Efficient Algorithms for Solving Structured Eigenvalue Problems Arising in the Description of Electronic Excitations

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

von **M. Sc. Carolin Penke**

geb. am **09.03.1991** in Melle

genehmigt durch die Fakultät für Mathematik

der Otto-von-Guericke-Universität Magdeburg

Gutachter: **Prof. Dr. Peter Benner**

Prof. Dr. Daniel Kressner

Dr. Chao Yang

eingereicht am: **20.12.2021**

Verteidigung am: **03.05.2022**

PUBLICATIONS

The results presented in this thesis were achieved during the author's time as a PhD student at the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg in the period of May 1, 2017, to May 31, 2021. This includes a three-month research stay at the National Institute of Informatics in Tokyo from January 7 to April 4, 2019. The author is the main contributor in the following publications produced during this time.

[BKP17] P. Benner, M. Köhler, and C. Penke. GPU-accelerated and storage-efficient implementation of the QR decomposition. In S. Bassini, M. Danelutto, P. Dazzi, G. R. Joubert, and F. Peters, editors, *Parallel Computing is Everywhere, Proceedings of the International Conference on Parallel Computing, ParCo 2017, 12-15 September 2017, Bologna, Italy*, volume 32 of *Advances in Parallel Computing*, pages 329–338. IOS Press, 2017.

[BMP18] P. Benner, A. Marek, and C. Penke. Improving the performance of numerical algorithms for the Bethe-Salpeter eigenvalue problem. *Proc. Appl. Math. Mech.*, 18(1):e201800255, 2018.

[PMV+20] C. Penke, A. Marek, C. Vorwerk, C. Draxl, and P. Benner. High performance solution of skew-symmetric eigenvalue problems with applications in solving the Bethe-Salpeter eigenvalue problem. *Parallel Comput.*, 96:102639, 2020.

[BP21a] P. Benner and C. Penke. Efficient and accurate algorithms for solving the Bethe-Salpeter eigenvalue problem for crystalline systems. *J. Comput. Appl. Math.*, 2021.

[BP21b] P. Benner and C. Penke. GR decompositions and their relations to Cholesky-like factorizations. *Proc. Appl. Math. Mech.*, 20(1):e202000065, 2021.

[BNP22a] P. Benner, Y. Nakatsukasa, and C. Penke. Stable and efficient computation of generalized polar decompositions. *SIAM J. Matrix Anal. Appl.*, 2022. Accepted.

[BNP22b] P. Benner, Y. Nakatsukasa, and C. Penke. A structure-preserving divide-and-conquer method for pseudosymmetric matrices. e-print 2203.08900, arXiv, 2022. Handed in.

These publications, except for [BKP17], found their way into this thesis and make up large parts of it. The introduction (Chapter 1), the physical preliminaries (Chapter 3) and the conclusions (Chapter 10) are not found in the listed publications. The mathematical preliminaries (Chapter 2) compile basics from all publications above and present additional material.

Theoretical results given in [BP21a] are compiled in Chapter 4. Chapter 5 is based on [BMP18] and [PMV+20]. Algorithms given in [BP21a] are presented in Chapter 6. Chapter 7 constitutes an extended version of [BP21b] and contains parts of [BNP22a]. Chapter 8 is based on [BNP22a]. A paper based on Chapter 9 is handed in and available as a preprint [BNP22b].

ABSTRACT

This dissertation is located in the field of numerical linear algebra. It proposes algorithms for solving structured eigenvalue problems arising in electronic structure computations. Matrices arising in linear-response time-dependent density functional theory and many-body perturbation theory, in particular in the Bethe-Salpeter approach, show a 2×2 block structure. We show that they can also be characterized with the help of non-Euclidian scalar products.

The motivation to devise new algorithms, instead of using general purpose eigenvalue solvers, comes from the need to solve large problems on high performance computers. This requires parallelizable and communication-avoiding algorithms and implementations.

After giving some mathematical, computational and physical background knowledge, this thesis documents the extension of an optimized HPC library for solving skew-symmetric eigenvalue problems on distributed memory machines. This procedure is used to solve a specific form of Bethe-Salpeter eigenvalue problem under certain definiteness conditions.

For crystalline systems, the periodicity of the lattice may be exploited to arrive at a different form of the Bethe-Salpeter eigenvalue problem. Here, it is possible to reduce the matrix dimension by half. Under the mentioned definiteness conditions, the resulting eigenvalue problem is Hermitian positive definite. Accuracy may be improved by employing a singular value decomposition instead of solving an implicitly squared eigenvalue problem. A high performance implementation is not yet available but easy to realize with basic linear algebra routines.

In the second half of this thesis, the focus shifts towards the development of new algorithms, which do not yet have a high performance implementation but show high potential in this regard.

We develop algorithms for computing generalized concepts of polar decompositions and QR decompositions. Generalized polar decompositions are computed via iterations where the number of steps can be known a priori. The iteration steps employ routines for which communication-avoiding implementations exist. Further options of parallelization are available making the algorithm promising for high performance computing. Several heuristics to improve the accuracy of the involved computations are proposed.

The motivating eigenvalue problems, stemming from electronic structure theory,

fall into the category of pseudosymmetric eigenvalue problems. We develop a new algorithm for this class of problems, by generalizing ideas from the method of spectral divide-and-conquer for symmetric matrices. They are applied in a setting defined by indefinite scalar products. Pseudosymmetry is preserved throughout the computations. Generalized polar decompositions and generalized QR decompositions are necessary tools in the presented algorithm.

ZUSAMMENFASSUNG

Diese Dissertation ist im Feld der numerischen linearen Algebra angesiedelt. Sie stellt Algorithmen zur Lösung strukturierter Eigenwertprobleme, die sich bei der Berechnung der elektronischen Struktur von Materie ergeben, vor. Matrizen aus der zeitabhängigen Dichtefunktionaltheorie im Regime der linearen Antwort sowie der Vielteilchen-Störungstheorie, insbesondere im Bethe-Salpeter Ansatz, zeigen eine 2×2 Blockstruktur. Wir zeigen, dass diese Strukturen außerdem mithilfe von nicht-euklidischen Skalarprodukten charakterisiert werden können.

Die Motivation, neue Algorithmen zu entwickeln, statt allgemeine Löser zu verwenden, ergibt sich aus der Notwendigkeit, große Probleme auf Hochleistungsrechnern zu lösen. Hierzu werden parallelisierbare Algorithmen und Implementierungen benötigt, die außerdem unnötige Kommunikation vermeiden.

Nach einleitenden Voraussetzungen aus der Mathematik, der Informatik und der Physik dokumentiert diese Arbeit die Erweiterung einer optimierten HPC Bibliothek zur Lösung schiefsymmetrischer Eigenwertprobleme auf Maschinen mit verteiltem Speicher. Diese Prozedur wird genutzt, um eine spezifische Form des Bethe-Salpeter-Eigenwertproblems unter gewissen Definitheitsbedingungen zu lösen.

Für kristalline Systeme kann die Periodizität des Gitters genutzt werden, um zu einer anderen Form des Bethe-Salpeter-Eigenwertproblems zu gelangen. Hier ist es möglich, die Matrixdimension um die Hälfte zu verringern. Unter den genannten Definitheitsbedingungen ist das resultierende Eigenwertproblem hermitsch positiv definit. Die Genauigkeit kann verbessert werden, wenn eine Singulärwertzerlegung genutzt wird, statt implizit ein quadriertes Eigenwertproblem zu lösen. Eine Implementierung auf Hochleistungsrechnern ist noch nicht verfügbar, aber leicht umzusetzen mit grundlegenden Operationen der linearen Algebra.

Die zweite Hälfte dieser Arbeit konzentriert sich auf die Entwicklung neuartiger Algorithmen, die noch keine Implementierung für Hochleistungsrechner besitzen, diesbezüglich aber ein großes Potenzial zeigen.

Wir entwicklen Algorithmen zur Berechnung verallgemeinerter Konzepte von Polarzerlegungen und QR Zerlegungen. Verallgemeinerte Polarzerlegungen werden mithilfe von Iterationen berechnet, bei denen die Anzahl an Schritten a priori bekannt ist. Die Iterationsschritte nutzen Routinen, für die kommunikationsvermeidende Implementierungen existieren. Weitere Möglichkeiten zur Parallelisierung sind verfügbar und machen den Algorithmus vielversprechend für den Bereich des Hochleistungsrechnens.

Verschiedene Heuristiken werden vorgestellt, um die Genauigkeit der durchgeführten Berechnungen zu verbessern.

Die motivierenden Eigenwertprobleme aus der elektronischen Strukturtheorie fallen in die Kategorie der schiefsymmetrischen Eigenwertprobleme. Wir entwickeln einen neuen Algorithmus für diese Problemklasse, indem wir Ideen der spektralen Teile-und-Herrsche Methode für symmetrische Matrizen verallgemeinern. Sie werden angewandt in einem Umfeld, welches durch indefinite Skalarprodukte charakterisiert wird. Die Pseudosymmetrie bleibt während der Rechnungen erhalten. Verallgemeinerte Polarzerlegungen und verallgemeinerte QR Zerlegungen werden als Werkzeuge im präsentierten Algorithmus benötigt.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF SYMBOLS

$\mathbb{R}$, $\mathbb{C}$,	fields of real and complex numbers
$\mathbb{K}$	$\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$
$\mathbb{C}_+$, $\mathbb{C}_-$	open right/open left complex half plane
$\mathbb{R}_+$, $\mathbb{R}_-$	strictly positive/negative real line
$\mathbb{R}^n$, $\mathbb{C}^n$	vector space of real/complex n-tuples
$\mathbb{R}^{m\times n}$, $\mathbb{C}^{m\times n}$	real/complex $m \times n$ matrices
$\mathbb{Z}$	set of whole numbers
$\mathcal{P}_n$	set of polynomials of degree n.
$\mathcal{R}_{p,q}$	$:= \{r(x) = \frac{P(x)}{Q(x)} : P \in \mathcal{P}_p,\ Q \in \mathcal{P}_q\}$, set of rational functions of degree (p, q).
$\lvert\xi\rvert$	absolute value of real or complex scalar
$\arg \xi$	argument of complex scalar
i	imaginary unit ($\mathrm{i}^2 = -1$)
$\mathrm{Re}(A)$, $\mathrm{Im}(A)$	real and imaginary part of a complex quantity $A = \mathrm{Re}(A) + \mathrm{i}\,\mathrm{Im}(A) \in \mathbb{C}^{m\times n}$
$\overline{A}$	$:= \mathrm{Re}(A) - \mathrm{i}\,\mathrm{Im}(A)$, complex conjugate of $A \in \mathbb{C}^{m\times n}$
a_{ij}	the (i, j)-th entry of A
$A(i:j,:)$, $A(:,k:\ell)$	rows $i, \ldots, j$ of A and columns $k, \ldots, \ell$ of A
$A(i:j,k:\ell)$	rows $i, \ldots, j$ of columns $k, \ldots, \ell$ of A
A^{T}	the transpose of A
$\mathrm{colspan}(A)$	$:= \{v : v = \sum_{i=1}^{n} \lambda_i A(:,i),\ \lambda_i \in \mathbb{K}\}$, column space of A
A^{H}	$:= (\overline{A})^{\mathrm{T}}$, the complex conjugate transpose (also called Hermitian transpose)
A^*	transpose or Hermitian transpose, depending on context ($A^* = A^{\mathsf{T}}$ or $A^* = A^{\mathsf{H}}$)
A^{-1}	the inverse of nonsingular A

$A^{-\mathsf{T}}$, $A^{-\mathsf{H}}$, A^{-*}	the inverse of A^{T}, A^{H}, A^*
$A^\dagger$	$:= \hat{\Sigma}A^*\Sigma$, the pseudo-inverse of a $(\Sigma, \hat{\Sigma})$-orthogonal matrix A, where Σ, $\hat{\Sigma}$ denote signature matrices
I_n	identity matrix of dimension n
$\Lambda(A)$, $\Lambda(A, M)$	spectrum of matrix A/matrix pair (A, M)
$\lambda_j(A)$, $\lambda_j(A, M)$	j-th eigenvalue of A/(A, M)
$\rho(A, M)$	$:= \max_j \lvert\lambda_j(A, M)\rvert$, spectral radius of (A, M)
$\sigma_{\max}(A)$	the largest singular value of A
$\mathrm{tr}(A)$	$:= \sum_{i=1}^n a_{ii}$, trace of A
$\lVert u\rVert_p$	$:= (\sum_{i=1}^n \lvert u_i\rvert^p)^{1/p}$ for $u \in \mathbb{C}^n$ and $1 \le p < \infty$
$\lVert u\rVert_\infty$	$:= \max_i \lvert u_i\rvert$, the maximum norm of u
$\lVert A\rVert_p$	$:= \sup\{\lVert Au\rVert_p : \lVert u\rVert_p = 1\}$, subordinate matrix p-norm, $1 \le p \le \infty$
$\lVert A\rVert_{\mathrm{F}}$	$:= \sqrt{\sum_{i,j} \lvert a_{ij}\rvert^2} = \sqrt{\mathrm{tr}(A^*A)}$, the Frobenius norm of matrix $A \in \mathbb{C}^{m\times n}$
$\lVert u\rVert$, $\lVert A\rVert$	Euclidean vector or subordinate matrix norm $\lVert\cdot\rVert_2$
$\kappa_p(A)$	$:= \lVert A\rVert_p \lVert A^{-1}\rVert_p$, the p-norm condition number for nonsingular A
$\kappa(A)$, $\mathrm{cond}(A)$	the 2-norm condition number for regular A
$A > 0$, $A < 0$	A is symmmetric or Hermitian positive/negative definite
$A \ge 0$, $A \le 0$	A is symmmetric or Hermitian positive/negative semidefinite
$\mathrm{diag}(\sigma_1, \ldots, \sigma_n)$	$:= \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}$
I_n	Identity matrix of size $n \times n$, $I_n := \mathrm{diag}(1, \ldots, 1)$
e_i	i-th column of the identity matrix of appropriate size, i.e. a vector with entry 1 on position j and 0 as other entries
P_n	$:= \begin{bmatrix} e_1 & e_3 & \ldots & e_{2n-1} & e_2 & e_4 & \ldots e_{2n} \end{bmatrix}$, perfect shuffle permutation
$A \oplus B$, $\mathrm{diag}(A, B)$	$:= \begin{bmatrix} A & \\ & B \end{bmatrix}$, a block diagonal matrix with blocks A and B
K_n	$:= I_n \oplus -I_n$

J_n	$:= \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$
$\partial_{x_j} f := \frac{\partial}{\partial x_j} f$	partial derivative with respect to x_j of f
$\partial^2_{x_j} f := \partial_{x_j x_j} f$	second order partial derivative with respect to x_j of f
$\nabla f := (\partial_{x_1} f, \ldots, \partial_{x_n} f)^{\mathrm{T}}$	the gradient of f
$\Delta f := \sum_{i=1}^{n} \partial^2_{x_i} f$	the *Laplacian operator* applied to f
$\nabla^2 f$	$:= \begin{bmatrix} \partial^2_{x_1} f & \cdots & \partial_{x_1 x_n} f \\ \vdots & \ddots & \vdots \\ \partial_{x_n x_1} f & \cdots & \partial^2_{x_n} f \end{bmatrix}$, the *Hessian matrix* of f
$L^2(V, \mathbb{K})$	$:= \left\{ f : V \to \mathbb{K} \ : \ \int_V \lvert f(x) \rvert^2 \, dx < \infty \right\}$
$[a, b]$, $a, b \in \mathbb{R}$	$:= \{x \in \mathbb{R} : a \leq x \leq b\}$
(a, b), $a, b \in \mathbb{R}$	$:= \{x \in \mathbb{R} : a < x < b\}$
$\forall$	for all
`GEMM`	BLAS routine: **ge**neral **m**atrix **m**atrix multiplication
`GEMV`	BLAS routine: **ge**neral **m**atrix **v**ector multiplication
`SYR2`	BLAS routine: **sy**mmetric **r**ank-2 update
`SYMV`	BLAS routine: **sy**mmetric **m**atrix **v**ector multiplication
`POTRF`	LAPACK routine: **po**sitive definite matrix **tr**iangular (i.e. Cholesky) **f**actorization
`TRSM`	LAPACK routine: **tr**iangular matrix **s**olve applied on a **m**atrix
`GESVD`	LAPACK routine: **ge**neral matrix **s**ingular **v**alue **d**ecomposition
`PZHEEVD`	ScaLAPACK routine: **p**arallel double complex (**Z**) **He**rmitian **e**igen**v**alue **d**ecomposition
`PZHEEVR`	ScaLAPACK routine: **p**arallel double complex (**Z**) **He**rmitian **e**igen**v**alue and eigenvector computation in a **r**ange
`PDSSTRD`	ScaLAPACK-like routine: **p**arallel **d**ouble **s**kew-**s**ymmetric **tri**d**i**agonalization

CHAPTER 1

INTRODUCTION

1.1 The unreasonable effectiveness of mathematics in algorithm development

In an article published in 1960 renowned physicist Eugene Wigner muses on the remarkable connection between physics and mathematics [183]. Mathematics seems to provide a perfect tool for describing fundamental laws of nature. According to Wigner, this seems "unreasonably effective" as much of mathematics was developed in disregard for possible applications. Instead, a core motive was to strive for "a sense of formal beauty". He concludes with a widely cited sentiment:

> *The miracle of the appropriateness of the language of mathematics for the formulation of the laws of physics is a wonderful gift which we neither understand nor deserve.*

Wigner acknowledges the twofold role of mathematics in describing the physical world. As becomes apparent in the quote, he emphasizes its role as a language to describe fundamental laws. Its other role is to serve as a tool to evaluate said fundamental laws, in order to make predictions for specific situations. In particular, he refers to the mathematical description of quantum mechanics, which has only existed for 30 years at the time. The memories on the unique rush of major developments at the beginning of the century in this regard are still fresh. Wigner hopes for this success story to continue with similar breakthroughs, e.g. in the unification of quantum mechanics and general relativity. However, he also acknowledges that this is not guaranteed. Indeed, finding a *Theory of Everything* is a problem yet to be solved gratifyingly.

The second role of mathematics is to compute trajectories of reality resulting from the existing laws for given conditions, "merely serving as a tool", as Wigner puts it. From his words one may falsely conclude that the evaluation of the Schrödinger equation, the center piece of modern quantum mechanics, is a trivial task. Nothing could be further from the truth.

An analytic solution of the Schrödinger equation is only possible for a few toy problems and hydrogen forming the simplest molecule conceivable. Approximate solution

schemes suffer from the curse of dimensionality: For a molecule with N electrons, a partial differential equation in $3N$ dimensions needs to be solved.

The focus of research expanded in the six decades following Wigner and now also includes computational methods themselves as subjects of research. The present thesis is situated in this field. The rise of computing power motivated the development of algorithms to simulate more complex systems. Tangible results were delivered and in turn motivated the deployment of more and more powerful machines up to the massive supercomputers of today [172].

In Wigner's time, it must have seemed utterly infeasible to compute structural, chemical or optical properties of materials not yet existent in reality, but only in form of a sum of parameters fed to a computer program. In the research landscape of today this is being done in form of electronic structure calculations with great success, using the strongest supercomputers ever built.

Various electronic structure methods play an important role in current and future technological advances. In our age, a major focus is defined by the great challenges posed by the threat of climate change. Obvious applications of electronic structure theory include the study of new materials [176] for building solar cells. Pure perovskite or perovskite silicon tandem solar cells show a higher efficiency than their classic pure silicone counter parts [9]. Other applications related to the energy revolution include the study of storage technologies, or finding catalysts for the creation of green hydrogen or other power-to-gas processes. These topics are examples of the growing interest in *in silico* experiments in quantum chemistry.

Another major challenge was posed by the recent outbreak of the COVID-19 pandemic. Simulations of molecular processes aim for a better understanding of the involved biological structures and mechanism or evaluate substances with respect to their clinical potential [101].

With respect to this new context, i.e. the study of computational methods, Wigner's quoted sentiment still holds. Algorithms are developed and analyzed with the help of mathematical ideas that originated independently of the specific algorithmic context. Which mathematical concepts turn out to be useful depends on the level of abstraction chosen to describe a specific problem.

In order to successfully apply mathematics, abstractions of some degree have to be performed. The resulting mathematical problem can be considered detached from one specific context and is applicable to a more general range of settings. Generally, the process of abstraction will uncover mathematical concepts which have been studied in depth on their own, or with respect to other application contexts.

The degree of abstraction may be chosen with a certain level of freedom and there is a trade-off to be considered. On the one hand, if methods for a very abstract problem formulation are found, they can be applied in many contexts, which is an advantage. On the other hand, if certain specific aspects are kept, they can be exploited in order to improve general methods. Algorithms may become more accurate or show performance benefits at an implementation level.

Most electronic structure methods at their core solve a linear matrix eigenvalue

problem, which in the given context can be considered a high level of abstraction.[1] Over the last century, the standard eigenvalue problem has been studied in depth, and great algorithms have been developed for its solution. Optimized implementations within standard software are available.

Ground state methods, in general, lead to eigenvalue problems involving symmetric matrices. While it is possible to ignore the symmetry, it would be a mistake of cardinal importance. Because symmetry is an extremely common structure in eigenvalue problems, perhaps even more common than the absence of symmetry, there are many algorithms and implementations available [83].

We consider the computation of optical properties. Various methods of abstracting the problem at hand once again lead to a structured eigenvalue problem. The structure is slightly more complex than symmetry and in current implementations it is often ignored. This thesis aims to fill this gap in order to improve performance and accuracy at the back end of electronic structure codes.

The starting point is the *Bethe-Salpeter equation* (BSE), which is why the following is called the *Bethe-Salpeter eigenvalue problem*:

$$\begin{gathered} Hv = \lambda v, \\ H = \begin{bmatrix} A & B \\ -\overline{B} & -\overline{A} \end{bmatrix} = \begin{bmatrix} A & B \\ -B^{\mathsf{H}} & -A^{\mathsf{T}} \end{bmatrix}, \\ A = A^{\mathsf{H}},\ B = B^{\mathsf{T}} \in \mathbb{C}^{n\times n}. \end{gathered} \tag{1.1}$$

We call H a *Bethe-Salpeter matrix*. In this thesis, we are interested in finding all eigenvectors $v \in \mathbb{C}^{2n}$ and all corresponding eigenvalues $\lambda \in \mathbb{C}$ for given matrices A and B. For crystalline systems, the given periodicity can be exploited to form a slightly different variant:

$$\begin{gathered} Hv = \lambda v, \\ H = \begin{bmatrix} A & B \\ -B & -A \end{bmatrix} = \begin{bmatrix} A & B \\ -B^{\mathsf{H}} & -A^{\mathsf{H}} \end{bmatrix}, \\ A = A^{\mathsf{H}},\ B = B^{\mathsf{H}} \in \mathbb{C}^{n\times n}. \end{gathered} \tag{1.2}$$

Here, both blocks A and B are Hermitian. This contrasts with B being complex symmetric in the original variant (1.1). Throughout this thesis we call (1.1) Bethe-Salpeter eigenvalue problem of form I, and the alternative setup (1.2) Bethe-Salpeter eigenvalue problem of form II.

When the matrix of form I is constructed using real basis functions, both forms coincide and can be tackled with the same algorithms.

[1] Of course, even higher levels of abstractions are possible, for example in the form of generalized, polynomial or general non-linear eigenvalue problems. However, their consideration is not required to make the point given in the text. Choosing a general-purpose linear eigensolver, unsuited for the specific problem, in a higher level algorithm is a real danger, because well-developed implementations are available and easy to use. Developers, who have not considered the specific structure at hand, may opt for a black-box solver and suffer from an unnecessary loss of performance or accuracy. The accidental choice of a non-linear solver is not a realistic scenario, as they are not included in popular linear algebra packages.

This is the level of abstraction we focus on in this thesis, i.e. we do not make further assumptions on the construction of A and B. The only further assumption we use in some parts of this thesis is for matrices of form I

$$\begin{bmatrix} A & B \\ B^{\mathsf{H}} & A^{\mathsf{T}} \end{bmatrix} > 0, \tag{1.3}$$

i.e. that the matrix, where the signs of the lower half have been switched, is Hermitian positive definite. A similar condition for matrices of form II is given by

$$\begin{bmatrix} A & B \\ B & A \end{bmatrix} > 0. \tag{1.4}$$

The later parts of the thesis focus on the property of H to be *pseudosymmetric*. Here, we first ignore the more specific structure of H, i.e. the fact that the upper left and lower right block are related, and that B is symmetric, and develop a general method for pseudosymmetric eigenvalue problems. Due to the additional abstraction step, this method has the potential to be applicable in other contexts. The more specific aspects of H can be used to improve the abstract method with respect to performance and accuracy.

In the process of algorithm development, we consider the computation of generalized polar decompositions. The concepts of pseudosymmetry and generalized polar decompositions have not been developed with applications of electronic structure theory in mind. But, to say it with Wigner's words, they turn out to be "unreasonably effective" as a tool in devising structure-preserving algorithms.

1.2 Outline of this thesis

This work starts by giving preliminaries useful for understanding the remainder of the thesis. We give insights into the fields of numerical linear algebra and high performance computing in Chapter 2. The presented tools are utilized in later chapters in order to develop algorithms where the specific structures given in (1.1), (1.2), (1.3) and (1.4) are exploited.

In Chapter 3 we give some background information as to how the structure in (1.1) arises from certain integral equations, such as the Bethe-Salpeter equation, with certain choices to perform approximations.

Chapter 4 compiles existing and new results on the consequences of the Bethe-Salpeter structure given in (1.1) (form I) as well as the alternative structure given by crystalline systems (1.2) (form II). It becomes clear why the alternative setup is advantageous and should be pursued when possible. Here, it is possible to transform the problem such that a Hermitian or pseudo-Hermitian eigenvalue problem of size $n \times n$ is solved instead of larger $2n \times 2n$ problem.

If this is not possible, Chapter 5 provides a way to solve (1.1) in a high performance setting. The use of supercomputers becomes necessary because the dimension n of the

Bethe-Salpeter matrix grows quadratically with the number of considered basis states. For complex materials, many basis states are necessary to guarantee accurate results leading to extremely large matrices. We extend the massively parallel ELPA library for skew-symmetric matrices and use this feature to solve a Bethe-Salpeter eigenvalue problem of size 51 200 on up to 512 cores with good scalability.

Chapter 6 explores, how the results given in Chapter 4 can be translated into algorithms and gives numerical results. The chapter focuses on matrices of form II (1.2). This form also results from studying molecules, as the resulting Bethe-Salpeter matrices may be real in that case. We propose and compare various algorithms and show that a typically used matrix square root can be substituted by a much cheaper Cholesky factorization. The condition (1.4) may or may not hold, as both cases are considered here.

Chapters 7 and 8 explore two tools which are used in Chapter 9 to extend the class of spectral divide-and-conquer methods for structured matrices. Generalized QR decompositions (Chapter 7) and generalized polar decompositions (Chapter 8) are interesting mathematical concepts worth to be studied in their own right. In these three chapters (Chapter 7 to Chapter 9), we develop algorithms with a focus on numerical stability as well as parallelizability and communication avoidance. The last two aspects are important to assure that the proposed methods are suitable for high performance computing.

Chapter 10 summarizes the thesis and gives an overview on the proposed algorithms and under which conditions they are suitable. We reflect on the considered approaches and propose focus points for future research.

CHAPTER 2

MATHEMATICAL PRELIMINARIES AND HIGH PERFORMANCE COMPUTING

Contents

In this chapter, we collect some known results in (numerical) linear algebra on which the results and algorithms presented in this thesis are based. We give some background information regarding concepts and software in high performance computing, to facilitate a broader understanding of the presented ideas. For a more detailed treatment we refer to basic literature such as [83, 165, 173].

2.1 Structured matrices

The main focus of this thesis is to develop new algorithms for structured matrices. If the structure of a matrix is preserved and exploited over the course of a computation, this can have positive effects on the achieved accuracy and performance of an algorithm [109]. A useful way of categorizing various occurring structures is given in form of non-standard scalar products.

We introduce the notation and concepts following [117]. The authors in [46, 47, 125] treat scalar products induced by Hermitian matrices, while in [96, 97] a more general treatment is provided. The group theoretical foundations are laid out in [75] and used to provide a systematic derivation of a large class of matrix decompositions.

A nonsingular matrix M defines a scalar product on $\mathbb{C}^n$ $\langle ., . \rangle_M$, which is a bilinear or sesquilinear form, given by

$$\langle x, y \rangle_M = \begin{cases} x^{\mathsf{T}} M y \text{ for bilinear forms,} \\ x^{\mathsf{H}} M y \text{ for sesquilinear forms,} \end{cases} \tag{2.1}$$

for $x, y \in \mathbb{C}^n$.

Definition 2.1:
For a matrix $A \in \mathbb{C}^{n\times n}$, $A^{\star_M} \in \mathbb{C}^{n\times n}$ denotes the adjoint with respect to the scalar product defined by M. This is a uniquely defined matrix satisfying the identity

$$\langle Ax, y\rangle_M = \langle x, A^{\star_M} y\rangle_M$$

for all $x, y \in \mathbb{C}^n$. We call $A^{\star_M}$ the M-adjoint of A. ◊

It holds

$$A^{\star_M} = M^{-1}A^*M,$$

where $.^*$ can refer to the transpose $.^{\mathsf{T}}$ or Hermitian transpose $.^{\mathsf{H}}$, depending on whether a bilinear or a sesquilinear form is considered.

Definition 2.2:
We call the matrix $A \in \mathbb{C}^{n\times n}$

1. M-orthogonal if $A^{\star_M} = A^{-1}$ (given the inverse exists),
2. M-self-adjoint if $A = A^{\star_M}$,
3. M-skew-adjoint if $A = -A^{\star_M}$. ◊

M-orthogonal matrices are exactly those that leave the structure of M-self-adjoint or M-skew-adjoint matrices intact when applied as a similarity transformation.

Lemma 2.3:
Let T be an M-orthogonal matrix and A be self- or skew-adjoint with respect to a scalar product induced by M. Then the transformed matrix

$$\hat{A} = T^{-1}AT$$

is again self- or skew-adjoint with respect to M. ◊

Proof. We show that the transformed matrix $\hat{A} = T^{-1}AT$ is self-, respectively, skew-adjoint, i.e. that it holds $M\hat{A} = \pm\hat{A}^*M$. We use $MA = A^*M$, $MT^{-1} = T^*M$ and $MT = T^{-*}M$ to see

$$\begin{aligned} M\hat{A} &= M(T^{-1}AT) = T^*MAT = \pm T^*A^*MT \\ &= \pm T^*A^*T^{-*}M = \pm(T^{-1}AT)^*M = \pm\hat{A}^*M. \end{aligned}$$ □

For $M = I_n$ the scalar product defined in (2.1) is just the Euclidian scalar product of two vectors.

The most ubiquitous form of a structured matrix is the symmetric (or Hermitian) matrix, where a self-adjoint matrix A fulfills $A = A^*$. There is virtually no application field of linear algebra where these matrices do not play a central role. Exploiting this

structure is a key element in popular algorithms for various tasks, such as the QR, Lanczos or Conjugate Gradient algorithm [83].

The skew-adjoint counterpart is given by skew-symmetric matrices, fulfilling $A = -A^*$. An I_n-orthogonal matrix Q is just an orthogonal (or unitary) matrix fulfilling $T^{-1} = T^*$. Orthogonal transformations preserve symmetry, as predicted by Lemma 2.3.

Allowing negative signs on the diagonal of the inducing matrix M leads to a generalization of the Euclidian scalar product, which plays a central role in this thesis.

Definition 2.4:
A matrix

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}, \quad \sigma_j \in \{-1, 1\} \text{ for } j = 1, \ldots, n,$$

is called a *signature matrix*. ◊

Definition 2.5:
A matrix $A \in \mathbb{K}^{n\times n}$ is called pseudosymmetric (pseudo-Hermitian) if there exists a signature matrix Σ, such that A is self-adjoint with respect to the bilinear form (sesquilinear form) induced by Σ. ◊

Equivalently, pseudosymmetry can be defined by

$$\begin{aligned} &\Sigma A = (\Sigma A)^* \\ \Leftrightarrow\ &A\Sigma = (A\Sigma)^*. \end{aligned}$$

This is the case if and only if A is symmetric up to sign changes of certain rows or columns. Pseudo-skew-symmetry can also be defined but does not play an important role here.

Given a particular Σ, pseudosymmetry is preserved by Σ-orthogonal matrices. However, we will see later, that it makes sense to relax this condition for developing reasonable algorithms and allow permutations on the diagonal of Σ.

A specific signature matrix used to describe the structure of Bethe-Salpeter matrices is denoted

$$K_n = \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix}$$

in this thesis.

Another important structure-defining scalar product is induced by the skew-symmetric matrix

$$J_n = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}.$$

When the dimension is not important or clear from the context, we drop the index and use $K = K_n$ and $J = J_n$.

Real matrices, that are self-adjoint with respect to the bilinear form and complex matrices, that are self-adjoint with respect to the sesquilinear form induced by J, are called *Skew-Hamiltonian* [37]. They can equally be defined by

$$\begin{aligned} JA &= -(JA)^* \\ \Leftrightarrow AJ &= -(AJ)^*. \end{aligned}$$

This is the case if and only if A shows a certain block structure

$$A = \begin{bmatrix} B & C \\ D & B^* \end{bmatrix}, \quad C = -C^*, \ D = -D^*.$$

Matrices that are skew-adjoint with respect to J are called Hamiltonian matrices. They fulfill (and can be defined by)

$$\begin{aligned} JA &= (JA)^* \\ \Leftrightarrow AJ &= (AJ)^*. \end{aligned}$$

This is the case if and only if A has a block structure of the form

$$A = \begin{bmatrix} B & C \\ D & -B^* \end{bmatrix}, \quad C = C^*, \ D = D^*.$$

Hamiltonian matrices and their corresponding eigenvalue problems naturally arise in control theory and model order reduction, in particular in the solution of algebraic Riccati equations [126, 28]. Skew-Hamiltonian matrices also play an important role in this context. This is because a squared Hamiltonian matrix yields a skew-Hamiltonian matrix [174] and a Hamiltonian matrix can be extended to form a skew-Hamiltonian matrix of twice the size with related eigenvalues and eigenvectors [39]. The skew-Hamiltonian structure can be exploited more effectively than the Hamiltonian structure, which is used to formulate structure-preserving algorithms.

J-orthogonal matrices are also called *symplectic*. A symplectic matrix S fulfills

$$S^* J S = J$$

and preserves (skew-)Hamiltonian structure when applied as a similarity transformation as in Lemma 2.3.

For real matrices that permit one of the structures in Definition 2.2, there is no reason to distinguish between bilinear and sesquilinear form induced by M in (2.1). For complex matrices on the other side, the structures induced by the bilinear and by the sesquilinear form are not the same. Just as a complex matrix might be symmetric (not Hermitian), a complex matrix can be self-adjoint with respect to the bilinear form induced by J. This leads to the class of so-called J-symmetric matrices [34].

The structural concepts emerging from generalized scalar products can be generalized to work for rectangular, non-square matrices. In particular, we are interested in a notion generalizing "orthogonality" of a rectangular matrix. This refers to a matrix $Q \in \mathbb{C}^{m\times n}$ with orthogonal columns, i.e. fulfilling $Q^*Q = I_n$.

As two vector spaces of different dimensions now play a role, two distinct scalar products are considered. We give some clarifying notation following [97]. For a matrix $A \in \mathbb{C}^{m\times n}$, $A^{\star_{M,N}} \in \mathbb{C}^{n\times m}$ denotes the adjoint with respect to the scalar products defined by the nonsingular matrices $M \in \mathbb{C}^{m\times m}$, $N \in \mathbb{C}^{n\times n}$. This matrix is uniquely defined by satisfying the identity

$$\langle Ax, y\rangle_M = \langle x, A^{\star_{M,N}} y\rangle_N$$

for all $x \in \mathbb{C}^n, y \in \mathbb{C}^m$. We call $A^{\star_{M,N}}$ the (M,N)-adjoint of A and it holds

$$A^{\star_{M,N}} = N^{-1}A^*M. \tag{2.2}$$

Definition 2.6:

1. We call a matrix $A \in \mathbb{C}^{m\times n}$ (M,N)-*orthogonal* or an (M,N)-*isometry* with respect to the scalar products induced by M and N when

$$A^{\star_{M,N}}A = I_n.$$

2. A matrix A is called a *partial* (M,N)-*isometry* when

$$AA^{\star_{M,N}}A = A. \qquad \lozenge$$

For an (M,N)-isometry A, it holds that $m \geq n$ and that A has full column rank. Partial (M,N)-isometries are generalizations of this concept. A partial (M,N)-isometry A does not need to have full column or row rank. For example, $A = Q_1Q_2^{\mathsf{T}}$ for orthogonal $Q_1 \in \mathbb{C}^{m\times k}$, $Q_2 \in \mathbb{C}^{n\times k}$, $k < m$, $k < n$, is a partial (I_m, I_n)-isometry.

2.2 Eigenvalues and eigenvectors

In the previous section we introduced structured matrices. We stated our aim to develop algorithms that preserve and exploit the given structure. These algorithms are supposed to compute eigenvalues and eigenvectors.

Definition 2.7 (Eigenvalues and eigenvectors [173]):
Given a matrix $A \in \mathbb{C}^{n\times n}$, $\lambda \in \mathbb{C}$ is called an eigenvalue of A, and a nonzero vector $v \in \mathbb{C}^n$ is called an eigenvector of A corresponding to λ if it holds

$$Av = \lambda v. \qquad \lozenge$$

The eigenvalues of a matrix are given by the roots of its characteristic polynomial, i.e. for an eigenvalue λ it holds

$$\det(A - \lambda I) = 0.$$

The set of all eigenvalues of A is called the spectrum of A and is denoted $\Lambda(A)$.

A naive approach for computing the eigenvalues of a matrix explicitly forms the characteristic polynomial. Closed form solutions are generally not available for polynomials of a degree higher than four. The roots of the polynomial could be computed via a numerical scheme such as Newton's method. This is also a bad idea because polynomial root-finding is in general an ill-conditioned problem and practically impossible for polynomials of high order.

If A has n linearly independent eigenvectors $v_1, \ldots, v_n$ corresponding to $\lambda_1, \ldots, \lambda_n$ (i.e. it is *nondefective*), A can be *diagonalized* [165]. With the eigenvectors arranged as columns of a matrix $V = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}$ it holds

$$V^{-1}AV = \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}.$$

In Section 2.1 we introduced symmetric (respectively Hermitian) matrices as a fundamentally important structure in applied linear algebra. The spectral structure (i.e. the properties of eigenvalues and eigenvectors) makes the symmetric eigenvalue problem particularly nice to work with.

Theorem 2.8 (Spectral Theorem [165]):
A Hermitian matrix $A \in \mathbb{C}^{n\times n}$ has only real eigenvalues and is unitarily diagonalizable, i.e. there exists a matrix $Q \in \mathbb{C}^{n\times n}$, such that $Q^*Q = I_n$ and

$$Q^*AQ = \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \in \mathbb{R}^{n\times n}. \qquad \Diamond$$

The eigenvectors of a Hermitian matrix form an orthogonal set. Q in Theorem 2.8 contains the eigenvectors of A, which are scaled to be orthonormal.

A k-dimensional linear subspace $\mathcal{V} \subseteq \mathbb{C}^n$ can be represented by its basis set $v_1, \ldots, v_k$ forming a matrix $V = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$. $\mathcal{V}$ is called an invariant subspace of A if it holds

$$Av \in \mathcal{V} \quad \forall v \in \mathcal{V}.$$

In matrix notation, an invariant subspace of A is encoded in $V \in \mathbb{C}^{n\times k}$ if it holds

$$AV = VD$$

for an arbitrary $D \in \mathbb{C}^{k\times k}$. If A is diagonalizable, its invariant subspaces are spanned by subsets of its eigenvectors.

We now consider the field $\mathbb{K}$ as a placeholder for $\mathbb{C}$ or $\mathbb{R}$. This reflects that in the presented decompositions, no complex arithmetic is needed if the original matrix was real. We use $.^*$ to refer to $.^{\mathsf{T}}$ or $.^{\mathsf{H}}$.

2.3 Tools and decompositions

In this section we present some computational tools which form the backbone of many algorithms in numerical linear algebra. We start with Householder transformations and Givens rotations. They are orthogonal transformations and can be used to introduce zeros in matrices. Orthogonal transformation have a perfect condition number of 1. This means that relative errors (for example rounding errors) are not inflated when they are applied, making them very attractive for numerical computations.

Theorem 2.9 (Householder transformations [83]):
A *Householder transformation* or *Householder reflection* is a matrix of the form

$$H(v) = I - \beta v v^*.$$

Given a vector $a \in \mathbb{K}^n$, choosing $v = \texttt{house}(a) := a \pm e^{\mathrm{i}\phi}\|a\|_2 e_1$ with $\phi = \arg a_1$ and $\beta = \frac{2}{v^* v}$ defines a Householder transformation such that

$$H(v)a = \mp e^{\mathrm{i}\phi}\|a\|_2 e_1. \qquad \Diamond$$

In practice (see BLAS/LAPACK in Section 2.4) slightly altered variants of the Householder transformation are in use. v is scaled such that $v_1 = 1$ in order to store it conveniently. The complex variant may be slightly altered to result in $H(v)a = \|a\|_2 e_1$ [100]. This is achieved by a complex β, which leads to H being non-Hermitian.

Theorem 2.10 (Givens rotations [83]):
A *Givens rotation* is a matrix of the form

$$G(\theta, \phi) = \begin{bmatrix} c & s \\ -\overline{s} & c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta)e^{i\phi} \\ -\sin(\theta)e^{-i\phi} & \cos(\theta) \end{bmatrix} \in \mathbb{K}^{2\times 2}.$$

Given a vector $a = \begin{bmatrix} u \\ v \end{bmatrix} \in \mathbb{K}^2$,

$$\cos(\theta) = \frac{\|u\|_2}{\sqrt{\|u\|_2^2 + \|v\|_2^2}} \qquad \sin(\theta) = \frac{-\|v\|_2}{\sqrt{\|u\|_2^2 + \|v\|_2^2}}$$

and $\phi = \arg(u) - \arg(v)$, it holds

$$G(\phi)^* a = \begin{bmatrix} \|a\| e^{i\arg(u)} \\ 0 \end{bmatrix}$$

and $G^* G = I_2$. $\Diamond$

In the practical computation and application of G, there is no need to ever use trigonometric functions. For details on the implementation of both kinds of transformations, see [83]. Householder transformations or Givens rotations can be used to transform a matrix such that the first column becomes a multiple of e_1, i.e. there are only zeros

below the first entry. A second transformation can transform the second column, such that there are only nonzero values except the first two entries, without destroying the zeros in the first column. Continuing this process column by column leads to the *QR decomposition*. This decomposition is one of the most versatile tools in numerical linear algebra.

Theorem 2.11 ((Thin) QR decomposition):
For each $A \in \mathbb{K}^{m\times n}$, $m \geq n$, there exists an orthogonal (respectively unitary) matrix $Q \in \mathbb{K}^{m\times n}$ and an upper triangular matrix $R \in \mathbb{K}^{n\times n}$ such that

$$A = QR \qquad \lozenge$$

Proof. As described above one finds orthogonal transformations $Q_1, \ldots, Q_n$, that introduce zeros below the diagonal. The product of orthogonal matrices is again orthogonal. This procedure yields a (full) QR decomposition

$$Q_n^* \cdots Q_1^* A = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad \Leftrightarrow \quad A = Q_0 \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad Q_0 = Q_1 \cdots Q_n.$$

Truncating the matrix Q_0 to its first n columns yields the thin QR decomposition. □

In practical implementations, because of performance considerations (see Section 2.4 below), it makes sense to accumulate the Householder transformations for some steps and then apply them all at once to transform the remaining matrix. This can be achieved with the storage-efficient variant of the QR decomposition [157], which is also explained in [145].

Theorem 2.12 (Storage-efficient QR decomposition [145]):
Let $Q = H_1 \cdots H_r$, $H_i \in \mathbb{K}^{m\times m}$ be a product of Householder matrices and let the rectangular matrix $V \in \mathbb{K}^{m\times r}$ contain the Householder vectors

$$v_i = \begin{bmatrix} 0 & \cdots & 0 & 1 & v_{i,i+1} & \cdots & v_{i,m} \end{bmatrix}^\mathsf{T} \in \mathbb{K}^m, \quad V = \begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix}.$$

Then an upper triangular matrix $T \in \mathbb{K}^{r\times r}$ exists, such that

$$Q = I_m - VTV^*. \qquad \lozenge$$

The theorem can be proven by induction using the following lemma. It describes how the columns of the matrix T can be constructed one after the other.

Lemma 2.13 (Accumulation of storage-efficient QR decompositions [145]):
Let $Q = I_m - VTV^* \in \mathbb{K}^{m\times m}$ be orthogonal with $V \in \mathbb{K}^{m\times j}$ and $T \in \mathbb{K}^{j\times j}$ upper triangular. Suppose that $H = I_m - \beta vv^*$ is a Householder matrix with $v \in \mathbb{K}^m$ and $\beta = \frac{2}{v^*v}$. Then

$$QH = I_m - V_+ T_+ V_+^*,$$

where

$$V_+ = \begin{bmatrix} V & v \end{bmatrix} \in \mathbb{C}^{m\times j+1}, \qquad T_+ = \begin{bmatrix} T & -\beta TV^*v \\ 0 & \beta \end{bmatrix}. \qquad \lozenge$$

The previously presented tools and the QR decomposition find their main applications in eigenvalue and least square problems. We now take our attention to decompositions of symmetric matrices, typically used for solving linear systems of equations given as $Ax = b$.

Gaussian elimination is the classic approach for solving these systems and yields the LU decomposition of A. If A is symmetric (respectively Hermitian) positive definite, it is guaranteed to exist and it holds $U = L^{\mathsf{H}}$, yielding the *Cholesky factorization*.

Theorem 2.14 (Cholesky factorization [83]):
Let $A \in \mathbb{K}^{n\times n}$ be symmetric (Hermitian) positive definite. Then there is a unique decomposition

$$A = R^* R,$$

where R is upper triangular and has positive diagonal entries. ◊

We used the upper triangular factor R in Theorem 2.14 instead of a lower triangular L to comply with Matlab syntax. Here, `R=chol(A)` yields the upper triangular factor.

