p = k$ and $k_i = 1$ we get (3.5). Thus, the factorization (3.6) leads to a wide variety of—in terms of the result—equivalent potential implementations of the matrix vector product $H_{2^k}x$. For a more comprehensive discussion on how to find the best performing of these implementations see Johnson and Püschel [2000].

For the sake of simplicity we considered the FWHT as an introductory example to the way how the Kronecker product of formalism is applied in the context of high performance computing. In more complicated fast transforms like the *Fast Fourier Transform* (FFT) also certain permutation matrices are needed in order to describe the different variants of a certain algorithm. The fruitful relationship between the Kronecker product formalism and fast linear transforms was not utilized for a long time, in particular Cooley and Tukey [1965] did not expose this framework. While early work [Drubin, 1971] outlined already the connection between the Kronecker product formalism and the FFT, many seemingly independent algorithms for computing the FFT were developed. Long time after the first FFT algorithms were known, e. g., Van Loan [1992] elaborated in detail how the Kronecker product formalism can be used to unify the manifold of known approaches towards computing the FFT.

3.3. Approximation with Kronecker Products

As we discussed previously in Example 3.5 we can efficiently solve systems of the shape $(B \otimes C)x = d$ by an effort of the order of solving B and C , respectively. Consider we are given a matrix A and know matrices B and C such that $A \approx B \otimes C$. Then $B \otimes C$ has a certain potential for being a good preconditioner for A , e. g., due to the moderate costs of solving $(B \otimes C)x = d$. To successfully utilize this potential we have to provide efficient routines for approximating given matrices in an appropriate norm.

3.3.1. The Nearest Kronecker Product Problem

Problem 3.1. Given an arbitrary matrix $A \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}$, find two matrices $B \in \mathbb{R}^{m_1 \times n_1}$ and $C \in \mathbb{R}^{m_2 \times n_2}$ such that $\|A - B \otimes C\|_F$ is minimal. //

Van Loan and Pitsianis [1993] gave a closed form solution for [Problem 3.1](#) in terms of the singular value decomposition of a reordering of A , which we are going to present below. The principal concept of the approach of Van Loan and Pitsianis [1993] is best understood by an example.

Example 3.9. Given $A \in \mathbb{R}^{4 \times 4}$, we want to find $B, C \in \mathbb{R}^{2 \times 2}$ such that $\|A - B \otimes C\|_F$ is minimal. Explicitly, this problem reads as

$$\left\| \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix} - \begin{pmatrix} b_{1,1} \cdot c_{1,1} & b_{1,1} \cdot c_{1,2} & b_{1,2} \cdot c_{1,1} & b_{1,2} \cdot c_{1,2} \\ b_{1,1} \cdot c_{2,1} & b_{1,1} \cdot c_{2,2} & b_{1,2} \cdot c_{2,1} & b_{1,2} \cdot c_{2,2} \\ b_{2,1} \cdot c_{1,1} & b_{2,1} \cdot c_{1,2} & b_{2,2} \cdot c_{1,1} & b_{2,2} \cdot c_{1,2} \\ b_{2,1} \cdot c_{2,1} & b_{2,1} \cdot c_{2,2} & b_{2,2} \cdot c_{2,1} & b_{2,2} \cdot c_{2,2} \end{pmatrix} \right\|_F \rightarrow \min.$$

In [Example 3.5](#) we saw the connection between a Kronecker product $B \otimes C$ and the dyadic product $\text{vec}(B)\text{vec}(C)^T$ obtained by the vectorization operator. By considering a certain reordering $\mathcal{R}(A)$ of A and $\text{vec}(B)\text{vec}(C)^T$ instead of $B \otimes C$, we obtain the following characterization of the minimization problem under investigation.

$$\begin{aligned} \|A - B \otimes C\|_F &= \|\mathcal{R}(A) - \text{vec}(B)\text{vec}(C)^T\|_F \\ &= \left\| \begin{pmatrix} a_{1,1} & a_{2,1} & a_{1,2} & a_{2,2} \\ a_{3,1} & a_{4,1} & a_{3,2} & a_{4,2} \\ a_{1,3} & a_{2,3} & a_{1,4} & a_{2,4} \\ a_{3,3} & a_{4,3} & a_{3,4} & a_{4,4} \end{pmatrix} - \begin{pmatrix} b_{1,1} \\ b_{2,1} \\ b_{1,2} \\ b_{2,2} \end{pmatrix} \begin{pmatrix} c_{1,1} & c_{2,1} & c_{1,2} & c_{2,2} \end{pmatrix} \right\|_F \end{aligned}$$

Thus, minimizing $\|A - B \otimes C\|_F$ is equivalent to finding the best rank-1 approximation of the reordering $\mathcal{R}(A)$ of A . Since computing low rank approximations of a given matrix is a well-known problem for which a closed form solution exists (see [Section 2.3.2](#)), also the problem of finding an optimal approximate $B \otimes C$ can be efficiently solved. //

In the following we always assume that we want to approximate an $m \times n$ matrix A with $m = m_1 m_2$ and $n = n_1 n_2$ by the Kronecker product of an $m_1 \times n_1$ matrix B with an $m_2 \times n_2$ matrix C . Moreover, we interpret A as block matrix consisting of $m_2 \times n_2$ blocks. Under these assumptions the rearrangement operator from [Example 3.9](#) is in general defined as follows.

Definition 3.6. Let $A \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}$ be an arbitrary matrix and let $A_{i,j}$ denote its (i, j) -th $m_2 \times n_2$ block. Then, we define the reordering $\mathcal{R}(A)$ of A by

$$\mathcal{R}(A) = \begin{pmatrix} A_1 \\ \vdots \\ A_{n_1} \end{pmatrix}, \quad A_i = \begin{pmatrix} \text{vec}(A_{1,i})^T \\ \vdots \\ \text{vec}(A_{m_1,i})^T \end{pmatrix}.$$

//

We want to minimize $\|A - B \otimes C\|_F$, which is defined on the level of individual entries. Due to the structure of this problem a “blocked” formulation of this target norm seems to be appropriate. Concretely, observe that by definition of the Kronecker product,

$$\|A - B \otimes C\|_F = \sum_{i=1}^{m_1} \sum_{j=1}^{n_1} \|A_{i,j} - b_{i,j} C\|_F.$$

3. The Kronecker Product

Moreover, note the following relationship between the vectorization operator and the norms $\|\cdot\|_F$ and $\|\cdot\|_2$: Let M_1 and M_2 be arbitrary matrices of the same dimension, then $\|M_1 - M_2\|_F^2 = \|\text{vec}(M_1) - \text{vec}(M_2)\|_2^2$. These simple observations are sufficient to prove the following identity.

Lemma 3.9. *For matrices $A \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}$, $B \in \mathbb{R}^{m_1 \times n_1}$ and $C \in \mathbb{R}^{m_2 \times n_2}$ it holds that*

$$\|A - B \otimes C\|_F = \|\mathcal{R}(A) - \text{vec}(B)\text{vec}(C)^T\|_F.$$

Proof. By the previous observation it follows that

$$\begin{aligned} \|A - B \otimes C\|_F^2 &= \sum_{i=1}^{m_1} \sum_{j=1}^{n_1} \|A_{i,j} - b_{i,j}C\|_F^2 = \sum_{j=1}^{n_1} \sum_{i=1}^{m_1} \|\text{vec}(A_{i,j}) - b_{i,j}\text{vec}(C)\|_F^2 \\ &= \sum_{j=1}^{n_1} \sum_{i=1}^{m_1} \|\text{vec}(A_{i,j})^T - b_{i,j}\text{vec}(C)^T\|_2^2 \\ &= \sum_{j=1}^{n_1} \|\text{vec}(A_j) - B(:,j)\text{vec}(C)^T\|_2^2 \\ &= \|\mathcal{R}(A) - \text{vec}(B)\text{vec}(C)^T\|_F^2 \end{aligned}$$

□

Recall that a matrix A can be expressed as sum of rank-1 matrices in terms of the SVD $U\Sigma V^T$ of A (cf. [Section 2.3.2](#)). In particular, $\sum_{i=1}^k \sigma_i U(:,i)V(i,:)^T$ is the best rank- k approximation of a matrix A with respect to $\|\cdot\|_2$ and $\|\cdot\|_F$. Thus, the nearest Kronecker product problem can be solved as follows.

Theorem 3.10. *Let the matrix $A \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}$ be arbitrary and $\mathcal{R}(A) = U\Sigma V^T$ the SVD of its reordering $\mathcal{R}(A)$. Then, the matrices B and C defined as $\text{vec}(B) = \sigma_1 U(:,1)$ and $\text{vec}(C) = V(:,1)$ minimize $\|A - B \otimes C\|_F$.*

The SVD does not only lead to the optimal solution of the nearest Kronecker product problem but also gives us information on the quality of the approximation. This connection can be easily seen by noting that for an $m \times n$ matrix A , it holds that

$$\|A\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_{\min(m,n)}^2}.$$

Thus, $\|A - B \otimes C\|_F^2 = \|A\|_F^2 - \sigma_1^2$, where σ_1 denotes the dominating singular value of $\mathcal{R}(A)$, i. e., the quality of the approximation relies on size of the singular values of $\mathcal{R}(A)$. Moreover, it is not needed to compute the full SVD of $\mathcal{R}(A)$, since specialized methods for computing the dominating singular vectors exist [[Van Loan and Pitsianis, 1993](#)].

3.3.2. Related Problems

The Kronecker product of two matrices preserves most of the structure inherent into the respective factors (cf. [Section 3.1](#)). Thus, it is not surprising that in case we want

to approximate a certainly structured matrix A by the product $B \otimes C$, we can choose the optimal B and C similarly structured. In particular, this is true for non-negative and symmetric positive definite matrices [Van Loan and Pitsianis, 1993].

A noteworthy byproduct of the presented SVD solution of the nearest Kronecker product problem is that we can efficiently solve the problem of finding the best approximating sum of k Kronecker products, i. e., we can find matrices B_i, C_i , such that

$$\left\| A - \sum_{i=1}^k (B_i \otimes C_i) \right\|_F \rightarrow \min.$$

In full analogy to the case $k = 1$, the optimal matrices B_i and C_i are determined by the respective i -th singular vectors of $\mathcal{R}(A)$.

The nearest Kronecker product problem has also several natural generalizations. For instance, in the spirit of sparse approximate inverse preconditioners (cf. Section 2.2.2), the following problem is of interest.

Problem 3.2. Given an arbitrary matrix $A \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$, find two matrices $B \in \mathbb{R}^{n_1 \times n_1}$ and $C \in \mathbb{R}^{n_2 \times n_2}$ such that $\|A(B \otimes C) - \mathcal{I}_{n_1 n_2}\|_F$ is minimal. //

In contrast to Problem 3.1, Problem 3.2 does not have a closed form solution. Thus, it can only be approached as structured least squares problem. Moreover, both Problem 3.1 and Problem 3.2 have natural multi-factor analogues where an approximation of the form $\bigotimes_{i=1}^p B_i$ is sought. Neither of these problems can be solved efficiently [Van Loan, 2000].

4. Matrices & Hamming Distance

Definition 4.1. Let A_1, A_2, \dots, A_n be non-empty sets and $x, y \in A_1 \times A_2 \times \dots \times A_n$, i. e., $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ with $x_i, y_i \in A_i$. Then, we define by

$$d_H(x, y) = \sum_{i=1}^n \delta(x_i, y_i), \quad \delta(u, v) = \begin{cases} 1 & u = v \\ 0 & u \neq v \end{cases}$$

the *Hamming distance* on the set $A_1 \times A_2 \times \dots \times A_n$. //

Note that the Hamming distance $d_H(\cdot, \cdot)$ is a metric on $A_1 \times A_2 \times \dots \times A_n$ [cf. Hamming, 1950, § 5]. This follows directly from the definition since the Hamming distance is an extension of the *discrete metric* ($\delta(\cdot, \cdot)$ in Definition 4.1) to multi-dimensional sets.

Suppose we are given finite sets A_1, A_2, \dots, A_n . If H is an arbitrary matrix of dimension $|A_1| \cdot |A_2| \cdot \dots \cdot |A_n|$, we can interpret the row and column indices of H as elements of $A_1 \times A_2 \times \dots \times A_n$, i. e., as finite “strings”. In this section we investigate structured matrices H with the characteristic property that each entry $h_{i,j}$ of H solely depends on the Hamming distance between its indices (appropriately interpreted as strings). We will give a complete characterization of this kind of matrices, analyze their algebraic properties and derive efficient algorithms for operating with them. Moreover, we will show how to solve several approximation problems with connections to preconditioning within this class of matrices.

4.1. Prerequisites

In this section we develop all necessary definitions and discuss the problems we investigate later on. First of all, we formally introduce the aforementioned class of structured matrices we investigate throughout this chapter.

Definition 4.2. Let $\Omega = A_1 \times A_2 \times \dots \times A_n$ be an arbitrary signature with finite sets A_1, \dots, A_n . Then we call each element $s \in \Omega$ a *string*. Moreover, we call $\dim(\Omega) = n$ its *dimension* and $|\Omega| = |A_1| \cdot |A_2| \cdot \dots \cdot |A_n|$ is the number of strings over Ω . //

If A is an arbitrary finite set with $|A|$ elements, we can always identify A with the set $\mathbb{Z}_{|A|} = \{0, 1, \dots, |A| - 1\}$ by a natural bijection between A and $\mathbb{Z}_{|A|}$. Therefore, we will henceforth consider without loss of generality only signatures of the form $\Omega = \mathbb{Z}_{b_1} \times \mathbb{Z}_{b_2} \times \dots \times \mathbb{Z}_{b_n}$ with $b_i \in \mathbb{N}$. Thus, we can interpret each string over Ω also as a number in a mixed radix numeral system where the numerical base at the i -th position is given by b_i . It is well-known how to convert numbers from an arbitrary mixed radix numeral system to integers. Concretely, for $\Omega = \mathbb{Z}_{b_1} \times \mathbb{Z}_{b_2} \times \dots \times \mathbb{Z}_{b_n}$ this is done by

4. Matrices & Hamming Distance

the mapping.

$$\iota_\Omega: \Omega \rightarrow \mathbb{Z}_{|\Omega|}, (z_1, z_2, \dots, z_n) \mapsto z_n + \sum_{i=1}^{n-1} \left(z_{n-i} \cdot \prod_{j=n-i+1}^n b_j \right)$$

Let A be a matrix of dimension n and let $n = b_1 \cdot b_2 \cdot \dots \cdot b_k$ be a factorization of n . Then, since ι_Ω is a bijection, we can interpret by $\iota^{-1}(i-1)$ each row and column index of A as a string over $\mathbb{Z}_{b_1} \times \mathbb{Z}_{b_2} \times \dots \times \mathbb{Z}_{b_n}$.

Example 4.1. Let $\Omega = \mathbb{Z}_2^n$, then we get for $\omega = (\omega_1, \omega_2, \dots, \omega_n) \in \Omega$ the mapping

$$\iota_\Omega(\omega) = \iota_\Omega((\omega_1, \omega_2, \dots, \omega_n)) = \sum_{i=0}^{n-1} \omega_{n-i} \cdot 2^i$$

In particular, ι_Ω is the usual conversion between binary numbers and integers. //

Accordingly, we henceforth use for a fixed signature Ω and indices $1 \leq i, j \leq |\Omega|$, the abbreviation $d_H(i, j) := d_H(\iota_\Omega^{-1}(i-1), \iota_\Omega^{-1}(j-1))$. Thus, we interpret $d_H(\cdot, \cdot)$ as a function $d_H: \{1, 2, \dots, |\Omega|\}^2 \rightarrow \mathbb{Z}_{\dim(\Omega)+1}$ instead of a function $d_H: \Omega \times \Omega \rightarrow \mathbb{Z}_{\dim(\Omega)+1}$.

Definition 4.3 (Hamming distance-based matrices). For an arbitrary finite signature Ω , we define the set of *Hamming distance-based (d_H -based) matrices* over Ω by

$$\mathcal{H}(\Omega) := \left\{ H \in \mathbb{R}^{|\Omega| \times |\Omega|} \mid \exists \varphi: \mathbb{Z}_{\dim(\Omega)+1} \rightarrow \mathbb{R} \text{ with } h_{i,j} = \varphi(d_H(i, j)) \right\}$$

Therefore, $\mathcal{H}(\Omega)$ contains all matrices whose (i, j) -th entry solely depends on the Hamming distance between i and j , i. e., it solely depends on the Hamming distance between the strings $\iota_\Omega^{-1}(i-1)$ and $\iota_\Omega^{-1}(j-1)$. For a matrix $H \in \mathcal{H}(\Omega)$ we call the function φ which determines the entries of H its *associated function*. Furthermore, we define $\mathcal{H}(\emptyset) := \mathbb{R}$. //

Example 4.2. Consider arbitrary matrices $H \in \mathcal{H}(\mathbb{Z}_2)$ and $G \in \mathcal{H}(\mathbb{Z}_2^2)$. Then we observe the following generic structure in the matrices H and G

$$H = \begin{pmatrix} a & b \\ b & a \end{pmatrix} \rightsquigarrow \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & a & b \\ 1 & b & a \end{array} \quad G = \begin{pmatrix} c & d & d & e \\ d & c & e & d \\ d & e & c & d \\ e & d & d & c \end{pmatrix} \rightsquigarrow \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 00 & c & d & d & e \\ 01 & d & c & e & d \\ 10 & d & e & c & d \\ 11 & e & d & d & c \end{array}$$

Moreover, we can observe how the ordering of the integers determines the rich structure of the matrices H and G . //

Definition 4.4 (Hamming distance-based vector). For an arbitrary finite signature Ω , we say that a vector $x \in \mathbb{R}^{|\Omega|}$ is d_H -based, if there exists a matrix $H \in \mathcal{H}(\Omega)$ such that x is the first column of H , i. e., $x = H(:, 1)$. In particular, it holds that $x_i = \varphi(d_H(i, 1))$ where φ denotes the associated function of H . //

We first derive an auxiliary result on the number of elements with a certain fixed Hamming distance in d_H -based vectors for the case of signatures $\Omega = \mathbb{Z}_c^n$.

Lemma 4.1. *Consider natural numbers $c, n \in \mathbb{N}$ with $c > 1$ and let $x \in \mathbb{R}^{|\Omega|}$ be a d_H -based vector over $\Omega = \mathbb{Z}_c^n$ with associated function φ_x . Then, for $0 \leq k \leq c$, the value $\varphi_x(k)$ appears $\binom{n}{k}(c-1)^k$ times in x . Moreover, this implies that*

$$\sum_{k=0}^n \binom{n}{k} (c-1)^k = c^n. \quad (4.1)$$

Proof. We give a combinatorial proof of

$$|\{i \mid 1 \leq i \leq c^n \wedge d_H(i, 1) = k\}| = \binom{n}{k} (c-1)^k,$$

where $d_H(\cdot, \cdot)$ has to be interpreted with respect to $\Omega = \mathbb{Z}_c^n$. Recall that the Hamming distance $d_H(\cdot, \cdot)$ measures the number of different elements between the strings $\iota^{-1}(i-1)$ and $\iota^{-1}(0)$. That means in case of $d_H(i, 1) = k$, exactly $\binom{n}{k}$ different patterns of equal elements (i. e., zeros) can occur. This follows from the interpretation of the binomial coefficient $\binom{n}{k}$ as the number of distinct binary strings of length n with k bits set to one. Since the Hamming distance only counts different elements, we additionally have to account for the different possible entries in the non-equal places. The fact that they can be filled arbitrary from the set $\{1, \dots, c-1\}^k$, leads to the additional factor of $|\{1, \dots, c-1\}^k| = (c-1)^k$. The validity of (4.1) follows immediately, since for every $1 \leq i \leq c^n$ there exists exactly one $0 \leq k \leq n$ such that $d_H(i, 1) = k$. \square

While Definition 4.3 and Example 4.2 indicate that all matrices $H \in \mathcal{H}(\Omega)$ are well-structured, it is not clear whether or not $\mathcal{H}(\Omega)$ itself carries some (algebraic) structure. Clearly, the usual matrix operations are well defined on $\mathcal{H}(\Omega)$, since all matrices in $\mathcal{H}(\Omega)$ are square. Moreover, $\mathcal{H}(\Omega)$ always contains the zero matrix and the identity matrix of the according dimension.

Problem 4.1. Let $\mathcal{H}(\Omega)$ be the set of all d_H -based matrices over the signature Ω . Does $\mathcal{H}(\Omega)$ carry an algebraic structure? //

Later on, we prove for fixed signatures $\Omega = \mathbb{Z}_c^n$, that $\mathcal{H}(\Omega)$ equipped with the usual matrix operations is a commutative ring with identity. Thus, $(\mathcal{H}(\Omega), +)$ is an abelian group, $(\mathcal{H}(\Omega), \cdot)$ is a commutative semigroup and the multiplication distributes over the addition. Contrary, for more general signatures, we will see that $\mathcal{H}(\Omega)$ is not closed under multiplication. At least the additive structure of $\mathcal{H}(\Omega)$ can be easily understood as the following lemma shows.

Lemma 4.2. *Let Ω be an arbitrary signature. Then for all $H, G \in \mathcal{H}(\Omega)$ it follows that $H + G = G + H$ and $H + G, -H \in \mathcal{H}(\Omega)$.*

Proof. Let φ_H and φ_G denote the associated functions of H and G , respectively. Then $\varphi_{H+G} := \varphi_H + \varphi_G$ is the associated function of $H + G$, i. e., $H + G \in \mathcal{H}(\Omega)$. Moreover, the commutativity of $+$ in $\mathcal{H}(\Omega)$ follows from the commutativity of $+$ in \mathbb{R} . Accordingly the associated function of $-H$ is given by $-\varphi_H$ and therefore we conclude also $-H \in \mathcal{H}(\Omega)$. \square

4. Matrices & Hamming Distance

From a computational point of view, structured matrices raise the question whether or not the complexity of certain operations can be (substantially) reduced by exploiting the available structure. By definition, each $H \in \mathcal{H}(\Omega)$ has only $\dim(\mathcal{H}(\Omega)) + 1 = \mathcal{O}(\log(|\Omega|))$ degrees of freedom. Therefore, the construction of efficient algorithms for these structured matrices seems to be feasible. On the other hand, the structure of the matrices in $\mathcal{H}(\Omega)$ does not allow for an immediate derivation of fast algorithms since the blocks of the matrices in $\mathcal{H}(\Omega)$ are, except of the principal structure, not directly related (cf. [Example 4.2](#)).

Problem 4.2. Let Ω be an arbitrary signature and $H, G \in \mathcal{H}(\Omega)$. Which effort does it take to compute $H + G$, $H \cdot G$, H^{-1} and Hx , respectively? //

Regarding the $\mathcal{O}(\log(|\Omega|))$ degrees of freedom in the matrices $H \in \mathcal{H}(\Omega)$, one would expect that matrix matrix operations in $\mathcal{H}(\Omega)$ are of a *polylogarithmic complexity* in $|\Omega|$, i. e., $\mathcal{O}(p(\log(|\Omega|)))$ where $p(\cdot)$ is a polynomial function. Accordingly, the complexity of the matrix vector multiplication Hx should be bounded by $\mathcal{O}(|\Omega| \log(|\Omega|))$. In the following, we construct algorithms which provably meet all the aforementioned complexity bounds.

For the sake of clarity we will first investigate the aforementioned problems for the special signature $\Omega = \mathbb{Z}_2^n$, i. e., the binary strings of length n . Concretely, we develop first all ideas and results for this special case and show subsequently how to extend the results to arbitrary finite signatures.

4.2. Binary Alphabets

In this section we consider for arbitrary $n \in \mathbb{N}$ signatures of the form $\Omega = \mathbb{Z}_2^n$. For a concise notation we define $\mathcal{B}_n := \mathcal{H}(\mathbb{Z}_2^n)$. In particular, we also define $\mathcal{B}_0 := \mathbb{R}$. First, we give general insights into the structure of the matrices in \mathcal{B}_n (cf. [Example 4.2](#)).