If the considered matrix is symmetric, nonsingular but not positive definite, the LDL^{T} *factorization* may be used.

Theorem 2.15 (LDL^{T} factorization [83]):
Let $A \in \mathbb{K}^{n\times n}$ be symmetric (Hermitian) and let all its leading principal submatrices be nonsingular. Then there is a unique factorization

$$A = LDL^*,$$

where L is unit lower triangular and D is diagonal. ◊

The LDL^{T} factorization with a strictly diagonal D is typically not used in modern algorithms, as it becomes unstable when small diagonal values appear [14]. Instead, D is allowed to be block-diagonal with 1×1 and 2×2 blocks, and a pivoting scheme is employed [53].

Theorem 2.16 (LDL^{T} factorization with pivoting [83]):
Let $A \in \mathbb{K}^{n\times n}$ be symmetric (Hermitian). Then there is a factorization

$$A = PLDL^* P^{\mathsf{T}},$$

where P is a permutation and L is unit lower triangular. D is block-diagonal with 1×1 or 2×2 blocks and a real diagonal. ◊

We call this factorization "LDL^{T} factorization with pivoting" or "block LDL^{T} factorization" in order to distinguish it from the "diagonal LDL^{T} factorization". The additional degrees of freedom destroy the uniqueness property but allow for a more stable computation. Several backward stable algorithms have been developed (see [51, 52]) and well-established implementations are available in software packages such as LAPACK and Matlab [14, 73]. In Matlab the implementation is given as the `ldl` command.

The LDL^{T} decomposition with pivoting can be used to compute a decomposition with a signature matrix at the center.

Lemma 2.17 (Scaled LDL^T decomposition with pivoting):
Let $A \in \mathbb{K}^{n\times n}$ be symmetric (Hermitian) and nonsingular. Then there is a decomposition

$$A = L\Sigma L^*,$$

where Σ is a signature matrix and $L = PL_0V$, where P is a permutation, L_0 is unit lower triangular and V is an orthogonal, block-diagonal matrix with 1×1 and 2×2 blocks. ◊

Proof. With the LDL^T decomposition with pivoting from Theorem 2.16 we have

$$A = PL_0D_0L_0^*P^\mathsf{T}$$

with unit lower triangular L_0 and permutation P. D_0 is block-diagonal, Hermitian and has an eigenvalue decomposition $D = VD_0V^*$, where V is orthogonal and D_0 is real diagonal (see Theorem 2.8). Furthermore, V is block diagonal with the same structure as D, which is easily seen: Let D_i, $i = 1, \dots, n_D$ be the ith block of D, i.e.

$$D = D_1 \oplus \cdots \oplus D_{n_D},$$

and V_i define the eigenvalue decomposition of D_i, i.e.

$$V_i^*D_iV_i = D_{0,i}.$$

Then $V := V_1 \oplus \cdots \oplus V_{n_D}$ fulfills

$$V^*DV = D_0 = D_{0,1} \oplus \cdots \oplus D_{0,n_D}.$$ □

The QR decomposition and the Cholesky factorization (and its generalization, the LDL^T decomposition) seem unrelated at first sight. Part of this thesis is concerned with a hidden connection between them (see Chapter 7) and how it is exploited in a generalized context to uncover useful properties of a structured matrix.

A decomposition which uncovers lots of useful properties of a general (not necessarily square) matrix is the *singular value decomposition* (SVD).

Theorem 2.18 (The singular value decomposition (SVD) [83]):
Let $A \in \mathbb{K}^{m\times n}$. Then there exists a decomposition

$$A = U\Sigma V^*, \qquad U \in \mathbb{K}^{m\times m}, \quad V \in \mathbb{K}^{n\times n}, \quad \Sigma \in \mathbb{R}^{m\times n},$$

where

$$\begin{cases} \Sigma = \Sigma_0, & \text{if } m = n, \\ \Sigma = \begin{bmatrix} \Sigma_0 \\ 0 \end{bmatrix} & \text{if } m > n, \\ \Sigma = \begin{bmatrix} \Sigma_0 & 0 \end{bmatrix} & \text{if } m < n, \end{cases}$$

with

$$\Sigma_0 = \operatorname{diag}\big(\sigma_1, \dots, \sigma_{\min(m,n)}\big), \quad \sigma_1 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0,$$
$$U = \big[u_1, \dots, u_m\big], \; U^*U = I_m, \quad V = \big[v_1, \dots, v_n\big], \; V^*V = I_n.$$ ◊

While the QR decomposition yields an orthogonal basis for the column space of a matrix, the SVD yields orthogonal bases for the column as well as the row space. Furthermore, a weight σ_i is assigned to each of the basis vector pairs (u_i, v_i). By omitting those vectors with small weight we are able to keep the essential information of a linear mapping (given by the matrix) while drastically reducing the storage requirements. This low-rank approximation property (mathematically given in form of the Eckart–Young–Mirsky theorem [74, 129]) is what makes the SVD extremely useful in data science (in form of principal component analysis [89]) and model order reduction [27].

One of the many applications of the SVD is to provide a construction and a proof of the existence of the *polar decomposition*. The polar decomposition has interesting approximation properties and can be used to solve the orthogonal Procrustes problem [95].

Theorem 2.19 (The polar decomposition [95]):
Let $A \in \mathbb{K}^{m\times n}$ with $m \geq n$. Then there is a decomposition

$$A = UH, \quad U^*U = I_n, \quad H = H^* \geq 0. \tag{2.3}$$

H is uniquely given as the principal matrix square root $H = (A^*A)^{\frac{1}{2}}$. If A has full column rank, H is positive definite and U is unique. ◊

Proof. Let $A = U_s \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V_s^*$ be an SVD and $A = U_{s,0}\Sigma V_s^*$ with $U_{s,0} = U_s(:, 1:n)$ its truncated version. Then

$$A = \underbrace{U_{s,0}V_s^*}_{=:U} \underbrace{V_s\Sigma V_s^*}_{=:H} = UH$$

gives a polar decomposition of A. If A has rank $r < n$, any U of the form

$$U = U_{s,0} \begin{bmatrix} I_r & \\ & W \end{bmatrix} V_s^*,$$

with arbitrary matrix W, together with uniquely defined, rank deficient $H = V_s\Sigma V_s^*$ forms a polar decomposition. If A has full rank so has H and $U = AH^{-1}$ is uniquely defined. □

The matrix polar decomposition generalizes the well-known scalar polar decomposition: A complex number $z \in \mathbb{C}$ can be decomposed in the form

$$z = re^{\mathrm{i}\phi}. \tag{2.4}$$

Here, $r = (\overline{z}z)^{\frac{1}{2}}$ corresponds to the Hermitian factor H. $e^{\mathrm{i}\phi}$ is a rotation in the complex plane and corresponds to the orthogonal factor U in the generalized version.

Further generalizations consider non-Euclidian scalar products [96, 97], as given in (2.1). The efficient computation of a certain class of the resulting generalized polar decompositions is the goal of Chapter 8 of this thesis.

A related scalar decomposition is defined using the sign function, i.e.

$$\operatorname{sign}(z) = \begin{cases} 1, & \operatorname{Re}(z) > 0, \\ -1, & \operatorname{Re}(z) < 0, \end{cases} \qquad z \in \mathbb{C},\ z \notin \mathrm{i}\mathbb{R}.$$

A complex number $z \notin \mathrm{i}\mathbb{R}$ has a sign decomposition

$$z = (z^2)^{-\frac{1}{2}} \operatorname{sign}(z)\,. \tag{2.5}$$

If $z \in \mathbb{R}$, the decompositions (2.4) and (2.5) are identical because $z^{\mathsf{H}} z = z^2$. In this case, $\operatorname{sign}(z)$ and the orthogonal factor of the polar decomposition coincide.

This notion carries over to higher dimensions, where it applies to Hermitian matrices. The scalar sign function is generalized to form the *matrix sign function*, which is applied to matrices. For a detailed treatment see [95, Chapter 5]. The matrix sign function has a strong connection to the polar decomposition [93].

Definition 2.20 (Matrix sign function):
Let a square matrix A without purely imaginary eigenvalues have a Jordan decomposition

$$A = Z \begin{bmatrix} J_+ & \\ & J_- \end{bmatrix} Z^{-1},$$

where $J_+ \in \mathbb{K}^{n_+ \times n_+}$ contains Jordan blocks associated with eigenvalues with positive real part and $J_- \in \mathbb{K}^{n_- \times n_-}$ contains Jordan blocks associated with eigenvalues with negative real part. Then the matrix sign function is defined as

$$\operatorname{sign}(A) = Z \begin{bmatrix} I_{n_+} & \\ & -I_{n_-} \end{bmatrix} Z^{-1}. \tag{2.6}$$

◊

Lemma 2.21:
Let $A \in \mathbb{K}^{n \times n}$ be a symmetric (Hermitian) matrix without purely imaginary eigenvalues, $S = \operatorname{sign}(A)$ and $A = UH$ be a polar decomposition. Then it holds

$$S = U. \qquad \lozenge$$

Proof. S can be expressed as $S = A(A^2)^{-\frac{1}{2}}$ [93]. If A has no purely imaginary eigenvalues, it is also nonsingular. U is unique and can be expressed as $U = AH^{-1} = A(A^*A)^{-\frac{1}{2}}$. Clearly, these expressions are identical if $A = A^*$. □

It follows from (2.6) that the matrix sign function can be used to acquire projectors onto invariant subspaces associated with the positive and negative parts of the spectrum. This property is the reason it was originally proposed in order to solve algebraic Riccati equations [149].

Lemma 2.22:
Let $S = \text{sign}(A)$ exist for $A \in \mathbb{K}^{n\times n}$. $P_+ = \frac{1}{2}(I_n + S)$ and $P_- = \frac{1}{2}(I - S)$ are projectors onto the invariant subspaces associated with eigenvalues of A on the open right and open left half-plane respectively. ◊

In order to acquire projections onto invariant subspaces associated with other eigenvalue subsets, the matrix sign function of a shifted $A + \sigma I$ can be used. Another possibility is to transform A before computing the matrix sign function in order to acquire subspaces associated with almost arbitrary regions of the eigenvalue spectrum [19]. What makes the matrix sign function a widely used algorithmic tool, is that there exist iterative methods for its computation [95, 102]. The most simple one can be derived as a Newton iteration to find the roots of $f(X) = X^2 - I$,

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \quad X_0 = A. \tag{2.7}$$

The same idea applied to $f(X) = X^*X - I$ yields the Newton iteration

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-*}), \quad X_0 = A. \tag{2.8}$$

(2.8) converges to the orthogonal polar factor U in (2.3). Many other iterative methods for computing the matrix sign function can be repurposed to form an iteration for computing the orthogonal polar factor [95]. These include the dynamically weighted Halley and the Zolotarev iteration, which are explored in more detail in Chapter 8.

2.4 High performance computing

In this section we give a brief overview on basic software packages and their central role in high performance computing.

BLAS stands for *Basic Linear Algebra Subroutines* [134, 43]. They define a reference behavior and an API for the most basic linear algebra algorithms involving matrices and vectors. The routines are divided into three classes [83]:

1. Level 1 BLAS routines involve a linear amount of data and a linear amount of arithmetic, i.e. $\mathcal{O}(n)$, where n is the dimension of the input data, e.g. vector length. These typically include scalar and vector operations. An example is the `axpy` operation

$$y \leftarrow \alpha x + y.$$

2. Level 2 BLAS routines involve a bilinear amount of data and a bilinear amount of arithmetic, i.e. $\mathcal{O}(mn)$, where m and n are the dimensions of the input data. These typically include matrix-vector operations. An example is the general matrix-vector product (`gemv` operation)

$$y \leftarrow \alpha Ax + \beta b.$$

3. Level 3 BLAS routines involve a bilinear amount of data and a trilinear amount of arithmetic, i.e. $\mathcal{O}(mnk)$, where m, n, k are the dimensions of the input data. These typically include matrix-matrix operations. An example is the general matrix-matrix product (`gemm` operation)

$$C \leftarrow \alpha AB + \beta C.$$

During the design of algorithms it is in most cases beneficial to maximize the portion of level 3 BLAS operations in comparison with the other levels in order to achieve high performance. The reason for this is the high arithmetic density of these operations. The amount of work ($\mathcal{O}(n^3)$ for square matrices) exceeds the amount of data that needs to be moved to the registers ($\mathcal{O}(n^2)$ for square matrices) by one order of magnitude. On modern processors this is the only way to achieve peak performance. Otherwise, performance is limited by the memory bandwidth [87, 185].

LAPACK (*Linear Algebra PACKage*) [13, 135] extends the idea of BLAS and includes more complex routines. These can for example be used to compute the decompositions presented in Section 2.3.

Reference implementations of BLAS and LAPACK are available at [134] and [135]. These define the specifications of the included routines but should not be used for software where the performance is relevant. Optimized BLAS implementations, that show the same behavior but have been tweaked at a low level to exploit a particular software architecture, are made available by chip vendors and open source initiatives [2, 3, 4, 175]. BLAS and LAPACK are usually shipped together.

These implementations are optimized to run on shared memory multi-core processors, such as the ones integrated in standard desktop PCs and laptops. Calling an optimized and multithreaded LAPACK or BLAS routine can already invoke a degree of parallelism on this level.

If several shared memory processors are connected via a network, one arrives at a distributed memory machine, also called computer cluster. The individual (shared memory) computers are called nodes. Current supercomputers interconnect hundreds to thousands of nodes [172].

The communication between nodes is realized by an implementation of the *Message Passing Interface* (MPI) standard [128]. As in the case of BLAS and LAPACK, vendor specific and open source libraries are available. The main idea is to start multiple instances of the same program running in parallel. The instances are called processes. Each process is aware of its unique ID and can behave differently according to that. The processes can communicate by passing messages. Multiple processes may run on the same node, so that they communicate via shared memory, or on different nodes, so that they communicate via the interconnect. The MPI library takes care of the technical details on how the message passing is realized on the operating system and the hardware level.

On top of MPI, libraries have been developed to realize the algorithms available in LAPACK, e.g. for eigenvalue computation, with the aim to be as convenient as the shared memory variants. ScaLAPACK (*Scalable Linear Algebra PACKage*) [42]

employs BLACS (*Basic Linear Algebra Communication Subroutines*), which uses MPI, and gives reference implementations for distributed memory versions of the LAPACK routines. Again, vendor-specific optimized libraries exist. The ELPA library [119] is an open source endeavor to provide an alternative to the eigensolvers in ScaLAPACK.

For distributed memory machines, the question arises on how to organize the storage of large dense matrices in memory. Each node holding a full copy might be impossible due to memory constraints and is wasteful, when one node only works on part of the matrix. The 2D block cyclic data layout is a one-size-fits-all solution to this problem introduced by ScaLAPACK and also used in ELPA. At the initialization stage of a program, the processes are arranged in form of an abstract $m_p \times n_p$ grid. For example, if a program is started to use 6 MPI processes using the console command

```
mpirun -np 6 [program]
```

a process grid may be defined (inside the program) in the form

$$\begin{pmatrix} P_{1,1} & P_{1,2} & P_{1,3} \\ P_{2,1} & P_{2,2} & P_{2,3} \end{pmatrix}.$$

A matrix A is divided into blocks A_{ij} of chosen size $m_b \times n_b$, e.g.

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} \\ A_{21} & A_{22} & A_{13} & A_{24} & A_{25} & A_{26} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} & A_{36} \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & A_{46} \end{bmatrix}.$$

Some blocks might be smaller than $m_b \times n_b$ when the dimensions of A are not a multiple of m_b and n_b. A matrix block $A_{i,j}$ belongs to process $P_{k,l}$, where

$$k = i \mod m_p, \qquad l = j \mod n_p.$$

In our example the process $P_{1,1}$ owns the matrix blocks $A_{1,1}$, $A_{1,4}$, $A_{3,1}$ and $A_{3,4}$. These matrix blocks are stored in form of one large array in the local memory of $P_{1,1}$. This way, one can easily benefit from fast BLAS level 3 performance if the algorithm is implemented in the right way.

The 2D block cyclic data layout is preferable to a non-cyclic layout because it guarantees better load balancing for a large variety of algorithms, while still having a blocked structure to facilitate BLAS level 3 on a local level.

We already presented two different approaches to realize parallelism on one node with shared memory: The use of multiple processes governed by MPI and the use of optimized BLAS libraries, which take care of the parallelism in the background. A third approach used in high performance computing is the *OpenMP* API [140]. Here, parallelism is realized within one process in the form of multithreading. The programmer determines parallel regions via compiler directives leading to a fork-join parallelism.

In recent years, *graphics processing units* (GPUs) and other forms of accelerators became as important as general purpose processors (*central processing units*, CPUs)

in the field of high performance computing. Originally invented to render complex graphics, GPUs are equipped with many more cores than CPUs but with less control infrastructure on the chip. This makes them well-suited for SIMD (single instruction multiple data) parallelism. The individual threads on the different GPU cores run independently and do the same computations but on different input data. By now GPUs have outgrown their original graphics purpose and have established themselves as central computational devices for linear algebra and machine learning workloads.

Nvidia is the current market leader in the field of GPUs and provides useful infrastructure to facilitate their programming. CUDA [136, 138] is a parallel computing platform and API. It allows the programmer to address the resources of a GPU from the serial code running on the CPU. In a widely adopted model, a CPU, for example on a node in a compute cluster, has access to one or multiple GPUs and offloads computations with high arithmetic intensity to the accelerator (i.e. the GPU). To realize operations in numerical linear algebra, the *cuBLAS* library [137] provides a BLAS-like interface on top of CUDA. Above, we pointed out the need for BLAS level 3 operations due to limited memory bandwidth. This applies even more in this context as the data in main memory first has to be transferred to the accelerator device.

We see that parallelism and data locality play an important role on many different levels. A programmer has a lot of freedom in choosing and combining the available tools such that an optimal performance for the given hardware can be achieved. The art of performance optimization takes place on the level of mathematical algorithm design, on the level of implementation, as well as on the level of a system's configuration.

CHAPTER 3

BASICS OF ELECTRONIC STRUCTURE THEORY

Contents

3.1 Introduction

The main goal of this thesis is to give insight and provide algorithms for eigenvalue problems arising in the description of excited states within electronic structure theory. Over the course of this project, it has turned out to be crucial to foster a mutually beneficial relationship between the fields of mathematics, in particular numerical linear algebra, and (solid-state) physics. In order to achieve this, one needs to bridge the existing gap between the fields. By this, we not only mean the basic domain knowledge necessary for understanding the nature of the problem, but also more subtle differences in the culture leading to different writing styles, as well as seemingly trivial things such as mathematical notation.

The power of applied mathematics lies in abstraction. It can derive deeper insights because a problem is detached from unimportant and distracting details obfuscating its true mathematical nature. The methods developed from these insights are applicable

to a wider range of problems because the abstract problem is more general and may contain other specific problems, perhaps even from completely different applications. The Bethe-Salpeter eigenvalue problem (1.1) is an abstraction in the sense that the "details" of how the block matrices A and B are constructed, and what the results mean in a physical sense, are declared unimportant. Further abstractions are made in Chapters 7 to 9, which may find applications in completely different fields.

However, our chosen level of abstraction, i.e. in form of the eigenvalue problem (1.1), is rather arbitrary. Determining which details are to be stripped away in an abstract formulation, is a big challenge. Details may seem unimportant to an applied scientist, and inaccessible for a mathematician due to a lack of domain knowledge. Those details can perhaps be exploited algorithmically, if both parties are aware of them.

This is why close interdisciplinary collaboration is needed. There is no strict border between the fields, where a concrete problem of physics turns into an abstract problem of mathematics. Rather, there is a whole border region whose landscape should be explored by experts from both sides, determining a good point for making abstractions.

This chapter aims to contribute towards making the field of electronic structure theory accessible for scientists with a perspective rooted in (numerical) linear algebra, but not necessarily with a background in physics. Mathematically speaking, this chapter is less rigorous than the following chapters where new results are presented. A mathematically sound introduction to quantum mechanics alone fills entire textbooks. We rather aim to give some context and intuition about where the eigenvalue problems treated in this thesis come from. We start by giving a concise overview on the ideas of quantum mechanics (Section 3.2). Methods for computing the ground state of a quantum system (Hartree-Fock and Density functional theory) are introduced in Section 3.3. Section 3.4 introduces the notion of electronic excitations and how related optical properties can be computed. We do not derive equations from many-body perturbation theory, such as the Bethe-Salpeter equation, but elaborate in Section 3.5 on how these equations lead to structured eigenvalue problems such as (1.1). Section 3.6 provides an alternative way of setting up a matrix eigenvalue problem for crystalline solids. This leads to a slightly different structure that can be exploited algorithmically in later chapters of this thesis.

3.2 A short introduction to quantum physics

In classical Newtonian mechanics an object such as an electron is characterized by a specific position and momentum which can in theory be measured with a 100% certainty. While Newton's laws, together with Einstein's relativity, accurately describe movements in the macroscopic world (see e.g. [141]), they fail to accurately predict results of experiments concerning microscopic quantities, such as the double-slit and the Stern-Gerlach experiment [85]. In response to this, quantum mechanics was developed in the 20th century. It replaced Newtonian mechanics as a fundamental theory of motion. While it is less intuitive to understand, it's predictions have been confirmed in countless experiments.

Every particle and most of reality can be described in terms of quantum mechanics. In this thesis we focus on electrons, as their behavior (the *electronic structure* of atoms, molecules and condensed matter) determines chemical and optical properties. An understanding of electronic structure can therefore lead to the development of new materials useful in industrial applications.

In the following we introduce some basic concepts of quantum mechanics, which can e.g. be found in [85] and [115]. We disregard spin in the following in order to achieve conceptual clarity. A more complete account can be found, e.g., in [153].

The quantum *state* of an electron is given in form of a wave function

$$\psi(x) \in \mathcal{H} = L^2(\mathbb{R}^3, \mathbb{C}) = \left\{ f : \mathbb{R}^3 \to \mathbb{C} \ : \ \int_{\mathbb{R}^3} |f(x)|^2 \, dx < \infty \right\}, \quad x \in \mathbb{R}^3. \tag{3.1}$$

$\mathcal{H}$ is referred to as the Hilbert space. The scalar product of two wave functions is defined as

$$\langle \psi | \phi \rangle = \int_{\mathbb{R}^3} \overline{\psi(x)} \phi(x) dx.$$

In the physics literature using $.^*$ to refer to the complex conjugation (and not necessarily to the conjugated transpose of a matrix) is very common.

$\psi(x) \in \mathcal{H}$ represents a physical state if it is normalized, i.e. we have

$$\langle \psi | \psi \rangle = 1. \tag{3.2}$$

From the scalar product notation stems another commonly used way to refer to the quantum state, introduced by Dirac [71]. A quantum state is expressed in form of a so-called ket $|\psi\rangle$. A ket refers to a vector living in the abstract Hilbert space, that is not yet expressed in a specific basis. We will see below, that the wave function (3.1) refers to the quantum state in a so-called position basis. The matching counterpart of a ket is a bra, denoted $\langle \psi |$. It is an element of the dual space $\mathcal{H}^*$, i.e. it is a linear functional $\langle \psi | : \mathcal{H} \to \mathbb{C}$. A bra applied on a ket forms a bra(c)ket, which explains the names, and is just a scalar product,

$$\langle \phi | \cdot | \psi \rangle = \langle \phi | \psi \rangle .$$

In addition to the concept of states we require the concept of linear *operators* to describe quantum mechanics. They act on elements of the Hilbert space and return another wave function,

$$Q : \mathcal{H} \to \mathcal{H}.$$

An operator is called Hermitian if it holds

$$\langle \psi | Q \phi \rangle = \langle Q \psi | \phi \rangle =: \langle \psi | Q | \phi \rangle , \qquad \forall \psi, \phi \in \mathcal{H}.$$

Hermitian operators represent *observables*, i.e. quantities that can be measured. The measurement provokes a *collapse of the wavefunction*, according to the Copenhagen

interpretation of quantum mechanics. Before the collapse, a measurable property can only be known in form of a probability distribution. After the collapse, the wave function instantaneously becomes an eigenfunction of the observable (given said eigenfunction is normalizable), and the corresponding eigenvalue is measured. The eigenfunction f_q fulfills the eigenvalue equation with eigenvalue q,

$$Qf_q = qf_q.$$

As predicted by Theorem 2.8, the eigenvalues of observables are real. Assuming a discrete spectrum $\Lambda(Q) = \{q_1, q_2, \dots\}$, the eigenfunctions $\{f_{q_1}, f_{q_2}, \dots\}$ corresponding to different eigenvalues are orthogonal, i.e. with normalization it holds

$$\langle f_{q_i}, f_{q_j} \rangle = \delta_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{3.3}$$

A measurement of q_n happens with a probability of $|c_n|^2$, where c_n is the coordinate of the quantum state $|\psi\rangle$ in the eigenfunction basis,

$$P(\text{observing } q_n \text{ after collapse}) = |c_n|^2, \quad \text{where } c_n = \langle f_{q_n} | \psi \rangle . \tag{3.4}$$

After the collapse, the quantum state is given by f_{q_n} and a second immediate measurement will yield the same result q_n with 100% certainty.

In case of a continuous spectrum, the eigenfunctions are not normalizable in the sense of (3.3). Let $z \in \mathbb{R}$ vary over $\mathbb{R}$, such that $q(z)$ gives the spectrum and $f_{q(z)}$ be the corresponding eigenfunction. The eigenfunctions can be normalized in the sense that

$$\langle f_{q(z)}, f_{q(\hat{z})} \rangle = \delta(z - \hat{z})$$

gives the well-known Dirac delta function. The Dirac delta function is defined as the distribution fulfilling

$$\delta(x) = \begin{cases} \infty & \text{if } x = 0, \\ 0 & \text{if } x \neq 0 \end{cases}, \quad \int \delta(x) dx = 1.$$

The "collapse" then refers to a narrow range around the measured quantity, depending on the measurement precision. The probability of getting a measurement in a certain interval is given via a probability density. For a given quantum state $|\psi\rangle$, the probability of getting a measurement result in the interval $[a, b]$ is

$$\begin{aligned} P(\text{observing } q \in [a,b] \text{ after collapse}) &= \int_a^b |c(z)|^2 \, dq(z), \\ &\text{where } c(z) = \langle f_{q(z)} | \psi \rangle . \end{aligned} \tag{3.5}$$

The role of the discrete index n in (3.4) is now played by the continuous variable z. Analogously to c_n, the function $c(z)$ is seen as containing the coordinates of $|\psi\rangle$ in the eigenfunction basis. The situation of a continuous spectrum arises for example when considering the position operator. To improve the clarity of notation we consider the one-dimensional problem in the following example, i.e. the particle is located on a line.

Example 3.1:
Let the quantum state of a particle be given by

$$\psi(x) \in \mathcal{H}^1 = L^2(\mathbb{R}, \mathbb{C}) = \left\{ f : \mathbb{R} \to \mathbb{C} \ : \ \int_{\mathbb{R}} |f(x)|^2 \, dx < \infty \right\}, \quad x \in \mathbb{R}.$$

The position operator $\hat{x}$ is given by

$$\begin{aligned} \hat{x} : \mathcal{H}^1 &\to \mathcal{H}^1, \\ \Psi(x) &\to x\Psi(x). \end{aligned}$$

It is obviously linear and Hermitian. All $z \in \mathbb{R}$ are eigenvalues and the (generalized) eigenfunctions are the delta functions

$$g_z(x) = \delta(x - z).$$

Now the wave function $\psi(x)$ can be interpreted as the coordinates of $|\psi\rangle$ in the basis given by the eigenfunctions of the position operator.

$$\psi(z) = \langle g_z | \psi \rangle = \int_{\mathbb{R}} g_z(x)\psi(x)dx.$$

Referring to (3.5), we see that $c(z) = \psi(z)$ and $q(z) = z$. Therefore, $|\psi(x)|^2 = \overline{\psi(x)}\psi(x)$ describes a probability distribution concerning the electron's position. This justifies the normalization condition (3.2): The probability of finding an electron *somewhere* in space is equal to one. ◊

We have seen in Example 3.1 how the state can be expressed with respect to position x. Just as well it can be represented in other bases such as the generalized eigenstates of the momentum operator. An extension to three dimensions is straightforward. Here, the generalized eigenfunctions of the position operator are given by delta functions defined on $\mathbb{R}^3$, and the position measurements are given as three-dimensional points in space.

The Hamiltonian operator H is an observable associated with the measurement of energy. In absence of a magnetic field it is given for an electron as

$$H : \psi(x) \to -\frac{\hbar^2}{2m_e}\Delta_x\psi(x) + V(x)\psi(x),$$

where $\hbar$ is the Planck's modified constant, m_e is the mass of an electron, and $V(x)$ is the potential energy function, describing the environment the electron is interacting with. In shorthand notation we write

$$H = -\frac{\hbar^2}{2m_e}\Delta_x + V(x).$$

From here on we use Hartree atomic units, i.e. set $\hbar = m_e = \frac{1}{4\pi\epsilon_0} = k_B = 1$, where $\frac{1}{4\pi\epsilon_0}$ is the Coulomb constant and k_B is the Boltzmann constant [160]. When results

are obtained from computations employing atomic units, they can easily be scaled to reflect proper units in a post-processing step. In this convention, the Hamiltonian operator becomes

$$H = -\frac{1}{2}\Delta_x + V(x), \tag{3.6}$$

The eigenvalue equation of this operator is the well-known time-independent Schrödinger equation

$$H\phi_E(x) = E\phi_E(x), \tag{3.7}$$

where $E \in \mathbb{R}$ is an eigenvalue of H, denoting possible energy levels of a system. The state corresponding to the lowest possible energy level is called the *ground state*.

The only missing puzzle piece for describing a quantum process concerns the evolution of a wave function in time. This behavior is described by the time-dependent Schrödinger equation

$$\mathrm{i}\frac{\partial\psi(x,t)}{\partial t} = H\psi(x,t). \tag{3.8}$$

We set

$$\psi(x,t) = \psi_E(x,t) = \phi_E(x)e^{-\mathrm{i}Et}, \tag{3.9}$$

and see that it solves (3.8). While this wave function does depend on t, the probability density associated with the electron's position does not:

$$|\psi(x,t)|^2 = \overline{\phi_E(x)}e^{\mathrm{i}Et}\phi_E(x)e^{-\mathrm{i}Et} = |\phi_E(x)|^2 .$$

The functions $\psi_E(x,t)$ defined in (3.9) are therefore called *stationary states*.

The spectrum of a Hamiltonian can have discrete as well as continuous parts. For now we assume that the spectrum of H is discrete. The eigenfunctions of the Hamiltonian (3.7) form a complete basis set, i.e. any initial state can be expressed as

$$\psi(x, t=0) = \sum_E c_E\phi_E(x).$$

Now the time evolution of the initial state according to the Schrödinger equation (3.8) is given by

$$\psi(x,t) = \sum_E c_E\psi_E(x,t) = \sum_E c_E\phi_E(x)e^{-\mathrm{i}Et}.$$

Getting solutions of the time-dependent Schrödinger equation (3.8) is therefore a trivial task, when the solutions of the time-independent Schrödinger equation (3.7) are known.

When n electrons are considered, the wave function has the form

$$\psi(x_1, x_2, \ldots, x_n) \in \mathcal{H}_n = L^2(\mathbb{R}^{3n}, \mathbb{C}). \tag{3.10}$$

Now, $|\psi(x_1, x_2, \ldots, x_n)|^2$ is associated with the probability density of finding electrons in positions x_1, $x_2, \ldots, x_n$. The Hamiltonian operator takes the form

$$H : \psi(x_1, \ldots, x_n) \to -\frac{1}{2}\sum_{i=1}^{n} \Delta_{x_i}\psi(x_1, \ldots, x_n) + V(x_1, \ldots, x_n)\psi(x_1, \ldots, x_n).$$

The solution of the eigenvalue problem (3.7) is highly non-trivial, in particular when more than one electron is involved.

In principal, all particles have a quantum nature and are described in form of a quantum state. However, atomic nuclei have a much higher mass, such that in comparison to the electrons, they can be regarded as a static potential. This is called the Born-Oppenheimer approximation and is applied in most of electronic structure theory. In this setting, the many-body Hamiltonian is given as [120, 144]

$$H = T + V_{EN} + V_{EE} + E_{NN}, \tag{3.11}$$

where T is the kinetic energy, V_{EN} is the potential created by the nuclei acting on the electrons, V_{EE} is the electron-electron interaction and E_{NN} is the classical nuclei-nuclei interaction. For n electrons and m nuclei with positions X_i and charges Z_i, $i = 1, \ldots, m$, they are given by

$$T = \sum_{i=1}^{n} -\frac{1}{2}\Delta_{x_i}, \quad V_{EN} = \sum_{j=1}^{m}\sum_{i=1}^{n} \frac{Z_i}{|x_i - X_j|},$$
$$V_{EE} = \sum_{j}^{n}\sum_{i}^{j-1} \frac{1}{|x_i - x_j|}, \quad E_{II} = \sum_{j}^{m}\sum_{i}^{j-1} \frac{Z_i Z_j}{|X_i - X_j|}.$$

Here, the nuclei-electron interaction V_{EN} is given as the bare nuclear Coulomb interaction but may be switched for a pseudopotential in numerical calculations to include the effect of core electrons. E_{II} is constant with respect to r_i and may also contain other terms contributing to the total energy. Complexity arises because the electrons are not independent of one another, i.e. each electron influences the electric potential determining the Hamiltonian operator.

3.3 Ground state methods in electronic structure theory

Computational methods are required to find approximate wave functions. For the ground state, i.e. the eigenstate corresponding to the lowest eigenvalue of the Hamiltonian, the Hartree-Fock (HF) method [170] and Density Functional Theory (DFT) [78] are being applied successfully.

3.3.1 Hartree-Fock

The Hartree-Fock methodology approximates the many-electron wave function as a product of independent, orthonormal wave functions, that has been antisymmetrized to fulfill the Pauli exclusion principle [170]. A many-body wave functions of this form is called a Slater determinant (see e.g. [104]). The ground state energy is given as the infimum of an energy functional, so that approximations of any form produce a ground state energy that is larger than or equal to the true one. A variational method is used to find the wave functions whose Slater determinant yields the lowest energy, i.e. the one closest to the true ground state energy. This is done by solving a nonlinear eigenvalue problem

$$\mathcal{F}(V)V = V\Lambda, \tag{3.12}$$

where V contains the eigenvectors and Λ is the diagonal matrix containing the eigenvalues. In actual computations, a wave function approximation is expanded in a finite-dimensional space and can be represented as a column vector. The Fock operator $\mathcal{F}$ is Hermitian and becomes a Hermitian matrix when expressed in finite-dimensional space. It depends on the eigenvectors, making the problem non-linear which results from the Hartree potential and the exchange operator. The Hartree-Fock equation (3.12) is typically solved in a series of Hermitian linear eigenvalue problems until self-consistency is achieved.

Hartree-Fock and its descendants are very popular in quantum chemistry as they provide good correspondence with experiments for many molecules. A major drawback of Hartree-Fock methods is that the interaction between electrons, called electronic correlation, is not completely taken into account.

This interaction has measurable influence in particular in crystalline systems, which is why in solid state physics *density functional theory* (DFT) is a more popular approach to compute ground state properties.

3.3.2 Density functional theory

Density functional theory is based on the following fact. For an accurate computation of the ground state, one does not need to know the complete many-body wave function (3.10), which is extremely demanding to compute or approximate [115]. Instead, one can use the *electron density*, which for a multidimensional wave function $\psi(r_1, \ldots, r_n)$ is given as

$$\rho(r) = n \int \cdots \int \overline{\psi(r, r_2, \ldots, r_n)}\psi(r, r_2, \ldots, r_n) dr_2 \ldots dr_n, \quad r, r_2, \ldots, r_n \in \mathbb{R}^3.$$

Hohenberg and Kohn [98] showed that the ground state wave function is completely determined by the ground state density. As in Hartree-Fock theory, DFT can be based on the expression of the ground state energy as the solution of an optimization problem, searching in the space of skew-symmetric (hence admissible) n-dimensional

wave functions $\mathcal{A}_n$.

$$E_{min} = \inf_{\psi \in \mathcal{A}_n} \langle \psi | H | \psi \rangle$$

We take the perspective of constrained optimization, which was developed by Levy [112] and Lieb [114]. E is expressed as a solution to an optimization problem on the set of admissible ground state densities. Given the formulation of the many-body Hamiltonian in (3.11), it holds

$$E_{min} = \inf_{\rho(r)} \inf_{\substack{\psi \in \mathcal{A}_n^0 \\ \psi \mapsto \rho(r)}} \langle \psi | T | \psi \rangle + E_H[\rho] + E_{xc}[\rho] + \int \rho(r) V_{ext}(r) dr + E_{II}. \tag{3.13}$$

Here,

$$E_H[\rho] = \frac{1}{2} \int \int \frac{\rho(r)\rho(r')}{|r - r'|} dr dr'$$

describes electron-electron repulsion and is called the Hartree energy. The remaining electron-electron interaction is captured in the exchange-correlation energy E_{xc}. The second (i.e. inner) infimum minimizes the resulting energy over all admissible wave functions ψ yielding a given density distribution $\rho(x)$ (denoted by $\psi \mapsto \rho(r)$). It can be shown that it is enough to consider the wave functions given by Slater determinants, which were introduced in the previous section. We call the set of all Slater determinants $\mathcal{A}_n^0$.

An n-dimensional Slater determinant is an anti-symmetrized product of n single particle wave functions. The optimization problem (3.13) can be reformulated, such that one is looking for the set of orthonormal wave functions $\psi_i(r)$ minimizing the Kohn-Sham energy $\mathcal{E}^{KS}$:

$$E_{min} = \inf_{\substack{\{\psi_i\}_{i=1}^n \\ \langle \psi_i | \psi_j \rangle = \delta_{ij}}} \mathcal{E}^{KS}(\{\psi_i\}) + E_{II},$$

$$\mathcal{E}^{KS}(\{\psi_i\}_{i=1}^n) = \frac{1}{2} \sum_{i=1}^n \int |\Delta_r \psi_i(r)|^2 dr + \int \rho(r) V_{ext}(r) dr + E_H[\rho] + E_{xc}[\rho].$$

Similar to the Hartree-Fock method one can solve the minimization problem by finding a stationary point of the corresponding Lagrangian. The resulting Euler-Lagrange equations are given by

$$H^{KS}[\rho]\varphi_i(r) = \sum_{j=1}^n \varphi_j(r) \lambda_{ij}, \qquad i = 1, \ldots, n, \tag{3.14}$$

where the operator

$$H^{KS}[\rho] = -\frac{1}{2}\Delta_r + V_{ext}(r) + V_H[\rho](r) + V_{xc}[\rho](r), \tag{3.15}$$

is the so-called Kohn-Sham Hamiltonian [107]. $V_H = \int \frac{\rho(r')}{|r-r'|} dr'$ is the Hartree potential and the functional derivative

$$V_{xc}[\rho] = \frac{\delta E_{xc}[\rho]}{\delta \rho}$$

is called the exchange-correlation potential. The matrix of Lagrange multipliers $[\lambda_{ij}]_{i,j=1}^n$ is Hermitian and can be diagonalized with an orthogonal transformation (cf. Theorem 2.8). This way, equation (3.14) is transformed into a non-linear eigenvalue problem, called the Kohn-Sham equation [107].

$$H^{KS}[\rho]\psi_i(r) = E_i^{KS}\psi_i(r), \qquad i = 1, \dots, n. \tag{3.16}$$

The eigenstates (eigenvectors) $\psi_i(r)$ are called Kohn-Sham orbitals. There is no point in explicitly computing the functions $\varphi_i(r)$ in (3.14), as they are just rotated versions of the set given by $\psi_i(r)$. Both sets yield the same density, i.e. they are equivalent and create the same Kohn-Sham Hamiltonian. The eigenvalue problem is non-linear because H_{KS} depends on the electron density, which in turn depends on the eigenvectors in form of

$$\rho(r) = \sum_{i=1}^{n} |\psi_i(r)|^2.$$

In order to solve the eigenvalue problem (3.16) to self-consistency, typically a series of Hermitian matrix eigenvalue problems is solved, representing the discretized, finite-dimensional form of the eigenvalue problem.

The Kohn-Sham Hamiltonian (3.15) has the same form as a classical Hamiltonian created by an external potential

$$V_{eff}(r) = V_{ext}(r) + V_H[\rho](r) + V_{xc}[\rho](r).$$

This means that the eigenvalue problem (3.16) is a time-independent Schrödinger equation and the Kohn-Sham orbitals can be seen as fictitious system of independent wave functions. Their combination in form of a Slater determinants yields the same electron density as the true solution of the original Schrödinger equation. Then the ground state energy must be the same.

The presented framework is in principle exact. All quantities are given explicitly with the exception of the exchange-correlation energy E_{xc} and consequently the exchange-correlation potential. The challenge in the successful application of DFT lies in the use of a suitable approximation for E_{xc}. The simplest and most widely used one is the local density approximation (LDA). It replaces the inhomogeneous system at point r with the homogeneous system of the same density. Surprisingly, this approximation often yields good results, even for very inhomogeneous systems [139].

Both Hartree-Fock and DFT compute eigenvalues and eigenstates of a fictitious system (see (3.16) for DFT). It is tempting to interpret these eigenvalues as excitation

levels (i.e. as eigenvalues of the original many-body Schrödinger equation), but this approach bears little justification and usually leads to wrong results [139].

The presented ground-state methods are unsuited for computing eigenstates corresponding to higher energy eigenvalues, called excited states. Excited states are required for the description of optical phenomena. The ground state may be used as a starting point for further computations as described in Section 3.4.

3.4 Computing optical properties

Electronic excitations are responsible for optical phenomena such as direct photoemission, inverse photoemission and absorption.

3.4.1 Optical properties and a Dyson-like equation

In this section, we compile equations for computing the frequency-dependent *reducible polarizability* $\chi(r, r', \omega)$ according to [139] and [155]. χ is needed to compute the dielectric function ϵ, which is used to compute the electron-loss spectrum. Furthermore, it can be used to compute the macroscopic dielectric function, which is needed for computing the optical absorption spectrum of solids. Alternatively, a modified polarizability $\overline{P}$ can be used to directly compute the macroscopic dielectric function. It is constructed from a modified Coulomb potential where the long range term is truncated. Similar equations hold for $\overline{P}$ as for χ and they can be solved in a similar manner (see details in [139]).

In case of a system consisting of independent (quasi-)particles, χ is constructed in form of

$$\begin{aligned}\chi_0(r, r', \omega) = &\sum_{\substack{i \in I_{occ}, \\ j \in I_{unocc}}} \frac{\varphi_i(r)\overline{\varphi_j}(r)\varphi_j(r')\overline{\varphi_i}(r')}{\omega - \omega_{ij}} \\ &+ \sum_{\substack{i \in I_{unocc}, \\ j \in I_{occ}}} \frac{\varphi_i(r)\overline{\varphi_j}(r)\varphi_j(r')\overline{\varphi_i}(r')}{-\omega - \omega_{ji}},\end{aligned} \tag{3.17}$$

where I_{occ} is the index set for which φ_i, $i \in I_{\text{occ}}$ describes an occupied orbital and I_{unocc} is the index set for which φ_i, $i \in I_{\text{unocc}}$ describes an unoccupied (also called virtual) orbital. $\omega_{ij} = \epsilon_j - \epsilon_a$ is the energy difference of the states φ_j and φ_i.

A corresponding four point quantity is constructed in form of [155]

$$\begin{aligned}\chi_0(r_1, r_2, r_3, r_4, \omega) = &\sum_{\substack{i \in I_{occ}, \\ j \in I_{unocc}}} \frac{\varphi_i(r_1)\overline{\varphi_j}(r_3)\varphi_j(r_2)\overline{\varphi_i}(r_4)}{\omega - \omega_{ij}} \\ &+ \sum_{\substack{i \in I_{unocc}, \\ j \in I_{occ}}} \frac{\varphi_i(r_1)\overline{\varphi_j}(r_3)\varphi_j(r_2)\overline{\varphi_i}(r_4)}{-\omega - \omega_{ji}}.\end{aligned} \tag{3.18}$$

The poles of χ in the frequency domain are the excitation energies of the system. If the Kohn-Sham orbitals from ground state DFT (see Section 3.3.2) are used to construct χ_0, it can be seen as a very rough approximation for χ. It can serve as a starting point for computing χ in form of a Dyson-like equation

$$\chi(r, r', \omega) = \chi_0(r, r', \omega) + \int\int \chi_0(r, r_1, \omega) C(r_1, r_2) \chi(r_2, r', \omega) dr_1 dr_2. \tag{3.19}$$

The specific kernel C depends on the chosen computational method. The two possible methods are the approach following from linear response *time-dependent density functional theory* (TDDFT) and the approach based on the *Bethe-Salpeter equation* (BSE), following from many-body perturbation theory. Some details are given in Sections 3.4.2 and 3.4.3.

Any two point operator $W(r, r')$ can approximately be expressed as a matrix $W = [W_{kl}]_{k,l=1}^n$ with respect to a finite-dimensional orthonormal basis set $\psi_k(r)$, $k = 1, \ldots, n$.

$$W_{kl} = \int\int W(r, r') \psi_k(r) \overline{\psi}_l(r') dr dr'.$$

In this representation, the integral equation (3.19) becomes a matrix equation

$$\chi = \chi_0 + \chi_0 C \chi. \tag{3.20}$$

Often, a matrix equation like (3.20) is used as a shorthand notation for integral equations like (3.19) without having a specific discretization in mind.

In the TDDFT approach, C takes the form of a two-point kernel, i.e. (3.19) holds as described above. In the Bethe-Salpeter approach on the other hand, a four-point kernel is needed to adequately describe the relationship between χ_0 and χ. In this case, equation (3.20) is seen as shorthand notation for the integral equation [155]

$$\begin{aligned}\chi(r_1, r_2, r_3, r_4, \omega) = \chi_0(r_1, r_2, r_3, r_4, \omega) + \int\int\int\int \chi_0(r_1, r_5, r_3, r_6, \omega) \\ \cdot C(r_6, r_7, r_5, r_8) \chi(r_8, r_2, r_7, r_4, \omega) dr_5 dr_6 dr_7 dr_8.\end{aligned} \tag{3.21}$$

Equation (3.21) can be regarded as a Bethe-Salpeter equation for χ.