Lemma 4.3. *Let $n \in \mathbb{N}$, then $B \in \mathcal{B}_n$ if and only if the following two conditions hold.*

(i). $B \in \mathbb{R}^{2^n \times 2^n}$ is block-symmetric in the sense that

$$B = \begin{pmatrix} C & D \\ D & C \end{pmatrix}, \quad C, D \in \mathcal{B}_{n-1}$$

(ii). Let φ_B , φ_C and φ_D denote the function associated with B , C and D , respectively. Then, it holds for all $0 \leq d < n$ that

$$\varphi_C(d) = \varphi_B(d), \quad \varphi_D(d) = \varphi_B(d + 1) \tag{4.2}$$

Proof. First, assume that $B \in \mathcal{B}_n$. Since $B \in \mathcal{B}_n$ is a $2^n \times 2^n$ matrix, we can always

split B up into four blocks.

$$B = \begin{pmatrix} C & D' \\ D & C' \end{pmatrix} \rightsquigarrow \begin{array}{c|ccc|ccc} & 00\dots 0 & \cdots & 01\dots 1 & 10\dots 0 & \cdots & 11\dots 1 \\ \hline 00\dots 0 & c_{1,1} & \cdots & c_{1,2^{n-1}} & d'_{1,1} & \cdots & d'_{1,2^{n-1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 01\dots 1 & c_{2^{n-1},1} & \cdots & c_{2^{n-1},2^{n-1}} & d'_{2^{n-1},1} & \cdots & d'_{2^{n-1},2^{n-1}} \\ \hline 10\dots 0 & d_{1,1} & \cdots & d_{1,2^{n-1}} & c'_{1,1} & \cdots & c'_{1,2^{n-1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 11\dots 1 & d_{2^{n-1},1} & \cdots & d_{2^{n-1},2^{n-1}} & c'_{2^{n-1},1} & \cdots & c'_{2^{n-1},2^{n-1}} \end{array}$$

The equalities $C = C'$ and $D = D'$ as well as the relationship between C and D follow directly from the (decimal) ordering of the binary numbers: If $0 \leq i < 2^{n-1}$ then $\iota^{-1}(i) = (0, b_2, \dots, b_n)$ and $\iota^{-1}(i + 2^{n-1}) = (1, b_2, \dots, b_n)$. That means we get for $1 \leq i, j \leq 2^{n-1}$ that $d_H(i + 2^{n-1}, j) = d_H(i, j + 2^{n-1}) = d_H(i, j) + 1$ and $d_H(i + 2^{n-1}, j + 2^{n-1}) = d_H(i, j)$. Thus, we deduce by

$$\begin{aligned} c_{i,j} &= \varphi_C(d_H(i, j)) = \varphi_B(d_H(i, j)) \\ c'_{i,j} &= \varphi_{C'}(d_H(i, j)) = \varphi_B(d_H(i + 2^{n-1}, j + 2^{n-1})) = \varphi_B(d_H(i, j)) \\ d_{i,j} &= \varphi_D(d_H(i, j)) = \varphi_B(d_H(i + 2^{n-1}, j)) = \varphi_B(d_H(i, j) + 1) \\ d'_{i,j} &= \varphi_{D'}(d_H(i, j)) = \varphi_B(d_H(i, j + 2^{n-1})) = \varphi_B(d_H(i, j) + 1) \end{aligned}$$

all claims.

In case conditions (i) and (ii) hold, $B \in \mathcal{B}_n$ follows immediately. \square

In order to understand the algebraic structure of \mathcal{B}_n and to be able to derive efficient algorithms for operating with the matrices in \mathcal{B}_n we will investigate the structure of the eigenvalues and eigenvectors of these matrices. As the next section shows the spectral properties of the matrices in \mathcal{B}_n can be generically analyzed.

4.2.1. Algebraic Properties

Definition 4.5. Recall the Walsh matrices H_{2^n} from [Definition 3.5](#). We denote by $W_n := 2^{-n/2}H_{2^n}$ the *normalized Walsh matrices*. //

Note that the normalized Walsh matrices W_n are symmetric orthogonal matrices, which directly follows from the generalized mixed product property of the Kronecker product since

$$W_1 W_1^T = (2^{-1/2}H_2)(2^{-1/2}H_2) = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = \mathcal{I}_2$$

and $W_n = 2^{-n/2}H_{2^n} = \bigotimes_{i=1}^n (2^{-1/2}H_2)$.

Lemma 4.4. *Let $B \in \mathcal{B}_n$ be arbitrary. Then, the normalized Walsh matrices W_n diagonalize B , i. e., the matrix $\Lambda = W_n B W_n$ is diagonal.*

4. Matrices & Hamming Distance

Proof. We proceed by induction over n . For $n = 0$ the claim trivially holds. If $n > 0$, we can split B up into four equally-sized blocks which leads to a block-symmetric representation of B (cf. [Lemma 4.3](#)). Assume the theorem holds for $n - 1$, then

$$\begin{aligned}
\Lambda &= W_n B W_n \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} W_{n-1} & W_{n-1} \\ W_{n-1} & -W_{n-1} \end{pmatrix} \cdot \begin{pmatrix} C & D \\ D & C \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} W_{n-1} & W_{n-1} \\ W_{n-1} & -W_{n-1} \end{pmatrix} \\
&= \frac{1}{2} \begin{pmatrix} W_{n-1}(C+D) & W_{n-1}(D+C) \\ W_{n-1}(C-D) & W_{n-1}(D-C) \end{pmatrix} \cdot \begin{pmatrix} W_{n-1} & W_{n-1} \\ W_{n-1} & -W_{n-1} \end{pmatrix} \\
&= \begin{pmatrix} W_{n-1}(C+D)W_{n-1} & 0 \\ 0 & W_{n-1}(C-D)W_{n-1} \end{pmatrix} \tag{4.3}
\end{aligned}$$

By [Lemma 4.2](#) both $C + D$ and $C - D$ are elements of \mathcal{B}_{n-1} and therefore, they are by the induction hypothesis diagonalized by W_{n-1} . Hence, $\Lambda = W_n B W_n$ is also diagonal. In particular, we observe the relationship

$$\begin{aligned}
\Lambda &= \begin{pmatrix} W_{n-1}C W_{n-1} + W_{n-1}D W_{n-1} & 0 \\ 0 & W_{n-1}C W_{n-1} - W_{n-1}D W_{n-1} \end{pmatrix} \\
&= \begin{pmatrix} \Lambda_C + \Lambda_D & 0 \\ 0 & \Lambda_C - \Lambda_D \end{pmatrix} = \begin{pmatrix} \Lambda_{C+D} & 0 \\ 0 & \Lambda_{C-D} \end{pmatrix} \tag{4.4}
\end{aligned}$$

□

In other words, [Lemma 4.4](#) shows that all matrices $B \in \mathcal{B}_n$ have the same eigenvectors. Concretely, the eigenvectors are given by the columns of the normalized Walsh matrices W_n . The proof of [Lemma 4.4](#) also exposes how the eigenvalues of a matrix $B \in \mathcal{B}_n$ recursively evolve from the eigenvalues of its blocks. As the next example indicates, this recursive process also produces a well structured output.

Example 4.3. Let $A_3 \in \mathcal{B}_3$ be arbitrary, i. e., A_3 is induced by the function φ with $\varphi(0) = a$, $\varphi(1) = b$, $\varphi(2) = c$ and $\varphi(3) = d$. Moreover, let $A_2 \in \mathcal{B}_2$ be the left upper 4×4 block of A_3 and let $A_1 \in \mathcal{B}_1$ be the left upper 2×2 block of A_2 . Then we observe the following behavior for the eigenvalues

$$\text{diag}(W_1 A_1 W_1)^T = \text{diag}\left(W_1 \begin{pmatrix} a & b \\ b & a \end{pmatrix} W_1\right)^T = \begin{pmatrix} a+b \\ a-b \end{pmatrix}$$

By the recursive relationship of the eigenvalues of A_2 (see (4.4)), we get

$$\text{diag}(W_2 A_2 W_2)^T = \text{diag}\left(W_2 \begin{pmatrix} a & b & b & c \\ b & a & c & b \\ b & c & a & b \\ c & b & b & a \end{pmatrix} W_2\right)^T = \begin{pmatrix} (a+b) + (b+c) \\ (a-b) + (b-c) \\ (a+b) - (b+c) \\ (a-b) - (b-c) \end{pmatrix} = \begin{pmatrix} a+2b+c \\ a-c \\ a-c \\ a-2b+c \end{pmatrix}$$

Utilizing the recursive relationship (4.4) again, leads to the eigenvalues of A_3

$$\text{diag}(W_3 A_3 W_3)^T = \begin{pmatrix} (a+2b+c) + (b+2c+d) \\ (a-c) + (b-d) \\ (a-c) + (b-d) \\ (a-2b+c) + (b-2c+d) \\ (a+2b+c) - (b+2c+d) \\ (a-c) - (b-d) \\ (a-c) - (b-d) \\ (a-2b+c) - (b-2c+d) \end{pmatrix} = \begin{pmatrix} a+3b+3c+d \\ a+b-c-d \\ a+b-c-d \\ a-b-c+d \\ a+b-c-d \\ a-b-c+d \\ a-b-c+d \\ a-3b+3c-d \end{pmatrix}$$

In this example we see that the diagonal matrix containing the eigenvalues of a d_H -based matrix is also well structured. In particular, we observe that its diagonal is a d_H -based vector. //

Lemma 4.5. *Let (x, y) be a pair of vectors $x, y \in \mathbb{R}^n$ which satisfy for all $1 \leq i < n$ the property $x_i + x_{i+1} = y_i - y_{i+1}$. Then it holds for the vectors $u = y + x$ and $v = y - x$ that $u(2:n) = v(1:(n-1))$.*

Proof. By definition of u and v , we get

$$\begin{aligned} u &= y + x = (y_1 + x_1, y_2 + x_2, \dots, y_n + x_n) \\ v &= y - x = (y_1 - x_1, \dots, y_{n-1} - x_{n-1}, y_n - x_n) \end{aligned}$$

Thus, the claim follows, since $y_{i+1} + x_{i+1} = y_i - x_i$ holds by assumption. \square

Lemma 4.6. *Consider the matrices $A^{(n)} \in \mathbb{R}^{(n+1) \times (n+1)}$ which are recursively defined by*

$$A^{(0)} := 1, \quad A^{(n)} := \begin{pmatrix} A^{(n-1)} & 0 \\ A^{(n-1)}(n, :) & 0 \end{pmatrix} + \begin{pmatrix} 0 & A^{(n-1)} \\ 0 & -A^{(n-1)}(n, :) \end{pmatrix}. \quad (4.5)$$

Then $A^{(n)}$ can also be decomposed in the form

$$A^{(n)} = \begin{pmatrix} A^{(n-1)}(1, :) & 0 \\ A^{(n-1)} & 0 \end{pmatrix} + \begin{pmatrix} 0 & A^{(n-1)}(1, :) \\ 0 & -A^{(n-1)} \end{pmatrix}. \quad (4.6)$$

Example 4.4. For $0 \leq n \leq 3$, consider the matrices $A^{(n)}$ defined in Lemma 4.6.

$$A^{(0)} = 1$$

$$A^{(1)} \stackrel{(4.5), (4.6)}{=} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$A^{(2)} \stackrel{(4.5)}{=} \begin{pmatrix} 1 & 1+1 & 1 \\ 1 & -1+1 & -1 \\ 1 & -1-1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \\ 1 & -2 & 1 \end{pmatrix} \stackrel{(4.6)}{=} \begin{pmatrix} 1 & 1+1 & 1 \\ 1 & 1-1 & -1 \\ 1 & -1-1 & 1 \end{pmatrix}$$

$$A^{(3)} \stackrel{(4.5)}{=} \begin{pmatrix} 1 & 2+1 & 1+2 & 1 \\ 1 & 0+1 & -1+0 & -1 \\ 1 & -2+1 & 1-2 & 1 \\ 1 & -2-1 & 1+2 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 3 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -3 & 3 & -1 \end{pmatrix} \stackrel{(4.6)}{=} \begin{pmatrix} 1 & 2+1 & 1+2 & 1 \\ 1 & 2-1 & 1-2 & -1 \\ 1 & 0-1 & -1-0 & 1 \\ 1 & -2-1 & 1+2 & -1 \end{pmatrix}$$

Note, the first (last) row of $A^{(n)}$ contains the (alternating) binomial coefficients $\binom{n}{i}$. //

4. Matrices & Hamming Distance

Proof of Lemma 4.6. Explicitly, the assertion of the lemma states that for all pairs of neighboring columns of $A^{(n)}$, i. e., all pairs $\{A^{(n)}(:, j), A^{(n)}(:, j+1)\}$ with $2 \leq j < n-1$, it holds that $A^{(n)}(2:n, j) + A^{(n)}(2:n, j+1) = A^{(n)}(1:n-1, j) - A^{(n)}(1:n-1, j+1)$ (cf. Example 4.4). Thus, in order to prove this lemma we have to inductively show that all neighboring columns of $A^{(n)}$ satisfy the assumptions of Lemma 4.5. Obviously (see Example 4.4), the two columns of $A^{(1)}$ satisfy the assumptions of Lemma 4.5. Therefore, for the inductive step, assume that all neighboring columns of $A^{(n-1)}$ satisfy the assumptions of Lemma 4.5. For short, we consider all pairs $\{x, y\}$ with $x = A^{(n)}(:, j)$, $y = A^{(n)}(:, j+1)$ and $1 \leq j \leq n$. We have to prove that $x_i + x_{i+1} = y_i - y_{i+1}$ for all $1 \leq i \leq n$. First, we compute for $1 \leq i < n$ by using the conventions $a_{i,0}^{(n-1)} = a_{i,n+1}^{(n-1)} = 0$ that

$$\begin{aligned} x_i + x_{i+1} &= (a_{i,j}^{(n-1)} + a_{i,j-1}^{(n-1)}) + (a_{i+1,j}^{(n-1)} + a_{i+1,j-1}^{(n-1)}) \\ &= (a_{i,j}^{(n-1)} + a_{i+1,j}^{(n-1)}) + (a_{i,j-1}^{(n-1)} + a_{i+1,j-1}^{(n-1)}) \\ &\stackrel{\text{i.h.}}{=} (a_{i,j+1}^{(n-1)} - a_{i+1,j+1}^{(n-1)}) + (a_{i,j}^{(n-1)} - a_{i+1,j}^{(n-1)}) \\ &= (a_{i,j+1}^{(n-1)} + a_{i,j}^{(n-1)}) - (a_{i+1,j+1}^{(n-1)} + a_{i+1,j}^{(n-1)}) = y_i - y_{i+1} \end{aligned}$$

Similarly, we compute for the case $i = n$ that

$$\begin{aligned} x_n + x_{n+1} &= (a_{n,j}^{(n-1)} + a_{n,j-1}^{(n-1)}) + (a_{n,j}^{(n-1)} - a_{n,j-1}^{(n-1)}) \\ &= a_{n,j}^{(n-1)} + a_{n,j}^{(n-1)} \\ &= (a_{n,j+1}^{(n-1)} + a_{n,j}^{(n-1)}) - (a_{n,j+1}^{(n-1)} - a_{n,j}^{(n-1)}) = y_n - y_{n+1} \end{aligned}$$

Therefore, all pairs of neighboring columns of $A^{(n)}$ satisfy the assumptions of Lemma 4.5, which completes the proof. \square

Theorem 4.7. *Let $B \in \mathcal{B}_n$, $n \geq 0$, be arbitrary and φ_B its associated function. Then B has at most $n+1$ different eigenvalues $\lambda_0, \lambda_1, \dots, \lambda_n$ and the multiplicity of λ_i is $\binom{n}{i}$. Moreover, the eigenvalues λ_i are given by*

$$(\lambda_0, \lambda_1, \dots, \lambda_n)^\top = A^{(n)}(\varphi_B(0), \varphi_B(1), \dots, \varphi_B(n))^\top \quad (4.7)$$

with the matrices $A^{(n)} \in \mathbb{R}^{(n+1) \times (n+1)}$ defined in (4.5).

Proof. We prove the validity of (4.7) inductively. The base case $n = 0$ is trivial. Assume that the claim holds for n and let $B \in \mathcal{B}_{n+1}$. Recall how the matrix of eigenvalues $\Lambda_B = W_{n+1} B W_{n+1}$ is structured (cf. (4.3) and (4.4)). By Lemma 4.3, we know that $\Lambda_C = W_n C W_n$ and $\Lambda_D = W_n D W_n$ are related as given in (4.2) and thus, together with the induction hypothesis we get for $\lambda_{d_H(i,1)}^C = [\Lambda_C]_{i,i}$ and $\lambda_{d_H(i,1)}^D = [\Lambda_D]_{i,i}$ the formula

$$\begin{aligned} (\lambda_0^C, \dots, \lambda_n^C)^\top &= A^{(n)}(\varphi_B(0), \dots, \varphi_B(n))^\top \\ (\lambda_0^D, \dots, \lambda_n^D)^\top &= A^{(n)}(\varphi_B(1), \dots, \varphi_B(n+1))^\top \end{aligned}$$

In order to prove the validity of (4.7) for $n+1$ we have to show for $\lambda^+ = \text{diag}(\Lambda_{C+D})$ and $\lambda^- = \text{diag}(\Lambda_{C-D})$ with

$$\begin{aligned} (\lambda_0^+, \lambda_1^+, \dots, \lambda_n^+) &= A^{(n)}(\varphi_B(0) + \varphi_B(1), \dots, \varphi_B(n) + \varphi_B(n+1))^\top \\ (\lambda_1^-, \dots, \lambda_n^-, \lambda_{n+1}^-) &= A^{(n)}(\varphi_B(0) - \varphi_B(1), \dots, \varphi_B(n) - \varphi_B(n+1))^\top \end{aligned}$$

that $\lambda_i^+ = \lambda_i^-$ for all $1 \leq i \leq n$. These equalities are an immediate consequence from the structure of $A^{(n+1)}$ (cf. Lemma 4.6): On the one hand it follows by (4.5) that

$$\begin{aligned} A^{(n)}(\varphi_B(0) + \varphi_B(1), \dots, \varphi_B(n) + \varphi_B(n+1))^T &= A^{(n)}(\varphi_B(0:n) + \varphi_B(1:n+1)) \\ &= A^{(n)}\varphi_B(0:n) + A^{(n)}\varphi_B(1:n+1) \\ &= A^{(n+1)}(1:n, :)\varphi_B(0:(n+1)). \end{aligned}$$

On the other hand, analogously, it follows by (4.6) that

$$A^{(n)}(\varphi_B(0) - \varphi_B(1), \dots, \varphi_B(n) - \varphi_B(n+1))^T = A^{(n+1)}(2:(n+1), :)\varphi_B(0:(n+1)).$$

Thus, the claimed equality holds since

$$(\lambda_1^+, \dots, \lambda_n^+) = (\lambda_1^-, \dots, \lambda_n^-) = A^{(n+1)}(2:n, :)\varphi_B(0:(n+1))$$

Therefore, we have shown that $\lambda_i^+ = \lambda_i^-$ for all $1 \leq i \leq n$, which implies for the eigenvalues $\lambda_{d_H(i,1)}^B = [\Lambda_B]_{i,i}$ of B that

$$\lambda^B = (\lambda_0^+, \lambda_1^\pm, \dots, \lambda_n^\pm, \lambda_{n+1}^-) = A^{(n+1)}(\varphi_B(0), \varphi_B(1), \dots, \varphi_B(n+1))^T.$$

This completes the inductive step as well as the proof of (4.7). Moreover, the claim on the multiplicity follows directly from Lemma 4.1 since $\text{diag}(\lambda_B)$ is a d_H -based vector. \square

Note that Theorem 4.7 only reveals how to compute the eigenvalues of a given matrix $B \in \mathcal{B}_n$ and how they are structured. On the other hand, it *does not* show that for arbitrary real numbers $\lambda_0, \lambda_1, \dots, \lambda_n$, we can find a matrix in \mathcal{B}_n with these eigenvalues. Therefore, we investigate if for an arbitrary d_H -based vector $\lambda \in \mathbb{R}^{2^n}$, it holds for $\Lambda = \text{diag}(\lambda)$ that $W_n \Lambda W_n \in \mathcal{B}_n$.

Example 4.5. Recall the factorization $2W_2 = H_4 = (H_2 \otimes \mathcal{I}_2)(\mathcal{I}_2 \otimes H_2)$ (cf. (3.5)) and let Λ be a diagonal matrix with $\Lambda_{i,i} = 2^{d_H(i,1)}$. We want to show that $B = W_2 \Lambda W_2^T$ is a d_H -based matrix, i. e., $B \in \mathcal{B}_2$. First we compute $M = (\mathcal{I}_2 \otimes H_2)\Lambda(\mathcal{I}_2 \otimes H_2)$

$$M = \begin{array}{c|c|c} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \\ \hline \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} & \begin{pmatrix} 1 & 2 & 0 & 0 \\ 1 & -2 & 0 & 0 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 2 & -4 \end{pmatrix} & \begin{pmatrix} 3 & -1 & 0 & 0 \\ -1 & 3 & 0 & 0 \\ 0 & 0 & 6 & -2 \\ 0 & 0 & -2 & 6 \end{pmatrix} \end{array}$$

As a second step we compute $4B = H_4 \Lambda H_4^T = (H_2 \otimes \mathcal{I}_2)M(H_2 \otimes \mathcal{I}_2)$

$$4B = \begin{array}{c|c|c} & \begin{pmatrix} 3 & -1 & 0 & 0 \\ -1 & 3 & 0 & 0 \\ 0 & 0 & 6 & -2 \\ 0 & 0 & -2 & 6 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \\ \hline \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} & \begin{pmatrix} 3 & -1 & 6 & -2 \\ -1 & 3 & -2 & 6 \\ 3 & -1 & -6 & 2 \\ -1 & 3 & 2 & -6 \end{pmatrix} & \begin{pmatrix} 9 & -3 & -3 & 1 \\ -3 & 9 & 1 & -3 \\ -3 & 1 & 9 & -3 \\ 1 & -3 & -3 & 9 \end{pmatrix} \end{array}$$

Thus, $B \in \mathcal{B}_2$ with $\varphi_B(0) = 9/4$, $\varphi_B(1) = -3/4$ and $\varphi_B(2) = 1/4$. //

4. Matrices & Hamming Distance

Theorem 4.8. Let $\lambda_0, \lambda_1, \dots, \lambda_n \in \mathbb{R}$ be arbitrary reals and $\Lambda \in \mathbb{R}^{2^n \times 2^n}$ a diagonal matrix with $\Lambda_{i,i} = \lambda_{d_H(i,1)}$, i. e., $\text{diag}(\Lambda)$ is a d_H -based vector. Then, $B = W_n \Lambda W_n \in \mathcal{B}_n$ and the values of the associated function φ_B of B are given by

$$(\varphi_B(0), \varphi_B(1), \dots, \varphi_B(n))^T = \tilde{A}^{(n)}(\lambda_0, \lambda_1, \dots, \lambda_n)^T,$$

with $\tilde{A}^{(n)} := 2^{-n} A^{(n)}$ and $A^{(n)}$ defined in (4.5).

Proof. First we define a truncated version of the factorization (3.5) of W_n .

$$W_n^j := 2^{-j/2} \prod_{i=n-j+1}^n (\mathcal{I}_{2^{i-1}} \otimes H_2 \otimes \mathcal{I}_{2^{n-i}}) = \mathcal{I}_{2^{n-j}} \otimes H_{2^j} \quad (4.8)$$

Note that $W_n^n = W_n$ and additionally we define $W_n^0 := \mathcal{I}_{2^n}$. As Kronecker product of symmetric matrices W_n^j is symmetric. Nevertheless, in light of (4.8), we distinguish in this proof between W_n^j and $(W_n^j)^T$, since the transposition reverses the order in the product.