It is also possible to consider the four-point quantities, even when in the TDDFT approach 2-point quantities would suffice. Then the four-point kernel C is given by

$$C(r_1, r_2, r_3, r_4, \omega) = C(r_1, r_4) \delta(r_1 - r_3) \delta(r_2 - r_4). \tag{3.22}$$

With $\chi_0(r_1, r_2, \omega) = \chi_0(r_1, r_2, r_1, r_2, \omega)$ and defining

$$\chi(r_1, r_2, \omega) = \chi(r_1, r_2, r_1, r_2, \omega),$$

we arrive at (3.19) when using (3.22) in (3.21). A method solving (3.21) can therefore be used so solve (3.20).

Section 3.5 shows how the four-point variant (3.21) can be reformulated in form of a matrix eigenvalue problem of dimension $2n_{\text{occ}}n_{\text{unocc}}$, where $n_{\text{occ}} = |I_{\text{occ}}|$ is the number of occupied orbitals and $n_{unocc} = |I_{\text{unocc}}|$ is the number of unoccupied orbitals.

The method of solving the eigenvalue problem following from treating the 2-point TDDFT equation (3.20) as a four-point equation for molecules is called Casida formalism [64].

3.4.2 Kernel derivation from TDDFT

The Kohn-Sham equations (3.16) can be generalized to reflect a time-dependent Hamiltonian

$$H(t) = -\sum_i \Delta_{x_i} + V(x,t)$$

with a time-dependent potential $V(x,t)$. Runge and Gross [150] showed that two external potentials, that only differ by a time-dependent function, give rise to the same electron density $\rho(r,t)$. The wave functions only differ by a phase factor. The resulting framework is called *time-dependent density functional theory* (TDDFT).

Analogously to classical mechanics, an equation of motion is found by determining the wave function $\psi(x,t)$ in form of a stationary point of the action $A[\psi]$. This leads to the time-dependent Kohn-Sham equations, describing the time evolution of a wave function under changing external potential:

$$\mathrm{i}\frac{\partial}{\partial t}\psi_i(x,t) = \left[-\frac{1}{2}\Delta_x^2 + V_{\text{eff}}(x,t)\right]\psi_i(x,t), \tag{3.23}$$

where

$$V_{\text{eff}} = \int \frac{\rho(x',t)}{|x-x'|}dx' + V_{\text{xc}}[\rho(r,s)_{t_0<s<t}](r) + V_{\text{ext}}(x,t). \tag{3.24}$$

The exchange-correlation potential V_{xc} is dependent not just on the current density, but on the density evolution since time t_0. In the time-independent case, it is already difficult to approximate and in TDDFT further sacrifices have to be made.

A straightforward choice is the *local density approximation* (LDA), which accurately describes the homogeneous electron gas, but works surprisingly well for many other systems [139].

In order to compute excitation energies, one needs to apply concepts from perturbation theory. Here, the Hamiltonian is seen as a stationary Hamiltonian which is perturbed by a small time-dependent potential $\delta V_{ext}(t)$. In the linear regime, the response function is given by the reducible dynamic polarizability operator χ and defined as

$$\chi(r,t,r',t') = \frac{\partial\rho(r,t)}{\partial V_{ext}(r',t')},$$

i.e. it describes how the electron density $\rho(r,t)$ reacts to a change in the external potential. What makes this quantity difficult to compute is seen in (3.23) and (3.24). The external potential V_{ext} does not directly perturb the Kohn-Sham Hamiltonian but creates an effective potential V_{eff}, including the difficult-to-approximate exchange-correlation potential V_{xc}. If the effective potential was known, the resulting density could easily be computed via the irreducible dynamic polarizability operator defined in this context as

$$\chi_0(r,t,r',t') = \frac{\partial\rho(r,t)}{\partial V_{eff}(r',t')}.$$

The operator χ_0 is the linear response of the fictitious system described by the Kohn-Sham orbitals ψ_i determined by equation (3.23). It can be shown to be computed as presented in equation (3.17).

Using these definitions, (3.20) can be shown to hold with a frequency-dependent kernel

$$C(r_1,r_2,\omega) = v(r_1,r_2) + f_{xc}(r_1,r_2,\omega). \tag{3.25}$$

$f_{xc}(r_1,r_2,\omega)$ is given by a Fourier transform of

$$f_{xc}(r,t,r',t') = \left.\frac{\partial V_{xc}[\rho(r,t)]}{\partial\rho(r',t')}\right|_{V_{ext}=0}.$$

In the time-dependent LDA (TDLDA) approximation, it is frequency-independent and given as

$$f_{xc}^{TDLDA}(r_1,r_2) = \delta(r_1-r_2)\frac{\partial V_{xc}^{LDA}(\rho(r_1),r_1)}{\partial\rho(r_1)},$$

where V_{xc}^{LDA} is known explicitly.

In practical computations, approximations already have to be made in the computation of χ_0. Typically, Kohn-Sham orbitals from the ground-state calculations with an approximate exchange-correlation functional are used, together with corrected eigenvalues using the GW method. The approximations induced by the TDLDA come on top.

3.4.3 Kernel derivation from the Bethe-Salpeter equation

The Bethe-Salpeter equation [154] is a general equation describing a two-body system within quantum field theory. For a general treatment see e.g. [77]. It can be applied in many physical settings including high energy particle physics, nuclear theory and electronic structure theory [166]. In the latter setting, it can also be derived from Hedin's equations [90]. This has the advantage of pointing out recipes for approximations of the required quantities.

Hedin's equations are a closed set of five integral equations, accurately describing the many-body problem. They relate the important physical quantities of the self-energy

Σ, the dynamical screened interaction W, the time-ordered polarization operator P, the vertex function Γ and the Green's function G. One approach to solve them is to start with approximations for G and Σ and use them to compute the other quantities hoping to achieve self-consistency. Setting the starting points to $\Sigma = 0$ and $G = G_0$ which is constructed from Kohn-Sham orbitals and eigenvalues, one arrives at the Dyson equation [77, 90]

$$G = G_0 + G_0 \Sigma G,$$

where $\Sigma = iG_0W$ has been approximated using the GW approximation. Applying Hedin's equations for Π and P yields

$$P = \chi_0 - PW\chi_0, \tag{3.26}$$

where χ_0 is constructed from G_0. P and χ are by definition connected in form of

$$\chi = P + Pv\chi, \tag{3.27}$$

where v is the bare Coulomb potential $v(r_1, r_2) = |r_1 - r_2|^{-1}$.

The Dyson-like equation (3.20) can be derived from (3.26) and (3.27) with the kernel

$$C(r_1, r_2, r_3, r_4) = v(r_1, r_4)\delta(r_4 - r_2)\delta(r_3 - r_1) - W(r_2, r_4)\delta(r_4 - r_1)\delta(r_3 - r_2). \tag{3.28}$$

We described the setup of the kernel C in an abstract fashion and omitted details which may vary in different variants of the BSE-based method. We see that approximations have to be made at multiple points. This gives some degrees of freedom in the implementations of electronic structure codes.

3.5 The emergence of structured eigenvalue problems

In this section, we show how the Dyson-like equations (3.20) can be treated as a matrix eigenvalue problem. The resulting matrix will display two kinds of symmetries. One is induced by the choice of basis functions in order to represent the 4-dimensional quantities in 2-dimensional matrix form. The other one is induced by using a four-point kernel of the form (3.22) where $C(r_1, r_4) = C(r_4, r_1)$. This is the case for the TDDFT kernel described in Section 3.4.2 and the first addend of the BSE kernel (3.28). In the second addend, only the roles of r_1 and r_2 are switched and the same symmetries emerge.

Starting from (3.20) and making the dependency on ω explicit, we have [155]

$$\begin{aligned} & \chi(\omega) = \chi_0(\omega) + \chi_0(\omega)C\chi(\omega), \\ \Leftrightarrow\ & \chi(\omega) = \chi_0(\omega)(I + C\chi(\omega)) \\ \Leftrightarrow\ & \chi_0(\omega)^{-1}\chi(\omega) - C\chi(\omega) = I \\ \Leftrightarrow\ & \chi(\omega) = (\chi_0(\omega)^{-1} - C)^{-1}. \end{aligned} \tag{3.29}$$

Below we express the involved operators in a specific basis set. In this basis set $\chi_0(\omega)$ is diagonal and $\chi_0(\omega)^{-1}$ has the form

$$\chi_0(\omega)^{-1}=D+\omega K, \tag{3.30}$$

where $D=\mathrm{diag}(D_0,D_0)$ is real diagonal and not depending on ω and $K=\mathrm{diag}(I,-I)$.

Using (3.29), the matrix representation of $\chi(\omega)$ is now given by

$$\begin{aligned}\chi(\omega)&=(D+\omega K-C)^{-1}\\ &=-(\underbrace{K(C-D)}_{=:H_C}-\omega I)^{-1}K\end{aligned} \tag{3.31}$$

where we used $K^{-1}=K$. The poles of χ are the sough-after excitation values of the original system. From (3.31) we see that ω is a pole of χ when the matrix pencil $H_C-\omega I$ is not invertible, i.e. when ω is an eigenvalue of H_C. Furthermore, $\chi(\omega)$ can be computed when a full diagonalization $H_C=V\Lambda V^{-1}$ is available as

$$\begin{aligned}\chi(\omega)&=-(V\Lambda V^{-1}-\omega I)^{-1}K\\ &=-V(\Lambda-\omega I)^{-1}V^{-1}K.\end{aligned} \tag{3.32}$$

The microscopic dielectric function and the electron-energy-loss spectrum can be computed from $\chi(\omega)$. The macroscopic dielectric function can be computed from the microscopic dielectric function via a matrix inversion or alternatively more direct by considering a modified polarizability [139].

From (3.18) we see that the free four-point polarization propagator $\chi_0(r_1,r_2,r_3,r_4,\omega)$ is an element of the vector space

$$\mathcal{V}_{orig}:=\mathcal{H}\otimes\mathcal{H}\otimes\overline{\mathcal{H}}\otimes\overline{\mathcal{H}},$$

i.e. the tensor product of four Hilbert spaces. The vector space $\mathcal{H}$ is spanned by the functions $\varphi_i(r)$ which were used in (3.18) to construct χ_0. The dual space $\overline{\mathcal{H}}$ is spanned by the complex conjugated basis functions $\overline{\varphi_i(r)}$. The space $\mathcal{V}_{orig}$ is a subspace of $L^2(\mathbb{R}^{3\times4},\mathbb{C})$.

Suppose each $\mathcal{H}$ has dimension $2N$ (corresponding to N occupied and N unoccupied orbitals). Then $\mathcal{V}_{orig}$ has dimension $16N^4$. Let $\mathcal{H}$ have a given basis set $\{\varphi_i : i=1,...,2N\}$. A basis of the tensor product space $\mathcal{V}_{orig}$ is given by

$$\{\psi_{(klmn)}=\varphi_k(r_1)\varphi_l(r_2)\overline{\varphi_m}(r_3)\overline{\varphi_n}(r_4):\ k,l,m,n\in\{1,...,2N\}\},$$

i.e. all possible product combinations with factors from the individual basis sets.

$\chi_0(r_1,r_2,r_3,r_3,\omega)$ is represented in this basis as

$$\chi_0(r_2,r_2,r_3,r_4,\omega)=\sum_{k,l,m,n}^{2N}\chi_{0,klmn}\psi_{klmn}$$

Any element of $\mathcal{V}_{orig}$ can be represented by $16N^4$ parameters, given in the case of χ_0 by $\chi_{0,klmn}$. But in fact we know more about what χ_0 looks like and do not need to consider the whole space. Instead we consider a subspace $\mathcal{V}$ of dimension $4N^4$.

Let $\mathcal{H} = \mathcal{O} \oplus \mathcal{U}$, where $\mathcal{O}$ is the subspace spanned by the occupied orbitals, associated with index set $I_{\text{occ}} = \{1, \dots, N\}$, and $\mathcal{U}$ is the subspace spanned by the unoccupied orbitals associated with the index set $I_{\text{unocc}} = \{N+1, \dots 2N\}$,

$$\begin{aligned} \mathcal{O} &= \text{span}(\varphi_i : i \in I_{\text{occ}}), \\ \mathcal{U} &= \text{span}(\varphi_i : i \in I_{\text{unocc}}). \end{aligned}$$

Rewriting (3.18), $\chi_0(\omega)$ has the form [10]

$$\begin{aligned} \chi_0(r_1, r_2, r_3, r_4, \omega) = &\sum_{i=1}^{N} \sum_{j=N+1}^{2N} \frac{1}{\omega - \omega_{ij}} \psi_{(ijji)} \\ &+ \sum_{i=N+1}^{2N} \sum_{j=1}^{N} \frac{1}{-\omega - \omega_{ji}} \psi_{(ijji)}. \end{aligned} \tag{3.33}$$

χ_0 can also be expressed as an element of the lower-dimensional $\mathcal{V} = \mathcal{B} \otimes \overline{\mathcal{B}}$, where $\mathcal{B}$ is given as the transition subspace

$$\begin{aligned} \mathcal{B} = \mathcal{B}_0 \oplus \hat{\mathcal{B}}_0 &= (\overline{\mathcal{O}} \otimes \mathcal{U}) \oplus (\overline{\mathcal{U}} \otimes \mathcal{O}) \\ &= \text{span}(\Psi_{ij} := \varphi_i(r)\overline{\varphi_j}(r'),\ i = 1, ..., N,\ j = N+1, ..., 2N) \\ &\oplus \text{span}(\Psi_{ij} = \varphi_i(r)\overline{\varphi_j}(r'),\ i = N+1, ..., 2N,\ j = 1, ..., N) \end{aligned} \tag{3.34}$$

and has dimension $2N^2$. We map the double indices (i, j) to one superindex I

$$I = \begin{cases} (j - (N+1))N + i & \text{for } i \in \{1, .., N\},\ j \in \{N+1, ..., 2N\}, \\ N^2 + (j-1)N + i - N & \text{for } i \in \{N+1, .., 2N\},\ j \in \{1, ..., N\}. \end{cases} \tag{3.35}$$

An ordered basis set of $\mathcal{B}$ is given as

$$\{\Psi_I(r, r') := \Psi_{I(i,j)}(r, r') = \Psi_{i,j}(r, r') : I = 1, ..., 2N^2\}. \tag{3.36}$$

This yields a basis of the tensor product space $\mathcal{B} \times \overline{\mathcal{B}}$

$$\{\Psi_I(r_1, r_3)\overline{\Psi_J}(r_4, r_2) :\ I, J = 1, ..., 2N^2\}. \tag{3.37}$$

χ_0 can be represented with respect to this basis as

$$\chi_0(r_1, r_2, r_3, r_4, \omega) = \sum_{I,J}^{2N^2} \chi_{0,IJ} \Psi_I(r_1, r_3) \overline{\Psi_J(r_4, r_2)}. \tag{3.38}$$

$\hat{\chi_0}(\omega) := (\chi_{0,IJ})_{I,J=1,\ldots,2N^2}$ is a two-dimensional tensor, i.e. a matrix, representing $\chi_0(r_1, r_2, r_3, r_4, \omega)$. It follows from (3.33) that $\hat{\chi_0}(\omega)$ is diagonal and given as

$$\hat{\chi_0}(\omega) = \begin{bmatrix} D_1(\omega) & \\ & D_2(\omega) \end{bmatrix},$$

where

$$\begin{aligned} D_1(\omega) = \operatorname{diag}((\omega - \omega_{1,N+1})^{-1}, \ldots (\omega - \omega_{1,2N})^{-1}, (\omega - \omega_{2,N+1})^{-1}, \ldots, (\omega - \omega_{2,2N})^{-1}, \\ \ldots, (\omega - \omega_{N,N+1})^{-1}, \ldots, (\omega - \omega_{N,2N})^{-1}) \\ D_2(\omega) = \operatorname{diag}((-\omega - \omega_{1,N+1})^{-1}, \ldots (-\omega - \omega_{1,2N})^{-1}, (-\omega - \omega_{2,N+1})^{-1}, \\ \ldots, (-\omega - \omega_{2,2N})^{-1}, \ldots, (-\omega - \omega_{N,N+1})^{-1}, \ldots, (-\omega - \omega_{N,2N})^{-1}). \end{aligned}$$

We see that (3.30) holds indeed with

$$D_0 = \operatorname{diag}(\omega_{1,N+1}, \ldots, \omega_{1,2N}, \ldots, \omega_{N,N+1}, \ldots, \omega_{N,2N}).$$

What is left to do is to examine the matrix structure of the kernel C given in (3.25) or (3.28), discretized with respect to the orthogonal basis set (3.37). The matrix elements of a kernel of the form (3.22) are

$$C_{I,J} = \int\int\int\int C(r_1, r_2, r_3, r_4))\overline{\Psi_I(r_1, r_3)}\Psi_J(r_4, r_2)dr_1dr_2dr_3dr_4 \tag{3.39}$$

$$= \int\int C(r_1, r_2)\overline{\Psi_I(r_1, r_1)}\Psi_J(r_2, r_2)dr_1dr_2. \tag{3.40}$$

It is easily shown that from $C(r_1, r_2) = \overline{C(r_2, r_1)}$ follows

$$C_{IJ} = \overline{C_{JI}}, \tag{3.41}$$

i.e. the matrix is Hermitian. The symmetry is a property independent from the specifically chosen basis functions.

Another kind of symmetry is introduced by the basis functions because it holds

$$\Psi_I(r, r') = \overline{\Psi_{\hat{I}}}(r', r), \quad \hat{I} = \begin{cases} I + N^2 & \text{if } I \leq N^2, \\ I - N^2 & \text{if } I > N^2. \end{cases} \tag{3.42}$$

It follows

$$\overline{\Psi_I(r_1, r_3)}\Psi_J(r_4, r_2) = \overline{\Psi_{\hat{\jmath}}(r_2, r_4)}\Psi_{\hat{\imath}}(r_3, r_1)$$

and therefore we have with $C(r_1, r_2) = C(r_2, r_1)$

$$C_{I,J} = C_{\hat{\jmath},\hat{\imath}}. \tag{3.43}$$

Equation (3.43) implies a block matrix structure

$$C = \begin{bmatrix} A & B \\ D & A^{\mathsf{T}} \end{bmatrix}, \quad D = D^{\mathsf{T}},\ B = B^{\mathsf{T}}. \tag{3.44}$$

Because we know from equation (3.41) that the matrices are Hermitian, we arrive at the block structure

$$C = \begin{bmatrix} A & B \\ \overline{B} & \overline{A} \end{bmatrix}, \ A = A^{\mathsf{H}},\ B = B^{\mathsf{T}}. \tag{3.45}$$

In the Bethe-Salpeter approach, the static approximation of the screened Coulomb interaction in four-point form is given as [139]

$$W(r_1, r_2, r_3, r_4) = W(r_2, r_4)\delta(r_4 - r_1)\delta(r_3 - r_2)$$

and yields a matrix form

$$\begin{aligned} W_{IJ} &= \int\int\int\int W(r_1, r_2, r_3, r_4))\overline{\Psi_I(r_1, r_3)}\Psi_J(r_2, r_4)dr_1dr_2dr_3dr_4 \\ &= \int\int W(r_2, r_4)\overline{\Psi_I(r_4, r_2)}\Psi_J(r_2, r_4)dr_2dr_4. \end{aligned}$$

Here we also see that $W_{IJ} = \overline{W_{JI}}$, i.e. the matrix is Hermitian. The symmetry following from the particular choice of basis functions for the tensor space (3.43), i.e. $W_{IJ} = W_{\hat{j}\hat{I}}$, also holds for W, such that we arrive at the same structure.

With the Kernel C in matrix form (3.45), the matrix H_C defined in (3.31) has the form

$$H_C = \begin{bmatrix} A & B \\ -\overline{B} & -\overline{A} \end{bmatrix}, \qquad A = A^{\mathsf{H}},\ B = B^{\mathsf{T}}$$

This is what we called the Bethe-Salpeter matrix in the thesis introduction in (1.1). Developing algorithms to solve the eigenvalue problem for this matrix that exploit the structure and are suitable for high performance computing, is a major focus of this thesis.

3.6 Exploiting crystalline structure

In this section, we assume that the Dyson-like equation (3.20) is to be solved for a crystalline system. Here, the atoms are arranged in form of a Bravais lattice [161], defined by basis vectors $a_1, a_2, a_3 \in \mathbb{R}^3$ as

$$\mathcal{L} = \{R = n_1a_1 + n_2a_2 + n_3a_3 | n_1, n_2, n_3 \in \mathbb{Z}\}.$$

The positions of the atoms can be considered fixed. They determine the state of an electron within the system in form of a periodic potential

$$V(r) = V(r + R) \qquad \forall R \in \mathcal{L}.$$

Bloch's theorem states that the eigenstates of the Hamiltonian operator determined by a periodic potential have the form [115]

$$\varphi_{n,k}(r) = e^{\mathrm{i}k\cdot r} u_{n,k}(r), \tag{3.46}$$

where $k \in \mathbb{R}^3$, $n = 0, 1, 2, \ldots$ and $u_{n,k}(r)$ is periodic with respect to the lattice, i.e.

$$u_{n,k}(r) = u_{n,k}(r + R) \qquad \forall R \in \mathcal{L}.$$

The states (3.46) are called Bloch states. The corresponding eigenvalues (energy levels) are given by $\epsilon_{n,k}$. From considerations involving the time reversal operator it follows that [151, 155]

$$\overline{\varphi_{n,k}(r)} = \varphi_{n,-k}. \tag{3.47}$$

As the φ_i in (3.17) are eigenstates of the Kohn-Sham Hamiltonian, they can be expressed in form of a Bloch state (3.46).

$$\varphi_i(r) = \varphi_{n(i),k(i)}(r).$$

We define the subspace spanned by occupied and unoccupied orbitals corresponding to a wave vector k in the Brillouin zone Ω^*.

$$\begin{aligned}
\mathcal{O}_k &= \operatorname{span}(\varphi_{n,k}(r) : n \in I_{occ} = \{1, \ldots, N_{\mathrm{sol}}\}), \\
\mathcal{U}_k &= \operatorname{span}(\varphi_{n,k}(r) : n \in I_{unocc} = \{N_{\mathrm{sol}} + 1, \ldots, 2N_{\mathrm{sol}}\}).
\end{aligned}$$

Assuming a zero momentum transfer in the excitation (corresponding to a direct band gap), only $\varphi_{n,k}$ with the same crystal momentum k contribute to the polarizability, i.e. χ_0 (see (3.18)) can be expressed as [155]

$$\begin{aligned}
\chi_0(r_1, r_2, r_3, r_4, \omega) = &\sum_{k\in\Omega^*} \sum_{\substack{i\in I_{occ},\\ j\in I_{unocc}}} \frac{\varphi_{i,k}(r_1)\overline{\varphi_{j,k}}(r_3)\varphi_j(r_2)\overline{\varphi_{i,k}}(r_4)}{\omega - \omega_{ij,k}} \\
&+ \sum_{k\in\Omega^*} \sum_{\substack{i\in I_{unocc},\\ j\in I_{occ}}} \frac{\varphi_{i,k}(r_1)\overline{\varphi_{j,k}}(r_3)\varphi_{j,k}(r_2)\overline{\varphi_{i,k}}(r_4)}{-\omega - \omega_{ji,k}},
\end{aligned}$$

where $\omega_{ij,k} = \epsilon_{j,k} - \epsilon_{i,k}$. By changing the summation index k in the second term (called antiresonant) to $-k$, which is also in the Brillouin zone, and applying (3.47) we arrive at

$$\begin{aligned}
\chi_0(r_1, r_2, r_3, r_4, \omega) = &\sum_{k\in\Omega^*} \sum_{\substack{i\in I_{occ},\\ j\in I_{unocc}}} \frac{\varphi_{i,k}(r_1)\overline{\varphi_{j,k}}(r_3)\varphi_{j,k}(r_2)\overline{\varphi_{i,k}}(r_4)}{\omega - \omega_{ij,k}} \\
&+ \sum_{k\in\Omega^*} \sum_{\substack{i\in I_{unocc},\\ j\in I_{occ}}} \frac{\overline{\varphi_{i,k}}(r_1)\varphi_{j,k}(r_3)\overline{\varphi_{j,k}}(r_2)\varphi_{i,k}(r_4)}{-\omega - \omega_{ji,k}}.
\end{aligned} \tag{3.48}$$

χ_0 can now be represented as a matrix acting on the transition subspace

$$\mathcal{B}_{\text{sol}} = \mathcal{B}_{\text{sol},0} \oplus \hat{\mathcal{B}}_{\text{sol},0}, \tag{3.49}$$

where

$$\begin{aligned}\mathcal{B}_{\text{sol},0} &= \oplus_{k\in\Omega^*}(\overline{\mathcal{O}}_k \otimes \mathcal{U}_k) \\ &= \text{span}(\Psi_{ij,k}(r,r') = \varphi_{i,k}(r)\overline{\varphi_{j,k}}(r') : i \in I_{occ},\ j \in I_{unocc},\ k \in \Omega^*)\end{aligned}$$

and

$$\begin{aligned}\hat{\mathcal{B}}_{\text{sol},0} &= \oplus_{k\in\Omega^*}(\overline{\mathcal{U}}_k \otimes \mathcal{O}_k) \\ &= \text{span}(\Psi_{ij,k}(r,r') = \overline{\varphi_{i,k}}(r)\varphi_{j,k}(r') : i \in I_{\text{unocc}},\ j \in I_{\text{occ}},\ k \in \Omega^*)\,.\end{aligned}$$

Let the set of possible values of k be encoded in the finite index set $I_k = \{1,\dots,N_k\}$. N_k represents the resolution of the discretization. This means that for complex systems a large number of k-points is required ($N_k \gg N$). We represent each k-point by an index $l \in I_k$. The index triple (i,j,l) is mapped on the superindex I analogously to (3.35). We define $I(i,j,l)$, where either $i \in I_{\text{occ}}$ and $j \in I_{\text{unocc}}$ or $i \in I_{\text{unocc}}$ and $j \in I_{\text{occ}}$, by

$$I(i,j,l) = \begin{cases}(l-1)N^2 + (j-(N+1))N + i & \text{for } i \in I_{occ}\, j \in I_{\text{unocc}}, \\ N_kN^2 + (l-1)N^2 + (j-1)N + i - N & \text{for } i \in I_{\text{occ}},\ j \in I_{\text{unocc}}.\end{cases}$$

An ordered basis set of $\mathcal{B}_{\text{sol}}$ is given as

$$\{\Psi_I(r,r') := \Psi_{I(i,j,l)}(r,r') = \Psi_{ij,k}(r,r') : I = 1,...,2N_kN^2\}. \tag{3.50}$$

The first half of basis functions corresponds to states where $i \in I_{\text{occ}}$, $j \in\ I_{\text{unocc}}$, and k covers all admissible k-points in the Brillouin zone. The second half covers $i \in I_{\text{unocc}}$, $j \in\ I_{\text{occ}}$ and again all possible k-points.

As in the general case discussed in Section 3.5, χ_0 (3.48) is diagonal with respect to the defined basis set (3.50).

The particular choice of basis functions introduces a symmetry in form of

$$\Psi_I(r,r') = \Psi_{\hat{I}}(r',r), \quad \hat{I} = \begin{cases} I + N_kN^2 & \text{if } I \le N_kN^2 \\ I - N_kN^2 & \text{if } I > N_kN^2 \end{cases}.$$

In contrast to (3.42), no complex conjugation is involved. It follows

$$\overline{\Psi_I(r_1,r_3)}\Psi_J(r_4,r_2) = \Psi_{\hat{J}}(r_2,r_4)\overline{\Psi_{\hat{I}}}(r_3,r_1)$$

and therefore we have, using the general definition of the matrix elements of C (3.40),

$$C_{I,J} = \overline{C_{\hat{J},\hat{I}}}. \tag{3.51}$$

In contrast to (3.44), a matrix block structure of the following form is implied by (3.51).

$$C = \begin{bmatrix} A & B \\ D & A^{\mathsf{H}} \end{bmatrix}, \quad D = D^{\mathsf{H}},\ B = B^{\mathsf{H}}.$$

Together with the fact, that C is Hermitian ((3.41) also holds for the newly defined basis set), we arrive at a structured matrix

$$C = \begin{bmatrix} A & B \\ B & A \end{bmatrix}, \quad A = A^{\mathsf{H}},\ B = B^{\mathsf{H}}.$$

The same ideas can be applied to the second part of the Bethe-Salpeter Kernel W, leading to the same structure. With this kernel, the Bethe-Salpeter matrix defined in (3.31) has the form

$$H_C = \begin{bmatrix} A & B \\ -B & -A \end{bmatrix}, \quad A = A^{\mathsf{H}},\ B = B^{\mathsf{H}}.$$

3.7 TDDFT vs. Bethe-Salpeter approach

The Dyson-like equation emerging from linear response TDDFT (3.20) can also be written down with 2-point quantities in form of (3.19) using the Kernel (3.25).

Extending it as a large $2n^2 \times 2n^2$ eigenvalue problem, i.e. using (3.22), leads to a much larger eigenvalue problem. This only makes sense under specific circumstances. In quantum chemistry, molecules are observed, instead of solid structures. Here, the basis functions can be chosen such that they have no imaginary components. This results in real matrices. Methods described in Chapter 6 can therefore be applied. These methods rely on the formation of $A - B$ and $A + B$. In [64] it is shown that $A - B$ is diagonal if the matrix was set up in this context. The presented approach relies on the matrix square root of $A - B$, also see Chapter 6. In this case, one only needs to compute the square roots of the diagonal entries. This is equivalent to the other presented approaches in Chapter 6, e.g. involving a Cholesky factorization. Furthermore, in quantum chemical applications usually a smaller number of states n is needed for modeling, and the quadratic size $2n^2$ of the eigenvalue problem is not yet prohibitively large. When $A - B$ is not diagonal, the square root approach should be avoided. Details are presented in Chapter 6.

For solids, it is not possible to only use real basis functions. Therefore, following the Casida formalism one arrives at a problem that is computationally equivalent to the BSE approach. The quality of TDDFT depends on the quality of the approximation used for the exchange-correlation potential. As pointed out in Section 3.3.2, this already poses a challenge in the non-time-dependent case. The Bethe-Salpeter approach on the other hand provides a clear path for better approximation by including larger parts of the corresponding Feynman diagrams [139].

In particular the inclusion of a strong electron-hole correlation is easier in the BSE approach. This is the case for materials such as bulk silicone. In [139] and [152] it is shown that a Bethe-Salpeter approach computes optical absorption spectra that better match experimental results than a TDDFT approach.

Both approaches can benefit from mutual insight. For example, approximate TDDFT exchange-correlation kernels can be deduced from a comparison with results from many-body perturbation theory.

3.8 Summary and conclusion

In this chapter, we discussed quantum physical ground state methods. They give a set of independent orbitals and eigenvalues. These are used to set up a function χ_0. The sought-after function is the (reducible) polarizability χ. It is related to χ_0 via a kernel C in form of a four-point integral equation, which can be considered a Bethe-Salpeter equation. C is deduced either from time-dependent density functional perturbation theory, or from many-body perturbation theory, i.e. Hedin's equations.

Rewriting the equation leads to an operator eigenvalue problem, which can be discretized to form a linear matrix eigenvalue problem. To this end, we used the idea that any four-point function can approximately be represented in form of a 4D tensor with respect to a set of orthogonal four-point basis functions. We used products of four one-point functions that are given by the ground state occupied orbitals and unoccupied orbitals. The 4D tensor is representable as a 2D tensor, i.e. a matrix when flattened. To this end we defined the product of two orbitals φ_i, φ_j as basis functions Ψ_I, where I is a superindex referring to all possible combinations of the pair i, j. In the product, only occupied and unoccupied states can be paired. We considered N occupied and N unoccupied states and arrived at a matrix of size $2N^2 \times 2N^2$.

The resulting structure shows two kinds of symmetries. One stems from the inherent symmetry of K, the other one comes from the redundancy inherent to the basis functions. Both symmetries together lead to the typical block matrix structure.

For solid materials, the time-inversion symmetry of basis functions in k-space can be applied. In the basis-related symmetry, the change introduces a complex conjugation. This leads to a slightly different block structure.

A conclusion to be drawn from this chapter is that the setup of the Bethe-Salpeter block matrix is only the last step in a series of mathematical considerations. The matrix eigenvalue problem is one way to make the problem tractable for computers. However, due to the N^2 scaling, the resulting matrix is extremely large, even for small scale problems. The symmetries in the matrix reflect the representation of redundant information in this approach. Future research, going beyond the scope of this thesis, should not just take the structured matrix as a starting point. Instead, the original 4-dimensional objects it represents should be studied further. If dimensionality reduction can be applied at this stage, much smaller matrices would need to be considered.

CHAPTER 4

PROPERTIES OF MATRICES WITH BETHE-SALPETER STRUCTURE

Contents

4.1 Introduction

We have seen in equation (3.32) that the discretized Bethe-Salpeter equation (3.20) for the reducible polarizability $\chi(\omega)$ can be solved by computing the eigenvalue decomposition of a structured matrix H.

With a general set of occupied and unoccupied orbitals as basis functions, H has the structure

$$H = H_1 = \begin{bmatrix} A & B \\ -B^{\mathsf{H}} & -A^{\mathsf{T}} \end{bmatrix} \in \mathbb{C}^{2n\times 2n}, \qquad A = A^{\mathsf{H}},\ B = B^{\mathsf{T}}. \tag{4.1}$$

In this thesis, we refer to Structure (4.1) as a *BSE matrix of form I*.

In crystalline systems, the given physical structure implies a periodic potential. Here, Bloch states are chosen as basis functions. The periodicity of crystalline systems implies a time-inversion symmetry in these states (see Section 3.6). This leads to H having the form

$$H = H_2 = \begin{bmatrix} A & B \\ -B & -A \end{bmatrix} \in \mathbb{C}^{2n\times 2n}, \qquad A = A^{\mathsf{H}},\ B = B^{\mathsf{H}}. \tag{4.2}$$

We call matrices of this form *BSE matrix of form II*.

An eigenvalue problem of this form is also called *linear response eigenvalue problem* [21, 22, 23, 20, 24]. It appears in the Casida formalism of time-dependent density functional theory [64] and in the random phase approximation concerning the excitations of atomic nuclei [142].

In most applications, apart from solid state physics, basis functions may be chosen to be real, leading to real valued matrices. In that case, form I and form II do not differ.

In the practice of computing excitation properties of materials, often there is even more structure available. It can be exploited for devising efficient algorithms. The Hermitian matrices

$$\mathcal{H}_1 := \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix} H_1 = \begin{bmatrix} A & B \\ B^{\mathsf{H}} & A^{\mathsf{T}} \end{bmatrix} > 0, \quad \mathcal{H}_2 := \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix} H_2 = \begin{bmatrix} A & B \\ B & A \end{bmatrix} > 0, \tag{4.3}$$

are typically positive definite [139, 155]. We refer to Bethe-Salpeter eigenvalue problems of form I or form II as "definite" when they fulfill this definiteness property (4.3).

In Section 4.2, we characterize the structure of the BSE matrices (4.1) and (4.2) by employing the concept of non-standard inner products. We compile important results regarding BSE matrices of form I and form II. We do not generally assume the definiteness property (4.3) to hold and explicitely state when it is assumed. An important new result concerning BSE matrices of form II is given in Section 4.3.

4.2 Results on the spectral structure of BSE matrices

Non-definite inner products, introduced in (2.1), provide a language to describe the structure of the BSE matrices in a more concise way, not relying on the matrix block structure. The following two matrices, and the inner products induced by them, play a central role:

$$J_n = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}, \qquad K_n = \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix}.$$

We drop the index when the dimension is clear from its context. The identities $J^{-1} = -J$ and $K^{-1} = K$ are regularly used in the following. The results compiled in this section are partly known but can now be proven easily using the notion of generalized inner products.

Theorem 4.1:
A matrix H is a BSE matrix of form I as given in (4.1) if and only if both of the following conditions hold.

1. H is skew-adjoint with respect to the complex bilinear form induced by J, i.e. $H = -H^{\star_J} = JH^{\mathsf{T}}J$.

2. H is self-adjoint with respect to the complex sesquilinear form induced by K, i.e. $H = H^{\star_K} = KH^{\mathsf{H}}K$. ◊

Proof. $JH^{\mathsf{T}}J = H$ is equivalent to JH being symmetric and $KH^{\mathsf{H}}K = H$ is equivalent to KH being Hermitian. We observe that JH is symmetric and KH is Hermitian, if H has BSE form I. Conversely, let $H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}$, $H_{ij} \in \mathbb{C}^{n\times n}$ and $JH = \begin{bmatrix} H_{21} & H_{22} \\ -H_{11} & -H_{12} \end{bmatrix}$ be symmetric. It follows

$$H_{21} = H_{21}^{\mathsf{T}}, \quad H_{12} = H_{12}^{\mathsf{T}}, \quad H_{11} = -H_{22}^{\mathsf{T}}. \tag{4.4}$$

Let $KH = \begin{bmatrix} H_{11} & H_{12} \\ -H_{21} & -H_{22} \end{bmatrix}$ be Hermitian. It follows

$$H_{11} = H_{11}^{\mathsf{H}}, \quad H_{22} = H_{22}^{\mathsf{H}}, \quad H_{21} = -H_{12}^{\mathsf{H}}. \tag{4.5}$$

Equations (4.4) and (4.5) give exactly BSE form I:

$$H = \begin{bmatrix} H_{11} & H_{12} \\ -H_{12}^{\mathsf{H}} & -H_{11}^{\mathsf{T}} \end{bmatrix} \text{ with } H_{11} = H_{11}^{\mathsf{H}},\ H_{12} = H_{12}^{\mathsf{T}}. \qquad \square$$

Theorem 4.2:
A matrix $H \in \mathbb{C}^{2n\times 2n}$ is a BSE matrix of form II as given in (4.2) if and only if both of the following conditions hold.

1. H is skew-adjoint with respect to the complex sesquilinear form induced by J, i.e. $H = -H^{\star_J} = JH^{\mathsf{H}}J$.
2. H is self-adjoint with respect to the complex sesquilinear form induced by K, i.e. $H = H^{\star_K} = KH^{\mathsf{H}}K$. $\lozenge$

Proof. The proof works exactly as the proof of Theorem 4.1, but here, the Hermitian transpose $.^{\mathsf{H}}$ is associated with J instead of the regular transpose $.^{\mathsf{T}}$. $\square$

This new characterization is now used to show that eigenvalues and eigenvectors also exhibit special structures. Matrices, that are skew-adjoint with respect to the sesquilinear form induced by J are called *Hamiltonian*, and play an important role in control theory and model order reduction (see e.g. [37]). The same property with respect to the bilinear form is called *J-symmetric* in [117] and explored further in [34].

The first two propositions of the following theorem are well known facts about Hamiltonian [37] and J-symmetric matrices [34].

Theorem 4.3:
Let $H \in \mathbb{C}^{2n\times 2n}$.

1. If H is skew-adjoint with respect to the sesquilinear form induced by J, i.e. $JH = -H^{\mathsf{H}}J$, then its eigenvalues come in pairs $(\lambda, -\bar{\lambda})$. If x is a right eigenvector of H corresponding to λ, then $x^{\mathsf{H}}J$ is the left eigenvector of H corresponding to $-\bar{\lambda}$.

2. If H is skew-adjoint with respect to the bilinear form induced by J, i.e. $JH = -H^{\mathsf{T}}J$, then its eigenvalues come in pairs $(\lambda, -\lambda)$. If x is a right eigenvector of H corresponding to λ, then $x^{\mathsf{T}}J$ is the left eigenvector of H corresponding to $-\lambda$.

3. If H is self-adjoint with respect to the sesquilinear form induced by K, i.e. $KH = H^{\mathsf{H}}K$, then its eigenvalues come in pairs $(\lambda, \bar{\lambda})$. If x is a right eigenvector of H corresponding to λ, then $x^{\mathsf{H}}K$ is the left eigenvector of H corresponding to $\bar{\lambda}$. ◊

Proof. 1. Using $H^{\mathsf{H}} = JHJ$ and $J^{-1} = -J$, we see that

$$Hx = \lambda x \quad \Leftrightarrow \quad x^{\mathsf{H}}JH = -\bar{\lambda}x^{\mathsf{H}}J.$$

2. Using $H^{\mathsf{T}} = JHJ$ and $J^{-1} = -J$, we see that

$$Hx = \lambda x \quad \Leftrightarrow \quad x^{\mathsf{T}}JH = -\lambda x^{\mathsf{T}}J.$$

3. Using $H^{\mathsf{H}} = KHK$ and $K^{-1} = K$, we see that

$$Hx = \lambda x \quad \Leftrightarrow \quad x^{\mathsf{H}}KH = \bar{\lambda}x^{\mathsf{H}}K.$$ □

Theorem 4.3 reveals that symmetries defined by the matrices J or K are reflected in connections between left and right eigenvectors of the considered matrix.

The BSE matrices show a symmetry with respect to two inner products (Theorem 4.1 and 4.2). This double-structure leads to eigenvalues that show up not only in pairs but in quadruples if they have a real and an imaginary component. Additionally, it yields a connection between right eigenvectors, clarified in the following theorem.

Theorem 4.4:
Let $H \in \mathbb{C}^{2n\times 2n}$ be self-adjoint with respect to the sesquilinear inner product induced by K and skew-adjoint with respect to (a) the sesquilinear inner product or (b) the bilinear inner product induced by J. Then

1. The eigenvalues of H come in pairs $(\lambda, -\lambda)$ if $\lambda \in \mathbb{R}$ or $\lambda \in \mathrm{i}\mathbb{R}$, or in quadruples $(\lambda, \bar{\lambda}, -\lambda, -\bar{\lambda})$.

2. a) If v is an eigenvector of H with respect to λ, then JKv is an eigenvector of H with respect to $-\lambda$.

   b) If v is an eigenvector of H with respect to λ, then $JK\bar{v}$ is an eigenvector of H with respect to $-\bar{\lambda}$. ◊

Proof. 1. The quadruple property comes from combining the propositions given in Theorem 4.3 (1. and 3. or 2. and 3., respectively). The pair property for real eigenvalues comes (a) from Theorem 4.3, proposition 1, or (b) from Theorem 4.3, proposition 2. The pair property for imaginary eigenvalues follows from Theorem 4.3, proposition 3 in both cases.

2. a) With $KJHJK = H$ we have

$$Hv = \lambda v \quad \Leftrightarrow \quad HJKv = -\lambda JKv.$$

b) With $KJHJK = \bar{H}$ we have

$$Hv = \lambda v \quad \Leftrightarrow \quad \bar{H}JKv = -\lambda JKv \quad \Leftrightarrow \quad HJK\bar{v} = -\bar{\lambda}JK\bar{v}. \qquad \square$$

The special case of (b) (i.e. for BSE matrices of form I, see equation (4.1)) has been proven in [34]. Our proof does not rely on the particular block structure of the matrix, but works with the given symmetries and is therefore more concise and easily extendable to other double-structured matrices.

The definiteness property (4.3) has consequences for the structure of the eigenvalue spectrum. To study these, we consider the more general class of Σ-Hermitian matrices. By this we mean matrices that are self-adjoint with respect to the inner product induced by a signature matrix Σ. K is a particular example of a signature matrix.

Theorem 4.5:
Let $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, $\sigma_i \in \{+1, -1\}$ be a signature matrix with p positive and $n - p$ negative diagonal entries. Let $H \in \mathbb{C}^{n\times n}$ be given such that ΣH is Hermitian positive definite. Then H is diagonalizable and its eigenvalues are real, of which p are positive and $n - p$ are negative. $\lozenge$

Proof. As ΣH is positive definite, and Σ is symmetric, they can be diagonalized simultaneously (see [83], Corollary 8.7.2), i.e. there is a nonsingular $X \in \mathbb{C}^{n\times n}$, such that

$$X^{\mathsf{H}}\Sigma HX = I_n, \tag{4.6}$$
$$X^{\mathsf{H}}\Sigma X = \Lambda, \tag{4.7}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n\times n}$ gives the eigenvalues of the matrix pencil $\Sigma - \lambda\Sigma H$. It follows from (4.7) and Sylvester's law of inertia that Λ has p positive and $n - p$ negative values. We have

$$X^{-1}HX = \Lambda^{-1},$$

i.e. H is diagonalizable and Λ^{-1} contains the eigenvalues of H. $\square$

The spectral structure of the BSE matrices given in practice follows immediately from the presented theorems and is summarized in the following lemma.

Lemma 4.6:
Let H be a BSE matrix of form I (see equation (4.1)) or form II (see equation (4.2)), such that the definiteness property (4.3) holds. Then the eigenvalues are real and come in pairs $\pm\lambda$. If v is an eigenvector associated with λ, then

1. $y = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} \bar{v}$ is an eigenvector associated with $-\lambda$ if H is of BSE form I.

2. $y = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} v$ is an eigenvector associated with $-\lambda$ if H is of BSE form II, $\lozenge$

We have seen in Sections 3.5 and 3.6 that for crystalline systems it is possible to either arrive at a Bethe-Salpeter eigenvalue problem of form I or form II, depending on the basis functions chosen for discretization. The basis functions leading to form II are the less intuitive choice and up to this point we did not provide a clear reason, why this form is preferable to form I. The reason is that for form II it is possible to devise algorithms that solve an eigenvalue problem of size $n \times n$ instead of the larger eigenvalue problem of size $2n \times 2n$. At its core this is possible due to the following observation which finds no analogue for BSE matrices of form I. It can e.g. be found in [159], albeit for real matrices.

Lemma 4.7:
Let H be a BSE matrix of form II (4.2).

1. With the matrix $Q = \frac{1}{2}\begin{bmatrix} I & I \\ -I & I \end{bmatrix}$ we have

$$Q^{-1}HQ = \begin{bmatrix} 0 & A+B \\ A-B & 0 \end{bmatrix}. \tag{4.8}$$

2. KH is positive definite if and only if $A+B$ and $A-B$ are positive definite. $\lozenge$

Lemma 4.7 says that a BSE matrix of form II is easily transformed to an antidiagonal block matrix. This matrix can be squared and one arrives at

$$\begin{bmatrix} 0 & A-B \\ A+B & 0 \end{bmatrix}^2 = \begin{bmatrix} (A-B)(A+B) & 0 \\ 0 & (A+B)(A-B) \end{bmatrix} = \begin{bmatrix} \hat{A} & 0 \\ 0 & -\hat{A} \end{bmatrix}. \tag{4.9}$$

So $\hat{A} = (A-B)(A+B)$ contains the squares of the eigenvalues of H. The second part of the lemma states that $\hat{A}$ is a matrix product consisting of two positive definite matrices. For the solution of this kind of product eigenvalue problem, efficient methods are available. Not only the eigenvalues of H can be deduced from $\hat{A}$ but also the eigenvectors of H can be constructed from the eigenvectors of $\hat{A}$. The computation is clarified in the next section.