In the following, we prove by induction over j that for all $0 \leq j \leq n$ it holds that $W_n^j \Lambda (W_n^j)^T$ is a $2^n \times 2^n$ block diagonal matrix with $2^j \times 2^j$ blocks $B_i^j \in \mathcal{B}_j$, $1 \leq i \leq 2^{n-j}$ on its diagonal, i. e.,

$$W_n^j \Lambda (W_n^j)^T = \begin{pmatrix} B_1^j & & & \\ & B_2^j & & \\ & & \ddots & \\ & & & B_{2^{n-j}}^j \end{pmatrix}, \quad B_i^j \in \mathcal{B}_j \quad (4.9)$$

Moreover, with $d = d_H(i, 1)$, the function associated with B_i^j is given by

$$\left(\varphi_{B_i^j}(0), \varphi_{B_i^j}(1), \dots, \varphi_{B_i^j}(j) \right)^T = \tilde{A}^{(j)}(\lambda_d, \lambda_{d+1}, \dots, \lambda_{d+j})^T. \quad (4.10)$$

For $j = 0$, the validity of (4.9) and (4.10) follows immediately by the assumptions on Λ . For the induction step assume that the claim holds for $j < n$. Then we get for $j + 1$

$$W_n^{j+1} \Lambda (W_n^{j+1})^T = \frac{1}{2} (\mathcal{I}_{2^{n-j}} \otimes H_2 \otimes \mathcal{I}_{2^j}) W_n^j \Lambda (W_n^j)^T (\mathcal{I}_{2^{n-j}} \otimes H_2 \otimes \mathcal{I}_{2^j})$$

By definition of the Kronecker product, $\mathcal{I}_{2^{n-j}} \otimes H_2 \otimes \mathcal{I}_{2^j}$ is a block diagonal matrix,

$$\mathcal{I}_{2^{n-j}} \otimes H_2 \otimes \mathcal{I}_{2^j} = \begin{pmatrix} \mathcal{I}_{2^j} & \mathcal{I}_{2^j} & & \\ \mathcal{I}_{2^j} & -\mathcal{I}_{2^j} & & \\ & & \ddots & \\ & & & \mathcal{I}_{2^j} & \mathcal{I}_{2^j} \\ & & & \mathcal{I}_{2^j} & -\mathcal{I}_{2^j} \end{pmatrix}$$

Therefore, block-wise multiplication leads to

$$W_n^{j+1} \Lambda (W_n^{j+1})^T = \begin{pmatrix} B_1^{j+1} & & & \\ & \ddots & & \\ & & & B_{2^{n-j-1}}^{j+1} \end{pmatrix}, \quad B_i^{j+1} = \frac{1}{2} \begin{pmatrix} B_{2^{i-1}}^j + B_{2^i}^j & B_{2^{i-1}}^j - B_{2^i}^j \\ B_{2^{i-1}}^j - B_{2^i}^j & B_{2^{i-1}}^j + B_{2^i}^j \end{pmatrix}$$

To complete the inductive step we have to show that B_i^{j+1} fulfills (4.10). First observe that $d_H(2i-1, 1) + 1 = d_H(2i, 1)$, since $2i-2$ is even and $2i-1$ is odd, i. e.,

$$\iota^{-1}(2i-2) = (b_1, \dots, b_j, 0), \quad \iota^{-1}(2i-1) = (b_1, \dots, b_j, 1)$$

Therefore, by the induction hypothesis, and $d = d_H(i, 1)$ we get the following overlapping conditions for the associated function $\varphi_{B_i^{j+1}}$ of B_i^{j+1} .

$$\begin{aligned} (\varphi_{B_i^{j+1}}(0), \varphi_{B_i^{j+1}}(1), \dots, \varphi_{B_i^{j+1}}(j)) &= \frac{1}{2} \tilde{A}^{(j)}(\lambda_d + \lambda_{d+1}, \dots, \lambda_{d+j} + \lambda_{d+j+1})^T \\ (\varphi_{B_i^{j+1}}(1), \dots, \varphi_{B_i^{j+1}}(j), \varphi_{B_i^{j+1}}(j+1)) &= \frac{1}{2} \tilde{A}^{(j)}(\lambda_d - \lambda_{d+1}, \dots, \lambda_{d+j} - \lambda_{d+j+1})^T \end{aligned}$$

By Lemma 4.6 all overlapping elements are indeed equal which leads to

$$(\varphi_{B_i^{j+1}}(0), \dots, \varphi_{B_i^{j+1}}(n+1)) = \tilde{A}^{(j+1)}(\lambda_d, \dots, \lambda_{d+j+1})^T$$

This completes the inductive step as well as the entire proof. \square

The two previous theorems show that for every matrix $B \in \mathcal{B}_n$ its eigenvalues and its associated function are linearly related through the matrices $A^{(n)}$. In particular, Theorem 4.7 and Theorem 4.8 together imply that $\tilde{A}^{(n)}A^{(n)} = 2^{-n}A^{(n)}A^{(n)} = \mathcal{I}_{n+1}$.

Example 4.6. Recall the matrix B from Example 4.5 with eigenvalues $\lambda = (1, 2, 4)$ and associated function $\phi = \varphi_B((0, 1, 2)) = (9/4, -3/4, 1/4)$. We explicitly check that $A^{(2)}$ and $\tilde{A}^{(2)}$ (cf. Example 4.4) relate those two vectors as shown in Theorem 4.7 and Theorem 4.8.

$$\begin{aligned} \lambda &= A^{(2)}\phi = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \\ 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} 9/4 \\ -3/4 \\ 1/4 \end{pmatrix} = \begin{pmatrix} 9/4 - 6/4 + 1/4 \\ 9/4 - 1/4 \\ 9/4 + 6/4 + 1/4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \\ \phi &= \tilde{A}^{(2)}\lambda = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \\ 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 + 4 + 4 \\ 1 - 4 \\ 1 - 4 + 4 \end{pmatrix} = \begin{pmatrix} 9/4 \\ -3/4 \\ 1/4 \end{pmatrix} \end{aligned}$$

//

Corollary 4.9. *The set \mathcal{B}_n is closed under multiplication and the multiplication in \mathcal{B}_n is commutative.*

Proof. By Theorem 4.7, we can find for all matrices $B, C \in \mathcal{B}_n$ diagonal matrices Λ_B, Λ_C whose diagonal is a d_H -based vector such that $B = W_n \Lambda_B W_n$ and $C = W_n \Lambda_C W_n$. Hence, the commutativity follows directly by the orthogonality of W_n .

$$B \cdot C = W_n \Lambda_B W_n W_n \Lambda_C W_n = W_n \Lambda_B \Lambda_C W_n = W_n \Lambda_C \Lambda_B W_n = C \cdot B$$

Moreover, since the diagonal of $\Lambda_B \Lambda_C$ is also a d_H -based vector, we know by Theorem 4.8 that $B \cdot C \in \mathcal{B}_n$. \square

Theorem 4.10. *With the usual matrix addition and multiplication $(\mathcal{B}_n, +, \cdot)$ is a commutative ring with identity. In particular, the zero matrix is the additive identity and \mathcal{I}_{2^n} is the multiplicative identity of $(\mathcal{B}_n, +, \cdot)$.*

4. Matrices & Hamming Distance

Proof. The zero matrix and \mathcal{I}_{2^n} are certainly elements of \mathcal{B}_n with associated functions $\varphi_0 \equiv 0$ and $\varphi_{\mathcal{I}_{2^n}}(0) = 1, \varphi_{\mathcal{I}_{2^n}}(d) = 0$ for all $1 \leq d \leq n$. Moreover, by [Lemma 4.2](#) we know that $(\mathcal{B}_n, +)$ is an abelian group and by [Corollary 4.9](#) we know that (\mathcal{B}_n, \cdot) is a commutative semigroup. That the multiplication distributes over the addition can be straightforwardly seen, since for arbitrary $B, C, D \in \mathcal{B}_n$ we conclude

$$\begin{aligned} B(C + D) &= W_n \Lambda_B W_n (W_n \Lambda_C W_n + W_n \Lambda_D W_n) \\ &= W_n \Lambda_B W_n (W_n (\Lambda_C + \Lambda_D) W_n) \\ &= W_n (\Lambda_B (\Lambda_C + \Lambda_D)) W_n \\ &= W_n (\Lambda_B \Lambda_C + \Lambda_B \Lambda_D) W_n \\ &= W_n (\Lambda_B \Lambda_C) W_n + W_n (\Lambda_B \Lambda_D) W_n = BC + BD \end{aligned}$$

Therefore, $(\mathcal{B}_n, +, \cdot)$ is a commutative ring with identity. \square

4.2.2. Fast Algorithms

We saw in [Lemma 4.4](#) that the matrices $H \in \mathcal{B}_n$ can be diagonalized in the form $\Lambda = W_n H W_n$, where the diagonal of Λ was shown to be a d_H -based vector in [Theorem 4.7](#). Thus, from a computational point of view, a matrix $H \in \mathcal{B}_n$ is most efficiently represented by a $(n + 1)$ -dimensional vector containing all its distinct eigenvalues. This representation can be obtained in $\Theta(n^2)$ time via [Theorem 4.7](#) and straightforwardly leads to efficient operations within \mathcal{B}_n .

Corollary 4.11. *Let $H, G \in \mathcal{B}_n$ be given and let $\lambda_H, \lambda_G \in \mathbb{R}^{n+1}$ denote the vectors containing the eigenvalues of H and G , respectively. Then,*

- (i). $\lambda_{H+G} = \lambda_H + \lambda_G$ contains the eigenvalues of $H + G$,
- (ii). $\lambda_{H \cdot G} = (\lambda_H(0)\lambda_G(0), \dots, \lambda_H(n)\lambda_G(n))$ contains the eigenvalues of $H \cdot G$,
- (iii). if H is regular, $\lambda_{H^{-1}} = (\lambda_H(0)^{-1}, \dots, \lambda_H(n)^{-1})$ contains the eigenvalues of H^{-1} .

Therefore, the operations $H + G$, $H \cdot G$ and H^{-1} can be computed in $\Theta(n)$ time.

[Corollary 4.11](#) does not surprise since all the involved objects are similarly structured. In the light of (matrix-free) iterative methods a particularly interesting question is whether or not the matrix vector product Hv with a matrix $H \in \mathcal{B}_n$ and an arbitrary vector $v \in \mathbb{R}^{2^n}$ can be efficiently evaluated. By utilizing results from Kronecker product theory (cf. [Chapter 3](#)) we show that the matrix vector product Hv with $H \in \mathcal{B}_n$ can be computed in $\Theta(n2^n)$ time.

First, we investigate how to efficiently create d_H -based vectors out of their associated function. The principal problem in the algorithmic treatment of d_H -based vectors is that the direct evaluation of the Hamming distance $d_H(i, 1)$ is a costly operation since it requires to extract the binary representation of the natural number i . Thus, analogously to the theoretical investigations in [Section 4.2.1](#), we aim at avoiding this operation. As a prerequisite, we study the following connection between the Kronecker product and d_H -based matrices.

Lemma 4.12. *Let $a, b \in \mathbb{R}$ and $n \in \mathbb{N}$. Then for all matrices*

$$K_n = \bigotimes_{i=1}^n \begin{pmatrix} a & b \\ b & a \end{pmatrix}, \quad V_n = \bigotimes_{i=1}^n \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix},$$

it holds that $K_n \in \mathcal{B}_n$ with associated function $\varphi_{K_n}(d) = a^{n-d} \cdot b^d$. Moreover, $\text{diag}(V_n) = K_n(:, 1)$ is a d_H -based vector with associated function φ_{K_n} .

Proof. We proceed by induction on n . For $n = 1$, we immediately see that $K_1 \in \mathcal{B}_1$. Assume that $K_n \in \mathcal{B}_n$. The matrix K_{n+1} is by definition a (symmetric) $2^n \times 2^n$ block matrix with diagonal blocks $C = aK_n$ and off-diagonal blocks $D = bK_n$, where $C, D \in \mathcal{B}_n$ by the assumption that $K_n \in \mathcal{B}_n$. Thus, according to [Lemma 4.3](#), we have to show that for all $0 < d < n$, it holds that $\varphi_C(d+1) = \varphi_D(d)$. This can be seen by induction, since

$$\varphi_C(d+1) = a(a^{n-(d+1)} \cdot b^{(d+1)}) = a^{n-d} \cdot b^{(d+1)} = b(a^{n-d} \cdot b^d) = \varphi_D(d).$$

Moreover, due to $\varphi_C(d) = a^{(n+1)-d} \cdot b^d$ and $\varphi_D(n) = b^{n+1}$ we conclude that the associated function of K_{n+1} is given by $\varphi_{K_{n+1}}(d) = a^{(n+1)-d} \cdot b^d$, which completes the inductive step. The second claim trivially holds for $n = 1$. Moreover, if we assume that $\text{diag}(V_n) = K_n(:, 1)$, then we see due to

$$K_{n+1} = \begin{pmatrix} aK_n & bK_n \\ bK_n & aK_n \end{pmatrix}, \quad V_{n+1} = \begin{pmatrix} aV_n & 0 \\ 0 & bV_n \end{pmatrix}$$

that also $V_{n+1} = K_{n+1}(:, 1)$ holds, which completes the proof. \square

Example 4.7. With the definitions from [Lemma 4.12](#) and $a = 1, b = 2$ we see that

$$\text{diag}(V_1) = (1, 2)^T, \quad \text{diag}(V_2) = (1, 2, 2, 4)^T, \quad \text{diag}(V_3) = (1, 2, 2, 4, 2, 4, 4, 8)^T.$$

This follows directly from the Kronecker product representation of V_i , which leads per definition to $\text{diag}(V_{i+1}) = (1 \cdot \text{diag}(V_i), 2 \cdot \text{diag}(V_i))$. By applying element-wise the function $x \mapsto \varphi(\log_2(x))$ on the elements of $\text{diag}(V_i)$, we finally obtain a d_H -based vector with associated function φ , e. g.,

$$(1, 2)^T \mapsto (\varphi(0), \varphi(1))^T, \quad (1, 2, 2, 4)^T \mapsto (\varphi(0), \varphi(1), \varphi(1), \varphi(2))^T.$$

Therefore, we can see that the efficient treatment of d_H -based vectors relies on the efficient treatment of the diagonal matrices V_i . //

The method to create d_H -based vectors sketched in [Example 4.7](#) obviously does not require any direct evaluation of $d_H(\cdot, \cdot)$. Straightforward algorithms (i. e., algorithms based on computing V_i as indicated in [Lemma 4.12](#)) for creating a d_H -based vector $h \in \mathbb{R}^{2^n}$ require $\Theta(n2^n)$ arithmetic operations. On the other hand, the recursive construction in [Example 4.7](#) is of a particularly simple form, which can be utilized in order to create d_H -based vectors even in linear time with respect to the vector dimension $\dim(h) = 2^n$. We propose [Algorithm 4.1](#) as a linear time algorithm for creating d_H -based vectors. [Algorithm 4.1](#) proceeds in two steps: First an auxiliary vector h with $h(i) = d_H(i, 1)$ is

Algorithm 4.1 Linear time creation of a d_H -based vector.

Input: $n \in \mathbb{N}$ and $\varphi: \mathbb{Z}_{n+1} \rightarrow \mathbb{R}$ **Output:** $h \in \mathbb{R}^{2^n}$ with $h(i) = \varphi(d_H(i, 1))$.

```

1:  $h(1) \leftarrow 0, p \leftarrow 1$ 
2: for  $i \leftarrow 1$  to  $n$  do
3:    $h((p+1):2p) \leftarrow h(1:p) + \mathbf{1} \triangleright \mathbf{1}$  denotes the vector of all ones,  $\mathbf{1} = (1, 1, \dots, 1)$ 
4:    $p \leftarrow 2p$ 
5: end for
6:  $h(1:p) = \varphi(h(1:p))$ 

```

computed (lines 1–5). This is done analogously to [Example 4.7](#) by iteratively combining the vector computed so far ($h(1:p)$) by a copy of itself which is additionally shifted by the vector $\mathbf{1} = (1, 1, \dots, 1)$. In particular (cf. [Example 4.7](#)), this auxiliary vector evolves as follows.

$$(0) \rightarrow (0, 1) \rightarrow (0, 1, 1, 2) \rightarrow (0, 1, 1, 2, 1, 2, 2, 3) \rightarrow \dots$$

Afterwards, the function φ is straightforwardly applied (line 6). This simple algorithm is of the desired linear time complexity.

Theorem 4.13. *For given $n \in \mathbb{N}$ and a function $\varphi: \mathbb{Z}_{n+1} \rightarrow \mathbb{R}$, [Algorithm 4.1](#) creates a d_H -based vector $h \in \mathbb{R}^{2^n}$, with associated function φ , in $\Theta(2^n)$ time.*

Proof. The correctness of [Algorithm 4.1](#) follows immediately from the considerations above. For the complexity, first note that the application of φ (line 6) is trivially of complexity $\Theta(2^n)$. In the loop (lines 2–5) we see that in iteration i exactly 2^{i-1} values are read and written, respectively. Thus, in total, $\sum_{i=1}^n 2^{i-1} = 2^n - 1 = \Theta(2^n)$ values are read and written, which proves the claim. \square

The existence of an efficient routine for creating a d_H -based vector h from its associated function φ_h clearly also implies a fast routine for evaluating the matrix vector product $\text{diag}(h)v$ with an arbitrary vector v of the according dimension.

Corollary 4.14. *Let the associated function of a d_H -based vector $h \in \mathbb{R}^{2^n}$ be given and $v \in \mathbb{R}^{2^n}$ be an arbitrary vector. Then the matrix vector product $\text{diag}(h)v$ can be computed in $\Theta(2^n)$ time.*

Proof. Use [Algorithm 4.1](#) and apply $h(i) \mapsto \varphi(h(i)) \cdot v(i)$ instead of $h(i) \mapsto \varphi(h(i))$ in line 6 of the algorithm. \square

In [Example 3.8](#) we have discussed how to efficiently evaluate the matrix vector product with the Walsh matrices H_{2^n} . For the sake of completeness a concrete implementation of the iterative algorithm (cf. (3.3)) for computing the matrix vector product with the normalized Walsh matrices W_n (cf. [Definition 4.5](#)) is given in [Algorithm 4.2](#).

A more detailed discussion of the properties of (a slight generalization of) [Algorithm 4.2](#) and its parallelization was given by Niederbrucker and Gansterer [[2011a](#)]. We conclude this section by the desired result on the efficiency of the matrix vector multiplication Hv with $H \in \mathcal{B}_n$.

Algorithm 4.2 Efficient computation of the matrix vector product $W_n v$

Input: $v \in \mathbb{R}^{2^n}$
Output: $W_n v$

- 1: **for** $i \leftarrow 1$ **to** 2^{n-1} **by** $2i$ **do**
- 2: **for** $j \leftarrow 1$ **to** 2^n **by** $2i$ **do**
- 3: $t_1 \leftarrow v(j:j+i)/\sqrt{2}$
- 4: $t_2 \leftarrow v(j:j+2i)/\sqrt{2}$
- 5: $v(j:j+i) \leftarrow t_1 + t_2$
- 6: $v(j:j+2i) \leftarrow t_1 - t_2$
- 7: **end for**
- 8: **end for**

Corollary 4.15. *Let $v \in \mathbb{R}^{2^n}$ be an arbitrary vector and $H = W_n \Lambda W_n \in \mathcal{B}_n$. Then, the matrix vector product Hv can be computed in $\Theta(n2^n)$ time by computing $W_n(\Lambda(W_n v))$.*

Proof. In case the matrix of eigenvalues Λ is unknown, the eigenvalues of H can be computed as a first step in $\Theta(n^2)$ time according to [Theorem 4.7](#). Furthermore, the multiplication with Λ can be carried out in linear time according to [Corollary 4.14](#). Moreover, the matrix vector products with W_n can be efficiently computed with [Algorithm 4.2](#). \square

Summarizing, we have seen in this section how the structure inherent in the matrices $H \in \mathcal{B}_n$ can be utilized to derive fast algorithms. In particular, the complexity of the basic matrix operations in \mathcal{B}_n solely depends on the $(n+1)$ degrees of freedom and not on the actual matrix dimension 2^n .

4.2.3. Approximation

So far we have analyzed the structure of the matrices in \mathcal{B}_n in detail and shown how to exploit their rich structure in order to derive efficient algorithms. Based on these insights, we will show in this section how to solve several approximation problems within \mathcal{B}_n . First, we consider a basic matrix approximation problem.

Problem 4.3 (Matrix approximation). Given an arbitrary matrix $A \in \mathbb{R}^{2^n \times 2^n}$. Find the best approximating d_H -based matrix $H \in \mathcal{B}_n$ which minimizes $\|A - H\|_F$. //

By definition of the set \mathcal{B}_n (cf. [Definition 4.3](#)) the $n+1$ individual entries are independent of each other. Thus, one would expect that the solution of [Problem 4.3](#) can be computed in terms of $n+1$ (simple) independent problems. This is indeed the case, as the following theorem shows.

Theorem 4.16. *Given an arbitrary matrix $A \in \mathbb{R}^{2^n \times 2^n}$. Then the matrix H^* with associated function φ_{H^*} ,*

$$\varphi_{H^*}(k) = \sum_{i,j: d_H(i,j)=d} a_{i,j} \binom{n}{d}^{-1} 2^{-n}$$

minimizes $\|A - H\|_F$ for $H \in \mathcal{B}_n$.

4. Matrices & Hamming Distance

Proof. We aim at minimizing the function

$$\|A - H\|_F^2 = \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} (a_{i,j} - h_{i,j})^2.$$

Note that $\|A - H\|_F^2$ is in fact a function $\mathbb{R}^{n+1} \rightarrow \mathbb{R}$, since the matrices $H \in \mathcal{B}_n$ have only $n + 1$ degrees of freedom h_d , where $h_d = h_{i,j}$ for all i, j with $d_H(i, j) = d$. The critical points fulfill $\nabla \|A - H\|_F = 0$. Thus, we get for each degree of freedom $0 \leq d \leq n$ an equation

$$\frac{\partial \|A - H\|_F^2}{\partial h_d} = \sum_{i,j: d_H(i,j)=d} -2(a_{i,j} - h_{i,j}) = 0$$

Since these $n + 1$ equations are independent we get by using $h_d = h_{i,j}$ in the sum

$$h_d = \sum_{i,j: d_H(i,j)=d} a_{i,j} \binom{n}{d}^{-1} 2^{-n} \quad (4.11)$$

Therefore, the associated function of the critical point is given by $\varphi(d) = h_d$ where h_d can be computed by (4.11). Moreover, the computed critical point is indeed a minimum since the Hessian matrix of $\|A - H\|_F$ is a diagonal matrix with positive entries and thus, positive definite. \square

A particular interesting approximation problem is the computation of so-called *approximate inverses*, e. g., due to their potential for being a reasonable preconditioner [cf. Grote and Huckle, 1997].

Problem 4.4 (Approximate Inverse). Given an arbitrary matrix $A \in \mathbb{R}^{2^n \times 2^n}$. Find a d_H -based matrix $H \in \mathcal{B}_n$ which minimizes $\|AH - \mathcal{I}_{2^n}\|_F$. //

Grote and Huckle [1997] showed how to compute *sparse approximate inverses*, i. e., approximate inverses with a predefined sparsity pattern. This can be done efficiently by observing for matrices $A, M \in \mathbb{R}^{m \times m}$ the relationship

$$\|AM - \mathcal{I}_m\|_F^2 = \sum_{i=1}^m \|(AM - \mathcal{I}_m)e_i\|_2^2,$$

where e_i denotes the i -th canonical basis vector, i. e., e_i is zero except of the i -th entry, which is one. This characterization shows that minimizing $\|AM - \mathcal{I}_m\|_F$ can be done by columnwise minimizing $\|AM(:, i) - e_i\|_2$ for all i , independently. On the other hand, this fact reveals that this approach due to Grote and Huckle [1997] is only suited for finding certainly structured (sparse) matrices with independent entries. Thus, Problem 4.4 can not be solved in this way, since the entries of the matrices $H \in \mathcal{B}_n$ are not independent.

A particularly simple kind of sparse matrices are diagonal matrices. Thus, consider the problem of finding for a given matrix $A \in \mathbb{R}^{m \times m}$ a diagonal matrix $D \in \mathbb{R}^{m \times m}$ such that $\|AD - \mathcal{I}_m\|_F$ is minimal. By the considerations above we have to find for each column $1 \leq i \leq m$ a value $D(i, i)$ such that

$$\|AD(:, i) - e_i\|_2^2 = (A(i, i)D(i, i) - 1)^2 + \sum_{j \neq i} (A(j, i)D(i, i) - 0)^2 \quad (4.12)$$

is minimal. Hence, by setting the derivative of (4.12) (with respect to $D(i, i)$) to zero we obtain the critical point

$$D(i, i) = \frac{A(i, i)}{\sum_{i=1}^n A(i, i)^2}.$$

This critical point is indeed a minimum and leads to the desired diagonal matrix since the second derivative of (4.12) is positive.