4.3 Solving a smaller product eigenvalue problem

In this section, we focus on BSE matrices of form II, paving the way for a new unified framework relating different solution strategies. In contrast to most of the literature, we do not require the definiteness property (4.3) here. The following two theorems play a central role in Chapter 6. The relate eigenvectors and corresponding real (Theorem 4.8) and complex eigenvalues (Theorem 4.9) of $(A+B)(A-B)$ (see equation (4.9)) to eigenvectors and eigenvalues of H.

Theorem 4.8:
Let H be a BSE matrix of form II (see Equation (4.2)) and $M_1 := A+B$, $M_2 := A-B$ be nonsingular. Let

$$M_1 M_2 v_1 = \mu v_1 \tag{4.10}$$

define a right eigenvector of the matrix product $M_1 M_2$, corresponding to an eigenvalue $\mu \in \mathbb{R}$, $\lambda_2 := v_1^{\mathsf{H}} M_2 v_1 \in \mathbb{R}$. Then

$$v_2 := \lambda_2^{-1} M_2 v_1 \tag{4.11}$$

is a left eigenvector of $M_1 M_2$ corresponding to μ with $v_1^{\mathsf{H}} v_2 = 1$.

With $Q := \frac{1}{2}\begin{bmatrix} I & I \\ -I & I \end{bmatrix}$ and $\lambda_1 := v_2^{\mathsf{H}} M_1 v_2 \in \mathbb{R}$ an eigenpair of H is given by

$$\lambda = \begin{cases} \mu^{\frac{1}{2}} & \text{if } (\lambda_1 > 0 \text{ and } \lambda_2 > 0) \text{ or } \operatorname{sign}(\lambda_1)\operatorname{sign}(\lambda_2) = -1, \\ -\mu^{\frac{1}{2}} & \text{if } \lambda_1 < 0 \text{ and } \lambda_2 < 0, \end{cases} \qquad v_\lambda = Q \begin{bmatrix} v_1 \lambda_1^{\frac{1}{4}} \lambda_2^{-\frac{1}{4}} \\ v_2 \lambda_1^{-\frac{1}{4}} \lambda_2^{\frac{1}{4}} \end{bmatrix}, \tag{4.12}$$

i.e. $Hv_\lambda = \lambda v_\lambda$. If γ is an eigenvalue of $M_1 M_2$ and v_γ is the corresponding constructed vector, it holds

$$v_\lambda^{\mathsf{H}} K v_\gamma = \begin{cases} 1 & \text{if } \lambda = \gamma \in \mathbb{R}, \\ 0 & \text{if } \lambda = \gamma \in \mathrm{i}\mathbb{R}, \\ 0 & \text{if } \lambda \neq \gamma, \end{cases} \quad v_\lambda^{\mathsf{H}} J v_\gamma = \begin{cases} 0 & \text{if } \lambda = \gamma \in \mathbb{R} \text{ or if } \lambda \neq \gamma, \\ \mathrm{i} & \text{if } \lambda = \gamma \in \mathrm{i}\mathbb{R} \text{ and } \lambda_1 > 0,\ \lambda_2 < 0, \\ -\mathrm{i} & \text{if } \lambda = \gamma \in \mathrm{i}\mathbb{R} \text{ and } \lambda_1 < 0,\ \lambda_2 > 0. \end{cases} \tag{4.13}$$

◊

Proof. It follows from (4.10) and (4.11), using $v_1^{\mathsf{H}} v_2 = 1$, that

$$M_1 v_2 = \lambda_1 v_1, \qquad \lambda_1 := \mu \lambda_2^{-1} = v_2^{\mathsf{H}} M_1 v_2. \tag{4.14}$$

For $\lambda \in \mathbb{C}$,

$$\begin{bmatrix} 0 & M_1 \\ M_2 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix}$$

holds if and only if

$$M_1 y = \lambda x \quad \text{and} \quad M_2 x = \lambda y. \tag{4.15}$$

This is achieved by $x := v_1 \lambda_1^{\frac{1}{4}} \lambda_2^{-\frac{1}{4}}$, $y := v_2 \lambda_1^{-\frac{1}{4}} \lambda_2^{\frac{1}{4}}$ and $\lambda = \lambda_1^{\frac{1}{2}} \lambda_2^{\frac{1}{2}}$. If λ_1 and λ_2 are both positive, or if they have opposing signs, it holds $(\lambda_1 \lambda_2)^{\frac{1}{2}} = \lambda_1^{\frac{1}{2}} \lambda_2^{\frac{1}{2}}$, so that we have

$\lambda = \lambda_1^{\frac{1}{2}}\lambda_2^{\frac{1}{2}} = \mu^{\frac{1}{2}}$. If λ_1 and λ_2 are both negative, it holds $(\lambda_1\lambda_2)^{\frac{1}{2}} = -\lambda_1^{\frac{1}{2}}\lambda_2^{\frac{1}{2}}$, so that $\lambda = \lambda_1^{\frac{1}{2}}\lambda_2^{\frac{1}{2}} = -\mu^{\frac{1}{2}}$. We observe that

$$Q^{-1}HQ = \begin{bmatrix} 0 & M_1 \\ M_2 & 0 \end{bmatrix}$$

and conclude

$$Hv_\lambda = HQ\begin{bmatrix} x \\ y \end{bmatrix} = Q\begin{bmatrix} 0 & M_1 \\ M_2 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \lambda Q\begin{bmatrix} x \\ y \end{bmatrix} = \lambda v_\lambda.$$

The orthogonality conditions (4.13) remain to be shown. We see with scaling factors $s_\lambda = \lambda_1^{\frac{1}{4}}\lambda_2^{-\frac{1}{4}}$, $s_\gamma = \gamma_1^{\frac{1}{4}}\gamma_2^{-\frac{1}{4}}$ and

$$v_\lambda = Q\begin{bmatrix} x_\lambda \\ y_\lambda \end{bmatrix}, \quad x_\lambda = v_{1,\lambda}s_\lambda, \quad y_\lambda = v_{2,\lambda}s_\lambda^{-1}, \qquad v_\gamma = Q\begin{bmatrix} x_\gamma \\ y_\gamma \end{bmatrix}, \quad x_\gamma = v_{1,\gamma}s_\gamma, \quad y_\gamma = v_{2,\gamma}s_\gamma^{-1},$$

that

$$v_\lambda^{\mathsf{H}}Kv_\gamma = \frac{1}{2}(\overline{s_\lambda}s_\gamma^{-1}v_{1,\lambda}^{\mathsf{H}}v_{2,\gamma} + \overline{s_\lambda^{-1}}s_\gamma v_{2,\lambda}^{\mathsf{H}}v_{1,\gamma}), \quad v_\lambda^{\mathsf{H}}Jv_\gamma = \frac{1}{2}(\overline{s_\lambda}s_\gamma^{-1}v_{1,\lambda}^{\mathsf{H}}v_{2,\gamma} - \overline{s_\lambda^{-1}}s_\gamma v_{2,\lambda}^{\mathsf{H}}v_{1,\gamma}).$$

For $\lambda \neq \gamma$, both expressions are zero because $v_{1,\lambda}^{\mathsf{H}}v_{2,\gamma} = 0$. For $\gamma = \lambda$ we have

$$v_\lambda^{\mathsf{H}}Kv_\lambda = \frac{1}{2}(\overline{s_\lambda}s_\lambda^{-1} + \overline{s_\lambda^{-1}}s_\lambda), \quad v_\lambda^{\mathsf{H}}Jv_\lambda = \frac{1}{2}(\overline{s_\lambda}s_\lambda^{-1} - \overline{s_\lambda^{-1}}s_\lambda).$$

If λ is real, then μ is positive and λ_1 and λ_2 have the same sign. Then it holds $s_\lambda = (\lambda_1\lambda_2^{-1})^{\frac{1}{4}}$. The factor s_λ is real and so it holds $\overline{s_\lambda}s_\lambda^{-1} = 1 = \overline{s_\lambda^{-1}}s_\lambda$, which shows $v_\lambda^{\mathsf{H}}Jv_\lambda = 0$ and $v_\lambda^{\mathsf{H}}Kv_\lambda = 1$. If λ is imaginary, then μ is negative and λ_1 and λ_2 have opposing signs. Here we have

$$\overline{s_\lambda}s_\lambda^{-1} = \begin{cases} \mathrm{i} & \text{if } \lambda_1 > 0, \lambda_2 < 0, \\ -\mathrm{i} & \text{if } \lambda_1 < 0, \lambda_2 > 0, \end{cases}$$

leading to the final results $v_\lambda^{\mathsf{H}}Jv_\lambda = \pm\mathrm{i}$ and $v_\lambda^{\mathsf{H}}Kv_\lambda = 0$. □

We have seen that the sign of of the eigenvalue of the computed eigenpair of H via Theorem 4.8 does not necessarily match the sign of the principal square root of the eigenvalue of the matrix product M_1M_2. In the definite case (4.3), i.e. where KH is positive definite we have positive definite M_1 and M_2. λ_1 and λ_2 are positive and the sign switch does not take place. If a sign switch occurs and the eigenvectors corresponding to the principal square root are of interest, they can easily be computed via Theorem 4.4, proposition 2, case (a).

The product eigenvalue problem resulting from Theorem 4.8 can also be interpreted as a generalized eigenvalue problem (M_2, M_1^{-1}). However, computing a matrix inverse is not necessary and the product eigenvalue problem can be tackled directly as we see in the following section.

Similarly to Theorem 4.8, complex eigenvalues can be related to a smaller product eigenvalue problem.

Theorem 4.9:
Let H be a BSE matrix of form II (see Equation (4.2)), with $M_1 := A + B$ and $M_2 := A - B$ nonsingular. Let $V_1 = \begin{bmatrix} v_{11} & v_{12} \end{bmatrix} \in \mathbb{C}^{n\times 2}$ contain a right eigenvector pair of the matrix $M_1 M_2$,

$$M_1 M_2 V_1 = V_1 M \tag{4.16}$$

where $M = \begin{bmatrix} \mu & 0 \\ 0 & \overline{\mu} \end{bmatrix}$ contains the eigenvalue pair. Then, with $\Lambda_2 := V_1^{\mathsf{H}} M_2 V_1 = \begin{bmatrix} 0 & \overline{\lambda_2} \\ \lambda_2 & 0 \end{bmatrix}$, the matrix

$$V_2 = \begin{bmatrix} v_{21} & v_{22} \end{bmatrix} := M_2 V_1 \Lambda_2^{-1} \in \mathbb{C}^{n\times 2},$$

contains a corresponding left eigenvector pair with $V_1^{\mathsf{H}} V_2 = I_2$. Define the scaling factors

$$\begin{gathered} \lambda_1 := v_{21}^H M_1 v_{22}, \quad \lambda_2 := v_{12}^H M_2 v_{11}, \quad \hat{\lambda} := \lambda_1^{1/4} \lambda_2^{-1/4}, \\ \hat{\Lambda}_1 := \begin{bmatrix} \hat{\lambda} & 0 \\ 0 & \overline{\hat{\lambda}} \end{bmatrix}, \quad \hat{\Lambda}_2 := \begin{bmatrix} 0 & \overline{\hat{\lambda}}^{-1} \\ \hat{\lambda}^{-1} & 0 \end{bmatrix}. \end{gathered} \tag{4.17}$$

Then, with $Q = \frac{1}{2}\begin{bmatrix} I & I \\ -I & I \end{bmatrix}$, two eigenvectors of H are given as columns of the matrix $V_\lambda = Q \begin{bmatrix} V_1 \hat{\Lambda}_1 \\ V_2 \hat{\Lambda}_2 \end{bmatrix}$ and we have

$$HV_\lambda = V_\lambda s \begin{bmatrix} \mu^{\frac{1}{2}} & \\ & \overline{\mu^{\frac{1}{2}}} \end{bmatrix}, \quad s = \begin{cases} -1 & \text{if } \arg\lambda_1 + \arg\lambda_2 < -\pi \text{ or } \arg\lambda_1 + \arg\lambda_2 > \pi, \\ 1 & \text{else.} \end{cases} \tag{4.18}$$

Furthermore, we have

$$V_\lambda^{\mathsf{H}} K V_\gamma = \begin{cases} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & \text{if } \{\lambda, \overline{\lambda}\} = \{\gamma, \overline{\gamma}\}, \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \text{if } \{\lambda, \overline{\lambda}\} \neq \{\gamma, \overline{\gamma}\}, \end{cases} \qquad V_\lambda^{\mathsf{H}} J V_\gamma = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{4.19}$$

$$V_\lambda^{\mathsf{H}} K v_\gamma = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathsf{T}}, \qquad V_\lambda^{\mathsf{H}} J v_\gamma = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathsf{T}}, \tag{4.20}$$

where V_γ contains two eigenvectors corresponding to a complex eigenvalue pair $\{\gamma, \overline{\gamma}\}$, constructed according to this theorem. v_γ is an eigenvector corresponding to a real or imaginary eigenvalue γ, constructed according to Theorem 4.8. ◊

Proof. It can be shown that

$$V_1^{\mathsf{H}} M_2 V_1 = \begin{bmatrix} 0 & \overline{\lambda_2} \\ \lambda_2 & 0 \end{bmatrix} =: \Lambda_2, \qquad V_2^{\mathsf{H}} M_1 V_2 = \begin{bmatrix} 0 & \lambda_1 \\ \overline{\lambda_1} & 0 \end{bmatrix} =: \Lambda_1$$

and

$$M_1 M_2 V_1 = V_1 \Lambda_1 \Lambda_2.$$

With (4.16) it follows that $\Lambda_1 \Lambda_2 = M$. Note that $Q^{-1} H Q = \begin{bmatrix} 0 & M_1 \\ M_2 & 0 \end{bmatrix}$ and

$$\begin{bmatrix} 0 & M_1 \\ M_2 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} \Lambda$$

holds for $X, Y \in \mathbb{C}^{n \times 2}$ if and only if

$$M_1 Y = X\Lambda \quad \text{and} \quad M_2 X = Y\Lambda. \tag{4.21}$$

Equation (4.21) can be manipulated to show that

$$\Lambda_1 \hat{\Lambda}_2 = s \hat{\Lambda}_1 M^{\frac{1}{2}} \quad \text{and} \quad \Lambda_2 \hat{\Lambda}_1 = s \hat{\Lambda}_2 M^{\frac{1}{2}} \tag{4.22}$$

$$\text{for } s = \begin{cases} -1 & \text{if } \arg\lambda_1 + \arg\lambda_2 < -\pi \text{ or } \arg\lambda_1 + \arg\lambda_2 > \pi, \\ 1 & \text{else.} \end{cases} \tag{4.23}$$

Using (4.22), we see that (4.21) holds for

$$X := V_1 \hat{\Lambda}_1, Y := V_2 \hat{\Lambda}_2,$$
$$\Lambda := s M^{\frac{1}{2}} = s \operatorname{diag}\left(\mu^{\frac{1}{2}}, \overline{\mu}^{\frac{1}{2}}\right).$$

In conclusion we have

$$H V_\lambda = HQ \begin{bmatrix} X \\ Y \end{bmatrix} = Q \begin{bmatrix} 0 & M_1 \\ M_2 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = Q \begin{bmatrix} X \\ Y \end{bmatrix} \Lambda = V_\lambda \Lambda.$$

The orthogonality conditions (4.19) remain to be shown. We observe $Q^{\mathsf{H}} K Q = \frac{1}{2} \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$ and with

$$V_\lambda = Q \begin{bmatrix} X_\lambda \\ Y_\lambda \end{bmatrix}, \quad X_\lambda = V_{1,\lambda} \hat{\Lambda}_1, \quad Y_\lambda = V_{2,\lambda} \hat{\Lambda}_2,$$
$$V_\gamma = Q \begin{bmatrix} X_\gamma \\ Y_\gamma \end{bmatrix}, \quad X_\gamma = V_{1,\gamma} \hat{\Gamma}_1, \quad Y_\gamma = V_{2,\gamma} \hat{\Gamma}_2,$$

we see

$$\begin{aligned} V_\lambda^{\mathsf{H}} K V_\gamma &= \frac{1}{2}(\hat{\Lambda}_1^H V_{1,\lambda}^H V_{2,\gamma} \hat{\Gamma}_2 + \hat{\Lambda}_2^H V_{2,\lambda}^H V_{1,\gamma} \hat{\Gamma}_1), \\ V_\lambda^{\mathsf{H}} J V_\gamma &= \frac{1}{2}(\hat{\Lambda}_1^H V_{1,\lambda}^H V_{2,\gamma} \hat{\Gamma}_2 - \hat{\Lambda}_2^H V_{2,\lambda}^H V_{1,\gamma} \hat{\Gamma}_1). \end{aligned} \tag{4.24}$$

This expressions are equal to 0 if $V_{2,\lambda}$ and $V_{1,\gamma}$ contain left and right eigenvectors corresponding to different eigenvalue pairs $\{\lambda, \overline{\lambda}\} \neq \{\gamma, \overline{\gamma}\}$ of $M_1 M_2$. If $\lambda = \gamma$ and if V_λ and V_γ were constructed in the same way, we have $\hat{\Lambda}_1 = \hat{\Gamma}_1$ and $\hat{\Lambda}_2 = \hat{\Gamma}_2$. The equations (4.24) simplify to $V_\lambda^{\mathsf{H}} K V_\lambda = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $V_\lambda^{\mathsf{H}} J V_\lambda = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. We used that $\hat{\Lambda}_1^H \hat{\Lambda}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ following from the definition (4.17). The orthogonality conditions (4.20) are shown in a similar way. □

CHAPTER 5

A MASSIVELY PARALLEL IMPLEMENTATION FOR BETHE-SALPETER EIGENVALUE PROBLEMS OF FORM I

Contents

5.1 Introduction

In this Chapter, we focus on ways to solve the Bethe-Salpeter eigenvalue problem, when it is not possible to exploit time-inversion symmetry of the basis functions in order to generate a matrix of form II. If it is possible to generate form II, it should be preferred. For an explanation see Chapter 4 and for resulting methods see Chapter 6.

We consider a BSE matrix of form I as given in (4.1) for which the definiteness

property (4.3) holds, i.e.

$$H_{BS} = \begin{bmatrix} A & B \\ -\bar{B} & -\bar{A} \end{bmatrix} = \begin{bmatrix} A & B \\ -B^{\mathsf{H}} & -A^{\mathsf{T}} \end{bmatrix}, \qquad A = A^{\mathsf{H}}, \quad B = B^{\mathsf{T}} \in \mathbb{C}^{n\times n}, \tag{5.1}$$

$$\begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix} H_{BS} = \begin{bmatrix} A & B \\ \bar{B} & \bar{A} \end{bmatrix} > 0. \tag{5.2}$$

The blocks A, B, are dense and extremely large up to a dimension of about $n = 50\,000$.

Because of the large matrix size, it is of imminent importance to develop algorithms that run in parallel and can exploit the computational resources available on a supercomputer. We aim for a solution method that preserves this structure under the influence of inevitable numerical errors, i.e. that guarantees that the eigenvalues come in pairs or quadruples, respectively (see Section 4.2). General methods for eigenvalue problems, such as the QR/QZ algorithm [83], destroy this property. In this case, it is not clear anymore which eigenpairs correspond to the same excitation state.

In Section 5.2, we will see how an eigenvalue problem of this form can be transformed into a skew-symmetric eigenvalue problem of equal size. Afterwards, we present an approach to solve skew-symmetric eigenvalue on high-performance architectures, implemented within the ELPA library [119]. Apart from the application in the Bethe-Salpeter approach, skew-symmetric eigenvalue problems also appear in many other areas and are worth to be studied in their own right. A matrix $A \in \mathbb{R}^{n\times n}$ is called skew-symmetric if $A = -A^{\mathsf{T}}$. We are interested in eigenvalues and eigenvectors of A.

The symmetric eigenvalue problem, i.e. the case $A = A^{\mathsf{T}}$, has been studied in depth for many years. It lies at the core of many applications in different areas such as electronic structure computations (see Section 3.3). Many methods for its solution have been proposed [83] and successfully implemented. Optimized libraries for many platforms are widely available [13, 42]. With the rise of more advanced computer architectures and more powerful supercomputers, the solution of increasingly complex problems comes within reach. Parallelizability and scalability become key issues in algorithm development. The ELPA library [119] is one endeavor to tackle these challenges and provides highly competitive direct solvers for symmetric (and Hermitian) eigenvalue problems running on distributed memory machines such as compute clusters.

The skew-symmetric case [181] lacks the ubiquitous presence of its symmetric counterpart and has not received the same extensive treatment. We close this gap by extending the ELPA methodology to the skew-symmetric case.

Our motivation stems from the connection to the Hamiltonian eigenvalue problem, which has many applications in control theory and model order reduction [37]. A real Hamiltonian matrix H is connected to a symmetric matrix M via

$$M = JH, \quad J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}.$$

If M is positive definite, the Hamiltonian eigenvalue problem can be recast into a skew-symmetric eigenvalue problem using the Cholesky factorization $M = LL^{\mathsf{T}}$. The

eigenvalues of H are given as eigenvalues of the skew-symmetric matrix $L^\mathsf{T}JL$ and eigenvectors can be transformed accordingly. This situation occurs for example in [159], where a structure-preserving method for the solution of the Bethe-Salpeter eigenvalue problem is described.

The remaining chapter is structured as follows. The solution approach of the Bethe-Salpeter eigenvalue problem via skew-symmetric matrices is presented in Section 5.2. Section 5.3 reintroduces the methods used by ELPA and points out the necessary adaptations to make them work for skew-symmetric matrices. Section 5.4 provides performance results of the ELPA extension, including GPU acceleration, and points out the speedup achieved in the context of the Bethe-Salpeter eigenvalue problem.

5.2 Solving the Bethe-Salpeter eigenvalue problem of form I

In general, we are interested in all eigenpairs of the matrix H_{BS} given in (5.1), as they contain valuable information on the excitations of the system. As shown in Chapter 4, the definiteness property (5.2) leads to H_{BS} having only real eigenvalues which come in pairs $\pm\lambda$. A structure-preserving method relying on this assumption is developed in [159] and has been made available as BSEPACK. It uses (partly modified) ScaLAPACK routines and runs in parallel on distributed memory systems. The main idea is to exploit a connection to a Hamiltonian eigenvalue problem given in the following theorem.

Theorem 5.1 (Theorem 2 in [159]):

Let $Q = \frac{1}{\sqrt{2}}\begin{bmatrix} I & -\mathrm{i}I \\ I & \mathrm{i}I \end{bmatrix}$, then Q is unitary and

$$Q^\mathsf{H}\begin{bmatrix} A & B \\ -\bar{B} & -\bar{A} \end{bmatrix} Q = i\begin{bmatrix} \mathrm{Im}(A+B) & -\mathrm{Re}(A-B) \\ \mathrm{Re}(A+B) & \mathrm{Im}(A-B) \end{bmatrix} =: \mathrm{i}H,$$

where H is real Hamiltonian, i.e. $JH = (JH)^\mathsf{T}$. ◊

Let

$$M = JH = \begin{bmatrix} \mathrm{Re}(A+B) & \mathrm{Im}(A-B) \\ -\mathrm{Im}(A+B) & \mathrm{Re}(A-B) \end{bmatrix} \tag{5.3}$$

be the symmetric matrix associated with the Hamiltonian matrix H. Its positive definiteness follows from property (5.2), which can be seen in the following way. Let the matrices K and $\mathcal{H}$ be given as

$$K = \begin{bmatrix} I_n & \\ & -I_n \end{bmatrix}, \qquad \mathcal{H} = \begin{bmatrix} A & B \\ \bar{B} & \bar{A} \end{bmatrix},$$

i.e. $H_{BS} = K\mathcal{H}$. With the matrix Q from Theorem 5.1 we have

$$M = -\mathrm{i}JQ^{\mathsf{H}}K\mathcal{H}Q. \tag{5.4}$$

It is easily verified that

$$-\mathrm{i}JQ^{\mathsf{H}}KQ = I_n,$$

i.e. $-\mathrm{i}JQ^{\mathsf{H}}K$ is the inverse of Q. The construction of M (5.4) can therefore be seen as a similarity transformation of $\mathcal{H}$. If $\mathcal{H}$ is positive definite, so is M. The method described in [159] relies on this property in order to guarantee the existence of the Cholesky factorization of M. It performs the following steps.

1. Construct M as in (5.3).
2. Compute a Cholesky factorization $M = LL^{\mathsf{T}}$.
3. Compute eigenpairs of the skew-symmetric matrix $L^{\mathsf{T}}JL$, where $J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$.
4. Perform the eigenvector back transformation associated with Cholesky factorization and transformation to Hamiltonian form (Theorem 5.1).

The main workload is given as the solution of a skew-symmetric eigenvalue problem (Step 3). As a proof of concept, solution routines for the symmetric eigenvalue problem from the ScaLAPACK reference implementation [42] were adapted to the skew-symmetric setting in [159]. Here, the matrix is reduced to tridiagonal form using Householder transformations. The tridiagonal eigenvalue problem is solved via bisection and inverse iteration.

The ScaLAPACK reference implementation is not regarded as a state-of-the art solver library. When performance and scalability are issues, one generally turns to optimized libraries such as ELPA [119] or vendor-specific implementations such as Intel's MKL. Within BSEPACK, ScaLAPACK can be substituted by ELPA working on skew-symmetric matrices. The resulting performance benefits are discussed in Section 5.4.2.

5.3 Solution method

5.3.1 Solving the symmetric eigenvalue problem in ELPA

The ELPA library [11, 15, 119] is a highly optimized parallel MPI-based code [128]. It shows great scalability over thousands of CPU cores and contains low-level optimizations targeting various compute architectures [110]. When only a portion of eigenvalues and eigenvectors are required, this is exploited algorithmically and results in performance benefits. We briefly describe the well-established procedure employed by ELPA. This forms the basis of the method for skew-symmetric matrices described in the next subsection.

ELPA contains functionality to deal with symmetric-definite generalized eigenvalue problems. In this chapter, we focus on the standard eigenvalue problem for simplicity. This is reasonable as it is the most common use case and forms the basis of any method for generalized problems. We only consider real skew-symmetric problems. The reason is that any real skew-symmetric or complex skew-Hermitian problem can be transformed into a Hermitian eigenvalue problem by multiplying it with the imaginary unit i. This problem can be solved using the available ELPA functionality for complex matrices. For the real case, this induces complex arithmetic, which should obviously be avoided, but for complex matrices this is a viable approach.

We consider the symmetric eigenvalue problem, i.e. the orthogonal diagonalization of a matrix,

$$Q^\mathsf{T} A Q = \Lambda,$$

where $A = A^\mathsf{T} \in \mathbb{R}^{n \times n}$ is the matrix whose eigenvalues are sought. We are looking for the orthogonal eigenvector matrix Q and the diagonal matrix Λ containing the eigenvalues. The solution is carried out in the following steps:

1. Reduce A to tridiagonal form, i.e. find an orthogonal transformation Q_{trd} such that

   $$A_{trd} = Q_{trd}^\mathsf{T} A Q_{trd}$$

   is tridiagonal. This is done by accumulating Householder transformations

   $$Q_{trd} = Q_1 Q_2 \cdots Q_{n-1},$$

   where $Q_i = I - \tau_i v_i v_i^\mathsf{T}$ represents the i-th Householder transformation that reduces the i-th column and row of the updated $Q_{i-1}^\mathsf{T} \cdots Q_1^\mathsf{T} A Q_1 \cdots Q_{i-1}$ to tridiagonal form. The matrices Q_i are not formed explicitly but are represented by the Householder vectors v_i. These are stored in place of the eliminated columns of A.

2. Solve the tridiagonal eigenvalue problem, i.e. find orthogonal Q_{diag} such that

   $$\Lambda = Q_{diag}^\mathsf{T} A_{trd} Q_{diag}.$$

   In ELPA, this step employs a tridiagonal divide-and-conquer scheme.

3. Transform the required eigenvectors back, i.e. perform the computation

   $$Q = Q_{trd} Q_{diag}.$$

The ELPA solver comes in two flavors which define the details of the transformation steps, i.e. Steps 1 and 3. The solver ELPA1 works as described, the reduction to tridiagonal form is performed in one step. The ELPA2 approach splits the transformations into two parts. Step 1 becomes

1. a) Reduce A to banded form, i.e. compute orthogonal Q_{band} such that

$$A_{band} = Q_{band}^{\mathsf{T}} A Q_{band}$$

is a band matrix.

b) Reduce the banded form to tridiagonal form, i.e. compute orthogonal Q_{trd} such that

$$A_{trd} = Q_{trd}^{\mathsf{T}} A_{band} Q_{trd}$$

is tridiagonal.

Accordingly, the back transformation step is split into two parts:

3. a) Perform the back transformation corresponding to the band-to-tridiagonal reduction

$$\tilde{Q} = Q_{trd} Q_{diag}.$$

b) Perform the back transformation corresponding to the full-to-band reduction

$$Q = Q_{band} \tilde{Q}.$$

The benefit of the two-step approach is that more efficient BLAS-3 procedures can be used in the tridiagonalization process (see Section 2.4) and an overlap of communication and computation is possible. As a result, a lower runtime can generally be observed in the tridiagonalization, compared to the one-step approach. This comes at the cost of more operations in the eigenvector back transformation due to the extra step that has to be performed. Therefore, the ELPA2 approach is superior to the ELPA1 variant in particular when only a portion of the eigenvectors is sought. In the context of skew-symmetric eigenvalue problems, this becomes pivotal as the purely imaginary eigenvalues come in pairs $\pm\lambda\mathrm{i}$, $\lambda \in \mathbb{R}$. The eigenvectors are given as the complex conjugates of each other. It is therefore enough to compute half of the eigenvalues and eigenvectors.

Both approaches are extended to skew-symmetric matrices in this work.

5.3.2 Solving the skew-symmetric eigenvalue problem

Like a symmetric matrix, a skew-symmetric matrix can be reduced to tridiagonal form using Householder transformations. A Householder transformation represents a reflection onto a scaled first unit vector e_1. Let H be a transformation that acts on a vector v such that $Hv = \alpha e_1$ (see Theorem 2.9). Obviously $-v$ is transformed to $H(-v) = -\alpha e_1$ by the same H. Therefore all tridiagonalization methods that work on symmetric matrices, such as the ones implemented in ELPA, can in principle work on skew-symmetric matrices as well.

A skew-symmetric tridiagonal matrix is related to a symmetric one via the following observation [181].

Lemma 5.2:
With the unitary matrix $D = \mathrm{diag}\{1, \mathrm{i}, \mathrm{i}^2, \ldots, \mathrm{i}^{n-1}\}$, where i denotes the imaginary unit, $\alpha_j \in \mathbb{R}$, it holds

$$-\mathrm{i}D^{\mathsf{H}} \begin{bmatrix} 0 & \alpha_1 & & \\ -\alpha_1 & 0 & \ddots & \\ & \ddots & \ddots & \alpha_{n-1} \\ & & -\alpha_{n-1} & 0 \end{bmatrix} D = \begin{bmatrix} 0 & \alpha_1 & & \\ \alpha_1 & 0 & \ddots & \\ & \ddots & \ddots & \alpha_{n-1} \\ & & \alpha_{n-1} & 0 \end{bmatrix}. \qquad \lozenge$$

After the reduction to tridiagonal form, the symmetric tridiagonal system is solved using a divide-and-conquer method [15]. As a first step of the back transformation, the resulting (real) eigenvectors have to be multiplied by the (complex) matrix D. Then the back transformations corresponding to the tridiagonalization take place. Algorithm 5.1 outlines this process. It is very similar to the method employed for symmetric eigenvalue problems. The differences are the addition of step 3 and changes in the implementation, which are given in detail in Sections 5.3.3.1 and 5.3.3.2.

In ELPA2, the transformation steps (1 and 4 in Algorithm 5.1) are both split into two parts as described in Section 5.3.1.

5.3.3 Implementation

Extending ELPA for skew-symmetric matrices means adding the back transformation step involving D. In contrast to symmetric matrices, skew-symmetric matrices have complex eigenvectors and strictly imaginary eigenvalues. Computationally complex values are introduced in Algorithm 5.1 with D in step 3. Further transformations have to be performed for the real and the imaginary part individually. It is preferable to set up an array with complex data type entries representing the eigenvectors as late as possible, so that we can benefit from efficient routines in double precision. The routines for the eigenvector back transformation corresponding to tridiagonalization do not change, because all they do is to apply Householder transformations to non-symmetric (and non-skew-symmetric) matrices. They are applied on the real and imaginary part independently, realizing the complex back transformation in real arithmetic. The symmetric tridiagonal eigensolver can be used as is. Making it aware of the zeros on the diagonal might turn out to be numerically or computationally beneficial.

We now examine the implementation of the two tridiagonalization approaches in ELPA1 and ELPA2 in more detail. At many points in the original implementation, symmetry of the matrix is assumed in order to avoid unnecessary computations and to efficiently reuse data available in the cache. In this section, we recollect some details of the tridiagonal reduction in order to point out these instances. Here, the implicit assumptions can be changed from "symmetric" to "skew-symmetric" by simple sign changes.

ELPA is based on the well established and well documented 2D block-cyclic data layout introduced by ScaLAPACK for load balancing reasons (see Section 2.4). It is therefore compatible with ScaLAPACK and can act as a drop-in replacement, while

Algorithm 5.1: Solving a skew-symmetric eigenvalue problem.

Data: $A = -A^{\mathsf{T}} \in \mathbb{R}^{n\times n}$

Result: Unitary eigenvectors $Q \in \mathbb{C}^{n\times n}$, $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ such that $Q^{\mathsf{H}} A Q = \mathrm{diag}\{\lambda_1 \mathrm{i}, \ldots, \lambda_n \mathrm{i}\}$.

1 Reduce A to tridiagonal form, i.e. generate Q_{trd} such that

$$Q_{trd}^{\mathsf{T}} A Q_{trd} = A_{trd} = \begin{bmatrix} 0 & \alpha_1 & & \\ -\alpha_1 & 0 & \ddots & \\ & \ddots & \ddots & \alpha_{n-1} \\ & & -\alpha_{n-1} & 0 \end{bmatrix}.$$

2 Solve the eigenvalue problem for the symmetric tridiagonal matrix $-\mathrm{i}D^{\mathsf{H}} A_{trd} D$, where $D = \mathrm{diag}\{1, \mathrm{i}, \mathrm{i}^2, \ldots, \mathrm{i}^n\}$, i.e. generate Q_{diag} such that

$$Q_{diag}^{\mathsf{T}} \begin{bmatrix} 0 & \alpha_1 & & \\ \alpha_1 & 0 & \ddots & \\ & \ddots & \ddots & \alpha_{n-1} \\ & & \alpha_{n-1} & 0 \end{bmatrix} Q_{diag} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}.$$

3 Back transformation corresponding to symmetrization (see Lemma 5.2), i.e. compute

$$Q \leftarrow D Q_{diag} \in \mathbb{C}^{n\times n}$$

4 Back transformation corresponding to band-to-tridiagonal reduction, i.e. compute

$$Q \leftarrow Q_{trd} Q$$

no ScaLAPACK routines are used by ELPA itself. In general, each process works on the part of the matrix that was assigned to it. This chunk of data resides in the local memory of the process. Communication between processes is realized via MPI. Each process calls serial BLAS routines. Additional CUDA and OpenMP support is available.

5.3.3.1 Tridiagonalization in ELPA1

In ELPA1, the tridiagonalization is realized in one step using Householder transformations. The computation of the Householder vectors is not affected by the symmetry of a matrix. Essentially, the tridiagonalization of a matrix comes down to a series of rank-2 updates [121], described in the following. Given a Householder vector v, the update of the trailing submatrix is performed as

$$
\begin{aligned}
A &\leftarrow (I - \tau v v^\mathsf{T}) A (I - \tau v v^\mathsf{T}) && (5.5)\\
&= A + v \underbrace{(0.5\tau^2 v^\mathsf{T} A v v^\mathsf{T} - \tau v^\mathsf{T} A)}_{u_1^\mathsf{T}} + \underbrace{(0.5\tau^2 v v^\mathsf{T} A v - \tau A v)}_{u_2} v^\mathsf{T} && (5.6)\\
&= A + v u_1^\mathsf{T} + u_2 v^\mathsf{T} && (5.7)\\
&= A + \begin{bmatrix} v & u_2 \end{bmatrix} \begin{bmatrix} u_1 & v \end{bmatrix}^\mathsf{T}. && (5.8)
\end{aligned}
$$

For symmetric matrices it holds $u_1 = u_2$. This is assumed in the original ELPA implementation. For skew-symmetric matrices it holds $u_1 = -u_2$. In ELPA1, the two matrices $\begin{bmatrix} v & u_2 \end{bmatrix}$ and $\begin{bmatrix} u_1 & v \end{bmatrix}^\mathsf{T}$ are stored explicitly. Actual updates are performed using `GEMM` and `GEMV` routines. The matrices differ in the data layout, i.e. which process owns which part of the matrix. After the vector u_1 is computed, it is transposed and redistributed to represent u_2 in $\begin{bmatrix} v & u_2 \end{bmatrix}$. Here, for the skew-symmetric variant, a sign change is introduced. The skew-symmetric update now reads

$$
A \leftarrow A + \begin{bmatrix} v & -u_1 \end{bmatrix} \begin{bmatrix} u_1 & v \end{bmatrix}^\mathsf{T}.
$$

During the computation of u_1, symmetry is assumed in the computation of $A^\mathsf{T} v$. In particular, the code assumes that an off-diagonal matrix tile is the same as in the transposed matrix. Another sign change corrects this assumption for skew-symmetric matrices.

5.3.3.2 Tridiagonalization in ELPA2

In ELPA2, the tridiagonalization is split into two parts. First, the matrix is reduced to banded form, then to tridiagonal form. For the reduction to banded form, the Householder vectors are computed by the process column owning the diagonal block. They are accumulated in a triangular matrix $T \in \mathbb{R}^{nb \times nb}$, where nb is the block size. The product of Householder matrices is stored via its storage-efficient representation [157]

$$
Q = H_1 \cdots H_{nb} = I - V T V^\mathsf{T}, \tag{5.9}
$$

where $V = \begin{bmatrix} v_1 & \cdots & v_{nb} \end{bmatrix}$ contains the Householder vectors. $H_i = I - \tau_i v_i v_i^\mathsf{T}$ is the Householder matrix corresponding to the i-th Householder transformation.

In this context, the update of the matrix A takes the following shape, analogous to the direct tridiagonalization described in Section 5.3.3.1.

$$A \leftarrow (I - VTV^\mathsf{T})^\mathsf{T} A (I - VTV^\mathsf{T}) \tag{5.10}$$

$$= A + V \underbrace{\left(0.5 T^\mathsf{T} V^\mathsf{T} A V T V^\mathsf{T} - T^\mathsf{T} V^\mathsf{T} A\right)}_{U_1^\mathsf{T}} + \underbrace{\left(0.5 V T^\mathsf{T} V^\mathsf{T} A V T - A V T\right)}_{U_2} V^\mathsf{T} \tag{5.11}$$

$$= A + \begin{bmatrix} V & U_2 \end{bmatrix} \begin{bmatrix} U_1 & V \end{bmatrix}^\mathsf{T}. \tag{5.12}$$

It holds $U_1 = U_2$ if A is symmetric, and $U_1 = -U_2$ if A is skew-symmetric. Each process computes the relevant parts of U_1 in a series of (serial) matrix operations and updates the portion of A that resides in its memory. Here, the symmetry of A is assumed and exploited at various points in the implementation. Sign changes have to be applied at these instances.

For the banded-to-tridiagonal reduction, the matrix is redistributed in the form of a 1D block-cyclic data layout. Each process owns a diagonal and a subdiagonal block. The reduction of a particular column introduces fill-in in the neighboring block. The "bulge-chasing" is realized as a pipelined algorithm, where computation and communication can be overlapped by reordering certain operations [15, 16].

The update of the diagonal blocks takes the same form as in ELPA1 (equations (5.5) to (5.8)). Here, no matrix multiplication is employed but BLAS-2 routines are used working directly with the Householder vectors. It holds $u_1 = u_2$ for symmetric A and $u_1 = -u_2$ for skew-symmetric A. In the symmetric case, the update is realized via a symmetric rank-2 update (`SYR2`). We implemented a skew-symmetric variant of this routine which realizes the skew-symmetric rank-2 update $A \leftarrow A - vu^\mathsf{T} + uv^\mathsf{T}$. For the setup of u, a skew-symmetric variant of the BLAS routine performing a symmetric matrix vector product (`SYMV`) is necessary.

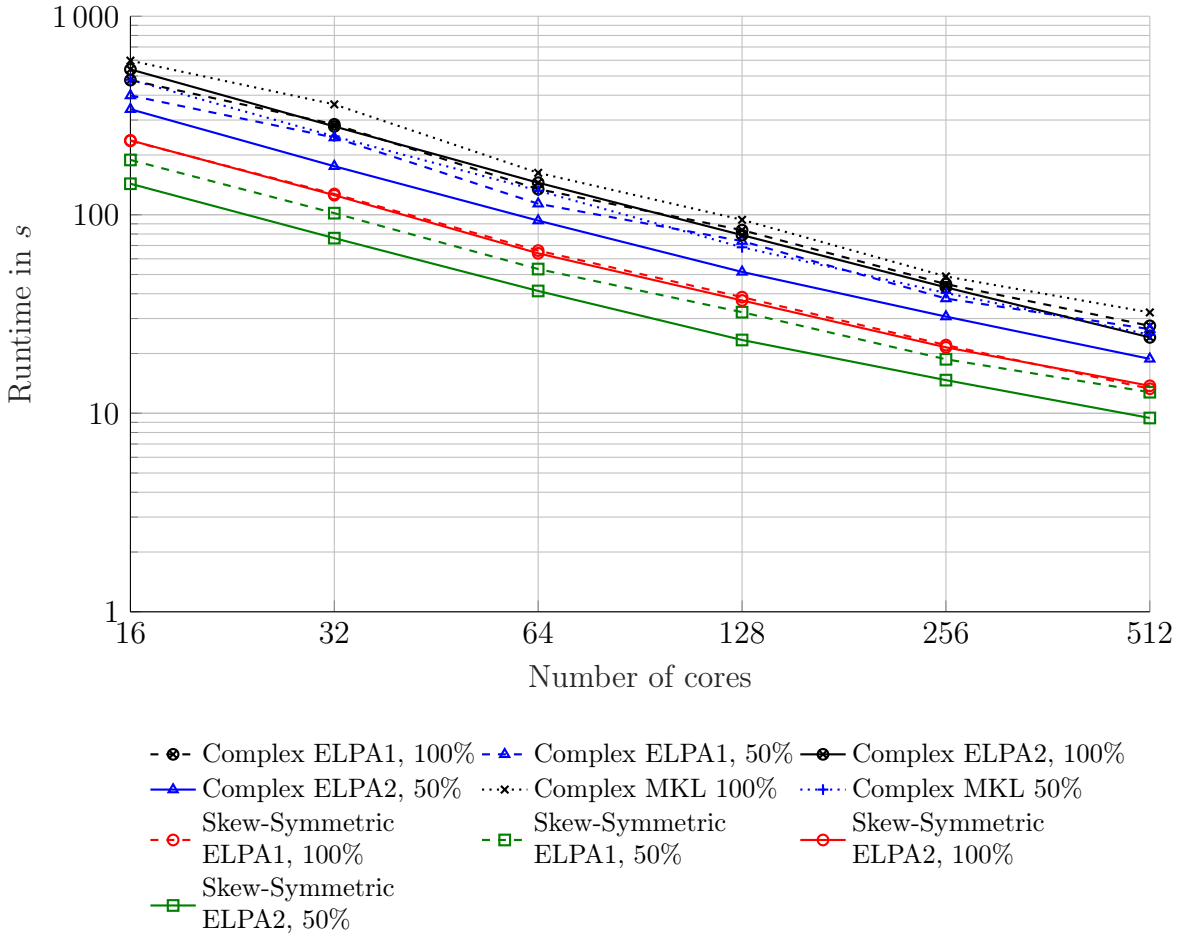
The other parts of Algorithm 5.1 are adopted from the symmetric implementation without changes. The computation of Householder vectors, the accumulation of the Householder transformations (see Lemma 2.13) in a triangular matrix and the update of the local block during reduction to banded form do not have to be changed compared to symmetric ELPA. This is because they act on the lower part of the matrix so that possible (skew-)symmetry has no effect.

5.4 Numerical experiments

5.4.1 ELPA benchmarks

In this section we present performance results for the skew-symmetric ELPA extension. All test programs are run on the `mechthild` compute cluster, located at the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg, Germany.

Figure 5.1: Scaling of the ELPA solver for skew-symmetric matrices. For comparison, the runtimes for the alternative solution method via complex Hermitian solvers are included. Here, ELPA and Intel's MKL 2018 routines `PZHEEVD` and `PZHEEVR` are used. The matrix has a size of $n = 20\,000$.



Up to 32 nodes are used, which consist of 2 Intel Xeon Silver 4110 (Skylake) processors with 8 cores each, running at 2.1 GHz. The Intel compiler, MPI library and MKL in the 2018 version are used in all test programs. The computations use randomly generated skew-symmetric matrices in double precision.

Figure 5.1 shows the resulting performance and the scaling properties of ELPA for a medium sized skew-symmetric matrix ($n = 20\,000$). As an alternative to the approach described in this work, the skew-symmetric matrix can be multiplied with the imaginary unit i. The resulting complex Hermitian matrix can be diagonalized using available methods in ELPA or Intel's ScaLAPACK implementation shipped with the MKL. This represents the only previously available approach to solve skew-symmetric eigenvalue problems in a massively parallel high-performance setting.

For skew-symmetric matrices, only 50% of eigenvalues and eigenvectors need to be computed, as they are purely imaginary and come in pairs $\pm\lambda\mathrm{i}, \lambda \in \mathbb{R}$. The runtime

Table 5.1: Execution time speedups achieved by different aspects of the solution approach with varying number of cores.

#Cores	Compl. ELPA2 100% vs. Compl. MKL 100 %	Compl. ELPA2 50% vs. Compl. MKL 50%	Skew-Sym. ELPA2 50% vs. Compl. ELPA2 50%	Skew-Sym. ELPA2 50% vs. Compl. MKL 50%
16	1.10	1.41	2.33	3.28
32	1.29	1.41	2.30	3.24
64	1.11	1.40	2.32	3.25
128	1.18	1.33	2.20	2.93
256	1.17	1.28	2.16	2.76
512	1.21	1.51	1.87	2.82

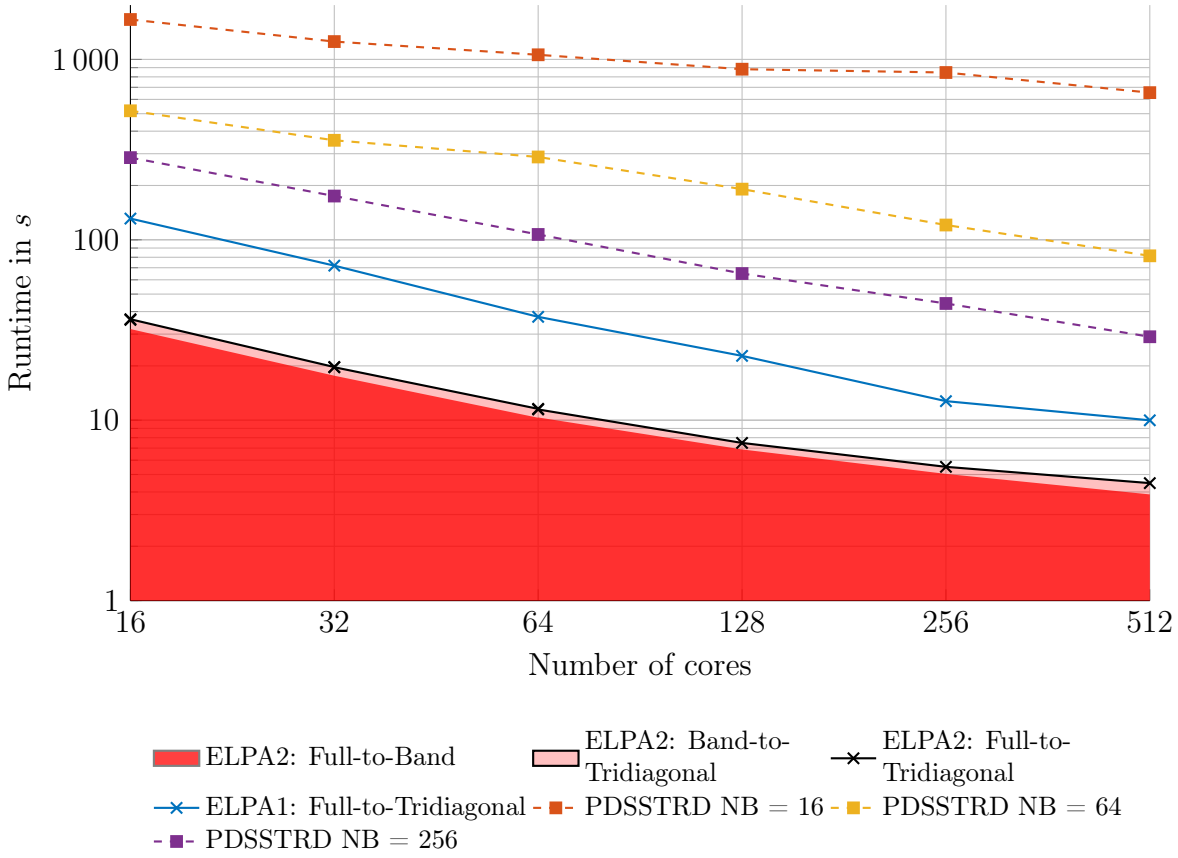
measurements for 100% are included for reference.

Figure 5.1 shows that all approaches display good scalability in the examined setting. Skew-symmetric ELPA runs 2.76 to 3.28 times faster than the complex MKL based solver, where both only compute 50% of eigenpairs. The data gives further insight into how this improvement is achieved. Table 5.1 compares the runtimes for different solvers and presents the achieved speedups. When we compare complex 100% solvers, ELPA already improves performance by a factor of 1.1 to 1.29 (column 2 in Table 5.1). When all eigenpairs are computed, ELPA1 and ELPA2 yield very similar runtime results which is why only ELPA2 is considered in Table 5.1. The two-step approach employed by ELPA2 pays off in particular when not all eigenpairs are sought, which is the case here. When complex 50% solvers are compared (ELPA2 vs. MKL, column 3 in Table 5.1), the achieved speedup increases to a value between 1.28 and 1.51. The largest impact on the performance is caused by avoiding complex arithmetic. This is represented by the speedup achieved by the skew-symmetric 50% ELPA2 implementation compared to the complex 50% ELPA2 implementation (column 4 of Table 5.1). This accounts for an additional speedup of 1.87 to 2.33.

The tridiagonalization is an essential step in every considered solution scheme and contributes a significant portion of the execution time. The fewer eigenpairs are sought, the more dominant it becomes with respect to computation time. Figure 5.2 displays the runtimes and scalability of available tridiagonalization techniques for skew-symmetric matrices. As an alternative implementation to the presented approaches there is a tridiagonalization routine `PDSSTRD` shipped in BSEPACK [159]. It is an adapted version of the ScaLAPACK reference implementation.

All discussed implementations are based on the 2D-block-cyclic data distribution established by ScaLAPACK. Here, the matrix is divided into blocks of a certain size NB. The blocks are distributed to processes organized in a 2D grid in a cyclic manner (see Section 2.4). Typically, the block size is a parameter chosen once in a software project. The data redistribution to data layouts defined by other block sizes is avoided as this involves expensive all-to-all communication. The main disadvantage of the

Figure 5.2: Scaling of the tridiagonalization in two steps (ELPA2) and one step (ELPA1). We compare it to the runtimes of the tridiagonalization routine for skew-symmetric matrices `PDSSTRD` available in BSEPACK [159] for different block sizes NB. The matrix size is $n = 20\,000$.



`PDSSTRD` routine is that it is very susceptible to the chosen block size, both with regard to scalability and overall performance. This makes it less suitable to be included in larger software projects, where the block size is a parameter predefined by other factors. ELPA (both the one and two-step version) on the other hand does not have this problem and performs equally well for all data layouts.

Figure 5.2 also displays the advantage of the two-step tridiagonalization over the one-step approach. Here, the performance is dominated by the first step, i.e. the reduction to banded form.

In the context of electronic structure computations, the matrices of interest can become extremely large. Figure 5.3 displays the achieved runtime improvements for larger matrices up to a size of $n = 125\,000$ when using 256 CPU cores. The individual speedups are presented in Table 5.2. For large matrices we achieve a speedup of up to 3.67 compared to the available MKL routine.

Figure 5.3: Runtimes for solving eigenvalue problems of larger sizes. 256 CPU cores were used, i.e. 16 nodes on the `mechthild` compute cluster.

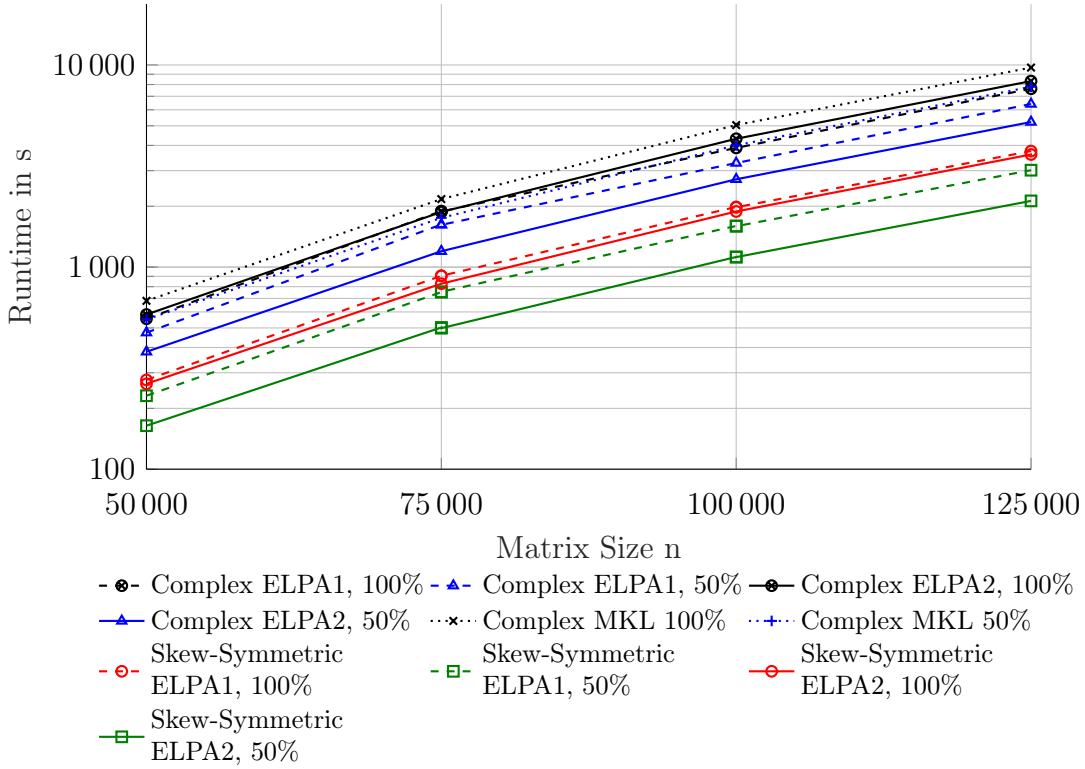


Table 5.2: Execution time speedups achieved by different aspects of the solution approach with varying matrix size. 256 CPU cores were used, i.e. 16 nodes on the `mechthild` compute cluster.

Matrix size	Compl. ELPA2 100% vs. Compl. MKL 100 %	Compl. ELPA2 50% vs. Compl. MKL 50%	Skew-Sym. ELPA2 50% vs. Compl. ELPA2 50%	Skew-Sym. ELPA2 50% vs. Compl. MKL 50%
50 000	1.17	1.45	2.32	3.35
75 000	1.16	1.46	2.39	3.50
100 000	1.17	1.47	2.42	3.57
125 000	1.17	1.49	2.46	3.67

5.4.1.1 GPU acceleration

For the 1-step tridiagonalization approach (ELPA1), there is a GPU-accelerated version available that gets shipped with the ELPA library [111]. The design approach is to stick with the same code base as the CPU-only version, and offload compute-intense parts, such as BLAS-3 operations, to the GPU in order to benefit from its massive parallelism. This is done using the CUBLAS library provided by NVIDIA. Because ELPA2 employs more fine-grained communication patterns, this approach works best for ELPA1. Here, the performance can benefit when the computational intensity is high enough, i.e. when big chunks of data are being worked on by the GPU.

Figure 5.4 shows the performance that can be achieved on one node of the `mechthild` compute cluster, that is equipped with an NVIDIA P100 GPU as an accelerator device. The GPU version is based on ELPA1 and therefore does not benefit from the faster tridiagonalization in ELPA2 (see Figure 5.2 and the discussion in the previous section). Despite this fact, the GPU-accelerated ELPA1 version eventually outperforms the ELPA2 CPU-only version, if the matrix is large enough. In our case the turning point is at around $n = 15\,000$. For smaller matrices the additional work of setting up the CUDA environment and transferring the matrix counteracts any possible performance benefits and results in a larger runtime. For matrices of size $n = 32\,768$ employing the GPU can reduce the runtime from 570 seconds to 328 seconds, i.e. by 41%.

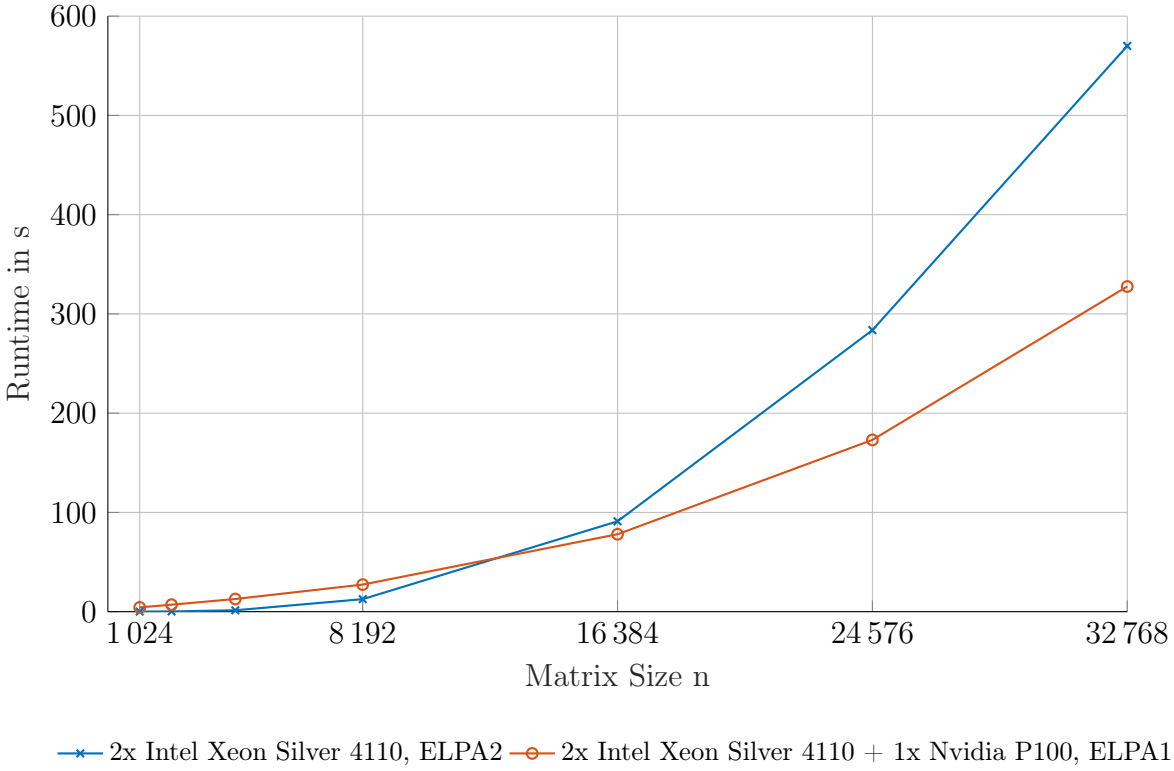
The take-away message of these results is the following. If nodes equipped with GPUs are available and to be utilized, it is important to make sure each node has enough data to work on. This way, the available resources are used most efficiently.

5.4.2 Accelerating BSEPACK

We consider the performance improvements that can be achieved by using the newly developed skew-symmetric eigenvalue solver in the BSEPACK [159] software, described in Section 5.2. In this procedure, Step 3, the computation of eigenpairs of the skew-symmetric matrix $L^\mathsf{T} J L$, is now performed by the ELPA library.

We evaluate the achieved perfomance for a matrix created by considering hexagonal boron nitride. It is a material with a wide band gap and strong electron-hole interactions resulting in the formation of bound excitons. For materials like this, the Bethe-Salpeter approach was developed, because other methods do not take the electron-hole interaction into account. This is why it is studied widely in experiments as well as theoretically [63, 44, 82, 79, 67, 108, 6]. According to previous studies, excitonic effects and the optical absorption spectrum can be computed accurately via the Bethe-Salpeter approach. The first Brillouin zone is discretized, such that there are $N_k = 16 \times 16 \times 4$ possible values for the wave vector k (see Section 3.6). We choose $N = 5$ occupied and unoccupied "orbitals", i.e. the five lowest conduction and the five highest valence bands. These choices lead to a matrix dimension of the matrix blocks A, B of $n = N^2 \cdot N_k = 25\,600$. The whole matrix H_{BSE} has dimension 51 200. In the computation of the Bethe-Salpeter matrix, the basis functions are given as plane waves with a cut-off of 387 eV. The static dielectric function is expanded in plane

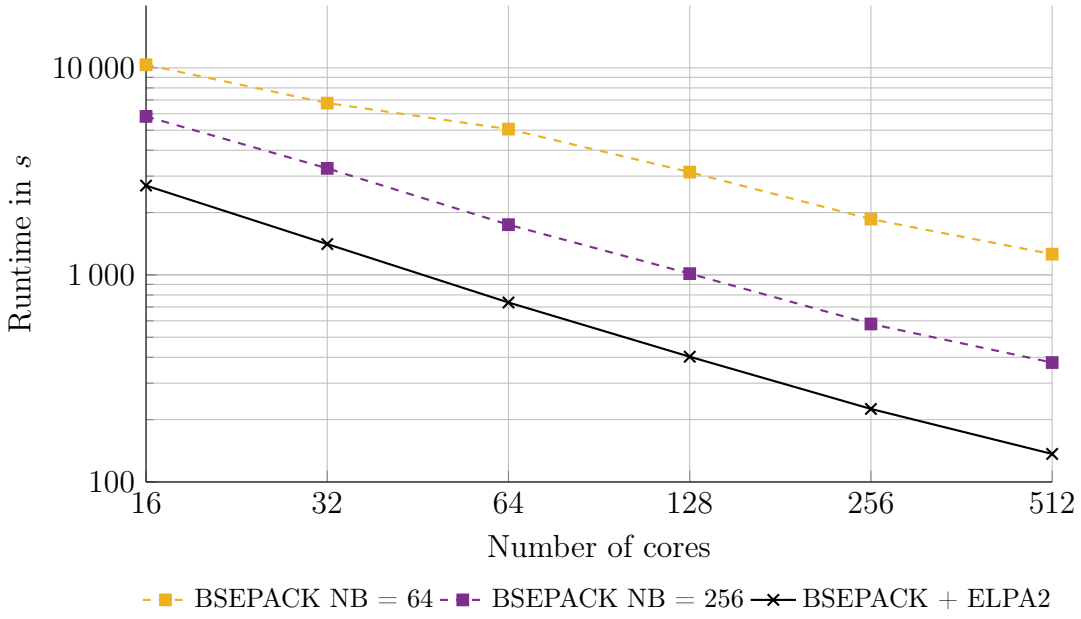
Figure 5.4: Runtimes for solving eigenvalue problems on one node on the `mechthild` compute cluster employing a GPU.



waves with a cutoff of 132 eV and obtained from ABINIT [84]. The EXC code [1] is used to construct the Bethe-Salpeter matrix.

Figure 5.5 displays the achieved runtimes of BSEPACK for this fixed-size matrix for different core counts. We compare the original version and a version that employs ELPA. The performance of the original solver is highly dependent on the chosen block size (see also Figure 5.2). This parameter determines how the matrix is distributed to the available processes in the form of a 2D block-cyclic data layout. The default is given as $NB = 64$, but choosing a larger block size can increase the performance dramatically, as can be seen in Figure 5.5 for $NB = 256$. Typically, software packages (e.g. [86, 178]) developed for electronic structure computations are large and contain many features, implementing methods for different quantities of interest. The block size is typically predetermined by other considerations. It would mean a serious effort to change it, in order to optimize just one building block of the software. Furthermore, the optimal block size of the original BSEPACK is probably dependent on the given hardware and the given matrix size. Autotuning frameworks could help, but are also very costly and impose an additional implementation effort. A software, that does not show this kind of runtime dependency is greatly preferable. Employing ELPA for the main computational task in BSEPACK fulfills this requirement. The performance of ELPA is nearly independent of the chosen NB, because the block size on the node

Figure 5.5: Scaling of the direct, complex BSEPACK eigenvalue solver for computing the optical absorption spectrum of hexagonal boron nitride. The Bethe-Salpeter matrix (5.1) has a size of 51 200.



level for optimal cache use is decoupled from the block size defining the multi-node data layout.

The ELPA-accelerated version is up to 9.22 times as fast as the original code with the default block size. Even when the block size is increased, using the new solver always yields a better performance. In the case of $NB = 256$, the ELPA-version still performs up to 2.76 times as fast. Choosing even larger block sizes has in general no further positive effect on the performance of the original BSEPACK. Employing ELPA also leads to an improved scalability over the number of cores.

5.5 Conclusions

We have presented a strategy to extend existing solver libraries for symmetric eigenvalue problems to the skew-symmetric case. Applying these ideas to the ELPA library, makes it possible to compute eigenvalues and eigenvectors of large skew-symmetric matrices in parallel with a high level of efficiency and scalability. We benefit from the maturity of the ELPA software project, where many optimizations have been realized over the years. All of these, including GPU support, find their way into the presented skew-symmetric solver. It is always possible to solve a complex Hermitian eigenvalue problem instead of a skew-symmetric one. Our newly developed solver outperforms this strategy, implemented via Intel MKL ScaLAPACK, by a factor of 3. We also observe an increase in performance concerning the Bethe-Salpeter eigenvalue problem. Here, we improve the runtime of available routines by a factor of almost 10, making the

BSEPACK library with ELPA a viable choice as a building block for larger electronic structure packages.

CHAPTER 6

METHODS FOR SOLVING THE BETHE-SALPETER EIGENVALUE PROBLEM OF FORM II

Contents

6.1 Introduction

We have seen in Chapter 4 that it is advisable to set up the Bethe-Salpeter matrix in form II, if it is available, rather than form I, i.e.

$$H_{BS} = \begin{bmatrix} A & B \\ -B & -A \end{bmatrix} \in \mathbb{C}^{2n\times 2n}, \quad A = A^{\mathsf{H}},\ B = B^{\mathsf{H}}. \tag{6.1}$$

Theorem 4.8 and Theorem 4.9 pointed out that in this case it is possible to work on a product eigenvalue problem of half the size instead of the large eigenvalue problem.

The definiteness property

$$KH_{BS} = \begin{bmatrix} A & B \\ B & A \end{bmatrix} > 0, \qquad K = \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix}, \tag{6.2}$$

has a direct effect on the matrices involved in the smaller eigenvalue problem. If it holds, all eigenvalues are real and Theorem 4.9 is not required. The corresponding smaller product eigenvalue problem then has two positive definite factors and can be tackled in fairly straightforward ways. The resulting algorithms are presented in Section 6.2. We present a concise overview of direct methods exploiting the structure

given in (6.1) and improve upon a currently used approach in terms of performance and accuracy. Results and algorithms may partly be known but have not yet been derived in a unified fashion as presented here. This contribution is particularly valuable because the eigenvalue problem appears in many loosely connected fields, spanning nuclear physics, various strands of electronic structure theory and quantum chemistry. Our unified framework shows how techniques that have been developed multiple times in different fields are all based on the same principles. This eases the switch from one technique to a better suited one in the context of complex high performance implementations. The proposed methods are well-suited for high performance computing as they rely on basic linear algebra building blocks for which high performance implementations are readily available.

If the property (6.2) does not hold, the methods available in the literature become very scarce. In Section 6.4 we show how Theorem 4.9 allows us to extend the algorithms from the definite to an indefinite setting.

Iterative methods [22, 158, 91] are certainly attractive in the context of large, possibly sparse matrices and when only the few smallest eigenpairs are sought. It is, however, worthwhile not to neglect direct approaches leading to full diagonalization. When all eigenpairs are of interest, or when it is not clear how many eigenpairs are required a-priori, a direct method is expected to have a favorable performance. Other reasons include ease of implementation, testing and benchmark purposes. Typically, a direct approach is established first and iterative methods are compared to this baseline in terms of performance and accuracy. In order to guarantee a fair comparison, the direct approach should not be needlessly laborious. This is not the case in current implementations [64, 155, 178], which employ a matrix square root. In this thesis, we show that a cheaper Cholesky factorization works just as well and point out the mathematical structure underlying the used algorithms.

6.2 Solving the definite Bethe-Salpeter Eigenvalue problem of form II

6.2.1 Algorithms

We have seen in Theorem 4.8 that the BSE problem of form II with size $2n \times 2n$ can be interpreted as a product eigenvalue problem with two Hermitian factors of size $n \times n$. In this section we assume that the definiteness property (6.2) holds. Then the factors of the product eigenvalue problem are Hermitian positive definite.

In practice, the complete set of eigenvectors provides insight to excitonic effects. To compute them, left and right eigenvectors of the smaller product eigenvalue problem are needed. Product eigenvalue problems are well studied, see e.g. [109]. A general way to solve these problems, which takes the product structure into account to improve numerical properties, is the periodic QR algorithm. This tool can be used for solving general Hamiltonian eigenvalue problems [39]. In this work we focus on non-iterative methods that work for Hermitian factors and transform the problem such that it can

be treated using available HPC libraries like ScaLAPACK or ELPA (see Section 2.4).

The algorithms presented in this section compute the positive part of the spectrum of a BSE matrix and the corresponding eigenvectors. If the eigenvectors corresponding to the negative mirror eigenvalues are of interest, they can easily be computed by employing proposition 2. a) in Theorem 4.4.

A widely used approach for solving the BSE problem of form II [64, 155, 178] relies on the computation of the matrix square root of $M_2 = A - B$. We present this method in the following and relate it to the framework given by Theorem 4.8.

Starting with the eigenvectors of the product eigenvalue problem defined in (4.10), we see

$$M_1 M_2 V_1 = V_1 \Lambda^2 \Leftrightarrow M_2^{\frac{1}{2}} M_1 M_2^{\frac{1}{2}} \hat{V} = \hat{V} \Lambda^2, \tag{6.3}$$

$$V_2^{\mathsf{H}} M_1 M_2 = \Lambda^2 V_2^{\mathsf{H}} \Leftrightarrow \hat{V}^{\mathsf{H}} M_2^{\frac{1}{2}} M_1 M_2^{\frac{1}{2}} = \Lambda^2 \hat{V}^{\mathsf{H}}, \tag{6.4}$$

where $\hat{V} = M_2^{\frac{1}{2}} V_1 = M_2^{-\frac{1}{2}} V_2$ contains the eigenvectors of the Hermitian matrix $M_2^{\frac{1}{2}} M_1 M_2^{\frac{1}{2}}$. The following theorem clarifies how this approach leads to a simple algorithm.

Theorem 6.1:
Let H, M_1, M_2 and Q be given as in Theorem 4.8 and property (6.2) hold, $S = M_2^{\frac{1}{2}}$ and

$$S M_1 S \hat{V} = \hat{V} \Lambda^2$$

be an eigenvalue decomposition with

$$\hat{V}^{\mathsf{H}} \hat{V} = I, \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}_+^{n \times n}.$$

Then

$$V = Q \begin{bmatrix} S^{-1} \hat{V} \Lambda^{\frac{1}{2}} \\ S \hat{V} \Lambda^{-\frac{1}{2}} \end{bmatrix}$$

contains eigenvectors of H corresponding to Λ, i.e. $HV = V\Lambda$. ◊

Proof. We see from the equivalences (6.3) and (6.4) that right and left eigenvectors of $M_1 M_2$ are given by $V_1 = S^{-1}\hat{V}$ and $V_2 = S\hat{V}$. It holds $V_1^{\mathsf{H}} V_2 = I$ so that we are in a position to apply Theorem 4.8. The scaling factors λ_1 and λ_2 for each eigenvector follow from observing $M_2 V_1 = V_2$ and $M_1 V_2 = V_1 \Lambda^2$. So we have $\lambda_1 = \lambda^2$, $\lambda_2 = 1$ for each eigenvalue λ. The eigenvector matrix V_1 needs to be scaled by $\Lambda_1^{\frac{1}{4}} \Lambda_2^{-\frac{1}{4}} = \Lambda^{\frac{1}{2}}$, the eigenvector matrix V_2 needs to be scaled by $\Lambda^{-\frac{1}{2}}$. □

The essential computational effort of this approach is the computation of the matrix square root and the solution of a Hermitian eigenvalue problem. Computing the (principal) square root of a matrix is a nontrivial task, compare e.g. [95, Chapter 6],

with a (perhaps surprisingly) high computational demand. Its efficient computation has been an active area of research. Given a Hermitian positive definite matrix C, its principal square root S, such that $S^2 = C$, can be computed by diagonalizing $M = V_C D_C V_C^{\mathsf{H}}$, and taking the square roots of the diagonal entries of D_C. Then the square root is given by $S := V_C D_C{}^{\frac{1}{2}} V_C^{\mathsf{H}}$. The main computational effort is therefore the subsequent solution of two Hermitian eigenvalue problems.

We now lay out how the product eigenvalue problem (4.10) can be solved by using Cholesky factorizations. Let the Cholesky factorization $M_2 = LL^{\mathsf{H}}$ be given.

Starting with the product eigenvalue problem (4.10), we see

$$M_1 M_2 V_1 = V_1 \Lambda^2 \Leftrightarrow L^{\mathsf{H}} M_1 L \hat{V} = \hat{V} \Lambda^2,$$
$$V_2^{\mathsf{H}} M_1 M_2 = \Lambda^2 V_2^{\mathsf{H}} \Leftrightarrow \hat{V}^{\mathsf{H}} L^{\mathsf{H}} M_1 L = \Lambda^2 \hat{V}^{\mathsf{H}},$$

where $\hat{V} = L^H V_1 = L^{-1} V_2$ contains the eigenvectors of the Hermitian matrix $L^{\mathsf{H}} M_1 L$.

The analogy to Theorem 6.1 is given in the following. The proof is omitted as it is very similar to the one of Theorem 6.1. The role of S is just taken over by L and L^{H} respectively.

Theorem 6.2:
Let H, M_1, M_2 and Q be given as in Theorem 4.8, and property (6.2) hold, $LL^{\mathsf{H}} = M_2$ be a Cholesky decomposition and $L^{\mathsf{H}} M_1 L \hat{V} = \hat{V} \Lambda^2$ be an eigenvalue decomposition with $\hat{V}^{\mathsf{H}} \hat{V} = I$, $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}_+^{n \times n}$. Then

$$V = Q \begin{bmatrix} L^{-\mathsf{H}} \hat{V} \Lambda^{\frac{1}{2}} \\ L \hat{V} \Lambda^{-\frac{1}{2}} \end{bmatrix}$$

contains eigenvectors of H corresponding to Λ, i.e. $HV = V\Lambda$. ◊

Comparing algorithms resulting from Theorems 6.1 and 6.2, we see that the essential work in both algorithms is solving Hermitian positive definite $n \times n$ eigenvalue problems. The Cholesky variant solves one explicitly, the square root variant solves one for computing the matrix square root, which is then used to set up the matrix for the second eigenvalue problem. Then both left and right eigenvectors of the product eigenvalue problem can be inferred from the computed eigenvectors.

Both approaches compute the squared eigenvalues of the original problem. In the numerical linear algebra community this procedure is well known to limit the attainable accuracy [174]. The methods essentially work on the (transformed) matrix product $M_1 M_2$. It corresponds to the squared matrix H^2 as

$$Q^{-1} H^2 Q = \begin{bmatrix} M_1 M_2 & \\ & M_2 M_1 \end{bmatrix},$$

where

$$Q = \frac{1}{2} \begin{bmatrix} I & I \\ -I & I \end{bmatrix}$$

is again given as in Theorem 4.8.

H belongs to the class of Hamiltonian matrices (see Section 4.2). When the eigenvalues are computed from the squared matrix H^2, employing a backward-stable method, the computational error can be approximated using first-order perturbation theory [184, 174, 30]. It is given as

$$|\lambda - \hat{\lambda}| \approx \epsilon \frac{\|H\|_2}{s(\lambda)} \min\left\{\frac{\|H\|_2}{\lambda}, \frac{1}{\sqrt{\epsilon}}\right\}, \tag{6.5}$$

where λ denotes an exact eigenvalue of H, $\hat{\lambda}$ the corresponding computed value, $s(\lambda)$ the condition number of the eigenvalue, and ϵ the machine precision. Unless λ is very large, the expression is dominated by $\frac{\sqrt{\epsilon}\|H\|_2}{s(\lambda)}$. Essentially, the number of significant digits of the eigenvalues is halved, compared to direct backward-stable methods. For example, applying the QR algorithm on the original matrix H would yield an approximate error of $\frac{\epsilon\|H\|_2}{s(\lambda)}$. It fails, however, to preserve and exploit the structure of the problem and is undesirable from a numerical as well as from a performance point of view. A remedy is given by making use of the singular value decomposition (SVD).

Given the Cholesky factorizations $L_1 L_1^{\mathsf{H}} = M_1$, $L_2 L_2^{\mathsf{H}} = M_2$ and the SVD $U\Lambda V^{\mathsf{H}} = L_1 L_2^{\mathsf{H}}$, the product eigenvalue problem can be rewritten in the form

$$M_1 M_2 V_1 = V_1 \Lambda^2 \Leftrightarrow L_1^{\mathsf{H}} L_2 (L_1^{\mathsf{H}} L_2)^{\mathsf{H}} \hat{V}_1 = \hat{V}_1 \Lambda^2, \tag{6.6}$$

$$V_2^{\mathsf{H}} M_1 M_2 = \Lambda^2 V_2^{\mathsf{H}} \Leftrightarrow \hat{V}_2^{\mathsf{H}} (L_1^{\mathsf{H}} L_2)^{\mathsf{H}} (L_1^{\mathsf{H}} L_2) = \Lambda^2 \hat{V}_2^{\mathsf{H}}, \tag{6.7}$$

where $\hat{V}_1 = L_1^{-1} V_1$, $\hat{V}_2 = L_2^{-1} V_2$. The diagonal matrix Λ contains the positive eigenvalues of the BSE matrix, i.e. the square roots of the eigenvalues of the matrix product $M_1 M_2$. The details of the eigenvector computation are given in the following theorem.

Theorem 6.3:
Let H, M_1, M_2 and Q be given as in Theorem 4.8, and property (6.2) hold, $L_1 L_1^{\mathsf{H}} = M_1$, $L_2 L_2^{\mathsf{H}} = M_2$ be Cholesky decompositions and $L_1^{\mathsf{H}} L_2 = U_{SVD} \Lambda V_{SVD}^{\mathsf{H}}$ be a singular value decomposition, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}_+^{n \times n}$. Then

$$V = Q \begin{bmatrix} L_1 U_{SVD} \Lambda^{-\frac{1}{2}} \\ L_2 V_{SVD} \Lambda^{-\frac{1}{2}} \end{bmatrix}$$

contains eigenvectors of H corresponding to Λ, i.e. $HV = V\Lambda$. ◊

Proof. We see from equivalences (6.6) and (6.7) that $V_1 = L_1 U_{SVD} \Lambda^{-\frac{1}{2}}$ and $V_2 = L_2 V_{SVD} \Lambda^{-\frac{1}{2}}$ are right and left eigenvectors of the matrix product $M_1 M_2$. It holds $V_1^{\mathsf{H}} V_2 = I$ and we can apply Theorem 4.8. We observe $M_2 V_1 = V_2 \Lambda$ and $M_1 V_2 = V_1 \Lambda$. So scaling factors are given as $\lambda_1 = \lambda_2 = 1$ for each λ. □

The main difference between the SVD-based algorithm and the other ones, from a numerical point of view, is that the eigenvalue matrix Λ is computed directly by the SVD and not as a square root of another diagonal matrix. The way real BSE matrices are treated in [159] is based on the same idea.

We can expect to see a higher accuracy in the eigenvalues than in the square root and the Cholesky approach, because the eigenvalues are computed directly, using a backward-stable method for the singular value decomposition. Perturbation theory [164] yields an approximate error of

$$|\lambda - \hat{\lambda}| \approx \epsilon \frac{\|H\|_2}{s(\lambda)}.$$

In the other approaches, a similar approximation only holds for the error of the squared eigenvalues λ^2, and translates in form of (6.5) to the non-squared ones.

6.2.2 Comparison

In recent years, various packages have been developed to facilitate the computation of the electronic structure of materials. See e.g. [86, 70, 156] or [171] for an overview. In particular, computing excited states via methods based on many-body perturbation theory has come into focus, as powerful computational resources become more widely available. Here, the Bethe-Salpeter approach constitutes a state-of-the art method for computing optical properties such as the optical absorption spectrum. To this end, an algorithm based on Theorem 6.1 is typically used to solve the resulting eigenvalue problem after the matrices A and B have been set up [155].

The main contribution of the previous section was to provide a unified frame of reference, which can be used to derive the currently used approach, based on Theorem 6.1, as well as two alternative ones, based on Theorems 6.2 and 6.3. Here, the similarities between the realizations of the different approaches become apparent. In all resulting algorithms we clearly see four steps.

1. *Preprocessing*: Setup a matrix M.
2. *Decomposition*: Compute spectral, respectively, singular value decomposition of M.
3. *Postprocessing*: Transform resulting vectors to (left and right) eigenvectors of matrix $(A + B)(A - B)$.
4. *Final setup*: Form eigenvectors of original BSE matrix.

A detailed compilation is given in Table 6.1. Seeing the algorithms side by side enables a direct comparison. The amount of flops (floating point operations) is based on estimates for sequential, non-blocked implementations [83], and lower order terms, i.e. $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$, are neglected. The preprocessing step is most expensive in the square root approach. Computing the square root of a Hermitian matrix involves the solution of a Hermitian eigenvalue problem. Additionally, the matrices S and M need to be set up, using 3 matrix-matrix products. This makes the preprocessing step even more expensive than the following "main" eigenvalue computation. The CHOL and the CHOL+SVD approach, on the other hand, only rely on one or two

Table 6.1: Algorithmic steps of the different methods. The number in parentheses estimates the number of flops (floating point operations), where lower-order terms are neglected.

	SQRT (Thm. 6.1)	CHOL (Thm. 6.2)	CHOL+SVD (Thm. 6.3)
1. Preprocessing	$S = (A-B)^{\frac{1}{2}}$, $(11n^3)$ $M = S(A+B)S$ $(4n^3)$	$LL^{\mathsf{H}} = A - B$, $(\frac{1}{3}n^3)$ $M = L(A+B)L^{\mathsf{H}}$ $(2n^3)$	$L_1 L_1^{\mathsf{H}} = A + B$, $(\frac{1}{3}n^3)$ $L_2 L_2^{\mathsf{H}} = A - B$, $(\frac{1}{3}n^3)$ $M = L_1^{\mathsf{H}} L_2$ (n^3)
2. Decomposition	$M = V_M \Lambda^2 V_M^{\mathsf{H}}$ $(9n^3)$	$M = V_M \Lambda^2 V_M^{\mathsf{H}}$ $(9n^3)$	$M = U_{SVD} \Lambda V_{SVD}^{\mathsf{H}}$ $(21n^3)$
3. Postprocessing	$V_1 := S^{-1} V_M \Lambda^{\frac{1}{2}}$, $(\frac{8}{3}n^3)$ $V_2 = S V_M \Lambda^{-\frac{1}{2}}$ $(2n^3)$	$V_1 = L^{-H} V_M \Lambda^{\frac{1}{2}}$, (n^3) $V_2 = L V_M \Lambda^{-\frac{1}{2}}$ (n^3)	$V_1 = L_1 U_{SVD} \Lambda^{-\frac{1}{2}}$, (n^3) $V_2 = L_2 V_{SVD} \Lambda^{-\frac{1}{2}}$ (n^3)
4. Final setup	$V = \begin{bmatrix} \frac{1}{2}(V_1 + V_2) \\ \frac{1}{2}(V_2 - V_1) \end{bmatrix}$.		

Cholesky factorizations and matrix multiplications, which are comparatively cheap to realize. The computational effort in the decomposition step is the highest in the CHOL+SVD step. The post-processing step again is most expensive in the SQRT approach, because the matrix S is a general square matrix, while the L matrices in CHOL and CHOL+SVD are triangular. In total, SQRT takes an estimated amount of $28\frac{2}{3}n^3$ flops, $CHOL$ takes $13\frac{1}{3}n^3$ flops and CHOL+SVD takes $24\frac{2}{3}n^3$ flops. The classical QR algorithm applied to the full, non-Hermitian matrix takes about $25(2n)^3 = 200n^3$ flops (not including the computation of eigenvectors from the Schur vectors).

An obvious solution approach for computing the eigenvalue decomposition of H_{BS} under condition (6.2) considers the equivalent Hermitian-definite eigenvalue problem

$$\begin{aligned} & H_{BS} x = \lambda x \\ \Leftrightarrow\ & K H_{BS} x = \lambda K x, \end{aligned}$$

defined by the matrix pencil (KH_{BS}, K). Using a Cholesky factorization of the positive definite matrix KH_{BS}, an equivalent Hermitian eigenvalue problem is formed (see e.g. [83, Algorithm 8.7.1]). Here, the available symmetry is exploited, but the method still acts on a large $2n \times 2n$ problem. It can be expected to perform $14(2n)^3 = 112n^3$ flops.

According to this metric, we expect the Cholesky and the SVD approach to perform faster than the square root approach. The actual performance of algorithms on current architectures is not simply determined by the number of operations performed, but by their parallelizability and communication costs. All presented approaches have a high computational intensity of $\mathcal{O}(n^3)$, such that the memory bandwidth is not likely to be a bottleneck. All methods rely on the same standard building blocks from numerical linear algebra, for which optimized versions (e.g. blocked variants for cache-efficiency) are available. This setting makes a fair comparison possible, where the arithmetic complexity has a high explanatory power.

To summarize, we expect CHOL to be about twice as fast as SQRT, while keeping the same accuracy. CHOL+SVD performs more computations than CHOL, and will take more time, but could improve the accuracy of the computations. It might be faster than SQRT, depending on how efficient the diagonalizations in SQRT and the SVD in CHOL+SVD are implemented.

The comparison in Table 6.1 is helpful when implementing the new approaches in codes that already use the square root approach. For the Cholesky approach, we need to substitute the computation of the matrix square root with the computation of a Cholesky factorization (LAPACK routine `POTRF`), compute the matrix M using triangular matrix multiplications (`TRMM`), and use a triangular solve (`TRSM`) and a triangular matrix multiplication (`TRMM`) in the post-processing step. For the CHOL+SVD approach, an additional Cholesky factorization is necessary and the Hermitian eigenvalue decomposition is substituted by a singular value decomposition (`zgesvd`). The post-processing involves two triangular matrix products instead of a matrix inversion and two general matrix products.

6.3 Numerical experiments

We implemented and compared serial versions of algorithms presented in Table 6.1 in MATLAB. They compute positive eigenvalues and associated eigenvectors of a BSE matrix $H \in \mathbb{C}^{2n\times 2n}$ of form II (6.1), which fulfills the definiteness property (6.2). The eigenvalues are given as a diagonal matrix $D \in \mathbb{R}^{n\times n}$. The eigenvectors $V \in \mathbb{C}^{2n\times n}$ are scaled such that K-orthogonality (guaranteed by equation (4.13) in Theorem 4.8) holds, i.e. $V^{\mathsf{H}}KV = I_n$. The K-orthogonality is an important property in the application. It is exploited in order to construct the polarizability operator ultimately used for the computation of the absorption spectrum.

We also include the MATLAB eigensolver `eig` for comparison. `eig` can either work on the BSE matrix H or solve the generalized eigenvalue problem for the matrix pencil (KH, K). In this formulation, both matrices are Hermitian and one is positive definite, which allows for a faster computation.

The experiments were performed on a laptop with an Intel(R) Core(TM) i7-8550U processor, running with 1.8 GHz on 4 cores, using MATLAB R2018a.

The first experiments aim to assess the accuracy of the computed eigenvalues. The matrices A and B are of size $n = 200$ and are created in the following way for a given

Table 6.2: Comparison of different methods for eigenvalue computation for a Bethe-Salpeter matrix of form II of size 400×400.

Method	Relative Error				Runtime
	$\kappa = 10$	$\kappa = 10^3$	$\kappa = 10^6$	$\kappa = 10^9$	
`eig`	1.28e-14	5.08e-14	3.82e-11	1.26e-08	62.7 ms
generalized `eig`	7.89e-15	6.67e-15	1.89e-11	1.97e-09	10.7 ms
`haeig`	4.73e-15	7.82e-15	4.32e-11	2.23e-08	50.9 ms
SQRT	5.45e-15	3.11e-12	4.64e-06	1.39e+00	5.87 ms
CHOL	4.23e-15	2.17e-12	1.32e-06	1.19e-05	3.09 ms
CHOL + SVD	1.23e-15	2.20e-14	2.53e-11	2.38e-09	4.28 ms

value $\kappa \in \mathbb{R}$. Let $d = \left[1, \ldots, \frac{1}{3}\kappa\right] \in \mathbb{R}^n$ be a vector with elements equally spaced between 1 and $\frac{1}{3}\kappa$. The BSE matrix is constructed as

$$H = \begin{bmatrix} A & B \\ -B & -A \end{bmatrix} := \begin{bmatrix} Q & 0 \\ 0 & Q \end{bmatrix}^{\mathsf{H}} \begin{bmatrix} \operatorname{diag}(d) & \frac{1}{2}\operatorname{diag}(d) \\ -\frac{1}{2}\operatorname{diag}(d) & -\operatorname{diag}(d) \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & Q \end{bmatrix},$$

where $Q \in \mathbb{C}^{n \times n}$ is a randomly generated, unitary matrix. It can be shown, that $\operatorname{cond}(H) = \kappa$ and the eigenvalues are given as $\frac{\sqrt{3}}{2}d$.