We can proceed analogously in the case that the diagonal of D is constrained to be a d_H -based vector which leads to the following theorem.

Theorem 4.17. *Let $A \in \mathbb{R}^{m \times m}$ with $m = 2^n$ be arbitrary. Then minimizing $\|AH - \mathcal{I}_m\|_F$ with $H \in \mathcal{B}_n$ is equivalent to minimizing $\|\tilde{A}\Lambda - \mathcal{I}_m\|_F$, where Λ is the matrix of eigenvalues of H . Moreover, the $n + 1$ eigenvalues of the matrix H^* which minimizes $\|AH - \mathcal{I}_m\|_F$ over \mathcal{B}_n are for $0 \leq d \leq n$ given by*

$$\lambda_d = \frac{\sum_{i: d_H(i,1)=d} \tilde{A}(i, i)}{\sum_{i: d_H(i,1)=d} \sum_{j=1}^m \tilde{A}(i, j)^2}, \quad \text{with } \tilde{A} := W_n A W_n.$$

Hence, by Theorem 4.8, H^* can be explicitly computed.

Proof. Because of $H \in \mathcal{B}_n$ we know that $H = W_n \Lambda W_n$. Hence, because $W_n W_n = \mathcal{I}_m$ and because $\|\cdot\|_F$ is invariant with respect to orthogonal transformations [Golub and Van Loan, 1996, § 2.5.2], it follows that

$$\begin{aligned} \|AH - \mathcal{I}_m\|_F &= \|AW_n \Lambda W_n - \mathcal{I}_m\|_F = \|W_n(W_n A W_n \Lambda - \mathcal{I}_m)W_n\|_F \\ &= \|W_n A W_n \Lambda - \mathcal{I}_m\|_F = \|\tilde{A}\Lambda - \mathcal{I}_m\|_F \end{aligned}$$

Since Λ is supposed to be diagonal, the problem $\|\tilde{A}\Lambda - \mathcal{I}_m\|_F$ can easily be solved. Analogously to (4.12) we obtain for each degree of freedom in Λ , i. e., $\lambda_d = \Lambda(i, i)$ with $d = d_H(i, 1)$ that

$$\sum_{i: d_H(i,1)=d} \|\tilde{A}\Lambda(:, i) - e_i\|_2^2 = \sum_{i: d_H(i,1)=d} \left((\tilde{A}(i, i)\lambda_d - 1)^2 + \sum_{j \neq i} \tilde{A}(j, i)^2 \lambda_d^2 \right).$$

Therefore, the claim follows by differentiating with respect to λ_d and reordering. \square

While Problem 4.3 and Problem 4.4 are common problems which can be studied for many classes of matrices we study in the following a more specific approximation problem which has an interesting connection to the nearest Kronecker product problem (see Section 3.3).

Problem 4.5. Given an arbitrary matrix $A \in \mathbb{R}^{2^n \times 2^n}$. Find a d_H -based matrix $H \in \mathcal{B}_n$ and a non-negative diagonal matrix D such that $\|A - HD\|_F$ is minimal. //

As in the case of the nearest Kronecker product problem (cf. Problem 3.1) we will see that applying a particular reordering \mathcal{CR} to A and HD transforms Problem 4.5 into a more accessible problem which can be efficiently solved. First, consider the following instructive example (cf. Example 3.9), before we give the precise definition of \mathcal{CR} in Definition 4.6.

4. Matrices & Hamming Distance

Example 4.8. Given $A \in \mathbb{R}^{4 \times 4}$, we want to find a d_H -based matrix $H \in \mathcal{B}_2$ and a diagonal matrix $D \in \mathbb{R}^{4 \times 4}$ such that $\|A - HD\|_F$ is minimal. Explicitly, this problem reads as

$$\left\| \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix} - \begin{pmatrix} h_1 & h_2 & h_2 & h_3 \\ h_2 & h_1 & h_3 & h_2 \\ h_2 & h_3 & h_1 & h_2 \\ h_3 & h_2 & h_2 & h_1 \end{pmatrix} \begin{pmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{pmatrix} \right\|_F \rightarrow \min.$$

Since the diagonal matrix D scales the columns of H , we can reorder the elements within the individual columns of H and A , respectively. Therefore, we consider a (column) reordering \mathcal{CR} which reorders H and A such that $\mathcal{CR}(H)$ consists of ‘‘constant rows’’.

$$\left\| \begin{pmatrix} a_{1,1} & a_{2,2} & a_{3,3} & a_{4,4} \\ a_{2,1} & a_{1,2} & a_{1,3} & a_{2,4} \\ a_{3,1} & a_{4,2} & a_{4,3} & a_{3,4} \\ a_{4,1} & a_{3,2} & a_{2,3} & a_{1,4} \end{pmatrix} - \begin{pmatrix} h_1 & h_1 & h_1 & h_1 \\ h_2 & h_2 & h_2 & h_2 \\ h_2 & h_2 & h_2 & h_2 \\ h_3 & h_3 & h_3 & h_3 \end{pmatrix} \begin{pmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{pmatrix} \right\|_F \rightarrow \min \quad (4.13)$$

In the form of (4.13), [Problem 4.5](#) is a structured rank-1 approximation problem: Find a d_H -based vector $h \in \mathbb{R}^4$ and a vector $d \in \mathbb{R}^4$ such that $\|\mathcal{CR}(A) - hd^T\|_F$ is minimal. We can further reduce the problem to a classical rank-1 approximation problem by combining the equal rows of the reordering $\mathcal{CR}(H)$ of H . In the special case at hand we obtain the rank-1 approximation problem

$$\left\| \begin{pmatrix} a_{1,1} & a_{2,2} & a_{3,3} & a_{4,4} \\ \frac{a_{2,1}+a_{3,1}}{2} & \frac{a_{1,2}+a_{4,2}}{2} & \frac{a_{1,3}+a_{4,3}}{2} & \frac{a_{2,4}+a_{3,4}}{2} \\ a_{4,1} & a_{3,2} & a_{2,3} & a_{1,4} \end{pmatrix} - \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} (d_1, d_2, d_3, d_4) \right\|_F \rightarrow \min. \quad (4.14)$$

That the problems (4.13) and (4.14) are indeed equivalent is a non-trivial fact which we show more generally in [Lemma 4.18](#). //

At the first sight (see (4.13)) the reordering \mathcal{CR} seems to be somewhat irregular. Moreover, the characterization based on the operation of \mathcal{CR} on matrices $H \in \mathcal{B}_n$ (cf. [Example 4.8](#)) shows that it is not unique. Following an idea by Niederbrucker and Gansterer [[2011b](#)] we obtain the following closed form definition.

Definition 4.6. For a matrix $A \in \mathbb{R}^{2^n \times 2^n}$ its reordering $\mathcal{CR}(A)$ is element-wise defined as $\mathcal{CR}(A)(i, j) := A((i - 1) \oplus (j - 1) + 1, j)$, where \oplus denotes the bit-wise XOR of two integers. That means $i \oplus j$ is the integer whose binary representation is the result of applying XOR element-wise on the binary representations of i and j , respectively. //

Since we will not use the explicit definition of \mathcal{CR} in the following, we refer for the correctness and derivation of this abstract representation to Niederbrucker and Gansterer [[2011b](#)]. Note that Niederbrucker and Gansterer [[2011b](#)] do not explicitly construct \mathcal{CR} as reordering but implicitly in terms of a (sparse) matrix vector multiplication routine. Moreover, since \mathcal{CR} solely reorders entries within their respective columns, it immediately follows that for matrices A, H, D as defined in [Problem 4.5](#) it holds that indeed

$$\|A - HD\|_F = \|\mathcal{CR}(A) - \mathcal{CR}(HD)\|_F = \|\mathcal{CR}(A) - \mathcal{CR}(H)D\|_F.$$

See also [Example 4.8](#) for an illustration of this property.

Definition 4.7. For given integers $k, n \in \mathbb{N}$ with $0 \leq k \leq n$ we define β_k^n as

$$\beta_k^n := \sum_{i=0}^k \binom{n}{i}$$

Thus, for $k > 0$, this means that $\beta_k^n - \beta_{k-1}^n = \binom{n}{k}$. //

Using [Definition 4.7](#) we can formalize the last transformation step [\(4.14\)](#) we pursued in [Example 4.8](#). In general, a d_H -based vector $h \in \mathbb{R}^{2^n}$ contains (at most) $n + 1$ distinct values given by its associated function φ_h . By definition (see also [Lemma 4.1](#)) the value $\varphi_h(k)$ appears $\binom{n}{k}$ times in h . Therefore, we consider the reordering \bar{h} of h whose elements are increasingly sorted with respect to φ_h^{-1} (cf. [\(4.13\)](#)). That means $\bar{h}(1) = \varphi_h(0)$ and $\bar{h}((\beta_{k-1}^n + 1) : \beta_k^n) = \varphi_h(k)$ for $0 < k \leq n$.

Lemma 4.18. For a matrix $A \in \mathbb{R}^{m \times m}$ with $m = 2^n$ consider the following two problems.

(i). Find a d_H -based vector $h \in \mathbb{R}^m$ and a vector $d \in \mathbb{R}^m$ such that $\|A - hd^T\|_F$ is minimal.

(ii). Find vectors $\bar{h} \in \mathbb{R}^{m+1}$ and $\bar{d} \in \mathbb{R}^m$ such that $\|\bar{A} - \bar{h}\bar{d}^T\|_F$ is minimal, where

$$\bar{A} \in \mathbb{R}^{(n+1) \times 2^n}, \quad \bar{A}(k, :) := \begin{cases} A(1, :) & k = 1 \\ \frac{1}{\beta_k^n - \beta_{k-1}^n} \sum_{i=\beta_{k-1}^n+1}^{\beta_k^n} A(i, :) & k > 1 \end{cases} \quad (4.15)$$

Then the problems (i) and (ii) are equivalent. Concretely, for the respective optimal solutions (h^*, d^*) and (\bar{h}^*, \bar{d}^*) it holds that $\bar{d}^* = d^*$ and that the associated function of h^* is given by $\varphi_{h^*}^*(k) = \bar{h}_k^*$.

Proof. The target function in problem (i) can be explicitly written as

$$\|A - hd^T\|_F^2 = \sum_{k=0}^n \sum_{j=1}^m \sum_{i: d_H(i,1)=k} (A(i, j) - h_k d(j))^2$$

where $h_l = h(i)$ for all i with $d_H(i, 1) = l$. Next, we derive the equation system a critical point as to fulfill. For the variables h_l with $0 \leq l \leq n$ we get

$$\begin{aligned} 0 &= \frac{\partial}{\partial h_l} \left(\sum_{k=0}^n \sum_{j=1}^m \sum_{i: d_H(i,1)=k} (A(i, j) - h_k d(j))^2 \right) \\ \Leftrightarrow 0 &= \sum_{j=1}^m \sum_{i: d_H(i,1)=l} -2(A(i, j) - h_l d(j))d(j) \\ \Leftrightarrow h_l \binom{n}{l} \sum_{j=1}^m d(j)^2 &= \sum_{j=1}^m \sum_{i: d_H(i,1)=l} A(i, j)d(j) \\ \Leftrightarrow h_l &= \frac{\sum_{j=1}^m d(j) \left(\sum_{i: d_H(i,1)=k} A(i, j) \binom{n}{k}^{-1} \right)}{\sum_{j=1}^m d(j)^2} \\ \Leftrightarrow h_l &= \frac{\sum_{j=1}^m d(j) \bar{A}(l+1, j)}{\sum_{j=1}^m d(j)^2} \end{aligned}$$

4. Matrices & Hamming Distance

Analogously, we obtain for all $d(l)$ with $1 \leq l \leq m$,

$$\frac{\partial}{\partial d(l)} \left(\sum_{k=0}^n \sum_{j=1}^m \sum_{i: d_H(i,1)=k} (A(i,j) - h_k d(j))^2 \right) = 0 \Leftrightarrow d(l) = \frac{\sum_{k=0}^n h(k) \bar{A}(k+1, l)}{\sum_{k=0}^n h(k)^2}.$$

By identifying $h_l = \bar{h}(l+1)$ in problem (ii) we obtain similarly

$$\begin{aligned} 0 &= \frac{\partial}{\partial h_l} \left(\sum_{k=0}^n \sum_{j=0}^m (\bar{A}(k+1, j) - h_k d(j))^2 \right) \\ \Leftrightarrow 0 &= \sum_{j=0}^m -2(\bar{A}(l+1, j) - h_l d(j)) d(j) \\ \Leftrightarrow h_l &= \frac{\sum_{j=0}^m d(j) \bar{A}(l+1, j)}{\sum_{j=0}^m d(j)^2} \end{aligned}$$

as well as

$$\frac{\partial}{\partial d(l)} \left(\sum_{k=0}^n \sum_{j=0}^m (\bar{A}(k+1, j) - h_k d(j))^2 \right) = 0 \iff d(l) = \frac{\sum_{k=0}^n h_k \bar{A}(k+1, l)}{\sum_{k=0}^n h_k^2}$$

In the notation used above, problem (i) and (ii) have identical critical points, which proofs the claimed equivalence. \square

Corollary 4.19. *For a given matrix $A \in \mathbb{R}^m$ with $m = 2^n$, the solution of [Problem 4.5](#) is given by the solution of the rank-1 approximation problem $\|\overline{\mathcal{CR}(A)} - h d^T\|_F \rightarrow \min$ where $\overline{\mathcal{CR}(A)}$ is the reduced form of $\mathcal{CR}(A)$ according to [\(4.15\)](#), $h \in \mathbb{R}^{n+1}$ and $d \in \mathbb{R}^m$. In particular, the associated function of the optimal H is given by $\varphi_H(x) = h(x+1)$ and the optimal D is given as $D = \text{diag}(d)$.*

Proof. This follows immediately by the fact that $\|A - HD\|_F = \|\overline{\mathcal{CR}(A)} - \mathcal{CR}(H)D\|_F$ (because \mathcal{CR} reorders elements within their column only) and [Lemma 4.18](#). \square

Since the solution of [Problem 4.5](#) relies on solving a certain rank-1 approximation problem similarly to the nearest Kronecker product problem ([Problem 3.1](#)), the same conditions on its efficiency apply. Concretely, efficient routines for the matrix vector products with $\overline{\mathcal{CR}(A)}$ and its transpose are required. We will see in the concrete application in which we consider [Problem 4.5](#) that the required matrix vector products can be performed in linear time only, which allows to solve this approximation problem efficiently.

4.3. Arbitrary Alphabets

In the previous [Section 4.2](#) we studied the special case $\mathcal{B}_n = \mathcal{H}(\mathbb{Z}_2^n)$ into great detail. In principle, most of the presented results can be analogously derived for $\mathcal{H}(\Omega)$ with

arbitrary signature $\Omega = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \cdots \times \mathbb{Z}_{a_1}$ ¹. We refrain in the following from developing all results from [Section 4.2](#) in the case of an arbitrary signature. Instead, we provide all major insights such that further details can easily be derived. In particular, we provide a full characterization of the spectral properties of the matrices $\mathcal{H}(\Omega)$, which is the essential ingredient in the analysis of d_H -based matrices (cf. [Section 4.2](#)). Moreover, we will also see that the more general setting discussed in this section does surprisingly not cause problems from a computational point of view, i. e., we can still provide highly efficient matrix operations. First, we exemplify how d_H -based matrices are structured beyond the case of \mathcal{B}_n .

Example 4.9. Consider matrices $H \in \mathcal{H}(\mathbb{Z}_4)$ and $G \in \mathcal{H}(\mathbb{Z}_3 \times \mathbb{Z}_2)$. Concretely, this means that we interpret the row/column indices of H as strings of length one over the alphabet \mathbb{Z}_4 . Moreover, the row/column indices of G are interpreted as strings of length two, where the alphabets for the individual elements are given by \mathbb{Z}_3 and \mathbb{Z}_2 , respectively. In particular, the matrices H and G can be imagined as follows (see also [Example 4.2](#)).

$$H = \begin{pmatrix} a & b & b & b \\ b & a & b & b \\ b & b & a & b \\ b & b & b & a \end{pmatrix} \rightsquigarrow \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & a & b & b & b \\ 1 & b & a & b & b \\ 2 & b & b & a & b \\ 3 & b & b & b & a \end{array}$$

$$G = \begin{pmatrix} c & d & d & e & d & e \\ d & c & e & d & e & d \\ d & e & c & d & d & e \\ e & d & d & c & e & d \\ d & e & d & e & c & d \\ e & d & e & d & d & c \end{pmatrix} \rightsquigarrow \begin{array}{c|cccccc} & 00 & 01 & 10 & 11 & 20 & 21 \\ \hline 00 & c & d & d & e & d & e \\ 01 & d & c & e & d & e & d \\ 10 & d & e & c & d & d & e \\ 11 & e & d & d & c & e & d \\ 20 & d & e & d & e & c & d \\ 21 & e & d & e & d & d & c \end{array}$$

For understanding the structure inherent in H and G recall that the Hamming distance $d_H(s_1, s_2)$ simply counts the number of different elements between the strings s_1 and s_2 (cf. [Definition 4.1](#)). These examples also suggest that for $\Omega = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \cdots \times \mathbb{Z}_{a_1}$ the matrices $H \in \mathcal{H}(\Omega)$ consist only of $n + 1$ values (degrees of freedom) as in the case of \mathcal{B}_n , i. e., the function φ_H associated with $H \in \mathcal{H}(\Omega)$ is a function $\varphi_H: \mathbb{Z}_{n+1} \rightarrow \mathbb{R}$. //

Lemma 4.20 (cf. [Lemma 4.3](#)). *Let $\Omega = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \cdots \times \mathbb{Z}_{a_1}$ and $m = a_1 \cdot a_2 \cdot \dots \cdot a_n$. Then, $H \in \mathcal{H}(\Omega) \subset \mathbb{R}^{m \times m}$ if and only if the following two conditions hold.*

¹We use this “reversed” notation throughout this section for the sake of a concise formal presentation. In particular, this notation is motivated by the fact that (mixed radix) numbers “grow to the left” with increasing value.

4. Matrices & Hamming Distance

(i). The matrix $H \in \mathbb{R}^{m \times m}$ is an $a_n \times a_n$ block matrix with blocks of size m/a_n ,

$$H = \begin{pmatrix} C & D & \cdots & D \\ D & C & \ddots & \vdots \\ \vdots & \ddots & \ddots & D \\ D & \cdots & D & C \end{pmatrix}, \quad C, D \in \mathcal{H}(\mathbb{Z}_{a_{n-1}} \times \mathbb{Z}_{a_{n-2}} \times \cdots \times \mathbb{Z}_{a_1}). \quad (4.16)$$

(ii). Let φ_H , φ_C and φ_D denote the function associated with H , C and D , respectively. Then, $\varphi_C(d) = \varphi_H(d)$ and $\varphi_D(d) = \varphi_H(d+1)$ holds for all $0 \leq d < n$.

Proof. Use the same proof as for [Lemma 4.3](#) by interpreting the column/row indices as mixed radix numbers instead of binary numbers. See also [Example 4.2](#) and [Example 4.9](#) for concrete examples. \square

In the case of $\mathcal{H}(\mathbb{Z}_2^n)$ the one dimensional base case $\mathcal{H}(\mathbb{Z}_2)$ could easily be understood. Since the properties of the matrices $\mathcal{H}(\mathbb{Z}_n)$ with general $n \in \mathbb{N}$ are less immediate, we first investigate the structures behind these matrices.

4.3.1. The Matrices $\mathcal{H}(\mathbb{Z}_n)$

Our ultimate goal is a characterization of the eigenvectors and eigenvalues of the matrices $\mathcal{H}(\Omega)$ with an arbitrary signature Ω . As a prerequisite we characterize the eigenvectors and eigenvalues of the matrices $\mathcal{H}(\mathbb{Z}_n)$.

Definition 4.8. For $n \in \mathbb{N}$ we define a matrix $V_n \in \mathbb{R}^{n \times n}$ by

$$[V_n]_{i,j} := \begin{cases} \frac{1}{\sqrt{n}} & j = 1 \\ \frac{1}{\sqrt{j(j-1)}} & j > 1 \wedge i < j \\ -\frac{j-1}{\sqrt{j(j-1)}} & j > 1 \wedge i = j \\ 0 & j > 1 \wedge i > j \end{cases}$$

Note that this definition leads for $n = 2$ to $V_2 = W_1$, i. e., V_2 is equal to the normalized Walsh matrix W_1 (cf. [Definition 4.5](#)). //

Example 4.10. First, we explicitly depict V_n according to [Definition 4.8](#) for $n \in \{2, 3, 4\}$.

$$V_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad V_3 = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \end{pmatrix}, \quad V_4 = \begin{pmatrix} \frac{1}{\sqrt{4}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & 0 & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & 0 & 0 & -\frac{3}{\sqrt{12}} \end{pmatrix}$$

Second, for $H \in \mathcal{H}(\mathbb{Z}_4)$ we compute that $HV_4 = V_4\Lambda$ with a diagonal matrix Λ .

$$\begin{array}{c|c} & \begin{pmatrix} \frac{1}{\sqrt{4}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & 0 & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & 0 & 0 & -\frac{3}{\sqrt{12}} \end{pmatrix} \\ \hline \begin{pmatrix} a & b & b & b \\ b & a & b & a \\ b & b & a & b \\ b & b & b & a \end{pmatrix} & \begin{pmatrix} \frac{1}{\sqrt{4}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & 0 & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{4}} & 0 & 0 & -\frac{3}{\sqrt{12}} \end{pmatrix} \begin{pmatrix} a+3b & 0 & \dots & 0 \\ 0 & a-b & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a-b \end{pmatrix} \end{array} \quad (4.17)$$

The following lemma reveals for any $n \in \mathbb{N}$ that the matrices V_n are orthogonal and that they diagonalize the matrices $\mathcal{H}(\mathbb{Z}_n)$. //

Lemma 4.21. *For any $n \in \mathbb{N}$, the matrix V_n is orthogonal and it diagonalizes all matrices $H \in \mathcal{H}(\mathbb{Z}_n)$, i. e., $V_n^T H V_n = \Lambda$ with a diagonal matrix $\Lambda \in \mathbb{R}^{n \times n}$. Moreover, $\Lambda(1, 1) = \varphi_H(0) + (n-1)\varphi_H(1)$ and $\Lambda(i, i) = \varphi_H(0) - \varphi_H(1)$ for $1 < i \leq n$.*

Proof. We prove the orthogonality of V_n by induction on n . Since V_2 is the normalized Walsh matrix W_1 (see [Definition 4.5](#)), we know that V_2 is orthogonal. Now assume that V_{n-1} is orthogonal. By [Definition 4.8](#) (see also [Example 4.10](#)) we see that V_n evolves out of V_{n-1} by: (1) changing the first column from $(1/\sqrt{n-1}, \dots, 1/\sqrt{n-1})^T$ to $(1/\sqrt{n}, \dots, 1/\sqrt{n})^T$, (2) appending the row $(1/\sqrt{n}, 0, \dots, 0, -(n-1)/\sqrt{n(n-1)})$, and (3) appending the column $(1/\sqrt{n(n-1)}, \dots, 1/\sqrt{n(n-1)}, -(n-1)/\sqrt{n(n-1)})^T$. First, we obtain for the n -th row and column of the product $V_n V_n^T$ that

$$\begin{array}{c|c} & \begin{pmatrix} \frac{1}{\sqrt{n}} & \dots & \frac{1}{\sqrt{n}} & \frac{1}{\sqrt{n}} \\ * & \dots & * & 0 \\ \vdots & \ddots & \vdots & \vdots \\ * & \dots & * & 0 \\ \frac{1}{\sqrt{n(n-1)}} & \dots & \frac{1}{\sqrt{n(n-1)}} & -\frac{n-1}{\sqrt{n(n-1)}} \end{pmatrix} \\ \hline \begin{pmatrix} \frac{1}{\sqrt{n}} & * & \dots & * & \frac{1}{\sqrt{n(n-1)}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{\sqrt{n}} & * & \dots & * & \frac{1}{\sqrt{n(n-1)}} \\ \frac{1}{\sqrt{n}} & 0 & \dots & 0 & -\frac{n-1}{\sqrt{n(n-1)}} \end{pmatrix} & \begin{pmatrix} * & \dots & * & 0 \\ \vdots & \ddots & \vdots & \vdots \\ * & \dots & * & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix} \end{array} \quad (4.18)$$

since $1/n - (n-1)/(n(n-1)) = 0$. Moreover, note that the block of V_n we sketched by $*$ in (4.18) is equal to $V_{n-1}(:, 2:n-1)$. Thus, since V_{n-1} is orthogonal and its first column is given by $(1/\sqrt{n-1}, \dots, 1/\sqrt{n-1})^T$ we see that the matrix product $V_{n-1}(:, 2:n-1)V_{n-1}^T(:, 2:n-1) = \mathcal{I}_{n-1} - 1/(n-1)$, where $1/(n-1)$ denotes the constant matrix with entry $1/(n-1)$. Therefore, because $(1/\sqrt{n})^2 + (1/\sqrt{n(n-1)})^2 = 1/(n-1)$, the orthogonality of the matrix V_n is proved (cf. (4.18)).