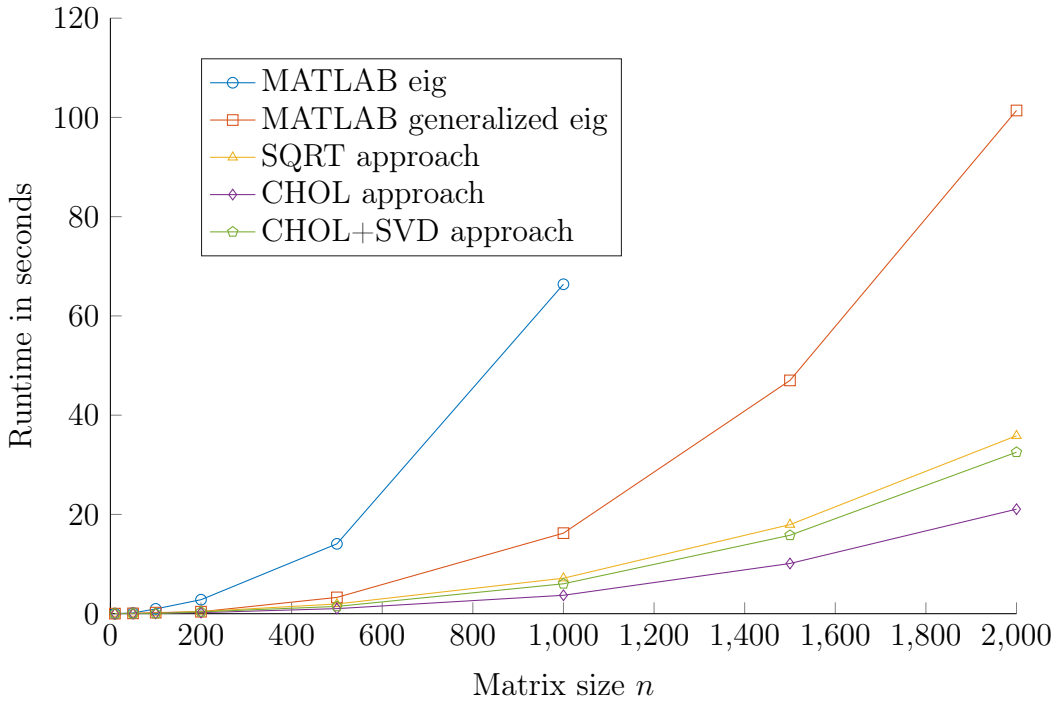
Table 6.2 shows the relative error in the smallest eigenvalue $\lambda = \frac{\sqrt{3}}{2}$, using the methods discussed in Section 6.2. We also include the routine `haeig` from the SLICOT package [40, 38]. Because `haeig` can only compute eigenvalues, not eigenvectors, we also only compute eigenvalues in the other methods in order to make the runtimes comparable.

The MATLAB `eig` function has the largest runtime. `haeig` is slightly faster, because it exploits the available Hamiltonian structure. However, the routine is not optimized for cache-reuse, which is why this effect can not be observed more clearly and vanishes for larger matrices. The generalized eigenvalue problem can be solved much faster, because it can be transformed to a Hermitian eigenvalue problem of size $2n \times 2n$. The other methods ultimately act on Hermitian matrices of size $n \times n$, which explains the much lower runtimes.

The observed eigenvalue errors comply with the error analysis given in Section 6.2.1. The currently used square root approach performs even worse than expected, yielding a completely wrong eigenvalue for matrices with a condition number $\kappa = 10^9$. In the application context, the small eigenvalues are of special interest. They correspond to bound exciton states, representing a strong electron-hole interaction. They are the reason why the Bethe-Salpeter approach is used instead of simpler schemes based on time-dependent density functional theory [152]. The smallest eigenvalues suffer the most from this numerical inaccuracy.

The second experiment aims to asses the runtime of the sequential implementations, including the eigenvector computation in the measurement. The matrices A and B are setup as random matrices, where the diagonal of A has been scaled up in order to

Figure 6.1: Runtimes for eigenvalue and eigenvector computation using different methods, $A, B \in \mathbb{C}^{n \times n}$ with varying matrix sizes.



guarantee the definiteness property (6.2). The measured runtimes are found in Figure 6.1 and serve as a rough indicator of computational effort.

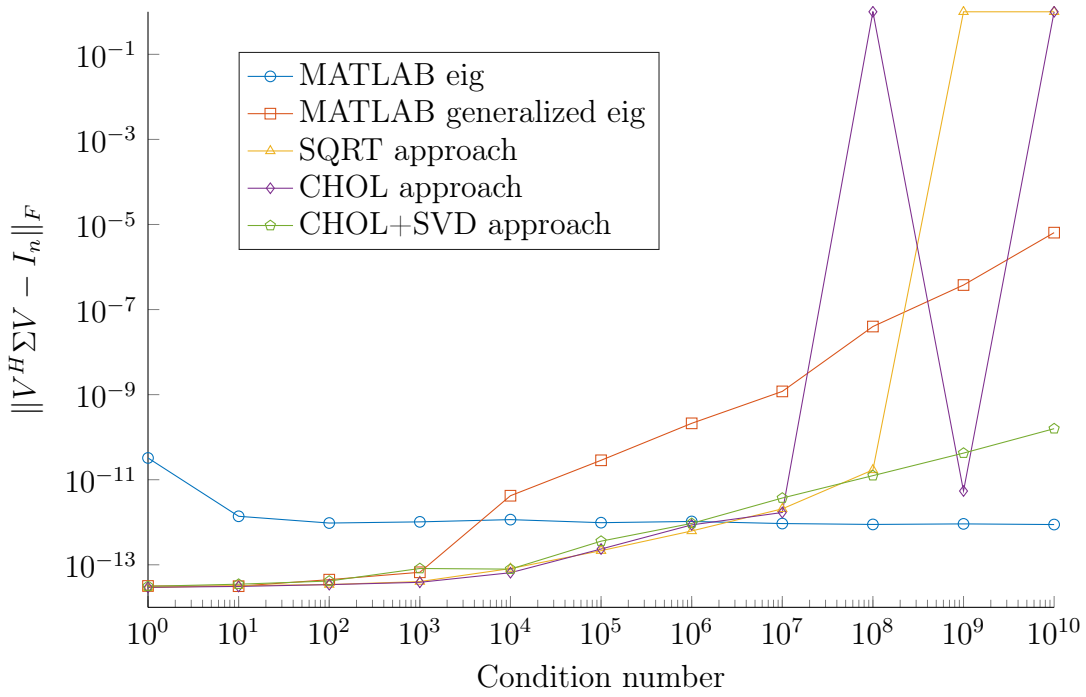
As expected, the Cholesky approach yields the fastest runtime of all approaches. The SVD approach also performs better than the square root approach. However, this picture could easily look different in another computational setup. An approach based on the `eig` command becomes prohibitively slow, when larger matrices are considered. Matrices in real applications become extremely large, up to dimensions of order 100 000, in order to get reasonable results. The effect would be even more drastic in a parallel setting, as the solution of a non symmetric dense eigenvalue problem is notoriously difficult to parallelize.

Figure 6.2 shows the achieved K-orthogonality of the eigenvector matrices for matrices with certain condition numbers. To this end, we manipulate the diagonal of the randomly generated matrix A such that badly conditioned BSE matrices H are generated. For the square root and the Cholesky approach, the K-orthogonality breaks down completely for badly conditioned matrices. This can have dramatic consequences and lead to completely wrong results, when further computations rely on this property.

To show the applicability to real life examples, we extracted a Bethe-Salpeter matrix corresponding to the excitation of Lithium-Fluoride from the `exciting` software package [86]. Computational details on how the matrix is generated can be found in the documentation[1]. Here, it is pointed out that a Tamm-Dancoff approximation,

[1] http://exciting-code.org/carbon-excited-states-from-bse

Figure 6.2: Deviation from K-orthogonality for different methods, $A, B \in \mathbb{C}^{200\times 200}$ with a certain condition number.



i.e. setting the off-diagonal block B to zero, already yields satisfactory results. The resulting 2560×2560 BSE matrix has a condition number (computed using `cond` in MATLAB) of 5.33. We do not expect the algorithms to suffer from the numerical difficulties observed in the first example.

The three smallest eigenvalues computed by different methods are found in Table 6.3. Indeed, all approaches coincide in the first 14 significant digits. The Tamm-Dancoff approximation (TDA) applies MATLAB `eig` on the diagonal Block A and provides eigenvalues, that are correct up to 4 significant digits, which is sufficient for practical applications. The measured runtimes reflect the results of the other experiments. Now the lack of low-level optimization in the `haeig` routine becomes apparent and leads to the lowest performance of all approaches.

6.4 Towards structure-preserving methods for non-definite problems

The algorithms presented in Section 6.2 assume property (6.2) to hold, i.e. KH is assumed to be positive definite. However, as pointed out in [22, 142], complex eigenvalues can occur in certain contexts and are related to finite lifetimes of particles. In this section, we point out how our framework can be used to gain insight in this case. In contrast to the definite case, the stable computation of required quantities can be

Table 6.3: Computed eigenvalues for the Lithium Fluoride example.

	λ_1	λ_2
`eig`	4.6423352497493209e-01	4.6524229149750918e-01
generalized `eig`	4.6423352497493126e-01	4.6524229149750407e-01
`haeig`	4.6423352497493725e-01	4.6524229149750940e-01
SQRT	4.6423352497493120e-01	4.6524229149750490e-01
CHOL	4.6423352497493031e-01	4.6524229149750573e-01
CHOL + SVD	4.6423352497493092e-01	4.6524229149750473e-01
TDA	4.6427305979874345e-01	4.6528180480128906e-01
	λ_3	Runtime
`eig`	4.6872644706731720e-01	32.49 s
generalized `eig`	4.6872644706732447e-01	10.62 s
`haeig`	4.6872644706732514e-01	71.43 s
SQRT	4.6872644706732541e-01	3.44 s
CHOL	4.6872644706732414e-01	2.06 s
CHOL + SVD	4.6872644706732453e-01	3.41 s
TDA	4.6877150201685513e-01	0.88 s

more challenging and routines are not as readily available. This section serves as a motivation for future algorithmic developments. This is why the numerical experiments in the previous section focus on the definite case.

In the definite case in Section 6.2, a Cholesky factorization was used to transform the eigenvalue problem. This role is now played by its generalized form, an LDL^{T} factorization. We use a scaled LDL^T decomposition to arrive at an eigenvalue problem of half the size. The resulting matrix is pseudo-Hermitian, i.e. Hermitian up to sign changes of certain rows or columns. The same idea is explored in [142]. Let $M_2 = LTL^H$ be a decomposition, where T is a signature matrix, i.e. $T = \operatorname{diag}(t_1, \ldots, t_n)$, $t_j \in \{1, -1\}$ for $j = 1, \ldots, n$ (see Lemma 2.17). Then we see

$$M_1 M_2 V_1 = V_1 D \quad \Leftrightarrow \quad L^{\mathsf{H}} M_1 L T \hat{V}_1 = \hat{V}_1 D, \tag{6.8}$$

where $\hat{V}_1 = L^H V_1$ contains the right eigenvectors of the pseudo-Hermitian matrix $L^{\mathsf{H}} M_1 L T$. Similarly, the left eigenvectors of $M_1 M_2$ show the relation

$$V_2^{\mathsf{H}} M_1 M_2 = D V_2^{\mathsf{H}} \quad \Leftrightarrow \quad \hat{V}_2^{\mathsf{H}} L^{\mathsf{H}} M_1 L T = D \hat{V}_2^{\mathsf{H}},$$

where $\hat{V}_2 = L^{-1} V_2$ contains the left eigenvectors of the pseudo-Hermitian matrix $L^{\mathsf{H}} M_1 L T$.

For Hermitian matrices occurring in the definite case, left and right eigenvectors are the same. For pseudo-Hermitian matrices this does not hold, but they are nevertheless related. This is pointed out in the following theorem stating a normal form for pseudo-Hermitian matrices, compare [122].

Theorem 6.4:
Let $M \in \mathbb{C}^{n\times n}$ be diagonalizable with k real positive, l real negative eigenvalues, and $2m$ complex eigenvalues and be selfadjoint with respect to a signature matrix Σ, i.e. $M\Sigma = (M\Sigma)^{\mathsf{H}}$, and define $S_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Then there exists an eigenvalue decomposition

$$V^{-1}MV = D := D_+ \oplus D_- \oplus D_{\text{complex}},$$
$$V^{\mathsf{H}}\Sigma V = S := \operatorname{diag}(\sigma_1, \ldots, \sigma_{k+l}, S_2, \ldots, S_2)\,.$$

The positive real eigenvalues of M are contained in

$$D_+ = \operatorname{diag}(\lambda_1, \ldots, \lambda_k)$$

and the negative real eigenvalues of M are contained in

$$D_- = \operatorname{diag}(\lambda_{k+1}, \ldots, \lambda_{k+l})\,.$$

The $2m$ complex eigenvalues of M are contained in

$$D_{\text{complex}} = \operatorname{diag}\big(\lambda_{k+l+1}, \overline{\lambda_{k+l+1}}, \ldots, \lambda_{k+l+m}, \overline{\lambda_{k+l+m}}\big)\,.$$

The ordered list $(\sigma_1, \ldots, \sigma_{k+l})$ contains signs ± 1. ◊

Given the decomposition from Theorem 6.4, the left eigenvectors of M are given as the columns of

$$V_2 = V^{-\mathsf{H}} = \Sigma V S.$$

Structure-preserving algorithms such as the HR algorithm [33, 182, 56, 57] can be employed to compute a structured eigenvalue decomposition. This idea is described for product eigenvalues in [49]. A well-known downside of this approach is that it is not guaranteed to be stable. A new class of algorithms for solving pseudo-Hermitian eigenvalue problems is presented in Chapter 9.

Given a structured decomposition, the following theorem explains how to use it to construct eigenvalues of H.

Theorem 6.5:
Let H be a BSE matrix of form II, with $M_1 := A + B$ and $M_2 = A - B$ nonsingular. Moreover, let $M_2 = LTL^{\mathsf{H}}$ be a decomposition, where T is a signature matrix, and assume

$$\hat{V}^{-1}M\hat{V} = D$$

to be the structured eigenvalue decomposition of $M = L^{\mathsf{H}}M_1LT$ given in Theorem 6.4, i.e.

$$\hat{V}^{-1} = S\hat{V}^{\mathsf{H}}T, \quad S = \operatorname{diag}(\sigma_1, \ldots, \sigma_{k+l}, S_2, \ldots, S_2)\,.$$

Then with $\hat{T} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{k+l}, I_2, \ldots, I_2)$ and $Q := \frac{1}{2}\begin{bmatrix} I & I \\ -I & I \end{bmatrix}$, the columns of the matrix

$$V = Q \begin{bmatrix} L^{-\mathsf{H}} \hat{V} D^{1/4} \\ L T \hat{V} \hat{T} D^{-1/4} \end{bmatrix},$$

provide eigenvectors of H fulfilling $HV = V\hat{\Lambda}$, where $\hat{\Lambda} = D^{\frac{1}{2}}\hat{T}$. The complete set of eigenvectors is given by

$$V_{\text{full}} = \begin{bmatrix} V & -\mathrm{i}JKV \end{bmatrix}, \tag{6.9}$$

which fulfills

$$V_{\text{full}}^{\mathsf{H}} K V_{\text{full}} = \begin{bmatrix} I_k & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_l & 0 \\ 0 & 0 & S_{2m} & 0 & 0 & 0 \\ 0 & 0 & 0 & -I_k & 0 & 0 \\ 0 & I_l & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -S_{2m} \end{bmatrix}, \quad S_{2m} = \mathrm{diag}(S_2, \ldots, S_2) \in \mathbb{R}^{2m \times 2m}. \tag{6.10}$$

♢

Proof. We see in equivalence (6.8) that $L^{-\mathsf{H}}\hat{V} =: V_1$ contains right eigenvectors of the matrix product $M_1 M_2$. Similarly, it can be seen that $L\hat{V}^{-\mathsf{H}} = LT\hat{V}S =: V_2$ contains left eigenvectors of $M_1 M_2$. We aim to apply Theorems 4.8 and 4.9 to compute the required scaling factors for V_1 and V_2. We observe that

$$M_1 V_2 = V_1 D S, \quad M_2 V_1 = V_2 S. \tag{6.11}$$

The eigenvector matrices $V_{1,\text{ri}} := V_1(:, 1 : k + l)$ and $V_{2,\text{ri}} := V_2(:, 1 : k + l)$ correspond to the real eigenvalues of M, and will turn out to correspond to real and imaginary eigenvalues of H. For the leading parts of S and T, we have

$$S_{\text{ri}} := S(1 : k + l, 1 : k + l) = \mathrm{diag}(\sigma_1, \ldots, \sigma_{k+l}) = \hat{T}_{\text{ri}} := \hat{T}(1 : k + l, 1 : k + l).$$

It follows with $D_{\text{ri}} = D(1 : k + l, 1 : k + l)$ that

$$M_1 V_{2,\text{ri}} = V_{1,\text{ri}} D_{\text{ri}} \hat{T}_{\text{ri}}, \quad M_2 V_{1,\text{ri}} = V_{2,\text{ri}} \hat{T}_{\text{ri}}.$$

According to Theorem 4.8, the scaling factors λ_1 and λ_2 are given by the diagonal elements of $D_{\text{ri}}\hat{T}_{\text{ri}}$ and $\hat{T}_{\text{ri}}$. Let μ be an eigenvalue of $M_1 M_2$ given as a diagonal element of D_{ri}. We consider the four possible cases

1. $\mu > 0$
   a) $\lambda_1 > 0$ and $\lambda_2 > 0 \Leftrightarrow$ Corresponding diagonal value in $\hat{T}_{\text{ri}}$ is $+1$,

b) $\lambda_1 < 0$ and $\lambda_2 < 0 \Leftrightarrow$ Corresponding diagonal value in $\hat{T}_{\text{ri}}$ is -1,

2. $\mu < 0$

a) $\lambda_1 < 0$ and $\lambda_2 > 0 \Leftrightarrow$ Corresponding diagonal value in $\hat{T}_{\text{ri}}$ is $+1$,

b) $\lambda_1 > 0$ and $\lambda_2 < 0 \Leftrightarrow$ Corresponding diagonal value in $\hat{T}_{\text{ri}}$ is -1.

In cases 1 (a) and 2 (a), the constructed vector corresponds to $\lambda = \mu^{\frac{1}{2}}$ with no sign switch occurring according to Theorem 4.8. In case 1 (b), a sign switch occurs according to Theorem 4.8. In these three cases the corresponding scaling factor is given as $\lambda_1^{\frac{1}{4}}\lambda_2^{-\frac{1}{4}} = \mu^{\frac{1}{4}}$. In case 2 (b), no sign switch occurs according to Theorem 4.8, but the proper scaling factor is given as $\lambda_1^{\frac{1}{4}}\lambda_2^{-\frac{1}{4}} = \overline{\mu^{\frac{1}{4}}} = -\mathrm{i}\mu^{\frac{1}{4}}$. If instead a scaling factor of $\mu^{\frac{1}{4}}$ is chosen, this yields the eigenvector corresponding to $-\mu^{\frac{1}{2}}$ as we see in the following. Let $v = Q\begin{bmatrix} v_1 s_1 \\ v_2 s_1^{-1} \end{bmatrix}$ be the properly constructed eigenvector H corresponding to μ with the scaling factor $s_1 = -\mathrm{i}\mu^{\frac{1}{4}}$. According to Theorem 4.4, 2. (a), the eigenvector corresponding to $-\mu^{\frac{1}{2}}$ is given by

$$JKv = Q\begin{bmatrix} v_1 s_1 \\ -v_2 s_1^{-1} \end{bmatrix} = -iQ\begin{bmatrix} v_1 \mu^{\frac{1}{4}} \\ v_2 \mu^{-\frac{1}{4}} \end{bmatrix}, \tag{6.12}$$

where we used $JKQ = QK$. Scaling this eigenvector with i gives the vector, that is constructed using $\mu^{\frac{1}{4}}$ as a scaling factor, as is proposed here. In summary, the proposed construction leads to a sign change whenever $\hat{T}$ has negative values and we have

$$HV_{\text{ri}} = V_{\text{ri}} D_{\text{ri}}^{\frac{1}{2}} \hat{T}_{ri}, \qquad V_{\text{ri}} = Q\begin{bmatrix} L^{-\mathsf{H}} \hat{V}_{\text{ri}} D_{\text{ri}}^{\frac{1}{4}} \\ LT\hat{V}_{\text{ri}} \hat{T}_{\text{ri}} D_{\text{ri}}^{-\frac{1}{4}} \end{bmatrix}. \tag{6.13}$$

The scaling matrices for the remaining vectors $V_{\text{c}} = V(:, k+l+1:n)$ associated with complex eigenvalues are constructed via Theorem 4.9. From (6.11) we see that the scaling factors for a given μ are given as $\lambda_1 = \mu$ and $\lambda_2 = 1$, resulting in $\hat{\lambda} = \mu^{\frac{1}{4}}$ and the scaling matrices

$$\hat{\Lambda}_1 = \begin{bmatrix} \mu^{\frac{1}{4}} & 0 \\ 0 & \overline{\mu^{\frac{1}{4}}} \end{bmatrix}, \qquad \hat{\Lambda}_2 = \begin{bmatrix} 0 & \overline{\mu^{-\frac{1}{4}}} \\ \mu^{-\frac{1}{4}} & 0 \end{bmatrix} = S_2 \hat{\Lambda}_1^{-1}.$$

The submatrices D_{c}, $\hat{V}_c$ $V_{1,\text{c}}$ and $V_{2,\text{c}}$ denote the trailing diagonal matrix or the trailing columns of the respective matrix, corresponding to the $2m$ complex eigenvalues of H with non-vanishing real and imaginary part. The scaling matrices for $V_{1,\text{c}}$ and $V_{2,\text{c}}$ are given as $\hat{\Lambda}_{1,\text{c}} = D_{\text{c}}^{\frac{1}{4}}$, $\hat{\Lambda}_{2,\text{c}} = S_{2m}\hat{\Lambda}_{1,\text{c}}^{-1}$. According to (4.18), no sign switch occurs with the given λ_1 and λ_2 and we have

$$HV_{\text{c}} = V_{\text{c}} D_{\text{c}}^{\frac{1}{2}}, \quad V_{\text{c}} = Q\begin{bmatrix} V_{1,\text{c}}\hat{\Lambda}_{1,\text{c}} \\ V_{2,\text{c}}\hat{\Lambda}_{2,\text{c}} \end{bmatrix} = Q\begin{bmatrix} L^{-\mathsf{H}}\hat{V}_{\text{c}} D_{\text{c}}^{\frac{1}{4}} \\ LT\hat{V}_{\text{c}} D_{\text{c}}^{-\frac{1}{4}} \end{bmatrix}. \tag{6.14}$$

Equations (6.13) and (6.14) together give the first part of the proposed theorem.

V_{full} provides a full set of eigenvectors according to Lemma 4.4, 2. (a). It remains to show the K-orthogonality condition (6.10). The upper left quadrant of $V_{\text{full}}^{\mathsf{H}} K V_{\text{full}}$ is given as proposed according to Theorems 4.8 and 4.9. The lower right quadrant is evaluated as $(-iJKV)^{\mathsf{H}} K(-iJKV) = -V^{\mathsf{H}} KV$. The remaining quadrants are determined by the entries $v_\lambda^{\mathsf{H}} K(-iJKv_\gamma) = \mathrm{i} v_\lambda^{\mathsf{H}} J v_\gamma$, where v_λ and v_γ refer to vectors given as columns of V. This term evaluates as 0 when $\lambda \neq \gamma$ or when the corresponding eigenvalue is not purely imaginary. For imaginary eigenvalues, we distinguish cases 2 (a) and 2 (b) defined above. In case 2 (a) (where $\lambda_1 < 0$ and $\lambda_2 > 0$), we have $\mathrm{i} v_\lambda^{\mathsf{H}} J v_\gamma = \mathrm{i}(-\mathrm{i}) = 1$ according to (4.13). In case 2 (b) (where $\lambda_1 > 0$ and $\lambda_2 < 0$), we used the scaling factor $\mu^{\frac{1}{4}}$ instead of $-\mathrm{i}\mu^{\frac{1}{4}}$ proposed by Theorem 4.8. Equation (6.12) states that the original vector, for which Theorem 4.8 holds, is given by $-\mathrm{i}KJv_\lambda$. So the J-orthogonality condition holds for $-\mathrm{i}KJv_\lambda$, i.e. in this case we have $(-\mathrm{i}KJv_\lambda)^{\mathsf{H}} J(-\mathrm{i}KJv_\lambda) = -v_\lambda^{\mathsf{H}} J v_\lambda = \mathrm{i}$. The quadrant entry then evaluates as $\mathrm{i} v_\lambda^{\mathsf{H}} J v_\gamma = 1$. □

The correct tool for generalizing the SVD-based approach in Theorem 6.3 is a generalized SVD (GSVD) given in form of Theorem 4.1 in [124]. The following version concerns the special case of diagonalizable matrices and gives a generalized SVD with respect to two signature matrices T_1 and T_2.

Theorem 6.6:
Let $A \in \mathbb{C}^{n\times n}$ be nonsingular and diagonalizable with k real positive, l real negative eigenvalues, and $2m$ complex eigenvalues and let T_1, $T_2 \in \mathbb{R}^{n\times n}$ be two signature matrices. Then there exist nonsingular matrices $U, V \in \mathbb{C}^{n\times n}$, such that

$$\begin{aligned} U^{\mathsf{H}} AV = \Lambda_r \oplus \Lambda_i \oplus \Lambda_c, \quad & U^{\mathsf{H}} T_1 U = \hat{T}_1 = \operatorname{diag}(s_1, \ldots, s_{k+l}, S_2, \ldots S_2)\,, \\ & V^{\mathsf{H}} T_2 V = \hat{T}_2 = \operatorname{diag}(\hat{s}_1, \ldots, \hat{s}_{k+l}, S_2, \ldots S_2)\,, \end{aligned} \tag{6.15}$$

where $\Lambda_r = \operatorname{diag}(\lambda_1, \ldots, \lambda_k)$, $\Lambda_i = \operatorname{diag}(\lambda_{k+1}, \ldots, \lambda_{k+l})$ are diagonal matrices containing positive real values,

$$\Lambda_c = \operatorname{diag}\big(\lambda_{k+l+1}, \overline{\lambda}_{k+l+1}, \ldots, \lambda_{k+l+m}, \overline{\lambda}_{k+l+m}\big)$$

is a diagonal matrix containing complex conjugate pairs of complex values, where $\arg \lambda_j \in (0, \pi/2)$ for $j = k + l + 1, , \ldots, k + l + m$. The diagonal values of $\hat{T}_2$ are signs, $s_j, \hat{s}_j \in \{1, -1\}$. They fulfill $s_j \hat{s}_j = 1$ for $j = 1, \ldots, k$ and $s_j \hat{s}_j = -1$ for $j = k{+}1, \ldots, k{+}l$. For the matrices $M = T_1 A T_2 A^{\mathsf{H}}$, $\hat{M} = T_2 A^{\mathsf{H}} T_1 A$, the decomposition presented in Theorem 6.4 is given as

$$U^{-1} M U = \Lambda_r^2 \oplus -\Lambda_i^2 \oplus (\Lambda_c^2)^{\mathsf{H}}, \quad V^{-1} \hat{M} V = \Lambda_r^2 \oplus -\Lambda_i^2 \oplus \Lambda_c^2. \qquad \lozenge$$

Using scaled LDL^{T} decompositions $M_1 = L_1 T_1 L_1^{\mathsf{H}}$, $M_2 = L_2 T_2 L_2^{\mathsf{H}}$, where T_1 ans T_2 are signature matrices, the product eigenvalue problem considered in Theorem 4.8 and Theorem 4.9 is rewritten as

$$M_1 M_2 V_1 = V_1 D \quad \Leftrightarrow \quad T_1 L_1^{\mathsf{H}} L_2 T_2 (L_1^{\mathsf{H}} L_2)^{\mathsf{H}} \hat{V}_1 = \hat{V}_1 D, \tag{6.16}$$

$$V_2^{\mathsf{H}} M_1 M_2 = D V_2^{\mathsf{H}} \quad \Leftrightarrow \quad \hat{V}_2^{\mathsf{H}} (L_1^{\mathsf{H}} L_2)^{\mathsf{H}} T_1 (L_1^{\mathsf{H}} L_2) T_2 = D \hat{V}_2^{\mathsf{H}}, \tag{6.17}$$

where $\hat{V}_1 = L_1^{-1}V_1$ and $\hat{V}_2 = L_2^{-1}V_2$.

Here, the connection to the GSVD (6.15) in Theorem 6.6 becomes apparent by setting $A = L_1^{\mathsf{H}}L_2$. The following theorem explores this idea and gives guidance for computing eigenpairs of the BSE matrix via the GSVD.

Theorem 6.7:
Let $M_1 = L_1T_1L_1^{\mathsf{H}}$, $M_2 = L_2T_2L_2^{\mathsf{H}}$ be decompositions, where T_1, T_2 are signature matrices and

$$\hat{U}^{\mathsf{H}}M\hat{V} = \Lambda = \Lambda_r \oplus \Lambda_i \oplus \Lambda_c$$

be the GSVD of $M = L_1^{\mathsf{H}}L_2$ with respect to T_1 and T_2 given in Theorem 6.6, i.e.

$$\begin{aligned}\hat{U}^{-1} &= \hat{T}_1\hat{U}^{\mathsf{H}}T_1, \quad \hat{T}_1 = \operatorname{diag}(s_1,\dots,s_{k+l},S_2,\dots,S_2)\,,\\ \hat{V}^{-1} &= \hat{T}_2\hat{V}^{\mathsf{H}}T_2, \quad \hat{T}_2 = \operatorname{diag}(\hat{s}_1,\dots,\hat{s}_{k+l},S_2,\dots,S_2)\,.\end{aligned}$$

Then

$$V = Q\begin{bmatrix}L_1\hat{U}(\Lambda^{-1/2})^{\mathsf{H}}\\ L_2\hat{V}\Lambda^{-1/2}P\end{bmatrix}, \quad P = I_k \oplus iI_l \oplus S_2 \oplus \cdots \oplus S_2$$

yields eigenvectors of H fulfilling $HV = V\hat{\Lambda}$, where

$$\hat{\Lambda} = \Lambda\hat{T}, \qquad \hat{T} = \operatorname{diag}(s_1,\dots s_{k+l},I_{2m})\,.$$

The complete set of eigenvectors is given by

$$V_{\text{full}} = \begin{bmatrix}V & \mathrm{i}JKV\end{bmatrix},$$

which satisfies

$$V_{\text{full}}^{\mathsf{H}}KV_{\text{full}} = \begin{bmatrix}I_k & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & I_l & 0\\ 0 & 0 & S_{2m} & 0 & 0 & 0\\ 0 & 0 & 0 & -I_k & 0 & 0\\ 0 & I_l & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & -S_{2m}\end{bmatrix}, \quad S_{2m} = \operatorname{diag}(S_2,\dots,S_2) \in \mathbb{R}^{2m\times 2m}. \tag{6.18}$$

◊

Proof. We use the GSVD of $M = L_1^{\mathsf{H}}L_2$ from Theorem 6.6, $\hat{U}^{\mathsf{H}}M\hat{V} = \Lambda$. Then it holds

$$T_1MT_2M^{\mathsf{H}} = \hat{U}(\Lambda^2)^{\mathsf{H}}S\hat{U}^{-1}, \qquad S = I_k \oplus -I_k \oplus I_{2m},$$

and the equations (6.16) are equivalent to

$$\hat{U}(\Lambda^2)^{\mathsf{H}}S\hat{U}^{-1}\hat{V}_1 = \hat{V}_1D, \qquad \hat{V}_1 = L_1^{-1}V_1.$$

This equation holds for $D = (\Lambda^2)^{\mathsf{H}} S$ and $V_1 = L_1 \hat{U} (\Lambda^{-\frac{1}{2}})^{\mathsf{H}}$. Similarly, it can be seen that (6.17) is solved by $V_2 = L_2 \hat{V} \Lambda^{-\frac{1}{2}}$ and $D = (\Lambda^2)^{\mathsf{H}} S$. $(\Lambda^{-\frac{1}{2}})^{\mathsf{H}}$ and $\Lambda^{\frac{1}{2}}$ are diagonal matrices scaling the eigenvectors and have been chosen such that it holds $V_1^{\mathsf{H}} V_2 = I_n$. Now we are in a position to apply Theorems 4.8 and 4.9. With some simple algebra we see that the scaling factors are given as (block-) diagonal elements of

$$\Lambda_1 = V_2^{\mathsf{H}} M_1 V_2 = \hat{T}_1 \Lambda, \qquad \Lambda_2 = V_1^{\mathsf{H}} M_2 V_1 = \Lambda \hat{T}_2.$$

The scaling factor corresponding to a real diagonal value λ_j, $j = 1, \ldots, k+l$, of Λ can be evaluated as

$$\lambda_{1,j}^{\frac{1}{4}} \lambda_{2,j}^{-\frac{1}{4}} = (\lambda_j s_j)^{\frac{1}{4}} (\lambda_j \hat{s}_j)^{-\frac{1}{4}} = \begin{cases} 1 & \text{if } s_j \hat{s}_j = 1, \\ \mathrm{i}^{-\frac{1}{2}} = \sqrt{2}^{-1} - \sqrt{2}^{-1}\mathrm{i} & \text{if } s_j = 1 \text{ and } \hat{s}_j = -1, \\ \mathrm{i}^{\frac{1}{2}} = \sqrt{2}^{-1} + \sqrt{2}^{-1}\mathrm{i} & \text{if } s_j = -1 \text{ and } \hat{s}_j = 1. \end{cases} \tag{6.19}$$

The 2×2 scaling blocks for complex eigenvalues evaluate as the identity matrix. Therefore, a non-trivial scaling is only needed for eigenvectors corresponding to imaginary eigenvalues ($s_j \hat{s}_j = -1$).

Equation (6.19) suggests for the case where $s_j = 1$ and $\hat{s}_j = -1$ to use the scaling factors $\mathrm{i}^{-\frac{1}{2}}$ for the corresponding vectors in V_1 and $\mathrm{i}^{\frac{1}{2}}$ in V_2. We further multiply all vectors with $\mathrm{i}^{\frac{1}{2}}$, such that only V_2 is scaled by i. This does not change the fact that they are normalized eigenvectors, but simplifies notation.

In the case $s_j = -1$, $\hat{s}_j = 1$ the wrong scaling factor has been chosen. This is the same situation as in the proof of Theorem 6.5. Similarly, it can be shown that using the "wrong" scaling factor leads to the computation of an eigenvector corresponding to an eigenvalue of switched sign. In the case $s_j = -1$, $\hat{s}_j = -1$, a sign switch occurs according to Theorem 4.8. No sign switch occurs for the complex eigenvalues according to (4.18), because we have $\lambda_1 = \lambda_2$ and $\arg \lambda_1 \in [0, \pi/2]$ according to the definition of the GSVD in Theorem 6.6. So in total, a sign switch occurs whenever $s_j = -1$. This is encoded in $\hat{\Lambda} = \Lambda \hat{T}$, where $\hat{T} = \mathrm{diag}(s_1, \ldots, s_{k+l}, I_{2m})$. The normalization condition (6.18) is shown similarly to the proof of Theorem 6.5. □

Table 6.4 summarizes and compares the two algorithmic approaches following from Theorems 6.5 and 6.7. Unfortunately, stable algorithms computing the necessary structured decompositions, in particular Theorem 6.6, are hard to find. Because of the missing established algorithmic pathways, we do not provide flop counts as in Table 6.1.

A method for computing the structured decomposition given in Theorem 6.4, needed for the LDL^{T} approach, is presented in Chapter 9.

6.5 Conclusions and discussion

We presented a unifying framework for solving the Bethe-Salpeter eigenvalue problem as it appears in the computation of optical properties of crystalline systems. Two presented methods are superior to the one often used in current implementations, which is

Table 6.4: Algorithmic steps of the different methods applicable in the non-definite setting, i.e. the definiteness property (6.2) does not necessarily hold.

	LDL^{T}	LDL^{T} + GSVD
1. Preprocessing	$LTL^{\mathsf{H}} = A - B$, $M = L(A+B)L^{\mathsf{H}}T$	$L_1T_1L_1^{\mathsf{H}} = A + B$, $L_2T_2L_2^{\mathsf{H}} = 0, A - B$, $M = L_1^{\mathsf{H}}L_2$
2. Decomposition	$M = \hat{V}D\hat{V}^{-1}$, where $\hat{V}^{\mathsf{H}}T\hat{V} = S$, $\Lambda = D^{\frac{1}{2}}$	$M = U_{GSVD}^{-\mathsf{H}}\Lambda V_{GSVD}^{-1}$, where $U_{GSVD}^{\mathsf{H}}T_1U_{GSVD} = \hat{T}_1$, $V_{GSVD}^{\mathsf{H}}T_2V_{GSVD} = \hat{T}_2$
3. Postprocessing	$V_1 = L^{-H}V_M\Lambda^{\frac{1}{2}}$, $V_2 = LTV_M\hat{T}\Lambda^{-\frac{1}{2}}$	$V_1 = L_1U_{GSVD}(\Lambda^{-\frac{1}{2}})^{\mathsf{H}}$, $V_2 = L_2V_{GSVD}\Lambda^{-\frac{1}{2}}P$, where $P = I_k \oplus iI_k \oplus S_{2m}$
4. Sign switches	$\hat{T}$ is the diagonal matrix containing the signature of S.	$\hat{T}$ is the diagonal matrix containing the signature of $\hat{T}_1$.
4. Final setup	$V = \begin{bmatrix} \frac{1}{2}(V_1+V_2) \\ \frac{1}{2}(V_2-V_1) \end{bmatrix}, \quad \hat{\Lambda} = \Lambda\hat{T}$	

based on the computation of a matrix square root. Computing the matrix square root constitutes a high computational effort for non-diagonal matrices. Our first proposed method substitutes the matrix square root with a Cholesky factorization which can be computed much easier. The total runtime is reduced by about 40% in preliminary experiments, while the same accuracy is achieved. In order to achieve a higher accuracy we proposed a second method, which also relies on Cholesky factorizations and uses a singular value decomposition instead of an eigenvalue decomposition.

We gave new theoretical results on structured matrices, which served as a foundation of the proposed algorithms. These results were further extended to be applicable in a setting where non-definite matrices and complex eigenvalues occur. The previously presented methods found direct analogies in this setting. Most methods available in the literature on the other hand rely on the definiteness property (6.2).

In the definite setting, many available methods, such as the iterative ones presented in [22, 158], are implicitly connected to the presented product eigenvalue structure given in Theorems 4.8 and 4.9. The contribution of this chapter in this regard is to clarify the common basis of methods that seem unconnected at first glimpse. The treatment of the non-definite case in Theorem 4.9 is completely new and can serve as a basis for further algorithm development. We only presented direct methods based on structured decompositions. Future research can explore the possibility of an iterative approach solving the indefinite product eigenvalue problem in Theorem 4.9.

A candidate might be based on an indefinite Lanczos method [143, 68, 62].

Iterative methods have benefits when only a fraction of the eigenpairs are sought. Typically, the eigenpairs corresponding to a few smallest eigenvalues already provide a lot of information about the considered system. It is also possible to reduce the computational costs of direct methods when only few eigenpairs are sought. For example, the needed decompositions can partially be computed in a divide-and-conquer fashion [133]. On an implementational level it is also possible to save some operations in the computation of symmetric eigenvalue decompositions in particular in the context of high performance computing [15].

CHAPTER 7

GR DECOMPOSITIONS AND THEIR RELATIONS TO CHOLESKY-LIKE FACTORIZATIONS

Contents

7.1 Introduction

We have seen in Chapter 4 that the Bethe-Salpeter matrices are self-adjoint with respect to two non-standard scalar products. Both of them share the property to be self-adjoint with respect to the sesquilinear form induced by

$$K_n = \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix},$$

i.e. they are pseudo-Hermitian.

The remaining three chapters of this thesis are directed towards developing new algorithms for solving eigenvalue problems for this class of matrices.

In this chapter, we examine generalized variants of the QR decomposition and how they are related to generalized variants of the Cholesky factorization. This connection is used to improve the accuracy of the computation of the QR-like decompositions (also called GR decompositions). Essentially, two rounds of reorthogonalization with respect to non-standard scalar products are performed.

The QR-like decomposition with respect to a signature matrix such as K_n is called hyperbolic (or indefinite) QR decomposition and will serve as a tool in the computation of the generalized polar decomposition in Chapter 8. The generalized polar decomposition in turn will serve as a tool to devise a new structure-preserving spectral divide-and-conquer method for pseudo-Hermitian eigenvalue problems in Chapter 9.

For a given full-rank matrix $A \in \mathbb{K}^{m\times n}$, where $\mathbb{K} = \mathbb{C}$ or $\mathbb{K} = \mathbb{R}$, $m \geq n$, we are interested in computing GR decompositions

$$A = GR, \quad G \in \mathbb{K}^{m\times n},\ R \in \mathbb{K}^{n\times n},$$

where G is an isometry with respect to given scalar products induced by matrices $M \in \mathbb{K}^{m\times m}$ and $N \in \mathbb{K}^{n\times n}$, see Definition 2.6.

We first examine GR decompositions with respect to $M = K_m$, $N = K_n$, yielding the hyperbolic QR decomposition in Section 7.2. In Section 7.3, $M = J_m$ and $N = J_n$ are considered, where

$$J_n = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}.$$

The resulting decomposition is called the symplectic QR decomposition.

The standard QR decomposition given in Theorem 2.11 is a representative of a GR decomposition defined by the Euclidian scalar product, i.e. $M = I_m$, $N = I_n$. Here, R is upper triangular. With respect to identity matrices, an isometry is a matrix with orthonormal columns. Typically, the QR decomposition is computed in a stable fashion by successively eliminating subdiagonal entries of the matrix using orthogonal transformations. This is called the Householder QR method in the following. Instead of Householder transformations, Givens rotations can be used, but they are less suited for a cache-aware, blocked implementation.

There is a well known connection to the Cholesky factorization. Let A have full column rank. It holds that $A = QR$ is a thin QR decomposition if and only if R defines a Cholesky decomposition $R^{\mathsf{H}}R = A^{\mathsf{H}}A$. Computing $Q := AR^{-1}$ provides an alternative to the column elimination approach. For tall and skinny matrices, this method takes half as many operations as the Householder QR method. However, it is known to be unstable.

The stability can be drastically improved by doing a second repetition, i.e. compute the QR decomposition of Q [187]. Implementing a shifting strategy further improves the method for ill-conditioned matrices [80]. If A is reasonably conditioned, more

specifically $\text{cond}(A) < \mathcal{O}(u^{-\frac{1}{2}})$, then $\|\hat{Q}^*\hat{Q} - I_n\|_{\text{F}} = \mathcal{O}(u)$ and $\|\hat{Q}\hat{R} - A\|_{\text{F}} = \mathcal{O}(u)$ can be achieved. This algorithm is called CholeskyQR2. The detailed procedure is given as

1. Compute Gram matrix $X_1 := A^*A$.
2. Compute Cholesky factor R_1, such that $R_1^*R_1 = X_1$.
3. $Q_1 := AR_1^{-1}$.
4. Compute Gram matrix $X_2 := Q_1^*Q_1$.
5. Compute Cholesky factor R_2, such that $R_2^*R_2 = X_2$.
6. $Q := Q_1R_2^{-1}$.
7. $R := R_2R_1$.

While the amount of flops of this method is comparable to the Householder approach, the CholeskyQR2 method is better suited for high performance computing, which has been shown in empirical tests [69, 81].

This raises the hope to improve stability in the setting defined by non-standard scalar products. The idea is to compute a GR decomposition via the corresponding Cholesky-like factorization and repeat this a second time. This method is investigated for the hyperbolic QR decomposition in Section 7.2.4 and for the symplectic QR decomposition in Section 7.3.4.

7.2 The hyperbolic QR decomposition

7.2.1 Definition

The hyperbolic QR decomposition is a well-known tool in numerical linear algebra and can be computed via successive column elimination, similar to the orthogonal QR decomposition. This method is described briefly in Section 7.2.2. In the same way as the orthogonal QR decomposition is connected to the Cholesky factorization, the hyperbolic QR decomposition is connected to the LDL^{T} factorization. This factorization exists in different variants: D can be strictly diagonal or block-diagonal. Each variant can be used to define a variant of the hyperbolic QR decomposition. Section 7.2.3 unravels the involved nuances. In Section 7.2.4, we investigate the possibility to compute the Hyperbolic QR decomposition via two subsequent LDL^{T} factorizations.

Let Σ be a given signature matrix. We search for a way to compute $(\Sigma, \hat{\Sigma})$-orthogonal bases (see Definition 2.6) which span a given subspace. While Σ is a given signature matrix, $\hat{\Sigma}$ can be another arbitrary signature matrix. $(\Sigma, \hat{\Sigma})$-orthogonal matrices are also called hyperexchange matrices [94] and can be used to solve indefinite least square problems [45].

The methods presented in this section take a rectangular matrix $A \in \mathbb{K}^{m\times n}$ and a signature matrix Σ as input and deliver two outputs. These are another signature matrix $\hat{\Sigma}$, and $H \in \mathbb{K}^{m\times n}$, which spans the same subspace as A and is $(\Sigma, \hat{\Sigma})$-orthogonal. Subspace representations of this kind will be used in the computation of generalized polar decompositions in Chapter 8 of this thesis. Furthermore, they are required in the structure-preserving spectral divide-and-conquer method for pseudosymmetric matrices developed in Chapter 9. A classic method for computing such a subspace representation uses the hyperbolic QR decomposition.

Theorem 7.1 (The hyperbolic QR decomposition [54]):
Let $\Sigma \in \mathbb{R}^{m\times m}$ be a signature matrix, $A \in \mathbb{K}^{m\times n}$, $m \geq n$. Suppose all the leading principal submatrices of $A^*\Sigma A$ are nonsingular. Then there exists a permutation matrix P, a signature matrix $\hat{\Sigma} = P^\mathsf{T}\Sigma P$, a $(\Sigma, \hat{\Sigma})$-orthogonal matrix $H \in \mathbb{K}^{m\times m}$ (i.e. $H^*\Sigma H = \hat{\Sigma}$), and an upper triangular matrix $R \in \mathbb{R}^{n\times n}$, such that

$$A = H \begin{bmatrix} R \\ 0 \end{bmatrix}. \qquad \lozenge$$

The hyperbolic QR decomposition is unique (i.e. $A = H_1R_1$ and $A = H_2R_2$, where H_1 and H_2 are $(\Sigma, \hat{\Sigma})$-orthogonal, implies $H_1 = H_2$, $R_1 = R_2$) when the diagonal values of R are restricted to be positive real [163].