For the second property, $V_n^T H V_n = \Lambda$, we prove $H V_n = V_n \Lambda$ in compliance with [Example 4.10](#) where the case $n = 4$ is depicted. We simply study how H operates on

4. Matrices & Hamming Distance

the columns of V_n . For the first column $V_n(:, 1) = (1/\sqrt{n}, \dots, 1/\sqrt{n})^\top$ the claim holds trivially with $\Lambda(1, 1) = \varphi_H(0) + (n-1)\varphi_H(1)$. For all other columns $1 < i \leq n$ we see by [Definition 4.8](#) (see also [Example 4.10](#)) that the columns sum up to zero. This shows for $v = HV_n(:, i)$ that $v(j) = 0$ for all $j > i$ since all non-zero values of $V_n(:, i)$ get multiplied by the same constant (cf. [\(4.17\)](#)). For the same reason, in the case $j = i$, it holds that $v(i) = -(i-1)/\sqrt{i(i-1)}(\varphi_H(0) - \varphi_H(1))$. Moreover, in the case $j < i$ we obtain

$$\begin{aligned} v(i) &= \frac{1}{\sqrt{i(i-1)}}\varphi_H(0) + (i-2)\frac{1}{\sqrt{i(i-1)}}\varphi_H(1) - \frac{(i-1)}{\sqrt{i(i-1)}}\varphi_H(1) \\ &= \frac{1}{\sqrt{i(i-1)}}(\varphi_H(0) - \varphi_H(1)) \end{aligned}$$

Thus, $v = V_n(:, i)(\varphi_H(0) - \varphi_H(1))$ and therefore, by setting $\Lambda(i, i) = \varphi_H(0) - \varphi_H(1)$, we end up with the claimed representation, which completes the proof. \square

Note that the choice of V_n is (besides of the first column) by no means unique since the vectors $\{V_n(:, i) \mid 1 < i \leq n\}$ span an $(n-1)$ -dimensional eigenspace. On the other hand, we will see that the particular choice of V_n we made in [Definition 4.8](#) is beneficial from a computational point of view. Concretely, [Section 4.3.3](#) will reveal that the matrix vector product $V_n v$ with $v \in \mathbb{R}^n$ can be computed in $\Theta(n)$ time.

4.3.2. Algebraic Properties

Based on the preceding results, it seems likely that all matrices $H \in \mathcal{H}(\Omega)$ have also the same eigenvectors in case of an arbitrary signature $\Omega = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \dots \times \mathbb{Z}_{a_1}$. We show in this section that this is indeed the case. Moreover, we will see that in case $\Omega = \mathbb{Z}_c^n$ the eigenvalues show an analogous structure to the binary case, whereas the general case $\Omega = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \dots \times \mathbb{Z}_{a_1}$ leads to less structure in the eigenvalues.

Corollary 4.22. *Let $H \in \mathbb{R}^{nk \times nk}$ be an $n \times n$ block matrix of the form*

$$H = \begin{pmatrix} \Lambda_C & \Lambda_D & \cdots & \Lambda_D \\ \Lambda_D & \Lambda_C & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Lambda_D \\ \Lambda_D & \cdots & \Lambda_D & \Lambda_C \end{pmatrix}$$

with diagonal matrices $\Lambda_C, \Lambda_D \in \mathbb{R}^{k \times k}$. Then it holds that

$$(V_n \otimes \mathcal{I}_k)^\top H (V_n \otimes \mathcal{I}_k) = \text{diag}(\Lambda_C + (n-1)\Lambda_D, \Lambda_C - \Lambda_D, \dots, \Lambda_C - \Lambda_D) = \Lambda. \quad (4.19)$$

In other words, H is diagonalized by $V_n \otimes \mathcal{I}_k$.

Proof. Recall that the vector Kronecker product $V_n \otimes \mathcal{I}_k$ (cf. [Definition 3.4](#)) “vectorizes” scalar computations. Here this means that we are exactly in the situation of [Lemma 4.21](#), except of the fact that H is composed by diagonal blocks instead of scalars. Thus, since the multiplication of diagonal matrices is commutative, the validity of the claim follows directly by (the proof of) [Lemma 4.21](#). \square

Theorem 4.23. *Let $\Omega = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \cdots \times \mathbb{Z}_{a_1}$ be an arbitrary signature. Then the matrix $V_\Omega := \bigotimes_{i=0}^{n-1} V_{a_{n-i}}$ diagonalizes all matrices $H \in \mathcal{H}(\Omega)$, i. e., $\Lambda = V_\Omega^\top H V_\Omega$ is a diagonal matrix and the columns of V_Ω are the eigenvectors of all $H \in \mathcal{H}(V_\Omega)$.*

Proof. We define $\Omega_k := \mathbb{Z}_{a_k} \times \mathbb{Z}_{a_{k-1}} \times \cdots \times \mathbb{Z}_{a_1}$ for all $1 \leq k \leq n$ and inductively show that V_{Ω_k} diagonalizes all matrices $H \in \mathcal{H}(\Omega_k)$. The base case $k = 1$ is covered by [Lemma 4.21](#) since $\Omega_1 = \mathbb{Z}_{a_1}$. Assume that the claim holds for k . Then $V_{\Omega_{k+1}} = V_{a_{k+1}} \otimes V_{\Omega_k}$ and we can decompose $V_{\Omega_{k+1}}$ in the form $V_{\Omega_{k+1}} = (\mathcal{I}_{a_{k+1}} \otimes V_{\Omega_k})(V_{a_{k+1}} \otimes \mathcal{I}_{|\Omega_k|})$. Consequently, we compute $V_\Omega^\top H V_\Omega$ in two steps. First, we compute $\tilde{H} = (\mathcal{I}_{a_{k+1}} \otimes V_{\Omega_k})^\top H (\mathcal{I}_{a_{k+1}} \otimes V_{\Omega_k})$ and subsequently, we compute $\Lambda = (V_{a_{k+1}} \otimes \mathcal{I}_{|\Omega_k|})^\top \tilde{H} (V_{a_{k+1}} \otimes \mathcal{I}_{|\Omega_k|})$. According to [Lemma 4.20](#) the matrices $H \in V_{\Omega_{k+1}}$ are block matrices with blocks $C, D \in \mathcal{H}(V_{\Omega_k})$.

$$\begin{aligned} \tilde{H} &= (\mathcal{I}_{a_{k+1}} \otimes V_{\Omega_k})^\top H (\mathcal{I}_{a_{k+1}} \otimes V_{\Omega_k}) = (\mathcal{I}_{a_{k+1}} \otimes V_{\Omega_k}^\top) H (\mathcal{I}_{a_{k+1}} \otimes V_{\Omega_k}) \\ &= \begin{pmatrix} V_{\Omega_k}^\top & 0 & \cdots & 0 \\ 0 & V_{\Omega_k}^\top & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & V_{\Omega_k}^\top \end{pmatrix} \begin{pmatrix} C & D & \cdots & D \\ D & C & \ddots & \vdots \\ \vdots & \ddots & \ddots & D \\ D & \cdots & D & C \end{pmatrix} \begin{pmatrix} V_{\Omega_k} & 0 & \cdots & 0 \\ 0 & V_{\Omega_k} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & V_{\Omega_k} \end{pmatrix} \\ &= \begin{pmatrix} V_{\Omega_k}^\top C V_{\Omega_k} & V_{\Omega_k}^\top D V_{\Omega_k} & \cdots & V_{\Omega_k}^\top D V_{\Omega_k} \\ V_{\Omega_k}^\top D V_{\Omega_k} & V_{\Omega_k}^\top C V_{\Omega_k} & \ddots & \vdots \\ \vdots & \ddots & \ddots & V_{\Omega_k}^\top D V_{\Omega_k} \\ V_{\Omega_k}^\top D V_{\Omega_k} & \cdots & V_{\Omega_k}^\top D V_{\Omega_k} & V_{\Omega_k}^\top C V_{\Omega_k} \end{pmatrix} = \begin{pmatrix} \Lambda_C & \Lambda_D & \cdots & \Lambda_D \\ \Lambda_D & \Lambda_C & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Lambda_D \\ \Lambda_D & \cdots & \Lambda_D & \Lambda_C \end{pmatrix} \end{aligned}$$

Since $\Lambda_C = V_{\Omega_k}^\top C V_{\Omega_k}$ and $\Lambda_D = V_{\Omega_k}^\top D V_{\Omega_k}$ are diagonal matrices (by the induction hypothesis), we can apply [Corollary 4.22](#) to compute $\Lambda = (V_{a_{k+1}} \otimes \mathcal{I}_{|\Omega_k|})^\top \tilde{H} (V_{a_{k+1}} \otimes \mathcal{I}_{|\Omega_k|})$. This completes the inductive step as well as the entire proof. \square

Therefore, for arbitrary signature Ω , all matrices $H \in \mathcal{H}(\Omega)$ have the same eigenvectors given by the columns of V_Ω . The corresponding eigenvalues can be explicitly computed by the recursive relationship given in [\(4.19\)](#). In the binary case we have shown that the diagonal of the matrix of eigenvalues is a d_H -based vector (cf. [Theorem 4.7](#)). This property does not generalize to arbitrary signatures Ω —as the following example indicates.

Example 4.11. Consider $H \in \mathcal{H}(\Omega)$ with $\Omega = \mathbb{Z}_3 \times \mathbb{Z}_2$, i. e., the associated function φ_H is given by $\varphi_H(0) = c$, $\varphi_H(1) = d$ and $\varphi_H(2) = e$. This means (cf. [Example 4.3](#)) that the eigenvalues of the two distinct 2×2 blocks C, D of H (cf. [Lemma 4.3](#)) are given by $(c + d, c - d)$ and $(d + e, d - e)$, respectively. Thus, by the recursive relationship [\(4.19\)](#), we obtain

$$\text{diag}(V_\Omega^\top H V_\Omega)^\top = \begin{pmatrix} (c + d) + 2(d + e) \\ (c - d) + 2(d - e) \\ (c + d) - (d + e) \\ (c - d) - (d - e) \\ (c + d) - (d + e) \\ (c - d) - (d - e) \end{pmatrix} = \begin{pmatrix} c + 3d + 2e \\ c + d - 2e \\ c - e \\ c + e \\ c - e \\ c + e \end{pmatrix} \neq \begin{pmatrix} \tilde{c} \\ \tilde{d} \\ \tilde{d} \\ \tilde{d} \\ \tilde{e} \\ \tilde{e} \end{pmatrix} \quad \text{for } d \neq e.$$

Thus, the diagonal of the matrix of eigenvalues is in general not a d_H -based vector. $//$

4. Matrices & Hamming Distance

An immediate consequence of the missing structure in the matrix of eigenvalues of the matrices $H \in \mathcal{H}(\Omega)$ is that the set $\mathcal{H}(\Omega)$ is in general not closed under multiplication.

Example 4.12. Consider $\Omega = \mathbb{Z}_3 \times \mathbb{Z}_2$ and let $G \in \mathcal{H}(\Omega)$ (cf. [Example 4.9](#)). Then $G \cdot G$ is in general not an element of $\mathcal{H}(\Omega)$. It suffices to show that $G \cdot G(:, 1)$ is not a d_H -based vector. Concretely, we obtain

$$G \cdot G(:, 1) = \begin{pmatrix} c & d & d & e & d & e \\ d & c & e & d & e & d \\ d & e & c & d & d & e \\ e & d & d & c & e & d \\ d & e & d & e & c & d \\ e & d & e & d & d & c \end{pmatrix} \begin{pmatrix} c \\ d \\ d \\ e \\ d \\ e \end{pmatrix} = \begin{pmatrix} c^2 + 3d^2 + 2e^2 \\ 2cd + 4de \\ 2cd + d^2 + 2de + e^2 \\ 2d^2 + 2ce + 2de \\ 2cd + d^2 + 2de + e^2 \\ 2d^2 + 2ce + 2de \end{pmatrix} \neq \begin{pmatrix} \tilde{c} \\ \tilde{d} \\ \tilde{d} \\ \tilde{e} \\ \tilde{d} \\ \tilde{e} \end{pmatrix} \text{ for } d \neq e.$$

Thus, the set $\mathcal{H}(\Omega)$ is in general not closed under multiplication. //

In the remainder of this section we consider signatures of the form $\Omega = \mathbb{Z}_c^n$ with arbitrary $c, n \in \mathbb{N}$. For signatures of this particular shape, almost all of the results we obtained for the binary case $\Omega = \mathbb{Z}_2^n$ in [Section 4.2.1](#) can be naturally generalized.

Lemma 4.24 (cf. [Lemma 4.5](#)). *Let $c \in \mathbb{N}$ be fixed and (x, y) be a pair of vectors $x, y \in \mathbb{R}^n$ which satisfy for all $1 \leq i < n$ the property $x_i + cx_{i+1} = y_i - y_{i+1}$. Then it holds for the vectors $u = y + cx$ and $v = y - x$ that $u(2 : n) = v(1 : (n - 1))$.*

Proof. By definition of u and v , we get

$$\begin{aligned} u &= y + cx = (y_1 + cx_1, y_2 + cx_2, \dots, y_n + cx_n) \\ v &= y - x = (y_1 - x_1, \dots, y_{n-1} - x_{n-1}, y_n - x_n) \end{aligned}$$

Thus, the claim follows, since $y_{i+1} + cx_{i+1} = y_i - x_i$ holds by assumption. □

Lemma 4.25 (cf. [Lemma 4.6](#)). *Consider for $c \in \mathbb{N}$ the matrices $A_c^{(n)} \in \mathbb{R}^{(n+1) \times (n+1)}$ which are recursively defined by*

$$A_c^{(0)} := 1, \quad A_c^{(n)} := \begin{pmatrix} A_c^{(n-1)} & 0 \\ A_c^{(n-1)}(n, :) & 0 \end{pmatrix} + \begin{pmatrix} 0 & cA_c^{(n-1)} \\ 0 & -A_c^{(n-1)}(n, :) \end{pmatrix}. \quad (4.20)$$

Then $A_c^{(n)}$ can also be decomposed in the form

$$A_c^{(n)} = \begin{pmatrix} A_c^{(n-1)}(1, :) & 0 \\ A_c^{(n-1)} & 0 \end{pmatrix} + \begin{pmatrix} 0 & cA_c^{(n-1)}(1, :) \\ 0 & -A_c^{(n-1)} \end{pmatrix}. \quad (4.21)$$

Proof. Use the proof of [Lemma 4.6](#) with [Lemma 4.24](#) instead of [Lemma 4.5](#). □

The next example indicates that the case $\Omega = \mathbb{Z}_c^n$ indeed provides structure in the eigenvalues, similar to the binary case $\Omega = \mathbb{Z}_2^n$.

Example 4.13. Consider $H \in \mathcal{H}(\Omega)$ with $\Omega = \mathbb{Z}_3^2$, i. e., the associated function φ_H is given by $\varphi_H(0) = a$, $\varphi_H(1) = b$ and $\varphi_H(2) = c$. This means (cf. [Example 4.3](#)) that the eigenvalues of the two distinct 3×3 blocks C, D of H (cf. [Lemma 4.3](#)) are given by $(a + 2b, a - b, a - b)$ and $(b + 2c, b - c, b - c)$, respectively. Thus, by the recursive relationship ([4.19](#)), we obtain

$$\text{diag}(V_\Omega^T H V_\Omega)^T = \begin{pmatrix} (a + 2b) + 2(b + 2c) \\ (a - b) + 2(b - c) \\ (a - b) + 2(b - c) \\ (a + 2b) - (b + 2c) \\ (a - b) - (b - c) \\ (a - b) - (b - c) \\ (a + 2b) - (b + 2c) \\ (a - b) - (b - c) \\ (a - b) - (b - c) \\ (a - b) - (b - c) \end{pmatrix} = \begin{pmatrix} a + 4b + 4c \\ a + b - 2c \\ a + b - 2c \\ a + b - 2c \\ a - 2b + c \\ a - 2b + c \\ a + b - 2c \\ a + b - 2c \\ a - 2b + c \\ a - 2b + c \end{pmatrix} = \begin{pmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{b} \\ \tilde{b} \\ \tilde{c} \\ \tilde{c} \\ \tilde{b} \\ \tilde{c} \\ \tilde{b} \\ \tilde{c} \end{pmatrix}.$$

The following theorem generalizes [Theorem 4.7](#) and shows that $\text{diag}(V_\Omega^T H V_\Omega)$ is for all $\Omega = \mathbb{Z}_c^n$ and $H \in \mathcal{H}(\Omega)$ a d_H -based vector. //

Theorem 4.26 (cf. [Theorem 4.7](#)). *Let $H \in \mathcal{H}(\mathbb{Z}_c^n)$ with $c, n \in \mathbb{N}$ and φ_H its associated function. Then H has at most $n + 1$ different eigenvalues $\lambda_0, \lambda_1, \dots, \lambda_n$ and the multiplicity of λ_i is $\binom{n}{i}(c - 1)^i$. Moreover, the eigenvalues λ_i are given by*

$$(\lambda_0, \lambda_1, \dots, \lambda_n)^T = A_c^{(n)} (\varphi_H(0), \varphi_H(1), \dots, \varphi_H(n))^T \quad (4.22)$$

with the matrices $A_c^{(n)} \in \mathbb{R}^{(n+1) \times (n+1)}$ defined in ([4.20](#)).

Proof. Use the proof of [Theorem 4.7](#) with [Lemma 4.25](#) instead of [Lemma 4.6](#). \square

While [Theorem 4.26](#) shows that the associated function of a matrix $H \in \mathcal{H}(\mathbb{Z}_c^n)$ can be arbitrarily chosen, the following example reveals that, in contrast to the binary case, the eigenvalues cannot be arbitrarily chosen in case of signatures $\Omega = \mathbb{Z}_c^n$. More specifically, [Theorem 4.8](#) cannot be generalized to the case $\mathcal{H}(\mathbb{Z}_c^n)$.

Example 4.14. Consider $\Lambda = \text{diag}(2, 1, 1)$ and $\Omega = \mathbb{Z}_3$, then $V_\Omega^T \Lambda V_\Omega \notin \mathcal{H}(\Omega)$. First, note that we can decompose V_Ω in the form

$$V_\Omega = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 0 & -2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{6}} \end{pmatrix} = \tilde{V}_\Omega D$$

Next, we compute $\tilde{V}_\Omega^T \Lambda \tilde{V}_\Omega$ (in order to obtain $V_\Omega^T \Lambda V_\Omega = D \tilde{V}_\Omega^T \Lambda \tilde{V}_\Omega D$)

$$\tilde{V}_\Omega^T \Lambda \tilde{V}_\Omega = \begin{pmatrix} 2 & 1 & 1 \\ 2 & -1 & 0 \\ 2 & 1 & -2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 0 & -2 \end{pmatrix} = \begin{pmatrix} 4 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 7 \end{pmatrix}.$$

Thus, $V_\Omega^T \Lambda V_\Omega \notin \mathcal{H}(\Omega)$ is not an element of $\mathcal{H}(\Omega)$, because its (off-)diagonal elements are not equal (cf. [Lemma 4.20](#)). //

4. Matrices & Hamming Distance

As [Example 4.14](#) depicts, we cannot generalize [Theorem 4.8](#) in order to obtain a linear mapping between the eigenvalues of a matrix $H \in \mathcal{H}(\mathbb{Z}_c^n)$ and its associated functions. On the other hand, the set $\mathcal{H}(\mathbb{Z}_c^n)$ is still closed under multiplication.

Lemma 4.27. *Let $c, n \in \mathbb{N}$ be arbitrary, $\Omega = \mathbb{Z}_c^n$, $H \in \mathcal{H}(\Omega)$ a d_H -based matrix and $x \in \mathbb{R}^{|\Omega|}$ a d_H -based vector (with respect to Ω). Then $y = Hx$ is also a d_H -based vector.*

Proof. We define for $0 \leq k \leq n$ the set $\Gamma_k := \{i \mid 1 \leq i \leq |\Omega| \wedge d_H(i, 1) = k\}$. Moreover, we denote by φ_H and φ_x the function associated with H and x , respectively. With this notation we obtain for $y = Hx$ the element-wise formula

$$y(i) = \sum_{k=1}^{|\Omega|} H(i, k)x(k) = \sum_{k=0}^n \varphi_x(k) \cdot \left(\sum_{j \in \Gamma_k} H(i, j) \right).$$

In order to complete the proof, we have to show that

$$\sum_{j \in \Gamma_k} H(i, j) = \sum_{j \in \Gamma_k} \varphi_H(d_H(i, j)) \quad (4.23)$$

solely depends on $d_H(i, 1)$ and not on the particular value of i .

Recall that the set Γ_k contains all indices whose string representation with respect to Ω contains exactly $n - k$ zeros. Therefore, the set Γ_k is invariant under “digit-permutations”, i. e., permutations which shuffles the elements in the string representation of $j \in \Gamma_k$. Similarly, the set Γ_k is closed under shifts with a fixed element, i. e., let $s \in \Gamma_k$ be fixed then $j + s \in \Gamma_k$ for all $j \in \Gamma_k$, where the addition has to be understood element-wise and modulo c . By definition, the string representations of arbitrary indices $1 \leq i, i' \leq |\Omega|$ with $d_H(i, 1) = d_H(i', 1)$ also contain the same number of zeros. Thus, we can for given indices i, i' with $d_H(i, 1) = d_H(i', 1) = k$ always find a digit-permutation π and a shift $s \in \Gamma_k$ such that $d_H(i, j) = d_H(i', \pi(j) + s)$ for all $j \in \Gamma_k$. Therefore, since Γ_k is closed under digit-permutations and shifts, the expression (4.23) is indeed solely dependent on $d_H(i, 1)$ and not on the particular value of i . \square

Corollary 4.28. *Let $c, n \in \mathbb{N}$ be arbitrary, then $\mathcal{H}(\mathbb{Z}_c^n)$ is closed under multiplication.*

Proof. We proceed by induction on n . The base case $n = 0$, i. e., $\mathbb{Z}_c^0 = \mathbb{R}$ is trivial. Assume that the claim holds for n . Then, by [Lemma 4.20](#), we conclude that for all $H, G \in \mathcal{H}(\mathbb{Z}_c^{n+1})$

$$H \cdot G = \begin{pmatrix} C & D & \cdots & D \\ D & C & \ddots & \vdots \\ \vdots & \ddots & \ddots & D \\ D & \cdots & D & C \end{pmatrix} \begin{pmatrix} C' & D' & \cdots & D' \\ D' & C' & \ddots & \vdots \\ \vdots & \ddots & \ddots & D' \\ D' & \cdots & D' & C' \end{pmatrix} = \begin{pmatrix} E & F & \cdots & F \\ F & E & \ddots & \vdots \\ \vdots & \ddots & \ddots & F \\ F & \cdots & F & E \end{pmatrix}$$

with $C, D, C', D' \in \mathcal{H}(\mathbb{Z}_c^n)$, $E = CC' + (c - 1)DD'$ and $F = CD' + C'D + (c - 2)DD'$. This means $H \cdot G$ fulfills property (i) in [Lemma 4.20](#). Property (ii) of the latter follows directly from [Lemma 4.27](#), since the first column of a d_H -based matrix fully determines its associated function as well as the associated functions of its sub-blocks. Thus, by [Lemma 4.20](#), the product $H \cdot G$ is an element of $\mathcal{H}(\mathbb{Z}_c^{n+1})$ for all $H, G \in \mathcal{H}(\mathbb{Z}_c^{n+1})$. \square

Theorem 4.29 (cf. [Theorem 4.10](#)). *Let $\Omega = \mathbb{Z}_c^n$ with $c, n \in \mathbb{N}$. With the usual matrix addition and multiplication $(\mathcal{H}(\Omega), +, \cdot)$ is a commutative ring with identity. In particular, the zero matrix is the additive identity and $\mathcal{I}_{|\Omega|}$ is the multiplicative identity of $(\mathcal{H}(\Omega), +, \cdot)$.*

Proof. Use the proof of [Theorem 4.10](#), where the fact that $\mathcal{H}(\Omega)$ is closed under multiplication follows by [Corollary 4.28](#). \square

We refrain in the following from generalizing the approximation results stated in [Section 4.2.3](#). The solutions to [Problem 4.3](#) and [Problem 4.5](#) can be straightforwardly generalized to arbitrary signatures, whereas this does not seem to be possible for [Problem 4.4](#), since its solution (cf. [Theorem 4.17](#)) required [Theorem 4.8](#) which does not hold for more general signatures. Moreover, we will solely consider the binary case in our practical considerations in [Chapter 5](#).