Remark 7.2:
The hyperbolic QR decomposition can be truncated to form a thin hyperbolic QR decomposition

$$A = H_0R,\ H_0 \in \mathbb{K}^{m\times n},\ R \in \mathbb{K}^{n\times n},\ H_0^*\Sigma H_0 = \hat{\Sigma}_0.$$

H_0 contains the first n columns of H and $\hat{\Sigma}_0$ contains the $n \times n$ leading submatrix of $\hat{\Sigma}$, where H and $\hat{\Sigma}$ are given in Theorem 7.1. $\lozenge$

7.2.2 Successive column elimination

The hyperbolic QR decomposition can be computed by accumulating transformations that introduce zeros below the diagonal, similar to the standard QR decomposition. We give a quick idea on how these elimination matrices are computed. For a more formal treatment, see e.g. [182]. For a given vector x and a given signature matrix Σ, we look for a transformation H such that $H^{-1}x = de_1$, where e_1 denotes the first unit vector and $H^\mathsf{H}\Sigma H = \hat{\Sigma}$ is another signature matrix. The two kinds of transformations used are orthogonal Householder transformations and hyperbolic Givens rotations. For illustrative purposes suppose $x \in \mathbb{C}^{2n}$ and $\Sigma = \mathrm{diag}(I_n, -I_n)$. Let

$$H_1 = \begin{bmatrix} H_+ & \\ & H_- \end{bmatrix}, \tag{7.1}$$

where H_+ and H_- are Householder transformations of dimension $n \times n$, such that $H_1^{-1}x = ae_1 + be_{n+1}$. We have $H_1^\mathsf{H}\Sigma H_1 = \Sigma$. The b entry in position $n+1$ is then

annihilated by a hyperbolic Givens rotation acting on rows 1 and $n+1$. The elimination $G^{-1}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix}$ is achieved by

$$G^{-1} = \begin{bmatrix} c & -s \\ -\overline{s} & c \end{bmatrix},$$
$$\text{where } \begin{cases} c = |a|/\sqrt{|a|^2 - |b|^2},\ s = e^{i\phi}|b|/\sqrt{|a|^2 - |b|^2} & \text{if } |a| > |b|, \\ c = |a|/\sqrt{|b|^2 - |a|^2},\ s = e^{i\phi}|b|/\sqrt{|b|^2 - |a|^2} & \text{if } |a| < |b|, \end{cases} \tag{7.2}$$

with $\phi = \arg a - \arg b$. The Givens matrix G is given as $G = \begin{bmatrix} c & \overline{s} \\ s & c \end{bmatrix}$. For the $|a| > |b|$ case we have

$$G^* \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} G = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{7.3}$$

For $|a| < |b|$ there is a sign switch in the signature matrix,

$$G^* \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} G = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{7.4}$$

If a and b are real then G is also real. Embedding G into a larger matrix H_2 (equal to the identity except in rows and columns 1 and $n+1$), gives the sought-after transformation $H = H_1 H_2$. Another signature matrix is given by $H^*\Sigma H = \hat{\Sigma}$, where $+1$ at diagonal position 1 and -1 at diagonal position $n+1$ have been interchanged if condition (7.4) takes effect. If condition (7.3) takes effect, the signature matrix does not change: $\Sigma = \hat{\Sigma}$.

The presented method works not only for the given specific signature matrix. For an arbitrary signature matrix Σ, the Householder matrix H_+ acts on the rows corresponding to positive entries of Σ, the Householder matrix H_- acts on the remaining rows. H_1 is set up accordingly. H_2 then acts on the remaining two entries and may or may not introduce a sign switch in the signature matrix. In the determination of the hyperbolic Givens rotation (7.2), the case $|a| = |b|$ is not covered and in this case, no suitable matrix G exists. The assumptions in Theorem 7.1 prevent this from happening. However, if a and b are close, G becomes ill-conditioned. This can lead to an instability in algorithms employing this kind of column elimination.

7.2.3 Connection to LDL^T factorizations

In order to overcome these potential instabilities, we take a look at the standard QR decomposition. Here, we can find the computation based on a Cholesky factorization as an alternative. In the indefinite setting an analogous connection exists between the hyperbolic QR factorization given in Theorem 7.1 and a scaled variant of the diagonal LDL^T factorization given in Theorem 2.15.

Lemma 7.3:
Let $A \in \mathbb{K}^{m\times n}$ have full column rank and a decomposition $A = HR$, where $H \in \mathbb{K}^{m\times n}$, $R \in \mathbb{K}^{n\times n}$. Then

$$H^*\Sigma H = \hat{\Sigma} \qquad \Leftrightarrow \qquad A^*\Sigma A = R^*\hat{\Sigma}R. \tag{7.5}$$

◊

Proof. For the implication from the left to the right statement, A does not need to have full rank. Inserting HR as A in $A^{\mathsf{H}}\Sigma A$, yields the right equation when the left equation is used. From right to left one inserts $H = AR^{-1}$ in $H^{\mathsf{H}}\Sigma H$ to find $\hat{\Sigma}$ by using the right equation. □

Remark 7.4:
If the right side of the equivalence (7.5) is given, $H = AR^{-1}$ can be recovered from A and R. In the case of signature matrices, the right side can be computed from an LDL^{T} decomposition $A^*\Sigma A = LDL^*$, where L is unit lower triangular, D is real diagonal. Then $R := |D|^{\frac{1}{2}}L^*$ and $\hat{\Sigma} := \operatorname{sign}(D)$ (containing the signs of the diagonal values in D) fulfill $A^*\Sigma A = R^*\hat{\Sigma}R$. ◊

Remark 7.4 points out how the (truncated) hyperbolic QR decomposition (Theorem 7.1) can be computed from the diagonal LDL^{T} decomposition. If instead the scaled LDL^{T} decomposition with pivoting (see Lemma 2.17) is used, one obtains the indefinite QR factorization given in the following Theorem. In contrast to the hyperbolic QR decomposition, it is not unique anymore. Additional degrees of freedom are introduced by the possibility of pivoting and because D is allowed to have 2×2 blocks on the diagonal. This allows for a more stable computation [14].

Theorem 7.5 (Indefinite QR decomposition [163]):
Let $\Sigma \in \mathbb{R}^{m\times m}$ be a signature matrix, $A \in \mathbb{K}^{m\times n}$, $m \geq n$. Suppose $A^*\Sigma A$ is nonsingular. Then there exists a factorization

$$A = HRP^{\mathsf{T}}, \qquad H \in \mathbb{K}^{m\times n},\ R \in \mathbb{K}^{n\times n},\ P \in \mathbb{R}^{n\times n}.$$

P is a permutation matrix, $P_\Sigma \in \mathbb{R}^{m\times n}$ contains n columns of an $m \times m$ permutation matrix and defines the signature matrix $\hat{\Sigma} = P_\Sigma^{\mathsf{T}}\Sigma P_\Sigma$. H is $(\Sigma, \hat{\Sigma})$-orthogonal (i.e. $H^*\Sigma H = \hat{\Sigma}$), and R is block-upper triangular with blocks of size 1×1 or 2×2. ◊

The difference between the indefinite QR factorization (Theorem 7.5) and the hyperbolic QR factorization (Theorem 7.1) is that pivoting is introduced, which results in the second permutation matrix P. Furthermore, 2×2 blocks may appear on the diagonal of R, and the assumption on $A^*\Sigma A$ is weaker. This decomposition can be computed via the successive use of transformation matrices, similar to the hyperbolic QR decomposition [162]. A perturbation analysis for the computation of the hyperbolic QR factorization (Theorem 7.1), i.e. the triangular case of the indefinite QR factorization in Theorem 7.5, is given in [163] and more recently in [113].

Computing the indefinite QR factorization via the LDL^{T} factorization employs the variant given in Lemma 2.17 and proceeds as follows.

1. Compute an LDL^T factorization $A^*\Sigma A = PLDL^*P^\mathsf{T}$, where D is block-diagonal.
2. Diagonalize D, i.e. compute unitary V, diagonal Λ such that $V\Lambda V^* = D$.
3. $R := |\Lambda|^{\frac{1}{2}} V^* L^*$.
4. $H := APR^{-1}$.

The unitary matrix V, computed in step 2, has the same block-diagonal structure as D.

7.2.4 Computing the indefinite QR factorization via two LDL^T decompositions

We follow the idea of the CholeskyQR2 algorithm and derive the indefinite variant. We call the algorithm LDLIQR2, standing for **LDL**$^\mathsf{T}$-based computation of the **I**ndefinte **QR** decomposition, applied **twice**. It computes a $(\Sigma, \hat{\Sigma})$-orthogonal basis of the subspace spanned by a matrix A. Σ is a given signature matrix and $\hat{\Sigma}$ is another signature matrix determined by the algorithm. It starts by computing the indefinite QR factorization

$$A = H_1 R_1 P_1^\mathsf{T}$$

via the LDL^T factorization with pivoting as described in the previous section. Then as a second step, the indefinite QR decomposition

$$H_1 = H R_2 P_2^\mathsf{T}$$

is computed using the same method. This yields a factorization

$$\begin{aligned} A = H R_2 P_2^\mathsf{T} R_1 P_1^\mathsf{T}, \text{ with } & R_1,\ R_2 \text{block upper triangular,} \\ & P_1,\ P_2 \text{ permutation matrices.} \end{aligned} \tag{7.6}$$

In exact arithmetic, the second step is redundant, as the hyperbolic QR decomposition of a $(\Sigma, \hat{\Sigma})$-orthogonal H is $H = HI$. In floating point arithmetic, however, we hope to see improvements regarding the accuracy of the computed factorization. P_2 will in practice often be the identity matrix. In this case, we have computed an instance of the indefinite QR factorization given in Theorem 7.5 with $R := R_2 R_1$, $P := P_1$. For our application we are just interested in a $(\Sigma, \hat{\Sigma})$-orthogonal basis, so the exact shape of R in a decomposition $A = HR$ does not matter. The method is formulated in Algorithm 7.1.

If one is only interested in computing H and $\hat{\Sigma}$, then Steps 4 and 8, computing R_1 and R_2, can be omitted. This is the case in the application considered in this thesis.

Algorithm 7.1: LDLIQR2: Compute $(\Sigma, \hat{\Sigma})$-orthogonal basis via double LDL^{T} factorization with pivoting.

Data: $A \in \mathbb{K}^{m\times n}$, with full column rank, $\Sigma \in \mathbb{R}^{n\times n}$ is a signature matrix.
Result: $(\Sigma, \hat{\Sigma})$-orthogonal $H \in \mathbb{K}^{m\times n}$ and R_1, P_1, R_2, $P_2 \in \mathbb{K}^{n\times n}$ yielding a decomposition (7.6).

`// First pass:`
1 $[L_1, D_1, P_1] \leftarrow \texttt{ldl}(A^*\Sigma A)$
2 $[V_1, \Lambda_1] \leftarrow \texttt{eig}(D)$ `// V_1 is block-diagonal.`
3 $H_1 \leftarrow AP_1L_1^{-*}V_1|\Lambda_1|^{-\frac{1}{2}}$
4 $R_1 \leftarrow |\Lambda_1|^{\frac{1}{2}}V_1^*L_1^*$
`// Second pass:`
5 $[L_2, D_2, P_2] \leftarrow \texttt{ldl}(H^*\Sigma H)$
6 $[V_2, \Lambda_2] \leftarrow \texttt{eig}(D)$ `// V_2 is block-diagonal.`
7 $H \leftarrow H_1P_2L_2^{-*}V_2|\Lambda_2|^{-\frac{1}{2}}$
8 $R_2 \leftarrow |\Lambda_2|^{\frac{1}{2}}V_2^*L_2^*$
`// Compute new signature matrix:`
9 $\hat{\Sigma} \leftarrow \Lambda_2|\Lambda_2|^{-1}$

7.3 The symplectic QR decomposition

7.3.1 Definition

The symplectic QR decomposition is introduced in [58] and serves as an important tool in model order reduction and control theory. Here, symplectic subspaces are used to solve algebraic Riccati equations [37, 109].

We now consider scalar products induced by J_n. A (J_m, J_n)-isometry $S \in \mathbb{K}^{2m\times 2n}$ ($\mathbb{K} = \mathbb{C}$ or $\mathbb{K} = \mathbb{R}$) fulfills the property $S^*J_mS = J_n$. The asterisk $.^*$ may refer to transposition ($.^* = .^{\mathsf{T}}$) or complex conjugate transposition ($.^* = .^{\mathsf{H}}$). When $\mathbb{K} = \mathbb{R}$ and $.^* = .^{\mathsf{T}}$, or when $\mathbb{K} = \mathbb{C}$ and $.^* = .^{\mathsf{H}}$, S is called (complex) *symplectic*. When $\mathbb{K} = \mathbb{C}$ and $.^* = .^{\mathsf{T}}$, the resulting isometries are referred to as *complex J-symmetric* [34]. In this work, the term symplectic may refer to all three cases.

The symplectic QR decomposition $A = SR$, as we use it in this thesis, is given in form of the following theorem. Here, we use the perfect shuffle permutation

$$P_n = \begin{bmatrix} e_1 & e_3 & \dots & e_{2n-1} & e_2 & e_4 & \dots e_{2n} \end{bmatrix},$$

where e_j denotes the unit vector, that contains 1 in entry j and 0 otherwise.

Theorem 7.6 (Symplectic QR decomposition [58, 76, 34]):
Let $A \in \mathbb{K}^{2m\times 2n}$, $m \geq n$, such that the principal minors of even dimension of

$P_n A^* J_m A P_n^\mathsf{T}$ are non-zero. Then A has a symplectic QR decomposition

$$A = SR, \quad R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \\ R_{21} & R_{22} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \ddots & \overset{0}{\ddots} \\ 0 & 0 \\ \underset{0}{\ddots} & \ddots \\ 0 & 0 \end{bmatrix}, \quad S^* J_m S = J_m, \tag{7.7}$$

where the diagonal values of R_{11} and R_{22} fulfill $R_{11,ii} = \pm R_{22,ii}$ for $i = 1, \ldots, n$. ◊

Proof. We describe the construction of the decomposition in Section 7.3.2, serving as a constructive proof. □

Remark 7.7 (Thin symplectic QR decomposition):
The symplectic QR decomposition can be truncated to form a thin symplectic QR decomposition

$$A = H_0 R, \; H_0 \in \mathbb{K}^{2m \times 2n}, \; R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_2 2 \end{bmatrix} = \begin{bmatrix} \ddots & \overset{0}{\ddots} \\ \underset{0}{\ddots} & \ddots \end{bmatrix} \in \mathbb{K}^{2n \times 2n}, \; H_0^* J_m H_0 = J_n.$$

For the orthogonal and hyperbolic QR decomposition this is done by taking the first n columns of the $n \times n$ matrix Q. Due to the different shape of R in the symplectic case, we have to chose the first n columns of the first and second half

$$H_0 = \begin{bmatrix} H(:, 1:n) & H(:, m+1:m+n) \end{bmatrix}. \qquad \Diamond$$

Remark 7.8:
The decomposition given in Theorem 7.6 can be made unique (i.e. $A = SR = \hat{S}\hat{R}$ implies $S = \hat{S}$, $R = \hat{R}$), e.g., by demanding the diagonal of R_{11} to contain only ones [58]. In the symplectic QR decomposition typically given in the literature [59], this is not the case and R has the form

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \\ R_{21} & R_{22} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \ddots & \ddots \\ 0 & 0 \\ \underset{0}{\ddots} & \ddots \\ 0 & 0 \end{bmatrix}. \tag{7.8}$$

Here, more degrees of freedoms are available. The diagonal values of R_{12} are not required to be zero, and the diagonal values of R_{22} are not fixed to mirror the ones of R_{11}. We choose them in a way such that a correspondence to the skew-symmetric Cholesky-like factorization can be established (see Section 7.3.3). ◊

7.3.2 Successive column elimination

Similar to the hyperbolic and orthogonal case, the symplectic QR decomposition can be computed by the successive introduction of zeros in the columns using symplectic transformations. The resulting matrix is of the form given in Remark 7.8. The missing conditions may be achieved in a post-processing step after the successive column elimination if they are relevant in the given context.

For illustrative purposes we consider real arithmetic (i.e. $\mathbb{K} = \mathbb{R}$) and set $.^* = .^\mathsf{T}$. The presented transformation matrices have complex analogues. The complex conjugate variants of these matrices lead to the complex J-symmetric decomposition [34]. Three different kinds of symplectic elimination matrices are needed. The first two are also orthogonal which makes them attractive from a numerical point of view.

The first kind of transformation performs the same orthogonal Householder transformation on the upper and the lower part of of the matrix, i.e.

$$H(k,v) = \begin{bmatrix} I_{k-1} & & & \\ & H_0(v) & & \\ & & I_{k-1} & \\ & & & H_0(v) \end{bmatrix}, \quad v \in \mathbb{R}^{n-k},\ k \in \{1,\dots,n\},$$

where $H_0(v) = I - \frac{2}{v^*v} vv^*$ is a regular Householder transformation (see Theorem 2.9).

The second kind of symplectic transformations needed are Givens rotations (see Theorem 2.10) that act on the entries k and $n+k$ of a vector of length $2n$:

$$G(k,\alpha) = \begin{bmatrix} I_{k-1} & & & & \\ & \cos\alpha & & \sin\alpha & \\ & & I_{n-1} & & \\ & -\sin\alpha & & \cos\alpha & \\ & & & & I_{n-k} \end{bmatrix}, \quad \alpha \in [0, 2\pi],\ k \in \{1,\dots,n\}.$$

Symplectic, orthogonal Householder transformations and Givens rotations can be combined to map a vector $a \in \mathbb{R}^{2n}$ onto the subspace

$$\operatorname{span}(e_1, \dots e_k, e_{n+1}, \dots, e_{n+k-1}),$$

where $k \in \{2,\dots,n\}$ is a given index. For $k = 1$, the vector is mapped onto $\operatorname{span}(e_1)$. This is achieved by an elementary orthogonal symplectic (EOS) matrix [109]

$$E_k(v_1, \alpha, v_2) = H(k, v_2)G(k, \alpha)H(k, v_1),$$

i.e. the concatenation of a Householder transformation, a Givens rotation and another Householder transformation. The Householder vector v_1 is chosen such that

$$H(k, v_1)a \in \operatorname{span}(e_1, \dots, e_{n+k}).$$

Now α, the parameter of the Givens rotation, is chosen such that the entry $n+k$ of $H(k, v_1)a$ is zeroed,

$$G(k, \alpha)H(k, v_1)a \in \operatorname{span}(e_1, \dots, e_{n+k-1}).$$

The defining vector v_2 of the second Householder transformation is chosen such that the entries $k+1$ to n of $G(k,\alpha)H(k,v_1)a$ are zeroed and we arrive at

$$H(k,v_2)G(j,\alpha)H(k,v_1)a \in \operatorname{span}(e_1,\dots e_k,e_{n+1},\dots,e_{n+k-1})$$

for $k \in \{2,\dots,n\}$ and

$$H(k,v_2)G(j,\alpha)H(k,v_1)a \in \operatorname{span}(e_1)$$

for the case $k=1$.

Applying EOS matrices defined by the first n columns of a matrix $A \in \mathbb{R}^{2m\times 2n}$ with the parameter $k=1,\dots,n$, one arrives at a matrix decomposition of the form

$$S^{-1}A = R = \begin{bmatrix} \ddots & \square \\ 0 & \square \\ \ddots & \square \\ 0 & \square \end{bmatrix}.$$

We see that no structure on the right half of A emerges. In order to introduce zeros in this part of A, orthogonal symplectic transformations do not suffice.

EOS matrices can be used to transform the first column of A to lie in $\operatorname{span}(e_1)$. The column $n+1$ can be transformed by a second EOS transformation to lie in $\operatorname{span}(e_1,e_2,e_{n+1})$, without destroying the structure in the first column. If now an EOS matrix with the parameter $k=2$ is used to transform the second column, the structure in the right half is lost. To prevent this from happening, it is necessary to get rid of the entry on the subdiagonal of the upper right block.

To achieve this, we introduce a third kind of symplectic transformation, which, in contrast to the previous two, is not orthogonal. A symplectic shear transformation [59] is given by

$$S(k,s) = \begin{bmatrix} I_{k-2} & & & & \\ & D(s) & & V(s) & \\ & & I_{n-2} & & \\ & & & D(s)^{-1} & \\ & & & & I_{n-k} \end{bmatrix},$$

$$D(s) = \begin{bmatrix} \frac{1}{(1+s^2)^{1/4}} & 0 \\ 0 & \frac{1}{(1+s^2)^{1/4}} \end{bmatrix}, \quad V(s) = \begin{bmatrix} 0 & \frac{s}{(1+s^2)^{1/4}} \\ \frac{s}{(1+s^2)^{1/4}} & 0 \end{bmatrix}.$$

Given $a \in \mathbb{R}^{2n}$, we have that the k-th entry of

$$y = S(k,s)a$$

is zero when

$$s = \begin{cases} -a_k/a_{n+k-1} & \text{if } a_{n+k-1} \neq 0, \\ 0 & \text{if } a_{n+k-1} = 0 \text{ and } a_k = 0. \end{cases}$$

If $a_{n+k-1} = 0$ and $a_k \neq 0$ is encountered, the symplectic QR decomposition does not exist. The conditions in Theorem 7.6 guarantee the existence and prevent this from happening in exact arithmetic. In floating point arithmetic however, values close to zero can cause instability. One approach to mitigate the issue is to use the degrees of freedom mentioned in Remark 7.8 to minimize the condition number of the diagonal blocks [76], which is not pursued in this work. Our approach is to exploit a connection to the skew-symmetric Cholesky factorization [31, 50] in Section 7.3.4.

The inverse of $S(k, s)$ is given by

$$S(k,s)^{-1} = \begin{bmatrix} I_{k-1} & & & & \\ & D(s)^{-1} & & -V(s) & \\ & & I_{n-2} & & \\ & & & D(s) & \\ & & & & I_{n-1-k} \end{bmatrix}.$$

Using the three kinds of symplectic transformations, the symplectic QR decomposition can be computed via successive column elimination. In a step k, columns k and $n+k$ are considered and transformed to

$$\begin{bmatrix} v_{1,k} & v_{2,k} \end{bmatrix}, \quad \begin{aligned} v_{1,k} &\in \operatorname{span}(e_1, \ldots, e_k, e_{n+1}, \ldots, e_{n+k-1}), \\ v_{2,k} &\in \operatorname{span}(e_1, \ldots, e_k, e_{n+1}, \ldots, e_{n+k}). \end{aligned}$$

The transformations are collected in S. The matrix A is updated accordingly until the form of R as in (7.8) is achieved.

Details are given in Algorithm 7.2. The functions `givens(.)` and `house(.)` refer to the computation of an appropriate Householder vector, see Theorem 2.9, and an appropriate angle for the Givens rotation, see Theorem 2.10. In actual implementations, the matrix elements of a Givens rotation are computed directly and the angle α is not needed. We use it in pseudocode for conceptual clarity.

After applying Algorithm 7.2 one arrives at a decomposition $A = SR$, where R has the form (7.8). With one more step, one arrives at the unique factorization given in Theorem 7.6. Let D_{ij} be a diagonal matrix with the same diagonal as a triangular block R_{ij}. The matrix

$$\hat{S} := \begin{bmatrix} D & -DD_{12}D_{22}^{-1} \\ 0 & D^{-1} \end{bmatrix}, \qquad D = \operatorname{sign}(D_{11}) \, |D_{11}|^{-\frac{1}{2}} \, |D_{22}|^{\frac{1}{2}}$$

is symplectic and transforms R such that the unique form (7.7) emerges. Here, $|D_{ij}|$ denotes a matrix which contains the absolute values of the entries of D_{ij}. The upper right block of $\hat{S}$ encodes n Gaussian eliminations introducing zeros on the diagonal of R_{12}. The factor $|D_{11}|^{-\frac{1}{2}} \, |D_{22}|^{\frac{1}{2}}$ represents scaling the rows of R such that the diagonal values of the upper left and lower right part have the same modulus. $\operatorname{sign}(D_{11})$ assigns a positive sign to the upper left block. With

$$\hat{S}^{-1} := \begin{bmatrix} D^{-1} & DD_{12}D_{22}^{-1} \\ 0 & D \end{bmatrix},$$

Algorithm 7.2: Compute the symplectic QR decomposition via successive column elimination.

Data: $A \in \mathbb{K}^{2m\times 2n}$ such that Theorem 7.6 holds.
Result: Symplectic matrix $S \in \mathbb{K}^{2m\times 2m}$, $R \in \mathbb{K}^{2m\times 2n}$ with form (7.8), such that $A = SR$.

$S \leftarrow I_{2m}$
for $k = 1 : n$ **do**
  `// Use elementary orthogonal symplectic matrix to create zeros in left half`
  $v \leftarrow \texttt{house}(A(m+k:2m, k))$
  $A \leftarrow H(k, v)A, \qquad S \leftarrow SH(k, v)^{-1}$
  $\alpha \leftarrow \texttt{givens}(A(k, k), A(m+k, k))$
  $A \leftarrow G(k, \alpha)A, \qquad S \leftarrow SG(k, \alpha)^{-1}$
  $v \leftarrow \texttt{house}(A(k:m, k))$
  $A \leftarrow H(k, v)A, \qquad S \leftarrow SH(k, v)^{-1}$
  if $k<m$ **then**
    `// Use elementary orthogonal symplectic matrix to create zeros in right half`
    $v \leftarrow \texttt{house}(A(m+k+1:2m, n+k))$
    $A \leftarrow H(k+1, v)A, \qquad S \leftarrow SH(k+1, v)^{-1}$
    $\alpha \leftarrow \texttt{givens}(A(k+1, n+k), A(m+k+1, n+k))$
    $A \leftarrow G(k+1, \alpha)A, \qquad S \leftarrow SG(k+1, \alpha)^{-1}$
    $v \leftarrow \texttt{house}(A(k+1:m, n+k))$
    $A \leftarrow H(k+1, v)A, \qquad S \leftarrow SH(k+1, v)^{-1}$
    `// Introduce zeros in upper right subdiagonal`
    $s = -A(k+1, n+k)/A(n+k)$
    $A \leftarrow S(k, s)A, \qquad S \leftarrow SS(k, s)^{-1}$
$R \leftarrow A$

the unique decomposition is given by

$$A = (S\hat{S}^{-1})(\hat{S}R) = S_u R_u,$$

where R_u has the form (7.7).

7.3.3 Connection to the skew-symmetric Cholesky-like factorization

Symplectic QR decompositions are related to factorizations of $A^{\mathsf{T}}JA$. The following lemma is an analogy of Lemma 7.3.

Lemma 7.9:
Let $A \in \mathbb{R}^{2m\times 2n}$, $m \geq n$, have full column rank and a decomposition $A = SR$ with

$S \in \mathbb{R}^{2m\times 2n}$, $R \in \mathbb{R}^{2n\times 2n}$. Then

$$S^\mathsf{T} J_m S = J_n \Leftrightarrow R^\mathsf{T} J_n R = A^\mathsf{T} J_m A. \tag{7.9}$$

◊

If a decomposition as in the right side of equivalence (7.9) is available, and R is nonsingular, it can be used to compute $S = AR^{-1}$. A factorization of this form is given by the skew-symmetric Cholesky-like factorization presented in [31, 50]. If it is formulated in form of the following Theorem, this becomes clear. This is a reformulated version of Theorem 2.2 in [31]. Again, the perfect shuffle P_n is used.

Theorem 7.10 (Skew-symmetric Cholesky-like factorization):
Let a skew-symmetric matrix $A \in \mathbb{R}^{2n\times 2n}$ be such that the principal submatrices of $P_n A P_n^\mathsf{T}$ of even size are nonsingular. Then A has a unique factorization

$$A = R^\mathsf{T} J_n R, \quad R = \begin{bmatrix} \ddots & \ddots & \\ 0 & \ddots & \ddots \\ & 0 & \ddots \end{bmatrix} = P_n^\mathsf{T} \hat{R} P_{n\cdot},$$

where the diagonal values of R_{11} are positive. The diagonals of R_{11} and R_{22} fulfill $R_{22,ii} = \pm R_{11,ii}$ for $j = 1, \ldots, n$. ◊

$\hat{R}$ in Theorem 7.10 is block upper triangular with 2×2 diagonal blocks on the diagonal.

For full-rank A, the decomposition $A = SR$ is a symplectic QR decomposition, as given in Remark 7.7, if and only if R defines a Cholesky-like decomposition of the skew-symmetric matrix $A^\mathsf{T} J_m A$ as in Theorem 7.10. With a given R, the symplectic factor S can be computed as $S = AR^{-1}$. Since R is just a permuted triangular matrix the linear solve is easy to compute.

[31] also introduces a variant of the skew-symmetric Cholesky-like factorization, where pivoting is used, yielding a better stability. The following theorem is a reformulated version of Theorem 2.3 in [31].

Theorem 7.11 (Skew-symmetric Cholesky-like factorization with pivoting):
Let a skew-symmetric matrix $A \in \mathbb{R}^{2n\times 2n}$ have even rank. Then there exists a permutation P such that A has a factorization

$$A = P^\mathsf{T} R^\mathsf{T} J R P, \quad R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} = \begin{bmatrix} \ddots & \ddots & \\ 0 & \ddots & \ddots \\ & 0 & \ddots \end{bmatrix} = P_n^\mathsf{T} \hat{R} P_{n\cdot}, \tag{7.10}$$

◊

where the diagonal values of R_{11} are positive or zero and the diagonals of R_{11} and R_{22} fulfill $R_{22,ii} = \pm R_{11,ii}$ for $j = 1, \ldots, n$.

The pivoting in form of a permutation matrix P, introduced in Theorem 7.11, increases the stability of the Cholesky-like decomposition. If we use the decomposition (7.10) and Lemma 7.9, we arrive at another variant of the symplectic QR decomposition.

Theorem 7.12 (Symplectic QR decomposition with pivoting):
Let $A \in \mathbb{R}^{2m\times 2n}$, such that $A^\mathsf{T} J_m A$ is nonsingular. Then there is a permutation P, such that A has a decomposition

$$A = SRP, \quad R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} = \begin{bmatrix} \ddots & \ddots \\ 0 & \ddots \\ & 0 \end{bmatrix}, \quad S^\mathsf{T} J_m S = J_n, \tag{7.11}$$

where R_{11} has a positive diagonal and the diagonal values of R_{11} and R_{22} fulfill $R_{22,ii} = \pm R_{11,ii}$ for $i = 1, \ldots, n$. ◊

An algorithm for computing the decomposition in Theorem 7.12 based on the skew-symmetric Cholesky-like factorization with pivoting in Theorem 7.11 takes the following form.

1. Compute Cholesky-like factorization $A^\mathsf{T} J_m A =: P^\mathsf{T} R^\mathsf{T} J_n R P$.
2. $S := AP^\mathsf{T} R^{-1}$.

In [31], an algorithm is devised to compute the triangular matrix $\hat{R} = P_n R P_n^\mathsf{T}$. We need to take care of the permutations and can use the algorithm to arrive at a permuted symplectic QR decomposition. There is no need to rewrite the algorithm to yield R instead of $\hat{R}$. The resulting procedure is the following.

1. Compute Cholesky-like factorization $A^\mathsf{T} J_m A =: Q^\mathsf{T} \hat{R}^\mathsf{T} \hat{J}_n \hat{R} Q$, where $\hat{J}_n = P_n J_n P_n^\mathsf{T}$, Q is a permutation.
2. $P := P_n^\mathsf{T} Q$.
3. $R := P_n^\mathsf{T} \hat{R} P_n$.
4. $S := AP^\mathsf{T} R^{-1}$.

7.3.4 Computing the symplectic QR decomposition via two Cholesky-like factorizations

As in the hyperbolic case, we draw inspiration from the CholeskyQR2 algorithm and repeat the procedure for the computed symplectic factor S. We call the resulting procedure `ssCholSQR2`, which stands for the computation of the **S**ymplectic **QR** decomposition via the **s**kew-**s**ymmetric **Chol**esky-like decomposition, applied **twice**.

First, the symplectic QR factorization of A is computed as

$$A = S_1 R_1 P_1$$

via the skew-symmetric Cholesky-like factorization with pivoting as described in the previous section. In a second step, the symplectic QR decomposition

$$S_1 = S R_2 P_2$$

is computed using the same method. We arrive at a a factorization

$$A = SR_2P_2R_1P_1, \text{ with } R_1,\ R_2 \text{ with form (7.10)}\ P_1,\ P_2 \text{ permutation matrices.} \tag{7.12}$$

The setting is completely analogous to the situation of the standard and the hyperbolic QR decomposition. We expect an improved accuracy after the second step, which would be redundant in exact arithmetic. The second pivoting step is often not necessary, such that one ends up with a symplectic QR decomposition as given in Theorem 7.12 with $R := R_2R_1$, $P := P_1$.

If one is interested in a symplectic subspace representation, then the shape of R does not matter. We give the details in Algorithm 7.3, where we use the notational shorthand

$$\hat{J}_n = P_nJ_nP_n^\mathsf{T} = \begin{bmatrix} 0 & 1 & & & & & \\ -1 & 0 & & & & & \\ & & 0 & 1 & & & \\ & & -1 & 0 & & & \\ & & & & \ddots & & \\ & & & & & 0 & 1 \\ & & & & & -1 & 0 \end{bmatrix}.$$

In Steps 1 and 5, the algorithm from [31] is used to compute the triangular matrix $\hat{R} = P_nRP_n^\mathsf{T}$. In Steps 3, 4, 7 and 8, the explicit forms of P_1, R_1, P_2 and R_2 from the decomposition (7.12) are computed. If only the symplectic matrix S is of interest, these steps are not necessary.

7.4 Numerical experiments

7.4.1 Hyperbolic QR decomposition

We implemented the LDLIQR2 Algorithm (Algorithm 7.1) in MATLAB R2018a. Random matrices of size 1000×1000 ($n = m = 500$) with a given condition number κ were generated using the command

```
A = gallery('randsvd', 2*n, kappa);
```

For the column elimination approach we used available MATLAB code [99], based on the works [12, 65, 92]. We compared this approach with the computation via the LDL^T decomposition presented in Section 7.2.3, referred to as "$1 \times LDL^\mathsf{T}$" and the LDLIQR2 algorithm (Algorithm 7.1), referred to as "$2 \times LDL^\mathsf{T}$". The quality of the computed factors H and R is evaluated in form of the residual $\|A - HR\|_F/\|A\|_F$ and the deviation from being $(\Sigma, \hat{\Sigma})$-orthogonal, i.e. $\|H^\mathsf{T}\Sigma H - \hat{\Sigma}\|_F$. Here, $\|.\|_F$ denotes the Frobenius norm. Results are found in Figure 7.1.

Algorithm 7.3: ssCholSQR2: Compute a symplectic basis via double skew-symmetric Cholesky-like factorization with pivoting.

Data: $A \in \mathbb{R}^{m\times n}$, with full column rank.
Result: Symplectic $S \in \mathbb{R}^{m\times n}$ and R_1, P_1, R_2, $P_2 \in \mathbb{R}^{n\times n}$ as in (7.12).
`// First pass:`
1 Compute factor $\hat{R}_1$, permutation Q_1 of a skew-symmetric Cholesky-like factorization such that

$$Q_1^\mathsf{T}\hat{R}_1^\mathsf{T}\hat{J}_n\hat{R}_1Q_1 = A^\mathsf{T}JA.$$

2 $S_1 \leftarrow AQ_1^\mathsf{T}\hat{R}_1^{-1}P_n$
3 $P_1 \leftarrow P_n^\mathsf{T}Q_1$
4 $R_1 \leftarrow P_n^\mathsf{T}\hat{R}_1P_n$
`// Second pass:`
5 Compute factor $\hat{R}_2$, permutation Q_2 of a skew-symmetric Cholesky-like factorization such that

$$Q_2^\mathsf{T}\hat{R}_2^\mathsf{T}\hat{J}_n\hat{R}_2Q_2 = Q_1^\mathsf{T}JQ_1.$$

6 $S \leftarrow SQ_2^\mathsf{T}\hat{R}_2^{-1}P_n$
7 $P_2 \leftarrow P_n^\mathsf{T}Q_2$
8 $R_2 \leftarrow P_n^\mathsf{T}\hat{R}_2P_n$

Figure 7.1: Numerical results for computing hyperbolic QR decompositions $A = HR$ of randomly generated matrices of size 1000×1000.

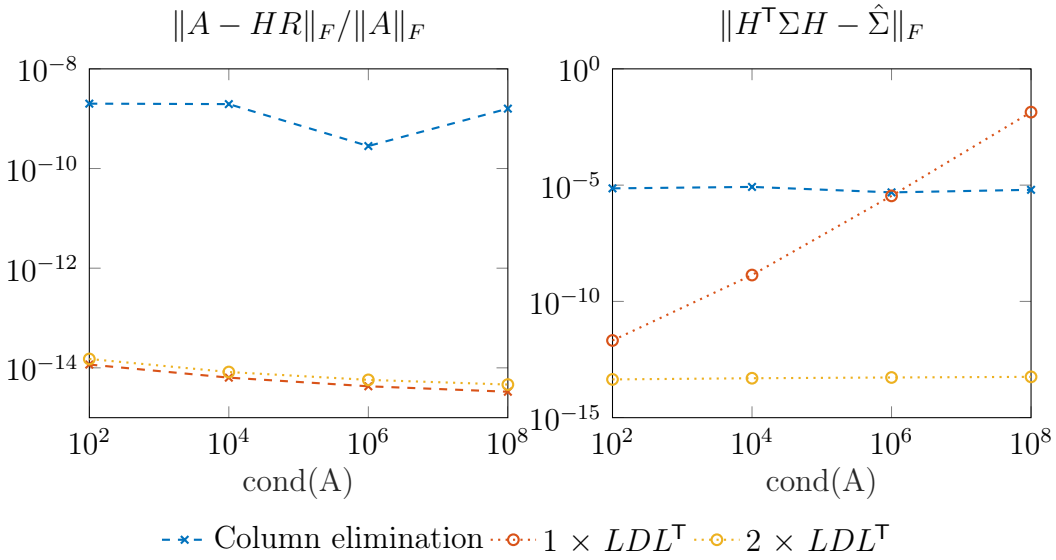


Figure 7.2: Numerical results for computing decompositions of randomly generated matrices of size 1000×1000.

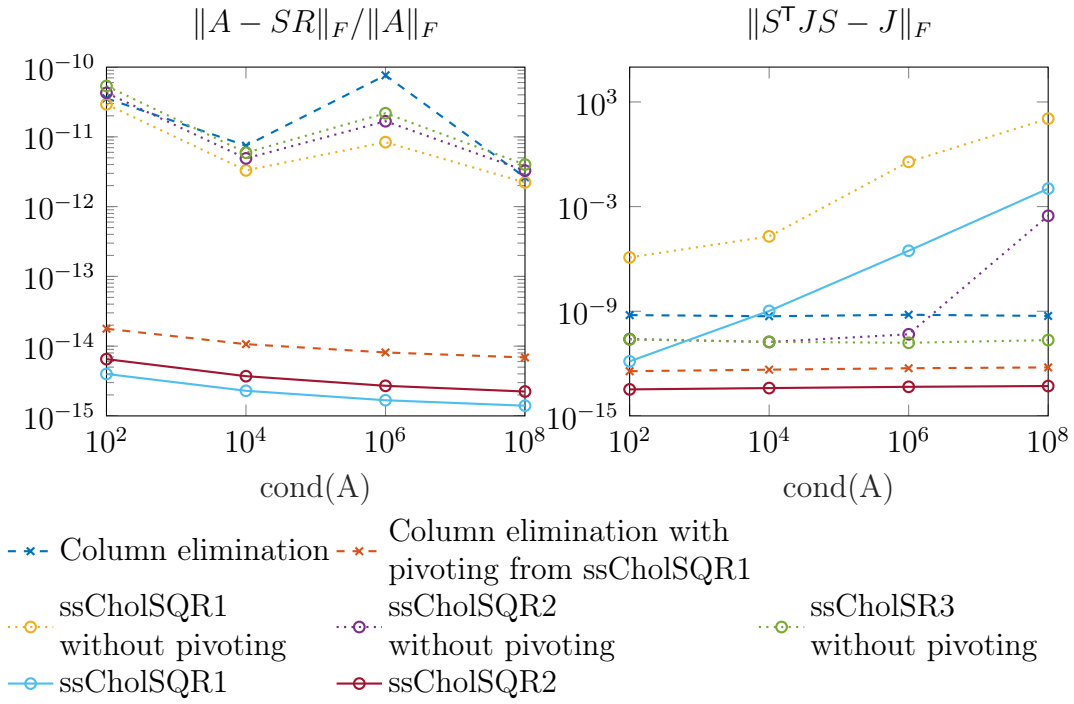


Figure 7.1 shows how the accuracy of a hyperbolic QR decomposition is improved by computing it via the LDL^T factorization. The residual is low for both LDL^T-based approaches, no matter the condition number of the matrix. A difference between both approaches is seen in the second figure regarding the deviation from generalized orthogonality. For well-conditioned matrices, both approaches yield better results than the elimination-based approach. However, the quality deteriorates for badly conditioned matrices, if only one LDL^T step is performed. Performing a second round solves this problem and yields consistently low deviations that are independent of the condition number. Using the $LDLIQR2$ algorithm is a promising approach when a high accuracy is mandatory.

7.4.2 Symplectic QR decomposition

We perform a similar experiment in the symplectic setting. Here, we consider several variants of the column elimination approach (Algorithm 7.2) and the approach based on the Cholesky-like factorization (Algorithm 7.3).

The matrices are generated as in the experiment regarding the hyperbolic QR decomposition in the previous section. Results on the residual and the (generalized) orthogonality are found in Figure 7.2.

Two variants are based on the skew-symmetric Cholesky-like decomposition with pivoting. Either one (`ssCholSQR1`), or two rounds (`ssCholSQR2`) of the process de-

scribed in Section 7.3.4 are performed.

We also consider variants of `ssCholSQR` without any pivoting, where we do up to three rounds of (re-)orthogonalization (`ssCholSR3`). Furthermore, we include an approach based on column elimination, where the matrix A is permuted according to the permutation resulting from `ssCholSQR1`. The pivoting in this test is known a-priori. This test serves the purpose to determine whether a clever way of pivoting could in principle improve the elimination-based approach.

ssCholSQR1 without pivoting gives a residual that is comparable to the column elimination approach but a worse symplecticity. It is therefore not recommended to compute the symplectic QR decomposition via a single skew-symmetric Cholesky-like factorization without pivoting. A second repetition (still without pivoting) only improves the symplecticity slightly. It still deteriorates for badly conditioned matrices. If we repeat the process one more time, the symplecticity seems reasonable, but the residual is still bad.

Pivoting seems to be a key element for good accuracy in the `ssCholSQR` algorithms. After one repetition with pivoting included, the residual is extremely low, but the symplecticity deteriorates for high condition numbers, becoming worse than for column elimination. A second repetition yields good results with respect to the residual and the symplecticity.

The last test performs a column elimination of the matrix with permuted columns. The permutation is the one previously computed by `ssCholSQR1`. This test shows what would be possible if a good pivoting strategy was known a-priori. The results are promising. The residual and the symplecticity do not deteriorate for badly conditioned matrices and are only about an order of magnitude worse than `ssCholSQR2`. In future research, this result could motivate the investigation of pivoting strategies for the column-elimination based approach.

7.5 Conclusions

In this section, we saw that connections exist between GR decompositions and analogues of the Cholesky factorization. We drew inspiration from the CholeskyQR2 algorithm and showed empirically, that a similar approach greatly improves the stability in the hyperbolic as well as in the symplectic setting.

We laid out the overall framework and showed that the computation via two Cholesky-like decompositions presents a way to avoid existing stability problems in the column elimination based approaches.

Computing the hyperbolic QR decomposition is useful in many applications [45, 186], which could benefit from the analysis given in this chapter. The LDLIQR2 method (Algorithm 7.1) is a promising new technique to tackle problems associated with the stability of the hyperbolic or indefinte QR decomposition.

This section can only serve as a first impulse for future research as a more thorough analysis of the stability regarding the laid out methods is missing and goes beyond the scope of this thesis.

Another point to be discussed concerns the suitability for high performance computing. While our presented algorithms were motivated by improved stability, CholeskyQR2, in the orthogonal setting, was motivated by improved performance in a highly parallel setting, where avoiding communication is key. In the orthogonal setting, the column elimination approach already yields a good stability. Our presented methods do not lent themselves to the high performance setting as naturally as the Cholesky decomposition because pivoting is involved. We have seen in particular in the experiments concerning the symplectic QR decomposition (see Figure 7.2) that pivoting is key for satisfactory results. However, pivoting always requires communication and could be an obstacle in the quest for high performance implementations. The design of pivoting strategies that yield satisfactory accuracy results while keeping communication at a minimum is a possible focus of future research. Shifting strategies, as employed in [80], should also be investigated.

CHAPTER 8

THE GENERALIZED POLAR DECOMPOSITION

Contents

8.1 Introduction

In this chapter we develop algorithms for computing generalized polar decompositions with a special focus on those related to scalar products induced by a signature matrix $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, $\sigma_j \in \{-1, 1\}$ for $j = 1, \ldots, n$. These decompositions may be considered in their own right and have interesting applications [105]. We are primarily interested in them as a tool to devise structure-preserving algorithms for structured eigenvalue problems, such as the Bethe-Salpeter eigenvalue problems introduced in Chapter 3 and examined in Chapter 4. The eigenvalue problems are given by matrices of the form

$$H = H_1 = \begin{bmatrix} A & B \\ -B^{\mathsf{H}} & -A^{\mathsf{T}} \end{bmatrix} \in \mathbb{C}^{2n\times 2n}, \qquad A = A^{\mathsf{H}},\ B = B^{\mathsf{T}}$$

or

$$H = H_2 = \begin{bmatrix} A & B \\ -B & -A \end{bmatrix} \in \mathbb{C}^{2n\times 2n}, \qquad A = A^{\mathsf{H}},\ B = B^{\mathsf{H}}.$$

As pointed out in Chapter 4, these matrices are pseudo-Hermitian. This is why this class of matrices receives special attention throughout this chapter. New structure-preserving algorithms for solving structured eigenvalue problems employ generalized polar decompositions and are presented in Chapter 9.

For $\mathbb{K} = \mathbb{C}$ or $\mathbb{K} = \mathbb{R}$, $m \geq n$, the polar decomposition of a matrix $A \in \mathbb{K}^{m\times n}$,

$$A = UH, \quad U^*U = I, \quad H = H^* \geq 0 \tag{8.1}$$

is given in Theorem 2.19. The matrix $U \in \mathbb{K}^{m\times n}$ is unitary and $H \in \mathbb{K}^{n\times n}$ is positive semidefinite. It is a well-known tool in numerical linear algebra.