4.3.3. Fast Algorithms

In principle we can straightforwardly employ the same strategies to obtain fast algorithms as in the binary case (cf. [Section 4.2.2](#)). The crucial property which is not yet clear for matrices $H \in \mathcal{H}(\Omega)$ is how to efficiently compute the matrix vector products with their respective matrix of eigenvectors V_Ω . As a prerequisite, we study how to efficiently compute $V_n v$ with $v \in \mathbb{R}^n$ (cf. [Definition 4.8](#)).

For a fast matrix vector product with V_n it is important to understand how its rows are structured (see also [Example 4.10](#)). In particular, we can express each row $V_n(i, :)$ with $2 \leq i \leq n$ in terms of the first row $V_n(1, :)$ by

$$V_n(i, :) = (V_n(1, 1), \underbrace{0, \dots, 0}_{i-2 \text{ times}}, -(i-1)V_n(1, i), V_n(1, (i+1):n)).$$

Thus, the product $V_n(i, :)v$ consists of three parts

$$V_n(i, :)x = V_n(1, 1)v(1) - (i-1)V_n(1, i)v(i) + V_n(1, (i+1):n)v((i+1):n).$$

By precomputing $s(i) = V_n(1, (i+1):n)v((i+1):n)$, this formula can be evaluated in constant time and therefore, the whole matrix vector product can be computed in linear time. [Algorithm 4.3](#) gives complete algorithm for efficiently computing $V_n v$ (in place) using the aforementioned ideas.

By having [Algorithm 4.3](#) for efficiently computing $V_n v$ at hand, the derivation of a fast routine for computing $V_\Omega v$ is straightforward. In particular, we exploit the Kronecker product representation of V_Ω in order to derive a fast algorithm (cf. [Section 3.2](#)). Consider signatures of the form $\Omega_n = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \dots \times \mathbb{Z}_{a_1}$, then we again use the idea of factorizing V_{Ω_n} to obtain

$$V_{\Omega_n} = (\mathcal{I}_{a_n} \otimes V_{\Omega_{n-1}})(V_{a_n} \otimes \mathcal{I}_{|\Omega_{n-1}|})$$

By the occurring parallel and vector Kronecker products (cf. [Definition 3.3](#) and [Definition 3.4](#)) an algorithm for computing $V_\Omega v$ consists of the following two steps:

Algorithm 4.3 Linear time computation of the matrix vector product $V_n v$

Input: $v \in \mathbb{R}^n$
Output: $V_n v$ (in v)

- 1: $s(1) \leftarrow V_n(1, 1) \cdot v(1)$
- 2: **for** $i \leftarrow 2$ **to** n **do**
- 3: $s(i) \leftarrow s(i - 1) + V_n(1, i) \cdot v(i)$
- 4: **end for**
- 5: **for** $i \leftarrow 1$ **to** n **do**
- 6: $v(i) \leftarrow s(1) - (i - 1) \cdot V_n(1, i) \cdot v(i) + (s(n) - s(i))$
- 7: **end for**

- (i). We compute $(V_{a_n} \otimes \mathcal{I}_{|\Omega_{n-1}|})v$, which means that we perform the multiplication with V_{a_n} in a vectorized fashion with vectors of length $|\Omega_{n-1}|$ instead of scalars. Nevertheless, we can employ [Algorithm 4.3](#) (in a vectorized form).
- (ii). The parallel Kronecker product $(\mathcal{I}_{a_n} \otimes V_{\Omega_{n-1}})$ simply states that we have to apply $V_{\Omega_{n-1}}$ on the a_n blocks of size $|\Omega_{n-1}|$ of the vector obtained in step (i).

This recursion obviously stops after n steps and can be naturally implemented in an iterative fashion (cf. [Section 3.2](#)). In particular, [Algorithm 4.4](#) depicts a complete implementation of the sketched Kronecker product driven algorithm.

Algorithm 4.4 Efficient computation of the matrix vector product $V_\Omega v$

Input: $\Omega = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \dots \times \mathbb{Z}_{a_1}$, $v \in \mathbb{R}^{|\Omega|}$
Output: $V_\Omega v$ (in v)

- 1: $b \leftarrow |\Omega|$
- 2: **for** $i \leftarrow n$ **down to** 1 **do**
- 3: $b \leftarrow b/a_i$
- 4: **for** $j \leftarrow 1$ **to** $|\Omega|$ **by** ba_i **do**
- 5: $s(1, :) \leftarrow V_n(1, 1) \cdot v(j:(j + b))$ $\triangleright s \in \mathbb{R}^{a_i \times b}$
- 6: **for** $i \leftarrow 2$ **to** a_i **do**
- 7: $s(i, :) \leftarrow s(i - 1) + V_n(1, i) \cdot v((j + (i - 1)b):(j + ib))$
- 8: **end for**
- 9: **for** $i \leftarrow 1$ **to** a_i **do**
- 10: $v((j + (i - 1)b):(j + ib)) \leftarrow s(1, :) + (s(n, :) - s(i, :))$
- 11: $- (i - 1) \cdot V_n(1, i) \cdot v((j + (i - 1)b):(j + ib))$
- 12: **end for**
- 13: **end for**
- 14: **end for**

Theorem 4.30. Let $\Omega = \mathbb{Z}_{a_n} \times \mathbb{Z}_{a_{n-1}} \times \dots \times \mathbb{Z}_{a_1}$ and $v \in \mathbb{R}^{|\Omega|}$ be arbitrary. Then, [Algorithm 4.4](#) computes the matrix vector product $V_\Omega v$ in $\Theta(n|\Omega|)$ time.

Proof. The for-loop ranging from lines 4–13 iterates $|\Omega|/(ba_i)$ times. Moreover, its body has an effort of $\Theta(ba_i)$, since simple vector operations with vectors of size b are

performed. Thus, this loop contributes—independent of the outer index i —an effort of $\Theta(|\Omega|)$. Therefore, the overall complexity is $\Theta(n|\Omega|)$. \square

[Theorem 4.30](#) particularly emphasizes, that the concrete shape of the signature Ω does not lead to problems from a computational point of view. Contrary, consider signatures $\Omega = \mathbb{Z}_c^n$, then $n|\Omega| = \log_c(|\Omega|)|\Omega|$, i. e., the algorithms become more efficient with larger alphabets. The correctness follows immediately by the considerations above. Moreover, note that lines 5–11 represent a vectorized version of [Algorithm 4.3](#).

5. Applications of Hamming Distance-based Matrices

In the following sections we consider several applications of the results on d_H -based matrices we developed in [Chapter 4](#). First and foremost we apply these results in several ways to the quasispecies model. In particular, we utilize d_H -based matrices in order to generalize the computationally tractable model assumptions and to design efficient preconditioners. Moreover, we also refer to applications beyond the quasispecies model. All experimental results in this chapter were obtained with MATLAB 7.11.0 (R2010b).

5.1. Generalizations in the Quasispecies Model

Recall (cf. [Section 1.3](#)) that computing the quasispecies requires the computation of the dominating eigenvector of a matrix $W = QF \in \mathbb{R}^{N \times N}$, where $N = 2^\nu$. The matrix Q describes thereby the underlying mutation model, whereas F consists of the fitness values describing the constitution of the different species. In general, the underlying ODE model (see [\(1.1\)](#)) poses the requirements that F is a diagonal matrix with strictly positive diagonal elements, whereas the matrix $Q \geq 0$ has to be nonnegative and column-stochastic, i. e., all columns of Q have to sum up to one. In contrast to this mathematical requirements for Q , commonly the so-called uniform error rate model [[Eigen, 1971](#)] is used as a mutation model. In this model a uniform mutation error rate $0 \leq p < 1$ is assumed, which leads to a matrix $Q \in \mathbb{R}^{N \times N}$ with elements

$$q_{i,j} = (1 - p)^{d_H(i,j)} \cdot p^{\nu - d_H(i,j)}. \quad (5.1)$$

Besides that this model is used throughout the theoretical literature (see [Schuster \[2008, 2011\]](#) and the references therein), all efficient computations are limited to this case, since the uniform error rate model is the only known model which leads to a Kronecker product representation of Q and consequently, to efficient solvers [[Niederbrucker and Gansterer, 2011a](#)].

Clearly, there is a huge gap between the model requirement that Q is column stochastic and the special case that Q is given by [\(5.1\)](#). The only practically motivated restriction we have to bear in mind is that the mutation error rate is modeled to be independent from the particular position a mutation takes place in the RNA sequence. Consequently, aiming at a simple Kronecker product representation of Q always ends up in the uniform error rate model. In contrast to that, as a substantial generalization, one can assume that the mutation matrix Q is an arbitrary d_H -based matrix, which leads to a fairly general modeling of the mutation process with all of the required properties. As the results in [Section 4.2.2](#) and [Section 4.3.3](#) depict, d_H -based matrices allow de-

5. Applications of Hamming Distance-based Matrices

spite their generality efficient algorithms of the same complexity as Kronecker product representations of Q allow.

Example 5.1. We consider two potential mutation matrices Q_1 and Q_2 in the case of a chain length of $\nu = 2$. The matrix Q_1 is based on the uniform error rate model with $p = 0.01$ and $Q_2 \in \mathcal{H}(\mathbb{Z}_2^2)$ is a particular example of a d_H -based matrix which does not allow for a simple Kronecker product representation.

$$Q_1 = \begin{pmatrix} 0.81 & 0.09 & 0.09 & 0.01 \\ 0.09 & 0.81 & 0.01 & 0.09 \\ 0.09 & 0.01 & 0.81 & 0.09 \\ 0.01 & 0.09 & 0.09 & 0.81 \end{pmatrix} = \bigotimes_{i=1}^2 \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} 0.90 & 0.04 & 0.04 & 0.01 \\ 0.04 & 0.90 & 0.01 & 0.04 \\ 0.04 & 0.01 & 0.90 & 0.04 \\ 0.01 & 0.04 & 0.04 & 0.90 \end{pmatrix} \neq \bigotimes_{i=1}^2 \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$$

Another example of a mutation model which can be expressed in term of d_H -based matrices would be to choose the mutation error rate exponentially decaying, i. e., choosing the associated function of Q as $\varphi_Q(d) = e^{-d}/c$ where c is a normalization constant. //

Generally, it can be easily seen from the structure of d_H -based matrices (cf. [Lemma 4.3](#) and [Lemma 4.20](#)) that they have constant column sums. This means, up to normalization any Hamming distance-based matrix, i. e., any model function of the form $\varphi(d_H(i, j))$, can be used within the quasispecies model. A particularly useful application of this observation is that we can also approximate a given matrix of mutation probabilities.

Example 5.2. Assume we are given an arbitrary column stochastic mutation matrix Q obtained, e. g., by practical observations or expert knowledge. Then [Theorem 4.16](#) allows to approximate Q by a d_H -based matrix \tilde{Q} in order to enable the investigation of (an approximation) of Q within the quasispecies model.

$$Q = \begin{pmatrix} 0.460 & 0.210 & 0.263 & 0.112 \\ 0.214 & 0.362 & 0.082 & 0.241 \\ 0.256 & 0.159 & 0.416 & 0.303 \\ 0.070 & 0.270 & 0.240 & 0.344 \end{pmatrix}, \quad \tilde{Q} = \begin{pmatrix} 0.3955 & 0.2496 & 0.2496 & 0.1058 \\ 0.2496 & 0.3955 & 0.1058 & 0.2496 \\ 0.2496 & 0.1058 & 0.3955 & 0.2496 \\ 0.1058 & 0.2496 & 0.2496 & 0.3955 \end{pmatrix}$$

It is important to note that for an arbitrary column stochastic matrix the optimal solution of the approximation problem considered in [Theorem 4.16](#) is a column stochastic matrix as well, which straightforwardly follows from its definition. //

Another aspect the results from [Chapter 4](#) allow to generalize is that we can not only consider binary alphabets but also larger alphabets, e. g., the usual four letter RNA alphabet [[Schuster, 2011](#)]. Even more generally, certainly constraint sequences where different elements of the sequences are taken from different alphabets can be considered. This might be of interest whenever for certain positions in the RNA sequence additional a priori knowledge is available.

Summarizing, for relatively low costs compared to the existing methods we can cover substantially more general scenarios by applying d_H -based matrices. In particular, this holds in terms of the mutation model as well as in terms of the manageable alphabets.

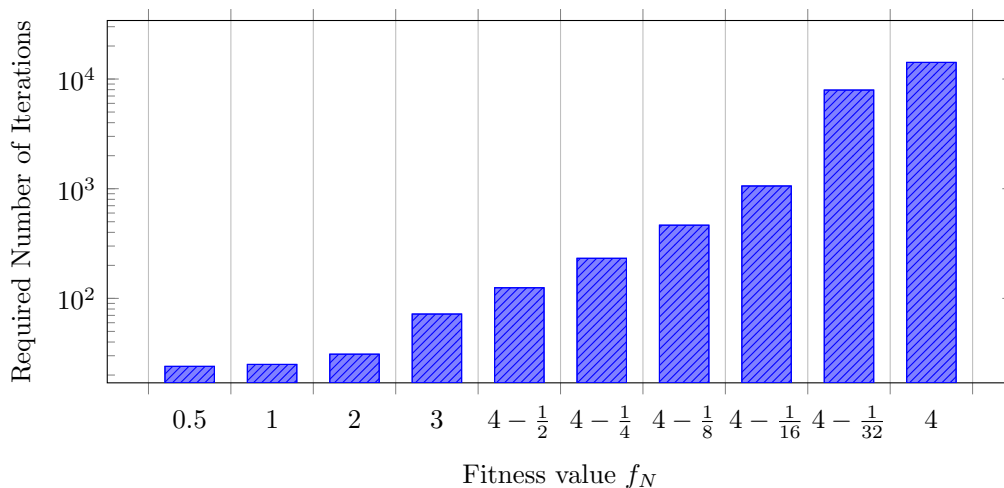


Figure 5.1.: Iterations required to approximate the dominating eigenvector up to $\varepsilon = 10^{-12}$ with the power method for different landscapes (choices of f_N).

5.2. Preconditioning and Shift-and-Invert Methods

As we have seen in [Section 2.3](#), with an efficient matrix vector multiplication routine at hand, it is very easy to compute the dominating eigenpair of a matrix via the power iteration. The crucial aspect in this context is that the eigenvector approximate delivered by the power iteration converges linearly with rate λ_2/λ_1 where λ_1 and λ_2 denote as before the two largest eigenvalues (in magnitude) of the respective matrix. This means that the convergence behavior of the power iteration depends in a particularly simple way on the spectral properties of the input matrix.

In case of the matrix $W = QF$ occurring in the quasispecies model, the spectral properties are mostly determined by the fitness landscape F . In case of the commonly considered single peak fitness landscape [[Schuster, 2011](#)] (see also [Figure 1.1](#)), the master sequence has a somewhat dominating eigenvalue which also leads to a well-separated dominating eigenvalue [[Niederbrucker and Gansterer, 2011a](#)]. On the other hand, other—especially practically relevant—fitness landscapes will in general not show a good separation of the dominating eigenvalue. Since the spectrum of W cannot be easily determined also the choice of the landscapes we consider requires attention.

In order to illustrate the effects of a badly separated spectrum we consider the following model landscape: Irrespective of the particular problem dimension N we consider a “double peak landscape”, where $f_1 = 4$, $f_N = 4 - \varepsilon$ and for all $1 < i < N$ the fitness value f_i is drawn uniformly from the unit interval $[0, 1]$. In principle, the idea behind this double peak fitness landscape is to introduce a certain symmetry into the problem which leads to two (almost) equally strong regimes and therefore, also to a bad separation of the dominating eigenvalue. [Figure 5.1](#) shows how certain choices of f_N influence the separation of the dominating eigenvalue. We can see that the number of iterations required to converge differs by several orders of magnitude. Therefore, also the time to converge increases by several orders of magnitude. In the following, we show how to solve also such more subtle problems with bad separation, by utilizing shift-and-invert methods, i. e., inverse iteration and Rayleigh quotient iteration (see [Section 2.3](#)).

5.2.1. Design of Preconditioners

Shift-and-invert methods require the solution of a linear system in each iteration. Moreover, the desired fast convergence of the eigenvalue iteration requires well chosen shifts, which eventually lead to ill-conditioned systems. While this problem is by far not as critical as it seems to be at a first glance [Trefethen and Bau, 1997, Exercise 27.5], the employed iterative solvers still greatly benefit from an appropriate preconditioning.

In its usual form the eigenvalue problem at hand involves a matrix $W = QF$ with a d_H -based matrix Q and a diagonal matrix F . For every d_H -based matrix H it holds by definition that $H - \mu\mathcal{I}$ is also a d_H -based matrix. This motivates that a good approximation of $QF - \mu\mathcal{I}$ could potentially be given in terms of a product HD with a d_H -based matrix H and a diagonal matrix D . Moreover, Corollary 4.19 shows how to determine H and D optimally in order to minimize $\|(QF - \mu\mathcal{I}) - HD\|_F$. Besides the hope for a reasonable approximation quality, it is clear that the intended preconditioner HD can be efficiently inverted since H is a d_H -based matrix (cf. Corollary 4.11) and D a diagonal matrix. The remaining problem is the concrete computation of H and D which requires the solution of a large rank-1 approximation problem involving a subtle reordering $\mathcal{CR}(QF - \mu\mathcal{I})$ of the matrix $QF - \mu\mathcal{I}$. Later, in Section 5.2.2 we will see that due to the specific structure of Q and F , this particular rank-1 approximation can be solved surprisingly efficient by employing the SVD Lanczos process (cf. Section 2.3.2).

Earlier in Section 1.3 we have seen that we can easily consider different formulations of the eigenvalue problem at hand, where the particular choice of the problem formulation results in different properties, e. g., symmetry vs. no symmetry. As the next lemma shows, finding a preconditioner for the case $W = QF$ is sufficient to obtain also analogous preconditioners in case of other problem formulations.

Lemma 5.1. *Let $W = QF$ be given from the quasispecies model, μ a real shift and P_R be a preconditioner targeting $QF - \mu\mathcal{I}$. Moreover, let $P_S := F^{\frac{1}{2}}P_R^{-1}F^{-\frac{1}{2}}$ and $P_L := FP_R^{-1}F^{-1}$. Then the matrices*

$$P_R^{-1}(QF - \mu\mathcal{I}), \quad P_S^{-1}(F^{\frac{1}{2}}QF^{\frac{1}{2}} - \mu\mathcal{I}), \quad P_L^{-1}(FQ - \mu\mathcal{I}) \quad (5.2)$$

are all similar, i. e., the matrices in (5.2) have the same set of eigenvalue and hence, also the same condition number with respect to the Euclidean norm.

Proof. We have to show that all of the matrices in (5.2) can be written in the form $M(P_R^{-1}(QF - \mu\mathcal{I}))M^{-1}$, where M can be an arbitrary invertible matrix. In particular it follows straightforwardly that

$$\begin{aligned} P_S^{-1}(F^{\frac{1}{2}}QF^{\frac{1}{2}} - \mu\mathcal{I}) &= F^{\frac{1}{2}}P^{-1}F^{-\frac{1}{2}}(F^{\frac{1}{2}}QF^{\frac{1}{2}} - \mu\mathcal{I}) = F^{\frac{1}{2}}(P_R^{-1}(QF - \mu\mathcal{I}))F^{-\frac{1}{2}}, \\ P_L^{-1}(FQ - \mu\mathcal{I}) &= FP^{-1}F^{-1}(FQ - \mu\mathcal{I}) = F(P_R^{-1}(QF - \mu\mathcal{I}))F^{-1}, \end{aligned}$$

which proves the similarity of the matrices in (5.2). \square

Due to the similarity of the different problem formulations, we consider in the following experiments only the standard formulation $W = QF$. Moreover, the different formulations also showed experimentally the expected analogous results. Before we study the behavior of the derived preconditioners in practice, we first investigate their computation via the SVD Lanczos process.

5.2.2. SVD Lanczos

As we have seen in [Example 2.1](#), the best rank-1 approximation of a matrix is given by its dominating singular vectors. In order to avoid the huge effort of computing the singular vectors via the SVD we pointed out the SVD Lanczos as an efficient alternative. First and foremost this method requires a fast matrix vector multiplication routine for the considered matrix and its transpose. At the first glance it seems to be an involved problem to provide such an efficient routine due to the subtle structure of the reordering $\overline{\mathcal{CR}}$ (see [Definition 4.6](#)). But as the following example depicts, this reordering matches well with the structure inherent in the matrices considered in the quasispecies model.

Example 5.3. Consider a chain length of $\nu = 2$ but an otherwise arbitrary mutation matrix Q and fitness landscape F . Then the shifted matrix $QF - \mu\mathcal{I}$ is structured like

$$\begin{aligned} QF - \mu\mathcal{I} &= \begin{pmatrix} q_{1,1}f_1 - \mu & q_{1,2}f_2 & q_{1,3}f_3 & q_{1,4}f_4 \\ q_{2,1}f_1 & q_{2,2}f_2 - \mu & q_{2,3}f_3 & q_{2,4}f_4 \\ q_{3,1}f_1 & q_{3,2}f_2 & q_{3,3}f_3 - \mu & q_{3,4}f_4 \\ q_{4,1}f_1 & q_{4,2}f_2 & q_{4,3}f_3 & q_{4,4}f_4 - \mu \end{pmatrix} \\ &= \begin{pmatrix} \varphi(0)f_1 - \mu & \varphi(1)f_2 & \varphi(1)f_3 & \varphi(2)f_4 \\ \varphi(1)f_1 & \varphi(0)f_2 - \mu & \varphi(2)f_3 & \varphi(1)f_4 \\ \varphi(1)f_1 & \varphi(2)f_2 & \varphi(0)f_3 - \mu & \varphi(1)f_4 \\ \varphi(2)f_1 & \varphi(1)f_2 & \varphi(1)f_3 & \varphi(0)f_4 - \mu \end{pmatrix}, \end{aligned}$$

where φ denotes the associated function of the d_H -based matrix Q . Applying the column reordering \mathcal{CR} (see [Definition 4.6](#)) to the matrix $QF - \mu\mathcal{I}$ leads to

$$\mathcal{CR}(QF - \mu\mathcal{I}) = \begin{pmatrix} \varphi(0)f_1 - \mu & \varphi(0)f_2 - \mu & \varphi(0)f_3 - \mu & \varphi(0)f_4 - \mu \\ \varphi(1)f_1 & \varphi(1)f_2 & \varphi(1)f_3 & \varphi(1)f_4 \\ \varphi(1)f_1 & \varphi(1)f_2 & \varphi(1)f_3 & \varphi(1)f_4 \\ \varphi(2)f_1 & \varphi(2)f_2 & \varphi(2)f_3 & \varphi(2)f_4 \end{pmatrix}.$$

Moreover the condensed reordering $\overline{\mathcal{CR}}$ (see [\(4.15\)](#)) is given by

$$\begin{aligned} \overline{\mathcal{CR}}(QF - \mu\mathcal{I})x &= \begin{pmatrix} \varphi(0)f_1 - \mu & \varphi(0)f_2 - \mu & \varphi(0)f_3 - \mu & \varphi(0)f_4 - \mu \\ \varphi(1)f_1 & \varphi(1)f_2 & \varphi(1)f_3 & \varphi(1)f_4 \\ \varphi(2)f_1 & \varphi(2)f_2 & \varphi(2)f_3 & \varphi(2)f_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \\ &= (\varphi(0), \varphi(1), \varphi(2))^T \cdot \sum_{i=1}^4 f_i x_i - (\mu, 0, 0)^T \cdot \sum_{i=1}^4 x_i. \end{aligned}$$

These considerations exemplify a particularly good match between the structure of $QF - \mu\mathcal{I}$ and the reordering $\overline{\mathcal{CR}}$. //

By generalizing the representation we derived in [Example 5.3](#) for $\nu = 2$, we obtain for general chain length ν the formula

$$\overline{\mathcal{CR}}(QF - \mu\mathcal{I})x = (\varphi(0), \varphi(1), \dots, \varphi(\nu))^T \cdot \sum_{i=1}^N f_i x_i - (\mu, 0, \dots, 0)^T \cdot \sum_{i=1}^N x_i.$$

5. Applications of Hamming Distance-based Matrices

Therefore, the matrix vector product required for the SVD Lanczos process can be computed in $\Theta(N)$ time. Analogously, a linear time matrix vector routine can be derived for the transpose of $\mathcal{CR}(QF - \mu\mathcal{I})$.

So far we know that a single iteration within the SVD Lanczos process can be efficiently implemented. The more subtle issue is to determine how many iterations are required in order to obtain a reasonable accurate approximation. In principle, the SVD Lanczos has (in exact arithmetic) an exact stopping criterion, whereas in practice less stringent criteria are required (cf. Algorithm 2.6). In particular, we use a pre-determined number of maximal iteration j_{\max} . In the absence of theoretical results, we depict in Figure 5.2 the approximation accuracy obtained by different choices of j_{\max} (averaged over 10 experiments per parameter constellation).

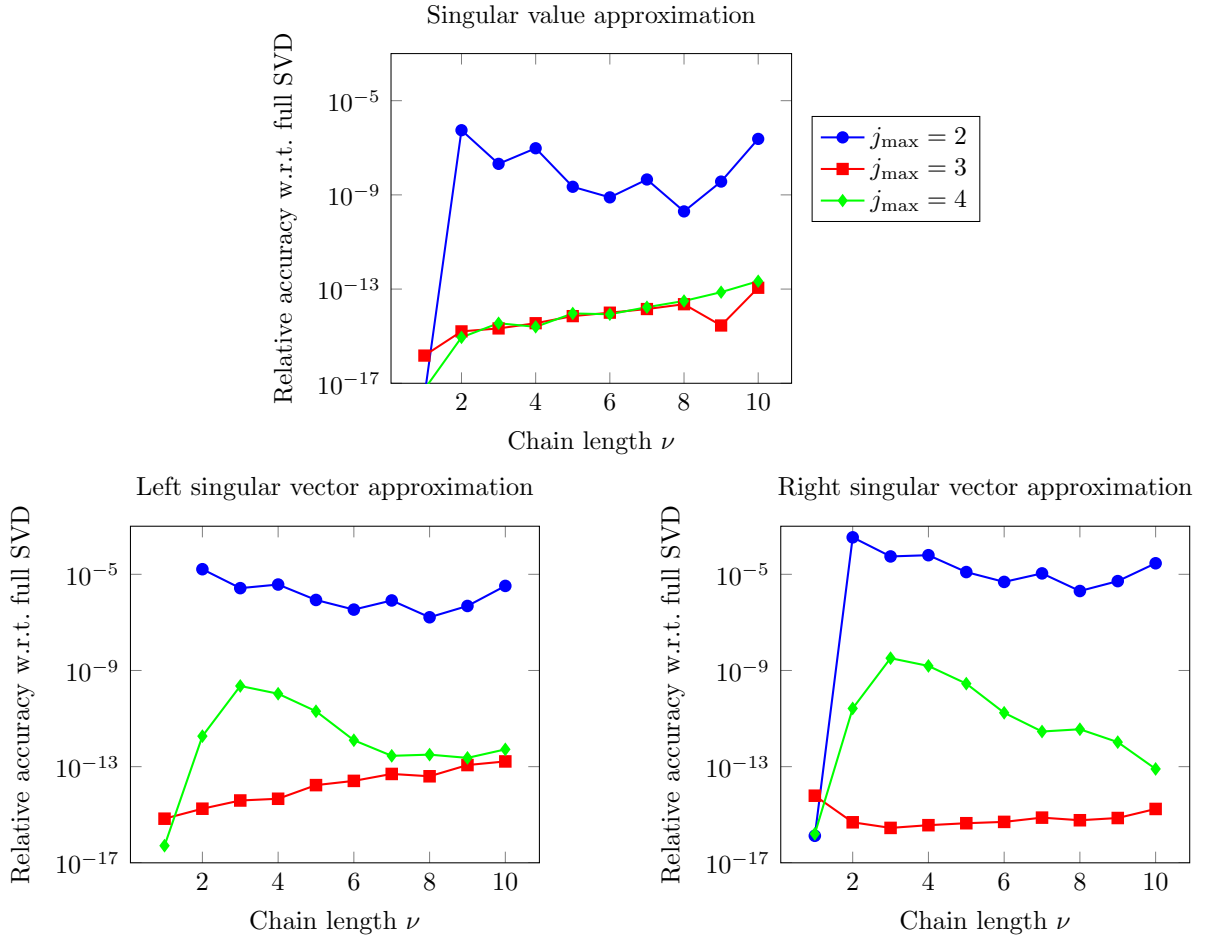


Figure 5.2.: Accuracy of the SVD Lanczos process for different numbers of maximal iterations j_{\max} , compared to the “correct” results obtained via the SVD.

As it can be seen in Figure 5.2, highly accurate results can be obtained with surprisingly little iterations. Following the results of Figure 5.2, we consider in all following experiments $j_{\max} = 3$. Complementary to the previously shown accuracy results, Figure 5.3 shows a performance comparison between computing the dominating singular vectors via the SVD and the SVD Lanczos process for different choices of j_{\max} (averaged over 10 experiments per parameter constellation). As expected, the SVD Lanczos pro-

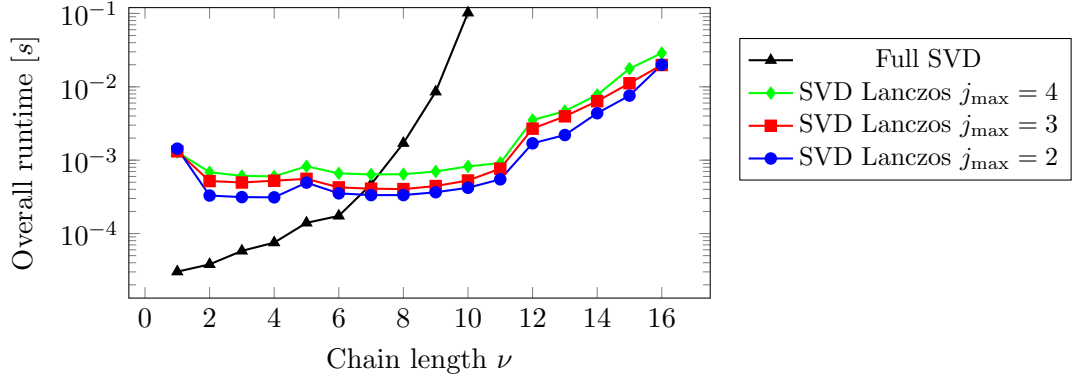


Figure 5.3.: Performance of the SVD Lanczos process for different number of iterations j_{\max} compared to the performance of the dominating singular value/vectors via the SVD.

cess shows a superior performance and scaling behavior. Additionally, its effort decently scales with the maximal number of iterations j_{\max} . Moreover, Figure 5.3 shows that the setup costs of the intended preconditioner are very low even for large chain lengths.

Summarizing, so far we obtained a structured preconditioner which can be efficiently inverted and we exemplified that its computation is very cheap. The missing property for being a “good” preconditioner is a (substantial) reduction of the number of iterations required to converge within a certain target accuracy.

5.2.3. Application to Shift-and-Invert Methods

In Section 2.3 we saw that the inverse iteration as well as the Rayleigh quotient iteration can be used to efficiently compute the dominating eigenpair of a matrix whose dominating eigenvalue is not well separated. Since the costs of iteratively solving a linear system in each iteration raises substantial costs, these methods only provide benefits over the simple power iteration in case the latter requires very large amounts of iterations. As we saw in Figure 5.1, certain landscapes easily lead to thousands of iterations and hence, power iteration-based approaches are very slow in these cases.

As a first step, we consider the effect of the previously introduced preconditioner detached from a particular shift-and-invert solver. Concretely, suppose we are given $W = QF$ where F represents a double peak landscape with $f_1 = f_N = 4$ and let λ_1 denote its dominating eigenvalue. Then we investigate the solution of $(W - 1.001\lambda_1)x = b$, where the right hand side b is randomly chosen from the uniform distribution in $[0, 1]^N$. Figure 5.4 depicts the number of BiCGSTAB iterations required to converge with and without preconditioning. We see that despite the overall number of required iterations is relatively small, still a considerable effect of the preconditioning is visible. Obviously, the results in Figure 5.4 are of a theoretical nature since only the number of required iterations is compared, whereas the overhead caused by the preconditioning is hidden. Hence, what really matters is overall execution time. Consequently, we show in Figure 5.5 the overall runtimes corresponding to the experiments shown in Figure 5.4. Moreover, we depict the ratio between the computation of the preconditioner and the overall time required to solve the system. The efficient implementation of the

5. Applications of Hamming Distance-based Matrices

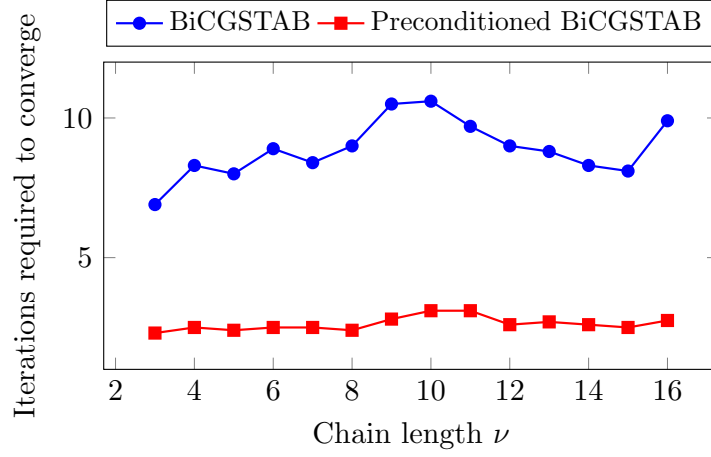


Figure 5.4.: Comparison between BiCGSTAB with and without preconditioning. We compare the number of iterations required to reach a target accuracy of $\varepsilon = 10^{-12}$

preconditioner pointed out in Figure 5.5 can be obtained as follows: First, recall that in the standard case we apply a preconditioner HD with a d_H -based matrix H and a diagonal matrix D to the matrix $QF - \mu\mathcal{I}$. Then we obtain by the eigendecomposition $H = W_\nu\Lambda W_\nu$ (cf. Lemma 4.4) that

$$D^{-1}H^{-1}(QF - \mu\mathcal{I}) = D^{-1}W_\nu\Lambda_H^{-1}W_\nu(W_\nu\Lambda_Q^{-1}W_\nu F - \mu\mathcal{I}) \quad (5.3)$$

$$= D^{-1}W_\nu\Lambda_H^{-1}(\Lambda_Q^{-1}W_\nu F - \mu W_\nu). \quad (5.4)$$

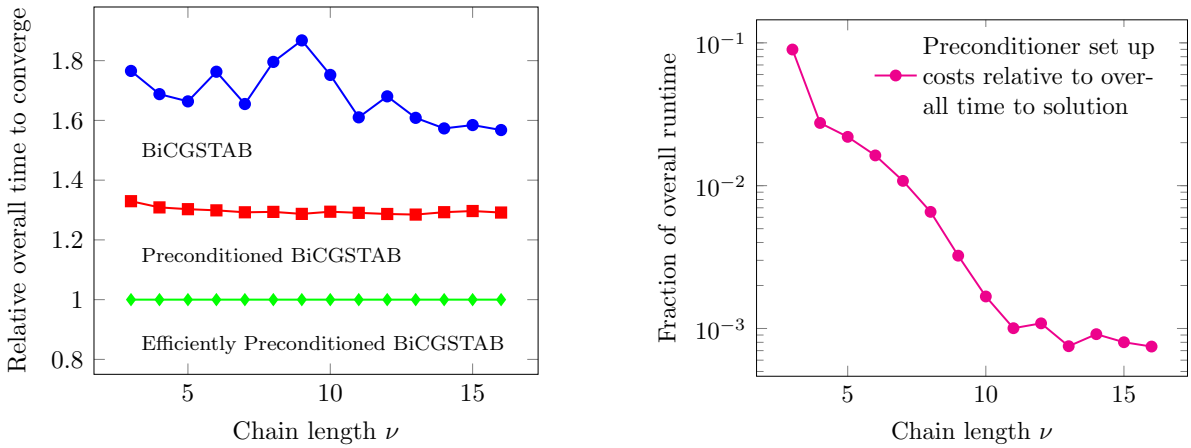


Figure 5.5.: Overall performance gains of preconditioned BiCGSTAB over BiCGSTAB without preconditioning. Besides a straightforward implementation of the preconditioner we also consider a slightly more efficient implementation which requires only three instead of four $\mathcal{O}(N \log N)$ operations.

The computationally dominating part in the application of the preconditioner are the matrix vector products with W_ν , since all other matrices are diagonal. Therefore,

applying the preconditioner in the form of (5.4) is advantageous over (5.3) because it saves one multiplication with W_ν .

What is left over is to depict the advantages of shift-and-invert methods over the power iteration in case of ill-conditioned problems. In particular, we compare these methods again in the case of a double peak landscape, i. e, a bad separation of the dominating eigenvalue. Finally, Figure 5.6 illustrates the performance gains with respect to the power iteration. In any case we see a substantial performance gain and observe

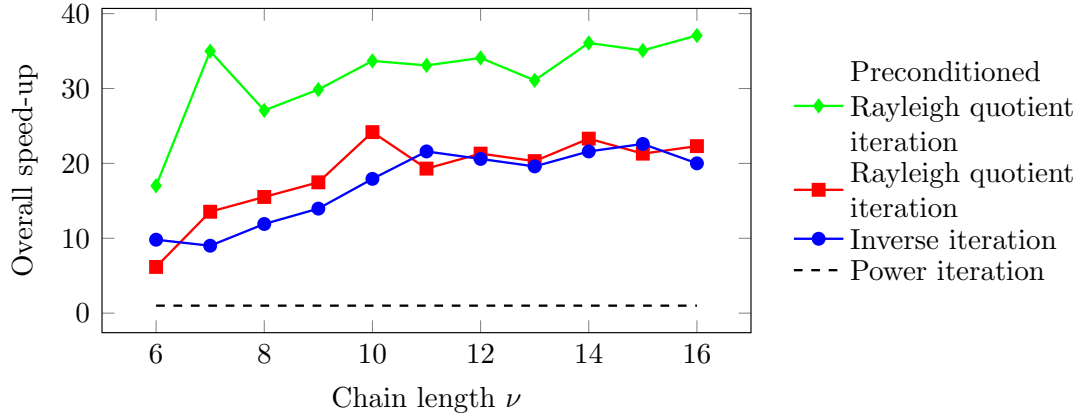


Figure 5.6.: Overall performance gain of different shift-and-invert methods compared to the power iteration. In case of the inverse iteration the shift is chosen as a coarse approximation of λ_1 computed via power iteration (the time spent with this initial step is taken into account).

that the Rayleigh quotient iteration utilizing a preconditioned solver performs best. In particular, all of the methods deliver performance gains of at least an order of magnitude. Beyond that, Figure 5.6 even indicates a growth of the gains with increasing chain length ν .

5.3. Further Applications

In the following, we consider three prototypical applications of the results about d_H -based matrices (cf. Chapter 4), which reach beyond the scope of the quasispecies model.

The Hamming distance is a widely applied distance measure between strings. Due to the string representation of the genetic code, the Hamming distance naturally appears at several places in (computational) biology. In particular, He, Petoukhov, and Ricci [2004] as well as He and Petoukhov [2010] derived matrix representations of the genetic code which are very closely related to what we called d_H -based matrices throughout this thesis. In many cases the matrices they consider are even d_H -based matrices. Moreover, they investigate (in terms of concrete examples) the structure and properties of these matrices. While their work has in general a lack of mathematical rigor and generality, it provides a variety of applications of the results we obtained in Chapter 4. More specifically, our results from Chapter 4 can be used to generalize and extend the observations for specific matrices which have been made by He, Petoukhov, and Ricci [2004] as well as He and Petoukhov [2010].

5. Applications of Hamming Distance-based Matrices

We have seen theoretically (cf. [Section 4.2.3](#)) as well as empirically (cf. [Section 5.2](#)) that several approximation problems involving d_H -based matrices can be efficiently solved. Therefore, due to the fact that they are easy to invert, d_H -based matrices have a certain potential for being a reasonable preconditioner for different classes of (data-sparse) matrices. For instance the classes of Toeplitz or block Toeplitz matrices are due to their constant diagonals closely related to d_H -base matrices and consequently, positive preconditioning effects can be anticipated. Due to their inherent “Kronecker structure”, d_H -based matrices can also be beneficial in the context of shifted Kronecker product systems [Moravitz Martin and Van Loan, 2007].

Last but not least we, consider the convergence analysis of distributed algorithms as a potential field of application. So-called gossip algorithms are very simple methods which allow—solely through nearest neighbor communication—to compute global quantities such as an average over a distributed network. The rigorous convergence analysis of these algorithms requires to investigate matrices with the same sparsity pattern as the adjacency matrix of the graph which models the underlying network [Boyd *et al.*, 2006]. Beyond that, information about the particular convergence rate which can be expected is given by certain spectral information [Boyd *et al.*, 2006]. Because (the adjacency matrices of) several common network topologies, e. g., hypercubes, are d_H -based matrices, our results on the structure and computation of the eigenvalues of d_H -based matrices can lead to more detailed insights into the convergence behavior of this kind of distributed algorithms.

6. Summary & Conclusions

We considered the problem of solving the eigenvalue problem occurring in the quasispecies model as a motivation throughout this thesis. More concretely, the objective was to compute the dominating eigenvector of certain structured large scale matrices. The treatment of the intended huge problem dimensions was only possible because the involved matrices are of a data-sparse nature. Our central aim was to extend prior work whose success is limited to matrices with a good separation of the dominating eigenvalue to the case of a bad separation—as it has to be expected in practice. In principle, we considered to employ shift-and-invert methods, which raise the need for solving a linear system in every iteration.

In order to meet the aforementioned goals we introduced the family of so-called Hamming distance-based matrices whose elements are characterized by the property that the (i, j) -th element solely depends on the Hamming distance between the indices i and j (appropriately interpreted as finite strings). We elaborated in great detail on these type of data-sparse matrices and developed a deep understanding of their structure. Moreover, we developed results on algebraic and algorithmic aspects as well as on the analytical solution of several approximation problems. Our interest in Hamming distance-based matrices was twofold: On the one hand, they allowed to substantially generalize the computationally tractable model assumptions in the quasispecies model. On the other hand, more importantly, results on the analytical solution of certain approximation problems involving Hamming distance-based matrices led to the design of an effective preconditioner for the problems at hand.

In particular, we showed that we can solve this approximation problem involving Hamming-distance-based matrices by reducing it to the problem of rank-1 approximation. Solving rank-1 approximation at a large scale is in principle a tricky problem but by employing the Lanczos SVD process it turned out that the particular rank-1 approximation problem we considered can be efficiently solved at an extreme dimensions as well. Notably, the (relative) overall effort for computing the resulting preconditioner even decreases with increasing dimension. Altogether, we observed for the resulting shift-and-invert solvers an overall performance gain of at least an order of magnitude compared to existing approaches.

Concluding, we saw that a comprehensive theory about the family of Hamming distance-based matrices allowed us to efficiently solve a large scale eigenvalue problem even in more general settings than it was previously the case. In the course of that, we could also derive an efficient data-sparse preconditioner for the linear systems occurring in the inner iteration of the employed shift-and-invert methods. Together, all these efforts led to substantial performance gains in the case of ill-conditioned problems.

A. Notation and Conventions

Notation A.1 (Matrices). For a matrix $A \in \mathbb{R}^{m \times n}$ we denote its entries by $a_{i,j}$, i. e.,

$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix}, \quad a_{i,j} \in \mathbb{R}.$$

Moreover, we also use $[A]_{i,j} = a_{i,j}$ to index the (i,j) -th entry of A . By $\text{tr}(A)$, $\text{rank}(A)$ and $\det(A)$, we denote the *trace*, *rank* and *determinant* of A , respectively. Moreover, in case A is a block matrix, we denote by $A_{i,j}$ its blocks, i. e.,

$$A = \begin{pmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{pmatrix}, \quad A_{i,j} \in \mathbb{R}^{m_i \times n_j}.$$

//

Notation A.2 (Explicit Matrix Multiplication). In case we explicitly compute a matrix product $P = A \cdot B \cdot C$ with three (or more) factors we use schemes of the form

$$P = A \cdot B \cdot C = \frac{A}{A} \left| \frac{B}{A \cdot B} \right| \frac{C}{A \cdot B \cdot C}$$

for the sake of a compact and easily readable exposition.

//

Notation A.3 (Diagonal Matrices). Let $A \in \mathbb{R}^{n \times n}$ be a square matrix, then $\text{diag}(A)$ denotes the vector formed by the entries on the main diagonal of A , i. e., $\text{diag}(A) = (a_{1,1}, a_{2,2}, \dots, a_{n,n})$. Conversely, if $v \in \mathbb{R}^n$ is a column or row vector, then

$$\text{diag}(v) = \text{diag}(v_1, v_2, \dots, v_n) = \begin{pmatrix} v_1 & 0 & \cdots & 0 \\ 0 & v_2 & 0 & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & v_n \end{pmatrix}.$$

Moreover, if $B_i \in \mathbb{R}^{m_i \times n_i}$ are matrices, then $\text{diag}(B_1, B_2, \dots, B_n)$ denotes the block diagonal matrix with the blocks B_i on its diagonal. By $\mathcal{I}_n \in \mathbb{R}^{n \times n}$ we denote the identity matrix in $\mathbb{R}^{n \times n}$, i. e., $\mathcal{I}_n = \text{diag}(1, 1, \dots, 1)$.

//

Notation A.4 (Submatrix Specification). For a matrix $A \in \mathbb{R}^{m \times n}$, and index sets $u \subseteq \{1, \dots, m\}$ and $v \subseteq \{1, \dots, n\}$, we denote by $A(u, v)$ the submatrix of A formed by picking the rows u and columns v of A , i. e.,

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix}, \quad A(\{1, 4\}, \{2, 3\}) = \begin{pmatrix} a_{1,2} & a_{1,3} \\ a_{4,2} & a_{4,3} \end{pmatrix}.$$

A. Notation and Conventions

We further use the *colon notation* to abbreviate index sets with consecutive indices. Concretely, for integers i and j , we denote the set $\{i, i + 1, \dots, j - 1, j\}$ by $i:j$. More generally, $i:k:j$ denotes the set $\{i, i + k, \dots, j - k, j\}$. To select all rows or columns from a matrix A we use $A(:, v)$ and $A(u, :)$, respectively. //

Notation A.5 (Matrices vs. Vectors). Let A be an arbitrary $m \times n$ matrix, then the *vectorization* of A is the column vector obtained by concatenating the columns of A .

$$\text{vec}(A) = \begin{pmatrix} A(:, 1) \\ A(:, 2) \\ \vdots \\ A(:, n) \end{pmatrix}$$

The natural inverse to the vectorization of a matrix is to construct a matrix out of a vector. Given a vector $x \in \mathbb{R}^{rc}$ we will denote by $x_{r \times c}$ the matrix

$$x_{r \times c} = \begin{pmatrix} x_1 & x_{r+1} & \cdots & x_{(c-1)r+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_r & x_{2r} & \cdots & x_{cr} \end{pmatrix}$$

//

Definition A.1 (Frobenius Norm). Let $A \in \mathbb{R}^{m \times n}$, then its *Frobenius norm*, $\|A\|_F$, is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{i,j}^2}.$$

//

Notation A.6 (Asymptotics). Let $f(n)$ and $g(n)$ be arbitrary functions in $n \in \mathbb{N}$. Then we define the following asymptotic notations.

- (i). *Upper bound*: $f(n) = \mathcal{O}(g(n))$, i. e., there exists an $c > 0$ such that there exists an $n_0 \geq 0$ such that $|f(n)| \leq c|g(n)|$ for all $n \geq n_0$.
- (ii). *Lower bound*: $f(n) = \Omega(g(n))$, i. e., $g(n) = \mathcal{O}(f(n))$.
- (iii). *Tight bound*: $f(n) = \Theta(g(n))$, i. e., $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(f(n))$.

//

B. Perron-Frobenius Theory

Definition B.1. We call a real matrix A *positive* ($A > 0$), if $a_{i,j} > 0$ for all i, j . Likewise, we call A *nonnegative* ($A \geq 0$) if $a_{i,j} \geq 0$ for all i, j . The same notation applies to vectors as well. //

The Perron-Frobenius theory is a particularly elegant piece of linear algebra which reveals, e. g., how the nonnegativity (positivity) of a matrix is reflected in its eigenvalues and eigenvectors, respectively. To indicate how well the Perron-Frobenius theory can be utilized in application contexts, we consider the convergence analysis of finite Markov chains as a running example throughout this appendix.

Example B.1 (Markov chain). A finite discrete-time Markov chain is a mathematical model consisting of a finite amount of states, where the current state can change at each discrete point of time. In particular, we consider so-called homogenous Markov chains, where at each point of time the probability for a state transition from state i to state j is given by $0 \leq p_{j,i} \leq 1$. Thus, the transition probabilities can be conveniently represented by a matrix P . In order to obtain a sound model, the *transition matrix* P has to be *column-stochastic*, i. e., each column of P sums up to one (is a *stochastic vector*). In particular, this means that for every node the sum over its (outgoing) transition probabilities equals to one.

Consider we are given a stochastic vector s whose elements s_i represent the probability that the Markov chain is initially in state i . Then the probability distribution of being in a certain state after k time steps is given by the vector $P^k s$, since a column-stochastic matrix times a stochastic vector is a stochastic vector. The following three question are of tremendous interest: Does the Markov chain converge, i. e., exists a *stationary distribution* $\pi = \lim_{k \rightarrow \infty} P^k s$? How does the stationary distribution π look? Does π dependent on the initial distribution s or not? As we will see, these question can easily be answered by utilizing results from the Perron-Frobenius theory. //

Historically, the Perron-Frobenius theory originates in investigations of positive matrices [Frobenius, 1908; Perron, 1907], which where later extended to nonnegative matrices [Frobenius, 1912]. In the following, we give a brief overview of the most fundamental notions and results of the Perron-Frobenius theory. A more detailed exposition can be found, e. g., in the textbooks of Meyer [2000, § 8] and Seneta [2006, § 1].

The core of the Perron-Frobenius theory is what is today commonly known as Perron-Frobenius theorem. Several version of this theorem exist which slightly differ in their assumptions and conclusions. First, we consider the Perron-Frobenius theorem for positive matrices.

Theorem B.1 (Perron-Frobenius, [Meyer, 2000, p. 667]). *Let $A \in \mathbb{R}^{n \times n}$ be a positive matrix, then the following assertions hold.*





B. Perron-Frobenius Theory

- (i). There exists a positive real eigenvalue λ of A such that $\lambda > |\mu|$ for all other eigenvalues $\mu \neq \lambda$ of A .
- (ii). The eigenvalue λ is a simple root of the characteristic polynomial of A .
- (iii). There is a strictly positive left eigenvector $u > 0$ with $u^T A = \lambda u^T$ and a strictly positive right eigenvector v with $Av = \lambda v$ associated with λ .
- (iv). Except for u and v all other eigenvectors of A contain at least one negative entry.
- (v). Let u and v be normalized such that $u^T v = 1$, then

$$\lim_{k \rightarrow \infty} A^k / \lambda^k = v u^T. \quad (\text{B.1})$$

Example B.2 (Markov chain, continued). In case we assume a positive transition matrix P (which means that any state change can happen with a certain probability), [Theorem B.1](#) straightforwardly delivers answers to all questions we raised in [Example B.1](#). Concretely, since P is column stochastic, its left eigenvector is given by the constant vector $\mathbf{1}$, with corresponding eigenvalue one. Thus, the spectral radius of P is one and $\lim_{k \rightarrow \infty} A^k = v \mathbf{1}^T$, i. e., the Markov chain converges. Moreover, since $v \mathbf{1}^T s = v$ for all stochastic vectors s , we see that the stationary distribution $\pi = v$ and that π is independent of the initial distribution.

In the following we will investigate up to which extent [Theorem B.1](#) can be generalized to nonnegative matrices. To illustrate the differences between certainly structured matrices, we consider the following four nonnegative transition matrices P_1 – P_4 .

	P_1	P_2	P_3	P_4
Matrix	$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$	$\begin{pmatrix} 0 & 1/2 \\ 1 & 1/2 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1/2 \\ 0 & 1/2 \end{pmatrix}$
Induced graph				
Eigenvalues	$\{1, 0\}$	$\{1, -1/2\}$	$\{1, -1\}$	$\{1, 1/2\}$
Eigenvectors	$\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1/2 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

Obviously, (some of) the conclusion of [Theorem B.1](#) do not hold in case of P_3 and P_4 . //

Nonnegative matrices can certainly be represented as limits of positive matrices. Thus, it does not surprise that for a nonnegative matrix $A \in \mathbb{R}^{n \times n}$ with spectral radius $\rho(A) \geq 0$, the real value $\lambda = \rho(A)$ is an eigenvalue of A and its corresponding eigenvector is nonnegative [[Meyer, 2000](#), p.670]. Moreover, a variational characterization of the spectral radius $\rho(A)$ of a square matrix $A \geq 0$ is given by the so-called Collatz-Wielandt formula [[Meyer, 2000](#), pp. 669–670]

$$\rho(A) = \max_{\substack{x \geq 0 \\ x \neq 0}} \min_{\substack{1 \leq i \leq n \\ x_i \neq 0}} \frac{[Ax]_i}{x_i} = \min_{x > 0} \max_{1 \leq i \leq n} \frac{[Ax]_i}{x_i}.$$

On the other hand, from an application point of view, it seems to be reasonable that [Theorem B.1](#) can be extended to certain classes of nonnegative matrices. Consider, e. g., a Markov chain with nonnegative transition matrix (cf. [Example B.2](#)). In case that it is possible to reach from each state (after a finite amount of transitions) every other state, one would expect that the stationary distribution should show a nonzero (potentially small) probability for every state. In the following, we make these considerations precise, which leads to the notion of *irreducible matrices*. Interestingly, its indeed only the pattern of nonzero values what matters in order to generalize the Perron-Frobenius theorem to nonnegative matrices and not the overall number of nonzero entries or their concrete values (cf. [Example B.2](#)).

We give in the following two equivalent characterization of irreducible matrices. A matrix $A \in \mathbb{R}^{n \times n}$ is called *reducible*, if there exists a permutation matrix P such that

$$P^T A P = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix},$$

with square matrices B and D . Consequently, a matrix is called *irreducible* if it is not reducible. A more insightful characterization can be given by utilizing notions from graph theory. Recall that a (directed) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes is fully determined by its *adjacency matrix* $A \in \{0, 1\}^{n \times n}$, i. e., $A_{i,j} = 1$, if and only if there is a directed edge between node i and node j . Therefore, given a matrix $A \in \mathbb{R}^{n \times n}$, we can define its *induced graph* by the adjacency matrix $A^{\mathcal{G}}$ with $A_{i,j}^{\mathcal{G}} = 1$, if and only if $A_{i,j} \neq 0$ and $A_{i,j}^{\mathcal{G}} = 0$ otherwise. Now a matrix $A \in \mathbb{R}^{n \times n}$ is irreducible, if and only if its induced graph is strongly connected [[Meyer, 2000](#), p. 671], i. e., there exists a directed path from every node to any other node. For nonnegative irreducible matrices [Theorem B.1](#) can be generalized as follows.

Theorem B.2 (Perron-Frobenius, [[Meyer, 2000](#), p. 673]). *Let $A \in \mathbb{R}^{n \times n}$ be a nonnegative irreducible matrix and let $\lambda = \rho(A)$ denote its spectral radius. Then the following assertions hold.*

- (i). *The spectral radius $\lambda > 0$ and λ is an eigenvalue of A .*
- (ii). *The eigenvalue λ is a simple root of the characteristic polynomial of A .*
- (iii). *There is a strictly positive left eigenvector $u > 0$ with $u^T A = \lambda u^T$ and a strictly positive right eigenvector v with $Av = \lambda v$ associated with λ .*
- (iv). *Except for u and v all other eigenvectors of A contain at least one negative entry.*
- (v). *If A has h eigenvalues on its spectral circle, then they are for $1 \leq i \leq h$ given by $\lambda_i = \lambda \omega^{(i-1)}$, where $\omega = e^{2\pi i/h}$ is a h -th root of unity.*

Thus, we observe two major difference between [Theorem B.1](#) and [Theorem B.2](#). First, there can exist several eigenvalues on the spectral circle in case of an irreducible matrix which are structured according to assertion (v) in [Theorem B.2](#). Moreover, the convergence property (v) in [Theorem B.1](#) is missing, which can indeed not be generalized to irreducible matrices, as the following example illustrates.

Example B.3 (Markov chain, continued). Consider the matrix P_3 from [Example B.2](#). The graph induced by P_3 is obviously strongly connected. Thus, P_3 is irreducible. On the other hand, note that $\lim_{k \rightarrow \infty} P_3^k$ does not exist, since for $j \in \mathbb{N}$ we see that $P_3^{2j} = P_3$ and $P_3^{2j+1} = \mathcal{I}_2$. Therefore, even though irreducible matrices preserve almost all properties of positive matrices, e. g., the positive eigenpair, they are not suited for deriving a convergence result of the kind we require here. //

As the previous example illustrates, the notion of irreducibility is not yet fully satisfactory in certain application contexts. To generalize [Theorem B.1](#) in its entirety to nonnegative matrices, the notion of *primitive matrices* plays an important role.

Definition B.2. We call a square nonnegative matrix $A \in \mathbb{R}^{n \times n}$ *primitive*, if there exists a constant $k \in \mathbb{N}$ such that A^k is a primitive matrix, i. e., $A^k > 0$. //

Corollary B.3 (Meyer [2000, p. 674]). *Theorem B.1 holds also in case we assume A to be primitive instead of strictly positive.*

It is noteworthy that this for applications very important generalization was already stated in the pioneering work of Perron [1907, p. 262]. The question which is left open is how the notions of irreducibility and primitivity are related to each other. We refrain here from a detailed discussion and instead, we give a very simple criterion to detect the primitivity of an irreducible matrix.

Lemma B.4 (Meyer [2000, p. 678]). *Let A be a nonnegative irreducible matrix with at least one positive diagonal element, then A is primitive.*

Example B.4 (Markov chain, continued). By the considerations above, the transition matrices given in [Example B.2](#) can easily be categorized (and analyzed). In particular, P_1 is positive, P_2 is primitive, P_3 is irreducible and P_4 is (only) nonnegative. Accordingly, either [Theorem B.1](#) (P_1, P_2), [Theorem B.2](#) (P_3) or neither of them (P_4) applies. //

Summarizing, we have seen that the Perron-Frobenius theorem provides several insights into the structure of the eigenvalues and eigenvectors of positive and nonnegative matrices, respectively. A particularly interesting variant of the theorem is the one for primitive matrices (cf. [Corollary B.3](#)), which allows for the same conclusion as the variant for positive matrices. The generalization of the Perron-Frobenius theorem to classes of nonnegative matrices required assumptions on the pattern of nonzero entries, which lead to the property of a matrix to be irreducible. In many applications this property, i. e., the strong connectivity of the induced graph, is given in a natural way and its verification is fairly straightforward. Moreover, the primitivity of an irreducible matrix is simply verified by the existence of a nonzero diagonal element.

Bibliography

- Bellman, R. E. (1961). *Adaptive control processes - A guided tour*. Princeton University Press (see p. 2).
ISBN: [0691079011](#)
- Benedetto, F. D., Fiorentino, G., and Serra, S. (1993). “C. G. preconditioning for Toeplitz matrices”. In: *Computers & Mathematics with Applications* **25**(6), pp. 35–45 (see p. 19).
DOI: [10.1016/0898-1221\(93\)90297-9](#)
- Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006). “Randomized Gossip Algorithms”. In: *IEEE Transactions on Information Theory* **52**(6), pp. 2508–2530 (see p. 80).
DOI: [10.1109/TIT.2006.874516](#)
- Chan, R. H. and Ng, K.-P. (1993). “Toeplitz preconditioners for Hermitian Toeplitz systems”. In: *Linear Algebra and its Applications* **190**, pp. 181–208 (see p. 19).
DOI: [10.1016/0024-3795\(93\)90226-E](#)
- Cooley, J. W. and Tukey, J. W. (1965). “An Algorithm for the Machine Calculation of Complex Fourier Series”. In: *Mathematics of Computation* **19**(90), pp. 297–301 (see pp. 31, 34).
DOI: [10.1090/S0025-5718-1965-0178586-1](#)
- Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2001). *Introduction to Algorithms*. 2nd ed. McGraw-Hill Higher Education (see p. 33).
ISBN: [0-07-013151-1](#)
- Cullum, J. K. and Willoughby, R. A. (2002). *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*. Society for Industrial and Applied Mathematics (see pp. 4, 14, 19).
DOI: [10.1137/1.9780898719192](#)
- Drake, J. (1993). “Rates of spontaneous mutation among RNA viruses.” In: *Proceedings of The National Academy of Sciences* **90**(9), pp. 4171–4175 (see p. 7).
DOI: [10.1073/pnas.90.9.4171](#)
- Drubin, M. (1971). “Kronecker Product Factorization of the FFT Matrix”. In: *IEEE Transactions on Computers* **C-20**(5), pp. 590–593 (see p. 34).
DOI: [10.1109/T-C.1971.223306](#)
- Eigen, M. (1971). “Selforganization of matter and the evolution of biological macromolecules”. In: *Naturwissenschaften* **58**(10), pp. 465–523 (see pp. 5, 6, 71).
DOI: [10.1007/BF00623322](#)
- Eigen, M. (2002). “Error catastrophe and antiviral strategy”. In: *Proceedings of the National Academy of Sciences of the United States of America* **99**(21), pp. 13374–13376 (see p. 7).
DOI: [10.1073/pnas.212514799](#)

Bibliography

- Eigen, M. and Schuster, P. (1977). “A principle of natural self-organization”. In: *Naturwissenschaften* **64**(11), pp. 541–565 (see p. 5).
DOI: [10.1007/BF00450633](https://doi.org/10.1007/BF00450633)
- Frigo, M. and Johnson, S. (2005). “The Design and Implementation of FFTW3”. In: *Proceedings of the IEEE* **93**(2), pp. 216–231 (see p. 31).
DOI: [10.1109/JPROC.2004.840301](https://doi.org/10.1109/JPROC.2004.840301)
- Frobenius, G. (1908). “Über Matrizen aus positiven Elementen”. In: *Sitzungsberichte der Preussischen Akademie der Wissenschaften* **1908**, pp. 471–476 (see p. 85).
URL: <http://archive.org/stream/sitzungsberichte1908deutsch>
- Frobenius, G. (1912). “Über Matrizen aus nicht negativen Elementen”. In: *Sitzungsberichte der Preussischen Akademie der Wissenschaften* **1912**, pp. 456–477 (see p. 85).
- Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. 3rd ed. The Johns Hopkins University Press (see pp. 1, 13, 21–23, 55).
ISBN: 0-8018-5413-X
- Grote, M. and Huckle, T. (1997). “Parallel Preconditioning with Sparse Approximate Inverses”. In: *SIAM Journal on Scientific Computing* **18**(3), pp. 838–853 (see pp. 18, 54).
DOI: [10.1137/S1064827594276552](https://doi.org/10.1137/S1064827594276552)
- Hackbusch, W. and Khoromskij, B. N. (2007). “Tensor-product approximation to operators and functions in high dimensions”. In: *Journal of Complexity* **23**(4–6), pp. 697–714 (see p. 2).
DOI: [10.1016/j.jco.2007.03.007](https://doi.org/10.1016/j.jco.2007.03.007)
- Hamming, R. W. (1950). “Error Detecting and Error Correcting Codes”. In: *Bell System Technical Journal* **29**(2), pp. 147–160 (see p. 39).
- Havlicek, H. (2006). *Lineare Algebra für Technische Mathematiker*. 1st ed. Berliner Studienreihe zur Mathematik. Band 16. Heldermann Verlag (see p. 3).
ISBN: 3-88538-116-8
- He, M. X., Petoukhov, S. V., and Ricci, P. E. (2004). “Genetic code, hamming distance and stochastic matrices”. In: *Bulletin of Mathematical Biology* **66**(5), pp. 1405–1421 (see p. 79).
DOI: [10.1016/j.bulm.2004.01.002](https://doi.org/10.1016/j.bulm.2004.01.002)
- He, M. and Petoukhov, S. (2010). “Genetic Codes, Matrices, and Symmetrical Techniques”. In: *Mathematics of Bioinformatics*, pp. 24–62 (see p. 79).
DOI: [10.1002/9780470904640.ch2](https://doi.org/10.1002/9780470904640.ch2)
- Hestenes, M. R. and Stiefel, E. (1952). “Methods of Conjugate Gradients for Solving Linear Systems”. In: *Journal of Research of the National Bureau of Standards* **49**(6), pp. 409–436 (see p. 14).
- Johnson, J. and Püschel, M. (2000). “In Search of the Optimal Walsh-Hadamard Transform”. In: *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)—Volume 6*, pp. 3347–3350 (see p. 34).
DOI: [10.1109/ICASSP.2000.860117](https://doi.org/10.1109/ICASSP.2000.860117)
- Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics (see pp. 85–88).
ISBN: 0-89871-454-0
- Moravitz Martin, C. and Van Loan, C. (2007). “Shifted Kronecker Product Systems”.

- In: *SIAM Journal on Matrix Analysis and Applications* **29**(1), pp. 184–198 (see p. 80).
DOI: [10.1137/050631707](https://doi.org/10.1137/050631707)
- Niederbrucker, G. and Gansterer, W.N. (2011a). “A Fast Solver for Modeling the Evolution of Virus Populations”. In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '11, 74:1–74:11 (see pp. 5, 8, 10, 11, 52, 71, 73).
DOI: [10.1145/2063384.2063483](https://doi.org/10.1145/2063384.2063483)
- Niederbrucker, G. and Gansterer, W.N. (2011b). “Efficient Solution of Evolution Models for Virus Populations”. In: *Procedia Computer Science* **4**. Proceedings of the International Conference on Computational Science, ICCS 2011, pp. 126–135 (see pp. 8–11, 56).
DOI: [10.1016/j.procs.2011.04.014](https://doi.org/10.1016/j.procs.2011.04.014)
- Nowak, M. and Schuster, P. (1989). “Error thresholds of replication in finite populations mutation frequencies and the onset of muller’s ratchet”. In: *Journal of Theoretical Biology* **137**(4), pp. 375–395 (see p. 8).
DOI: [10.1016/S0022-5193\(89\)80036-0](https://doi.org/10.1016/S0022-5193(89)80036-0)
- Perron, O. (1907). “Zur Theorie der Matrices”. In: *Mathematische Annalen* **64**(2), pp. 248–263 (see pp. 85, 88).
DOI: [10.1007/BF01449896](https://doi.org/10.1007/BF01449896)
- Püschel, M. *et al.* (2005). “SPIRAL: Code Generation for DSP Transforms”. In: *Proceedings of the IEEE* **93**(2), pp. 232–275 (see p. 31).
DOI: [10.1109/JPROC.2004.840306](https://doi.org/10.1109/JPROC.2004.840306)
- Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. 2nd ed. Society for Industrial and Applied Mathematics (see pp. 3, 4, 13–15, 17, 18).
ISBN: 0-89871-534-2
- Saad, Y. and Schultz, M.H. (1986). “GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems”. In: *SIAM Journal on Scientific and Statistical Computing* **7**(3), pp. 856–869 (see p. 14).
DOI: [10.1137/0907058](https://doi.org/10.1137/0907058)
- Schuster, P. (2006). “Prediction of RNA secondary structures: from theory to models and real molecules”. In: *Reports on Progress in Physics* **69**(5), pp. 1419–1477 (see p. 6).
DOI: [10.1088/0034-4885/69/5/R04](https://doi.org/10.1088/0034-4885/69/5/R04)
- Schuster, P. (2008). *The mathematics of Darwinian systems*. Appendix A4 to the book “From Strange Simplicity to Complex Familiarity: A Treatise on Matter, Information, Life and Thought” by Manfred Eigen. Oxford University Press, pp. 667–700 (see pp. 6, 7, 71).
DOI: [10.1093/acprof:oso/9780198570219.001.0001](https://doi.org/10.1093/acprof:oso/9780198570219.001.0001)
- Schuster, P. (2011). “Mathematical modeling of evolution. Solved and open problems”. In: *Theory in Biosciences* **130**(1), pp. 71–89 (see pp. 6–8, 71–73).
DOI: [10.1007/s12064-010-0110-z](https://doi.org/10.1007/s12064-010-0110-z)
- Seneta, E. (2006). *Non-Negative Matrices and Markov Chains*. Springer Series in Statistics. Springer (see p. 85).
DOI: [10.1007/0-387-32792-4](https://doi.org/10.1007/0-387-32792-4)

Bibliography

- Steeb, W.-H. (1997). *Matrix Calculus and the Kronecker Product with Applications and C++ Programs*. World Scientific Publishing Co., Inc. (see p. 25).
ISBN: 981-02-3241-1
- Swetina, J. and Schuster, P. (1982). “Self-replication with errors: A model for polynucleotide replication”. In: *Biophysical Chemistry* **16**(4), pp. 329–345 (see p. 8).
DOI: 10.1016/0301-4622(82)87037-3
- Thompson, C. J. and McBride, J. L. (1974). “On Eigen’s theory of the self-organization of matter and the evolution of biological macromolecules”. In: *Mathematical Biosciences* **21**(1–2), pp. 127–142 (see p. 6).
DOI: 10.1016/0025-5564(74)90110-2
- Trefethen, L. N. and Bau, D. (1997). *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (see pp. 13–16, 20–22, 74).
ISBN: 0-89871-361-7
- van der Vorst, H. A. (1992). “Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems”. In: *SIAM Journal on Scientific and Statistical Computing* **13**(2), pp. 631–644 (see p. 15).
DOI: 10.1137/0913035
- van der Vorst, H. A. (2004). “How to write a frequently-cited article”. In: *The Australian Mathematical Society Gazette* **31**(2), pp. 94–100 (see p. 15).
- Van Loan, C. F. (1992). *Computational Frameworks for the Fast Fourier Transform*. Society for Industrial and Applied Mathematics (see p. 34).
DOI: 10.1137/1.9781611970999
- Van Loan, C. F. (2000). “The ubiquitous Kronecker product”. In: *Journal of Computational and Applied Mathematics* **123**, pp. 85–100 (see pp. 11, 31, 37).
DOI: 10.1016/S0377-0427(00)00393-9
- Van Loan, C. F. and Pitsianis, N. (1993). “Approximation with Kronecker Products”. In: *Linear Algebra for Large Scale and Real Time Applications*. Ed. by M. S. Moonen, G. H. Golub, and B. L. R. De Moor, pp. 293–314 (see pp. 35–37).
ISBN: 0792321510