Classically, the SVD is the starting point for the computation of the polar decomposition, see proof of Theorem 2.19, and it can be regarded as a "tuned down" variant of the SVD. However, it is sensible to study the polar decomposition in its own right. It is of use in many applications, in particular because of its best-approximation properties. For a detailed treatment see [95, Chapter 8]. The SVD-based method is not very pleasing from an algorithmic point of view, as the polar decomposition contains less (but still very useful) information than the SVD. This route therefore computes more than might be necessary in a given application. In recent years, methods have been developed to compute the polar decomposition efficiently on modern computer architectures [130, 132, 131, 116]. In fact, the polar decomposition can now be seen as a first step towards computing the SVD of a general matrix [167]. Efficient algorithms for computing the SVD of large matrices on high performance architectures form an active field of research.

It is well known that the unitary polar factor of a Hermitian matrix coincides with the matrix sign function, see Lemma 2.21. The matrix sign function is a widely used tool for acquiring invariant subspaces of a matrix, see Lemma 2.22. This property is used to solve matrix equations [149, 41] and develop parallelizable algorithms for solving eigenvalue problems [17, 169]. Therefore, efficient iterations for computing the polar decomposition, such as the QDWH iteration [132], and its successor based on Zolotarev functions [131], can be used to improve these methods for Hermitian matrices.

The concept of polar decompositions can be generalized in terms of non-standard scalar product spaces introduced in Chapter 2. Let $A \in \mathbb{K}^{m\times n}$, and $M \in \mathbb{K}^{m\times m}$, $N \in \mathbb{K}^{n\times n}$ be nonsingular. For certain matrices $M \in \mathbb{K}^{m\times m}$, $N \in \mathbb{K}^{n\times n}$, the canonical generalized polar decomposition can be defined. M and N are required to form an *orthosymmetric pair*.

Definition 8.1 (Definition 3.2 in [97]):
$M \in \mathbb{K}^{m\times m}$ and $N \in \mathbb{K}^{n\times n}$ form an *orthosymmetric pair* if and only if

(a) $M^{\mathsf{T}} = \beta M$, $N^{\mathsf{T}} = \beta N$, $\beta = \pm 1$ for bilinear forms,

(b) $M^{\mathsf{H}} = \alpha M$, $N^{\mathsf{H}} = \alpha N$, $|\alpha| = 1$ for sesquilinear forms. ◊

Definition 8.2 (Definition 3.6 in [97]):
A matrix $A \in \mathbb{K}^{m\times n}$ has a *canonical generalized polar decomposition* with respect to an orthosymmetric pair of matrices $M \in \mathbb{K}^{m\times m}$ and $N \in \mathbb{K}^{n\times n}$, if there exists a partial (M,N)-isometry W and an N-self-adjoint matrix S, where all nonzero eigenvalues of S have a positive real part, such that

$$A = WS, \tag{8.2}$$

and $\operatorname{range}(W^{\star_{M,N}}) = \operatorname{range}(S)$. ◊

If A has full column rank, W is (M,N)-orthogonal. If additionally, A is square and $M = N$, then W is an N-automorphism.

In contrast to the standard polar decomposition, the existence of the (canonical) generalized polar decomposition can in general not be guaranteed. The following theorem clarifies this issue.

Theorem 8.3 (Theorem 3.9 in [97]):
A matrix $A \in \mathbb{K}^{m\times n}$ has a unique canonical generalized polar decomposition with respect to the orthosymmetric pair M, N if and only if all the following conditions apply.

1. The matrix $A^{\star_M}A$ has no eigenvalues on the negative real axis.
2. If zero is an eigenvalue of $A^{\star_{M,N}}A$, then it is semisimple.
3. $\ker(A^{\star_{M,N}}) = \ker(A)$. ◊

In case of existence, we have $S = (A^{*_{M,N}}A)^{\frac{1}{2}}$ and $W^{\star_{M,N}}WS = S$. Just as the standard polar decomposition, the generalized polar decomposition is related to the matrix sign function.

Theorem 8.4:
Let M be a nonsingular matrix and $A \in \mathbb{K}^{n\times n}$ be self-adjoint with respect to the scalar product induced by M. If A has no purely imaginary eigenvalues (including zero eigenvalues), $\operatorname{sign}(A)$ and the canonical generalized polar decomposition (with respect to M) $A = WS$ are well-defined and it holds

$$\operatorname{sign}(A) = W. \qquad ◊$$

Proof. The matrix sign function can be expressed as [95]

$$\operatorname{sign}(A) = A(A^2)^{-1/2}.$$

The generalized polar decomposition $A = WS$ is well-defined with a unique self-adjoint factor S if $M^{-1}A^*MA$ has no negative real eigenvalues. For self-adjoint matrices, it holds $M^{-1}A^*M = A$, so $M^{-1}A^*MA = A^2$ can only have negative real eigenvalues if A has purely imaginary eigenvalues. So $A = WS$ is well-defined. Using $S = (M^{-1}A^*MA)^{1/2}$, the factor W is given as

$$W = A(M^{-1}A^*MA)^{-1/2} = A(A^2)^{-1/2} = \operatorname{sign}(A)\,. \qquad \square$$

Because of this equivalence, finding efficient iterations for computing the generalized polar decomposition can lead to new methods for matrix equations and eigenvalue problems involving self-adjoint matrices.

The standard polar decomposition (8.1) can be used to solve the orthogonal Procrustes problem [95]. It consists of finding an orthogonal transformation that most closely maps one rectangular matrix to another. This task arises in the context of multidimensional scaling [48] and factor analysis [66], which form important sets of tools for data analysis in fields such as psychology and marketing.

Here, the non-orthogonal Procrustes problem arises as well, which calls for the polar factor with respect to an indefinite scalar product [105]. This is a specific version of the generalized polar decomposition (8.2).

In this chapter, we present some results on how generalized polar decompositions can be computed based on the dynamically weighted Halley (DWH) iteration. When this iteration employs a QR decomposition to realize an iteration step (then called QDWH iteration), it is successful in computing the standard polar decomposition in an efficient and stable way [132]. We focus on the important subclass of scalar products induced by signature matrices, i.e. diagonal matrices with $+1$ and -1 as diagonal values, denoted by Σ throughout this thesis.

As mentioned above, the ultimate goal of this thesis is to develop new algorithms for finding eigenvalues and eigenvectors of Bethe-Salpeter matrices. In chapters 4 and 6, we addressed that the Bethe-Salpeter matrix H is self-adjoint with respect to the signature matrix $K = I_n \oplus -I_n$ (i.e. it is pseudosymmetric), and often possesses an additional property: Due to physical constraints, KH is typically positive definite. Similar structures arise in different contexts of electronic structure theory [123, 72, 35]. We call pseudosymmetric matrices with this property definite pseudosymmetric matrices. For these matrices in particular, the convergence behavior of our proposed method will turn out to be as good as in the standard setting defined by the Euclidean scalar product.

Pseudosymmetric matrices also play a role in describing damped oscillations of linear systems. In [177] they are called J-Hermitian and definite pseudosymmetric matrices are called J-positive.

The remainder of this chapter is structured as follows. In Section 8.2, we recapitulate the central ideas of the QDWH algorithm. Section 8.3 shows how they can be applied in order to compute a generalized polar decomposition. We show general results and then restrict ourselves to scalar products induced by signature matrices. Here, inverses can be avoided by using the decompositions presented earlier in Section 7.2. The introduction of permuted graph bases can further improve the stability of the computation of the generalized polar factor. Details are found in Section 8.4. Section 8.5 gives numerical results on the question of stability and convergence. Conclusions and further research directions are discussed in Section 8.7.

8.2 The QDWH algorithm for computing the standard polar decomposition

Methods for the computation of the polar decomposition of a matrix (8.1) have been studied extensively in recent years. Once the orthogonal polar factor is computed, the symmetric factor can be recovered via $H = U^*A$. Numerical symmetry can be guaranteed by performing $H := (H + H^*)/2$ as an extra step.

A current state-of-the-art iterative method for computing the polar factor is the QDWH algorithm [130]. It is based on the well-known Halley iteration which is a member of the Padé family of iterations [103]. The Dynamically Weighted Halley (DWH) iteration introduces the weights a_k, b_k, $c_k \in \mathbb{R}^+$ and is given as

$$X_{k+1} = X_k(a_k I + b_k X_k^* X_k)(I + c_k X_k^* X_k)^{-1}, \quad X_0 = \frac{1}{\|A\|_2} A. \tag{8.3}$$

Convergence is globally guaranteed with an asymptotic cubic rate, provided A has full column rank. In order to choose the weights in an optimal fashion, iteration (8.3) is understood as an iteration acting on the singular values of the iterate X_k. Let $X_k = U_S D_k V_S{}^*$ be the SVD of X_k. Then one step of iteration (8.3) yields

$$X_{k+1} = U_S g_k(D_k) V_S{}^*, \tag{8.4}$$

where

$$g_k(x) = x \frac{a_k + b_k x^2}{1 + c_k x^2}.$$

The singular value $\sigma_{i,k+1}$ of X_{k+1} is hence given by a rational function acting on the singular value $\sigma_{i,k}$ of X_k,

$$\sigma_{i,k+1} = g_k(\sigma_{i,k}). \tag{8.5}$$

The singular values converge to 1 as X_k approaches the polar factor. Let $\ell (=: \ell_0)$ be a lower bound to the singular values of X_0. Due to the initial scaling with $1/\|A\|_2$ the singular values of X_0 lie between 0 and 1. A successful strategy for accelerating convergence can be developed by minimizing the distance of ℓ_k to 1 in each iteration. This line of thoughts leads to weights chosen as

$$a_k = h(\ell_k), \quad b_k = (a_k - 1)^2/4, \quad c_k = a_k + b_k - 1, \quad \ell_{k+1} = g_k(\ell_k), \tag{8.6}$$

where

$$h(\ell) = \sqrt{1 + d} + \frac{1}{2}\sqrt{8 - 4d + \frac{8(2 - \ell^2)}{\ell^2 \sqrt{1 + d}}}, \quad d = \sqrt[3]{\frac{4(1 - \ell^2)}{\ell^4}}. \tag{8.7}$$

The weights (8.6) are the solutions of an optimization problem. This is how they were introduced in [130]. Another derivation considers the best rank-(3,2) rational

approximation of the sign function. This leads to the same weights given in (8.6). The latter approach can be extended to rational approximations of higher order (Zolotarev functions) [131].

For a matrix A with condition number $\kappa_2(A) < 10^{16}$, convergence within 6 iterations can be guaranteed using these weights [130]. A rewrite of the iteration (8.3)

$$\begin{aligned} & X_k(a_k I + b_k X_k^* X_k)(I + c_k X_k^* X_k)^{-1} \\ = \; & \frac{b_k}{c_k} X_k + \left(a_k - \frac{b_k}{c_k}\right) X_k (I + c_k X_k^* X_k)^{-1} \end{aligned} \tag{8.8}$$

leads to two distinct implementation variants. The matrix $(I + c_k X_k^* X_k)$ is symmetric positive definite and its linear solve is done using a Cholesky factorization in the first variant.

$$\begin{cases} Z_k = I + c_k X_k^* X_k, \quad W_k = \mathrm{chol}(Z_k), \\ X_{k+1} = \frac{b_k}{c_k} X_k + \left(a_k - \frac{b_k}{c_k}\right) X_k W_k^{-1} W_k^{-*}. \end{cases} \tag{8.9}$$

It can also be shown that $X_k(I + c_k X_k^* X_k)^{-1}$ is equivalently computed via a QR decomposition, which leads to the QR-based Dynamically Weighted Halley (QDWH) iteration

$$\begin{cases} \begin{bmatrix} \sqrt{c_k} X_k \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R, \\ X_{k+1} = \frac{b_k}{c_k} X_k + \frac{1}{\sqrt{c_k}} \left(a_k - \frac{b_k}{c_k}\right) Q_1 Q_2^*. \end{cases} \tag{8.10}$$

This variant avoids inversion and is proven to be backward stable [132]. It has, however, a higher operation count than the Cholesky variant (8.9). This is why in practical implementations, the QR-based variant (8.10) is carried out in the first iterations and switches to the Cholesky variant (8.9) as soon as a reasonably conditioned iterate X_k is guaranteed. This way, numerical stability of the iteration is not compromised.

The two forms of the iteration represent the connection between the QR decomposition and the Cholesky factorization described in Chapter 7. They are two sides of the same coin. Either the QR decomposition of $A = \begin{bmatrix} \sqrt{c_k} X \\ I \end{bmatrix}$ is computed, leading to iteration (8.10), or the Cholesky factorization of $A^* A = I + c_k X^* X$ is computed and used for a linear solve, leading to iteration (8.9).

8.3 Computing generalized polar decompositions

8.3.1 The generalized QDWH algorithm

In this section, iterative methods for computing the generalized polar factor are constructed by exploiting a connection to the matrix sign function.

Theorem 8.5 (Theorem 5.1 in [96]):
Let $A = WS$ be a matrix with an existing canonical generalized polar decomposition with respect to the orthosymmetric pair M, N (see Definition 8.2). Let

$$X_{k+1} = g(X_k) = X_k h(X_k^2) \tag{8.11}$$

be an iteration that converges to $\operatorname{sign}(X_0)$, assuming it exists. $g(\cdot)$ and $h(\cdot)$ are matrix functions. Let $g(0) = 0$ and for sesquilinear forms assume that $g(X^{\star_N}) = g(X)^{\star_N}$ holds for all X in the domain of g. Then the iteration

$$Y_{k+1} = Y_k h(Y_k^{\star_{M,N}} Y_k), \qquad Y_0 = A,$$

converges to W with the same order of convergence as iteration (8.11) converges to $\operatorname{sign}(X_0)$. ◊

Iterations for the matrix sign function of the form (8.11) are very common and well-studied [95, Ch. 5]. They include the class of Padé iterations devised in [102]. Here, the iteration is given as a rational function of the form

$$X_{k+1} = X_k p_{lm}(I - X_k^2) q_{lm}(I - X_k^2)^{-1}, \qquad X_0 = A,$$

where $p_{lm}(\cdot)$ and $q_{lm}(\cdot)$ are explicitly given polynomials, yielding the Padé approximant of degree (l, m).

Choosing $l = m = 1$ leads to the Halley iteration, which also forms the basis of the QDWH algorithm presented in Section 8.2. In the context of the generalized polar decomposition, the DWH iteration follows from applying Theorem 8.5 and is given as

$$X_{k+1} = X_k(a_k I + b_k X_k^{\star_{M,N}} X_k)(I + c_k X_k^{\star_{M,N}} X_k)^{-1}, \quad X_0 = sA, \tag{8.12}$$

where $s \in \mathbb{K}$ is an arbitrary scaling factor, as any sA has the same polar factor W. A discussion on how to choose a beneficial s follows later. More explicitly, using $A^{\star_{M,N}} = N^{-1}A^*M$, the iteration (8.12) is given as

$$X_{k+1} = X_k(a_k I + b_k N^{-1} X_k^* M X_k)(I + c_k N^{-1} X_k^* M X_k)^{-1}, \quad X_0 = sA.$$

The generalization of the DWH algorithm given in the previous paragraphs is straightforward. We now investigate whether this iteration has attractive numerical properties and which circumstances can lead to an accelerated convergence. The key observation in the standard setting is that one iteration step acts as a rational function on the singular values of the iterate X_k (see equations (8.4) to (8.5)). A similar observation helps in the indefinite setting.

Corollary 8.6:
Let the canonical generalized polar decomposition $A = WS$ exist and be computed via an iteration $X_{k+1} = X_k h(X_k^{\star_{M,N}} X_k)$, $X_0 = A$, as given in Theorem 8.5. Then X_k has a canonical generalized polar decomposition

$$X_k = WS_k.$$

For the series of self-adjoint factors S_k it holds

$$S_{k+1} = S_k h(S_k^2). \tag{8.13}$$

◊

Proof. See proof of Theorem 5.1 in [97]. □

Using the Jordan canonical form $S = ZJZ^{-1}$, we see that (8.13) is equivalent to

$$S_{k+1} = Zg(J_k)Z^{-1} = ZJ_{k+1}Z^{-1},$$

with $g(x) = xh(x^2)$. Essentially, one iteration step for computing the generalized polar decomposition acts as a rational function on the eigenvalues of the self-adjoint factor S, such that they converge towards 1 (or stay 0 in the rank-deficient case). Note that all non-zero eigenvalues of S have positive real part and $S = (A^{\star_{M,N}}A)^{\frac{1}{2}}$ by definition.

In the standard setting outlined in Section 8.2, i.e. the case $M = I_m$, $N = I_n$, S is symmetric (respectively Hermitian) and has only real eigenvalues. These eigenvalues are the singular values of A. This property does not hold in the general case. Only the convergence of the real eigenvalues of S is guaranteed to benefit from choosing the weighing parameters as in the standard case.

The reason we are interested in developing this method further, lies in its possible applications laid out in Chapters 3 and 4. In the application in quantum physics, the relevant eigenvalues are in fact often real. This follows from physical constraints and does not follow directly from the given matrix structure. More specifically, it holds that ΣA is Hermitian and positive definite. We call a matrix with this property a definite pseudosymmetric matrix. The signature matrix is specifically given in this application by $\Sigma = K$. In this case, we expect great benefits from choosing the weighting parameters (8.6) and (8.7).

The scaling factor s in iteration (8.12) should be chosen in the following way: Let $sA = WS_s$ be the generalized polar decomposition of $X_0 = sA$. The polar factor W is the same as for A. The pseudosymmetric factor S_s is the scaled pseudosymmetric factor of $A = WS$, $S_s = sS$. Note that we consider the definite case here, where all eigenvalues of S are real. The scaling s should be chosen such that the absolute values of the eigenvalues of S_s lie between 0 and 1, i.e.

$$s \leq (|\lambda_{\max}(S)|)^{-1} = (|\lambda_{\max}|((\Sigma A^*\Sigma A)^{\frac{1}{2}}))^{-1}, \tag{8.14}$$

where $\lambda_{\max}$ refers to the eigenvalue with maximum absolute value. ℓ_0 should be a lower bound on the eigenvalue with smallest absolute value of S_s, i.e.

$$\ell_0 \leq |\lambda_{\min}|(S_s) = s|\lambda_{\min}|((\Sigma A^*\Sigma A)^{\frac{1}{2}}), \tag{8.15}$$

where $\lambda_{\min}$ refers to the eigenvalue with minimum absolute value. Computing values fulfilling properties (8.14) and (8.15) seems non-trivial, as computing S (after computing W via the iteration) is the goal of the algorithm and S is not known a-priori. The following lemma gives a remedy for square matrices.

Lemma 8.7:
Let $A \in \mathbb{K}^{n\times n}$ and $Q_1 \in \mathbb{K}^{n\times n}$, $Q_2 \in \mathbb{K}^{n\times n}$ be unitary. Then

$$|\lambda_{\max}((Q_1A^*Q_2A)^{\frac{1}{2}})| \leq \sigma_{\max}(A), \qquad |\lambda_{\min}((Q_1A^*Q_2A)^{\frac{1}{2}})| \geq \sigma_{\min}(A).$$

Again, $\lambda_{\max}$ and $\lambda_{\min}$ refer to an eigenvalue ordering by absolute value. ◊

Proof. Because the spectral norm is submultiplicative, we have

$$|\lambda_{\max}(Q_1A^*Q_2A)| \leq \sigma_{\max}(Q_1A^*Q_2A) \leq \sigma_{\max}(Q_1A^*Q_2)\sigma_{\max}(A) = \sigma_{\max}(A)^2,$$
$$|\lambda_{\min}(Q_1A^*Q_2A)| \geq \sigma_{\min}(Q_1A^*Q_2A) \geq \sigma_{\min}(Q_1A^*Q_2)\sigma_{\min}(A) = \sigma_{\min}(A)^2.$$

The proposition follows immediately. □

Lemma 8.7 for $Q_1 = Q_2 = \Sigma$ implies that s and ℓ_0 can be chosen as

$$s \approx 1/\sigma_{\max}(A), \qquad \ell_0 \approx s\sigma_{\min}(A) = 1/\mathrm{cond}_2(A) \tag{8.16}$$

in order to fulfill properties (8.14) and (8.15) in the case of square matrices.

As discussed, generalizing the ideas from QDWH guarantees favorable convergence properties for certain matrices. An additional benefit is that it leads to a new class of inverse-free iterations for computing the generalized unitary polar factor, as we see in the following. In the case of self-adjoint matrices, this polar factor coincides with the matrix sign function, which is relevant in many application areas. Avoiding the inverse opens up the possibility of more stable methods.

Here, the role of the orthogonal representations in QDWH is played by (M, N)-orthogonal matrices defined via two scalar products given by two matrices M and N. The following lemma provides a tool for substituting the inverse $(I + c_k X_k^{\star_{M,N}} X_k)^{-1}$ in iteration (8.12).

Lemma 8.8:
Let $M \in \mathbb{K}^{m\times m}$, $N \in \mathbb{K}^{n\times n}$ be nonsingular, and $M_2 := \begin{bmatrix} M & \\ & N \end{bmatrix}$. For $X \in \mathbb{K}^{m\times n}$, $\eta \in \mathbb{K}$, let $\begin{bmatrix} \eta X \\ I_n \end{bmatrix} = VR$ with $V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \in \mathbb{K}^{(m+n)\times n}$, $R \in \mathbb{K}^{n\times n}$ nonsingular, be a decomposition. Then

$$\eta X(I + |\eta|^2 X^{\star_{M,N}} X)^{-1} = V_1(V^{\star_{M_2,N}} V)^{-1} V_2^{\star_N}. \qquad \Diamond$$

Proof. It holds

$$\begin{aligned}
\eta X(I + |\eta|^2 X^{\star_{M,N}} X)^{-1} &= \eta X \left(\begin{bmatrix} \eta X \\ I \end{bmatrix}^{\star_{M_2,N}} \begin{bmatrix} \eta X \\ I \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} I & 0 \end{bmatrix} VR((VR)^{\star_{M_2,N}} VR)^{-1} \\
&= V_1((VR)^{\star_{M_2,N}} V)^{-1} \\
&= V_1(R^{\star_N} V^{\star_{M_2,N}} V)^{-1} \\
&= V_1(V^{\star_{M_2,N}} V)^{-1}(R^{-1})^{\star_N} \\
&= V_1(V^{\star_{M_2,N}} V)^{-1} V_2^{\star_N}.
\end{aligned}$$

In the last step we used $V_2 = R^{-1}$. □

For $M = I_m$, $N = I_n$ and orthogonal or unitary V, we have the known result

$$\eta X(I + |\eta|^2 X^* X)^{-1} = V_1 V_2^*,$$

given for example as Theorem 4.1 in [130]. The original QDWH algorithm is based on this result. A straightforward idea to generalize this approach would be to choose V to be (M_2, N)-orthogonal, i.e. $V^{\star_{M_2,N}} V = I$. The next lemma shows how we can relax this condition, while keeping the inverse easy to compute.

Lemma 8.9:
Let $M \in \mathbb{K}^{m\times m}$, $N \in \mathbb{K}^{n\times n}$ be nonsingular, $M_2 = \begin{bmatrix} M & \\ & N \end{bmatrix}$, and $V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \in \mathbb{K}^{(m+n)\times n}$ be $(M_2, \hat{N})$-orthogonal for a matrix $\hat{N} \in \mathbb{K}^{n\times n}$, i.e. $V^* M_2 V = \hat{N}$. Then

$$V_1 (V^{\star_{M_2,N}} V)^{-1} V_2^{\star_N} = V_1 V_2^{\star_{N,\hat{N}}}. \qquad \Diamond$$

Proof. From $V^* M_2 V = \hat{N}$, it follows $V^{\star_{M_2,N}} V = N^{-1}\hat{N}$ and therefore

$$V_1 (V^{\star_{M_2,N}} V)^{-1} V_2^{\star_N} = V_1 \hat{N}^{-1} V_2^* N = V_1 V_2^{\star_{N,\hat{N}}}. \qquad \square$$

8.3.2 Realizing the ΣDWH iteration

When a practical method for computing the $(M_2, \hat{N})$-orthogonal matrices in Lemma 8.9 is available, we can formulate a generalized QDWH algorithm. If applying N^{-1} is trivial to implement (e.g. N is involutory), this leads to an inverse-free computation, if the computation of the $(M_2, \hat{N})$-orthogonal matrix avoids inversion. We now leave the general framework and restrict ourselves to scalar products induced by signature matrices.

Chapter 7 (specifically Section 7.2) laid the groundwork for several options in the algorithm design realizing iteration (8.12) for computing the canonical generalized polar decomposition of $A \in \mathbb{K}^{m\times n}$ with respect to the signature matrices Σ_m and Σ_n. As signature matrices are involutory, the iteration is given as

$$X_{k+1} = X_k(a_k I + b_k \Sigma_n X_k^* \Sigma_m X_k)(I + c_k \Sigma_n X_k^* \Sigma_m X_k)^{-1}, \quad X_0 = A/\|A\|. \tag{8.17}$$

We call iteration (8.17) the ΣDWH iteration. The naive approach is to implement the iteration straightforward, using a linear solve employing the MATLAB backslash operator. However, there is a better way to exploit the structure at hand. To see this, we rewrite (8.17) as

$$\begin{aligned} & X_k(a_k I + b_k \Sigma_n X_k^* \Sigma_m X_k)(I + c_k \Sigma_n X_k^* \Sigma_m X_k)^{-1} \\ = \; & \frac{b_k}{c_k} X_k + (a_k - \frac{b_k}{c_k}) X_k (\Sigma_n + c_k X_k^* \Sigma_m X_k)^{-1} \Sigma_n. \end{aligned}$$

This is the indefinite analogue to equation (8.8). In the standard case, the Cholesky factorization is employed to exploit the symmetric structure in iteration (8.9). In the

indefinite case, this role is played by a scaled variant of the pivoted LDL^T factorization. Analogously to iteration (8.9), iteration (8.17) is equivalently given as

$$\begin{cases} Z_k = \Sigma_n + c_k X_k^* \Sigma_m X_k, \quad [L_k, D_k, P_k] = \mathrm{ldl}(Z_k), \\ X_{k+1} = \frac{b_k}{c_k} X_k + \left(a_k - \frac{b_k}{c_k}\right) X_k P_k L_k^{-*} D_k^{-1} L_k^{-1} P_k^\mathsf{T} \Sigma. \end{cases} \tag{8.18}$$

This approach is already more promising than the naive one because the structure of the involved matrices is exploited. This way, less computational work is needed and we may expect better accuracy.

We employ Lemma 8.8 and Lemma 8.9 to find an equivalent formulation of the DWH iteration (8.17), which in principle does not rely on computing inverses. The role of $\hat{N}$ in Lemma 8.9 is played by another signature matrix $\hat{\Sigma}_n$ of size $n \times n$. The formulation

$$\begin{cases} \begin{bmatrix} \sqrt{c_k} X_k \\ I \end{bmatrix} = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} R, \text{ where } \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}^* \begin{bmatrix} \Sigma_m & \\ & \Sigma_n \end{bmatrix} \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} = \hat{\Sigma}_n, \\ X_{k+1} = \frac{b_k}{c_k} X_k + \frac{1}{\sqrt{c_k}} \left(a_k - \frac{b_k}{c_k}\right) H_1 \hat{\Sigma}_n H_2^* \Sigma_n \end{cases} \tag{8.19}$$

is the analogue to the QR-based iteration (8.10) in the standard case. Instead of an orthogonal basis (using the QR decomposition), a $\left(\begin{bmatrix} \Sigma_m & \\ & \Sigma_n \end{bmatrix}, \hat{\Sigma}_n\right)$-orthogonal basis is computed. This can be done by computing the hyperbolic QR decomposition (Theorem 7.1) or the indefinite QR decomposition (Theorem 7.5). Here, methods exist that are based on successive column elimination and do not perform any matrix inversions (see Section 7.2.2). Computing the indefinite QR decomposition via an LDL^T factorization (i.e. employing Lemma 7.3) gives exactly the LDL^T based iteration (8.18). A promising way to compute the required basis is to employ the LDLIQR2 algorithm (Algorithm 7.1).

The resulting stability of iterations employing these different approaches is examined experimentally in the numerical experiments of Section 8.5.

8.4 Subspaces in the ΣDWH iteration

8.4.1 Permuted graph bases for general matrices

Looking at Lemma 8.8, we see that the factor R of the VR decomposition is in fact not referenced in order to rewrite part of the ΣDWH iteration. This suggests the idea to employ a well-conditioned basis of the subspace spanned by $\begin{bmatrix} \sqrt{c_k} X \\ I_n \end{bmatrix}$. The linear solve in one iteration step is not avoided completely but we hope to invert a better-conditioned matrix.

In the following we use $A \sim B$ to indicate that the columns of the two matrices A and B span the same subspace. A good candidate for providing a basis with desirable

properties are permuted graph bases. An n-dimensional subspace $\mathcal{U}$ is said to be represented in a *permuted graph basis* if

$$\mathcal{U} = \operatorname{colspan}\left(P^{\mathsf{T}} \begin{bmatrix} I_n \\ X \end{bmatrix}\right), \tag{8.20}$$

where P denotes a permutation. It is shown in [127] that for a given subspace, a matrix X and a permutation P exist, such that the entries of X are all smaller than 1. This leads to much better numerical properties when using this representation in numerical algorithms.

The computation of the entry-bound representations (8.20) is an NP-hard problem. However, [127] presents heuristic methods that compute representations, where for a given threshold value $\tau > 1$, $|x_{i,j}| < \tau$. This can be done with a reasonable amount of computational effort. In the worst case this is $\mathcal{O}(n^3 \log_\tau n)$ [147]. In practice, it is typically much lower, in particular when good starting guesses for P are available.

The following lemma is a reformulation of Lemma 8.8, where $M = \Sigma_m$ and $N = \Sigma_n$ are signature matrices and V is attained via Representation (8.20).

Lemma 8.10:
Let $\Sigma_m \in \mathbb{R}^{m\times m}$, $\Sigma_n \in \mathbb{R}^{n\times n}$ be signature matrices, For $X \in \mathbb{K}^{m\times n}$, $\eta \in \mathbb{K}$ let

$$\begin{bmatrix} I_n \\ \eta X \end{bmatrix} \sim V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = P^{\mathsf{T}} \begin{bmatrix} I \\ \hat{X} \end{bmatrix} \in \mathbb{K}^{(m+n)\times n},$$

where P is a permutation. Let

$$P \begin{bmatrix} \Sigma_n & \\ & \Sigma_m \end{bmatrix} P^{\mathsf{T}} = \begin{bmatrix} \hat{\Sigma}_n & \\ & \hat{\Sigma}_m \end{bmatrix}.$$

Then

$$\eta X(I + |\eta|^2 \Sigma_n X^* \Sigma_m X)^{-1} = V_2(\hat{\Sigma}_n + \hat{X}^* \hat{\Sigma}_m \hat{X})^{-1} V_1^* \Sigma_n. \qquad \Diamond$$

Proof. Let $\Sigma_2 := \begin{bmatrix} \Sigma_n & \\ & \Sigma_m \end{bmatrix}$. We follow the lines of the proof of Lemma 8.8 and observe

$$\begin{bmatrix} I \\ \eta X \end{bmatrix}^{\star_{\Sigma_2,\Sigma_n}} \begin{bmatrix} I \\ \eta X \end{bmatrix} = I + |\eta|^2 X^{\star_{\Sigma_m,\Sigma_n}} X.$$

As $\begin{bmatrix} I \\ \eta X \end{bmatrix}$ and V span the same subspace, there exists a nonsingular matrix R such that

$$\begin{bmatrix} I \\ \eta X \end{bmatrix} = VR.$$

Algorithm 8.1: Compute the generalized polar decomposition with respect to signature matrices, using permuted graph bases.

Data: $A \in \mathbb{K}^{m\times n}$, $\Sigma_m \in \mathbb{R}^{m\times m}$ $\Sigma_n \in \mathbb{R}^{n\times n}$: signature matrices, such that the canonical generalized polar decomposition of A exists (according to Theorem 8.5), s: estimate on $|\lambda_{\max}((\Sigma_n A^*\Sigma_m A)^{\frac{1}{2}})|^{-1}$,
ℓ: estimate on $s|\lambda_{\min}(\Sigma_n A^*\Sigma_m A)^{\frac{1}{2}}|$,
$\tau > 1$: threshold value for permuted graph basis.

Result: $A = WS$ is the canonical generalized polar decomposition with respect to Σ_m and Σ_n.

$W \leftarrow sA$

for $k = 1, 2, \ldots$ **do**

  Compute weighting parameters a, b, c and update ℓ from equations (8.6) and (8.7).

  Compute entry-bound permuted graph basis of $\operatorname{colspan}\left(\begin{bmatrix} I \\ \sqrt{c}W \end{bmatrix}\right)$, i.e.

$$\begin{bmatrix} I \\ \sqrt{c}W \end{bmatrix} \sim P^{\mathsf{T}} \begin{bmatrix} I \\ \hat{W} \end{bmatrix} =: \begin{bmatrix} V_1 \\ V_2 \end{bmatrix},$$
$$|\hat{W}_{ij}| < \tau \text{ for } i \in \{1, \ldots, m\},\ j \in \{1, \ldots, n\}.$$

  $\begin{bmatrix} \hat{\Sigma}_n & \\ & \hat{\Sigma}_m \end{bmatrix} \leftarrow P \begin{bmatrix} \Sigma_n & \\ & \Sigma_m \end{bmatrix} P^{\mathsf{T}}$

  Compute LDL^{T} factorization $\hat{\Sigma}_n + \hat{W}^*\hat{\Sigma}_m\hat{W} = PLDL^*P^{\mathsf{T}}$.

  $W \leftarrow \frac{b}{c}W + (a - \frac{b}{c})V_2 PL^{-*}D^{-1}L^{-1}P^{\mathsf{T}}V_1^*\Sigma_n$

Compute pseudosymmetric factor and ensure pseudosymmetry numerically:
$S \leftarrow \Sigma_n W^*\Sigma_m A, \qquad S \leftarrow (S + \Sigma_n S^*\Sigma_n)/2$

Exactly as in the proof of Lemma 8.8 (with the roles of V_1 and V_2 switched) it can be shown that

$$\begin{aligned} \eta X(I + |\eta|^2\Sigma_n X^*\Sigma_m X)^{-1} &= V_2(V^{\star_{\Sigma_2,\Sigma_n}}V)^{-1}V_1^{\star_{\Sigma_n}} \\ &= V_2(\Sigma_n \begin{bmatrix} I & \hat{X}^* \end{bmatrix} P\Sigma_2 P^{\mathsf{T}} \begin{bmatrix} I \\ \hat{X} \end{bmatrix})^{-1}\Sigma_n V_1^*\Sigma_n \\ &= V_2(\hat{\Sigma}_n + \hat{X}^*\hat{\Sigma}_m\hat{X})^{-1}V_1^*\Sigma_n. \end{aligned}$$ □

Algorithm 8.1 presents the details on how permuted graph bases can be used in the computation of generalized polar decomposition via the DWH iteration.

8.4.2 Permuted Lagrangian graph bases for pseudosymmetric matrices

As pointed out in Section 8.1, we are in particular interested in computing the generalized polar decomposition (with respect to a signature matrix) of pseudosymmetric matrices. A way to exploit this structure in the iteration can be found by considering Lagrangian subspaces, to which pseudosymmetric matrices can be linked.

Definition 8.11:
A subspace

$$\mathcal{U} = \operatorname{colspan}(U), \quad U \in \mathbb{K}^{2n\times n},$$

is called *Lagrangian* if it holds $U^*JU = 0$, where $J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$. ◊

A Lagrangian subspace can be represented by a *permuted Lagrangian graph basis*

$$\mathcal{U} = \operatorname{colspan}\left(\Pi^{\mathsf{T}} \begin{bmatrix} I \\ X \end{bmatrix}\right), \tag{8.21}$$

where $X = X^*$. Here, Π denotes a symplectic swap matrix [29]. A symplectic swap matrix is defined by a Boolean vector $v \in \{0,1\}^n$ and its complement $\hat{v} \in \{0,1\}^n$, $\hat{v}_i = 1 - v_i$. The corresponding symplectic swap matrix is defined as

$$\Pi_v = \begin{bmatrix} \operatorname{diag}(v) & \operatorname{diag}(\hat{v}) \\ -\operatorname{diag}(\hat{v}) & \operatorname{diag}(v) \end{bmatrix}. \tag{8.22}$$

It is shown in [127] that each Lagrangian subspace admits a representation (8.21), where X has no entries larger than $\sqrt{2}$.

As for general subspaces, there exist heuristics for computing a basis, such that the entries of X are bounded, within a reasonable amount of time. In this case $|x_{i,j}| < \tau$, where $\tau > \sqrt{2}$ is a given threshold value.

A Lagrangian subspace could of course be treated as a general subspace and admits a representation (8.20), with even smaller entries than in Representation (8.21). However, the structural property, i.e. the subspace being Lagrangian, is not encoded anymore in this representation. It is encoded in the symmetry of X, which can easily be enforced and preserved in the course of computations. This has numerical benefits, which typically outweigh the slightly larger entries in X.

The following lemma draws a connection between self-adjoint matrices and Lagrangian subspaces.

Lemma 8.12:
Let $M \in \mathbb{K}^{n\times n}$, $M = M^*$ be a nonsingular matrix. Let $X \in \mathbb{K}^{n\times n}$ be self-adjoint with respect to the scalar product induced by M. Then $\begin{bmatrix} M \\ X \end{bmatrix}$ spans a Lagrangian subspace. ◊

Proof. We have

$$\begin{bmatrix} M \\ X \end{bmatrix}^* \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix} \begin{bmatrix} M \\ X \end{bmatrix} = M^*X - X^*M = 0,$$

where we have used that $MX = X^*M$ is equivalent to X being self-adjoint with respect to M. □

The following lemma is a variant of Lemma 8.8 applied to square matrices, where the positions of the two matrix blocks are switched. The goal is to get to a formulation in which the subspace given in Lemma 8.12 appears.

Lemma 8.13:
Let $M, N \in \mathbb{K}^{n\times n}$ be nonsingular, N be M-orthogonal, i.e. $N^{\star_M}N = I$. $M_2 := \begin{bmatrix} M & \\ & M \end{bmatrix}$, $X \in \mathbb{K}^{n\times n}$. Let $\begin{bmatrix} N \\ \eta X \end{bmatrix} = VR$ with $V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \in \mathbb{K}^{2n\times n}$, $R \in \mathbb{K}^{n\times n}$ nonsingular be a decomposition. Then

$$\eta X(I + |\eta|^2 X^{\star_M} X)^{-1} = V_2(V^{\star_{M_2,M}} V)^{-1} V_1^{\star_M} N. \qquad \Diamond$$

Proof. We observe

$$\begin{bmatrix} N \\ \eta X \end{bmatrix}^{\star_{M_2,M}} \begin{bmatrix} N \\ \eta X \end{bmatrix} = N^{\star_M}N + |\eta|^2 X^{\star_M} X = I + |\eta|^2 X^{\star_M} X.$$

Following the proof of Lemma 8.8, we get

$$\eta X(I + |\eta|^2 X^{\star_M} X)^{-1} = V_2(V^{\star_{M_2,M}} V)^{-1}(R^{-1})^{\star_M} = V_2(V^{\star_{M_2,M}} V)^{-1} V_1^{\star_M} N.$$

In the last step we used $R^{-1} = N^{-1}V_1 = N^{\star_M}V_1$. □

Let us go back to the specific case of a scalar product induced by a signature matrix, i.e. $M := \Sigma$. In this case, Lemma 8.12 and Lemma 8.13 come together. Σ is symmetric, so Lemma 8.12 holds. So does Lemma 8.13 by setting $N := \Sigma$. The subspace in question can be represented by permuted Lagrangian graph bases. The situation is summarized in the following lemma.

Lemma 8.14:
Let $\Sigma \in \mathbb{R}^{n\times n}$ be a signature matrix. $\Sigma_2 := \begin{bmatrix} \Sigma & \\ & \Sigma \end{bmatrix}$, $X \in \mathbb{K}^{n\times n}$ be self-adjoint with respect to the scalar product induced by Σ, $\eta \in \mathbb{K}$. Let

$$\begin{bmatrix} \Sigma \\ \eta X \end{bmatrix} \sim \Pi^{\mathsf{T}} \begin{bmatrix} I \\ \hat{X} \end{bmatrix} =: V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

be a permuted Lagrangian graph basis, i.e. Π is a symplectic swap matrix and $\hat{X} = \hat{X}^*$. Then

$$\eta X(I + |\eta|^2 \Sigma X^* \Sigma X)^{-1} = V_2(\Sigma + \hat{X}\Sigma\hat{X})^{-1} V_1^*. \qquad \Diamond$$

Proof. Note that

$$V^{\star_{\Sigma_2,\Sigma}} V = \Sigma \begin{bmatrix} I_{2n} & \hat{X}^\mathsf{T} \end{bmatrix} \Pi \Sigma_2 \Pi^\mathsf{T} \begin{bmatrix} I_{2n} \\ \hat{X} \end{bmatrix} = I_{2n} + \Sigma \hat{X}^* \Sigma \hat{X} = I_{2n} + \Sigma \hat{X} \Sigma \hat{X}.$$

We have used $\Pi \Sigma_2 \Pi^\mathsf{T} = \Sigma_2$, which holds because $\Pi = \begin{bmatrix} V & \hat{V} \\ -\hat{V} & V \end{bmatrix}$ is a symplectic swap matrix as given in equation (8.22):

$$\Pi \Sigma_2 \Pi^\mathsf{T} = \begin{bmatrix} V & \hat{V} \\ -\hat{V} & V \end{bmatrix} \begin{bmatrix} \Sigma & \\ & \Sigma \end{bmatrix} \begin{bmatrix} V & \hat{V} \\ -\hat{V} & V \end{bmatrix}^\mathsf{T} = \begin{bmatrix} V\Sigma V + \hat{V}\Sigma\hat{V} & -V\Sigma\hat{V} + \hat{V}\Sigma V \\ -\hat{V}\Sigma V + V\Sigma\hat{V} & \hat{V}\Sigma\hat{V} + V\Sigma V \end{bmatrix} = \Sigma_2.$$

$V\Sigma V + \hat{V}\Sigma\hat{V} = \Sigma$ and $-V\Sigma\hat{V} + \hat{V}\Sigma V = 0$ hold because V and $\hat{V}$ pick up complementing rows and columns of Σ. Now applying Lemma 8.13 gives

$$\eta X (I + |\eta|^2 X^{\star_\Sigma} X)^{-1} = V_2 (\Sigma + \hat{X}^* \Sigma \hat{X})^{-1} V_1^*. \qquad \square$$

Algorithm 8.2 is a variant of Algorithm 8.1 using permuted Lagrangian graph bases. It computes the generalized polar decomposition of a pseudosymmetric matrix with respect to its defining signature matrix.

In the update step (Step 6 in Algorithm 8.1 and Step 5 in Algorithm 8.2), the structure of V_1 and V_2 should be taken into account for an efficient implementation. The rows of the identity matrix are distributed in V_1 and V_2 according to the permutation P or the symplectic swap Π. The remaining columns are given by $\hat{W}$. If this is taken care of, the matrix representing the subspace $V = \Pi^\mathsf{T} \begin{bmatrix} I \\ \hat{U} \end{bmatrix}$ never has to actually be formed. We can directly work on the matrices W and $\hat{W}$.

However, we may need to form a $n \times 2n$ matrix if a good starting guess for the permutation in the computation of the permuted graph basis is desired. For this task, a heuristic is proposed in [127] that includes a modified version of the QR factorization with column pivoting of an $n \times 2n$ matrix.

8.5 Numerical results

In this chapter, we have developed several variants of the ΣDWH iteration to compute the canonical generalized polar decomposition of a matrix with respect to signature matrices.

In general, the existence of the decomposition is not guaranteed, which is why we first examine pseudosymmetric matrices with respect to Σ. For these matrices, the generalized polar decomposition exists if and only if A has no purely imaginary eigenvalues (note that this is also required for $\operatorname{sign}(A)$ to exist, see Theorem 8.4). For randomly generated matrices, this is typically the case, which is why we observe convergence most times. Pseudosymmetric matrices represent an important class of matrices regarding the application potential of the developed methods, as pointed

Algorithm 8.2: Compute the generalized polar decomposition of a pseudosymmetric matrix with respect to a signature matrix, using permuted Lagrangian graph bases.

Data: Signature matrix $\Sigma \in \mathbb{K}^{n\times n}$, $A = \Sigma A^*\Sigma \in \mathbb{K}^{n\times n}$, such that A has no purely imaginary eigenvalues,
s: estimate on $|\lambda_{\max}((\Sigma A^*\Sigma A)^{\frac{1}{2}})|^{-1}$,
ℓ: estimate on $s|\lambda_{\min}(s(\Sigma A^*\Sigma A)^{\frac{1}{2}})|$,
$\tau > \sqrt{2}$: threshold value for permuted Lagrangian graph bases.

Result: $A = WS$ is the generalized polar decomposition with respect to Σ.

$W \leftarrow sA$

for $k = 1, 2, \ldots$ **do**

  Compute weighting parameters a, b, c and update ℓ from equations (8.6) and (8.7)

  Compute entry-bound permuted Lagrangian graph bases of $\operatorname{colspan}\left(\begin{bmatrix}\Sigma\\ \sqrt{c}W\end{bmatrix}\right)$, i.e.

$$\begin{bmatrix}\Sigma\\ \sqrt{c}W\end{bmatrix} \sim \Pi^{\mathsf{T}}\begin{bmatrix}I\\ \hat{W}\end{bmatrix} =: \begin{bmatrix}V_1\\ V_2\end{bmatrix}, \quad |\hat{W}_{ij}| < \tau \text{ for } i,j \in \{1,\ldots,n\}.$$

  Compute LDL^{T} factorization $\Sigma + \hat{W}^*\Sigma\hat{W} = PLDL^*P^{\mathsf{T}}$.

  $W \leftarrow \frac{b}{c}W + (a - \frac{b}{c})V_2PL^{-*}D^{-1}L^{-1}P^{\mathsf{T}}V_1^*$

Compute pseudosymmetric factor and ensure pseudosymmetry numerically:
$S \leftarrow \Sigma W^*\Sigma A, \qquad S \leftarrow (S + \Sigma S^*\Sigma)/2$

out in Section 8.1. For other matrices, which are not pseudosymmetric but yield a generalized polar decomposition with respect to Σ, similar results were observed in further tests. The experiments were performed on a laptop with an Intel® Core™ i7-8550U processor, running with 1.8 GHz on 4 cores, using MATLAB R2018a.

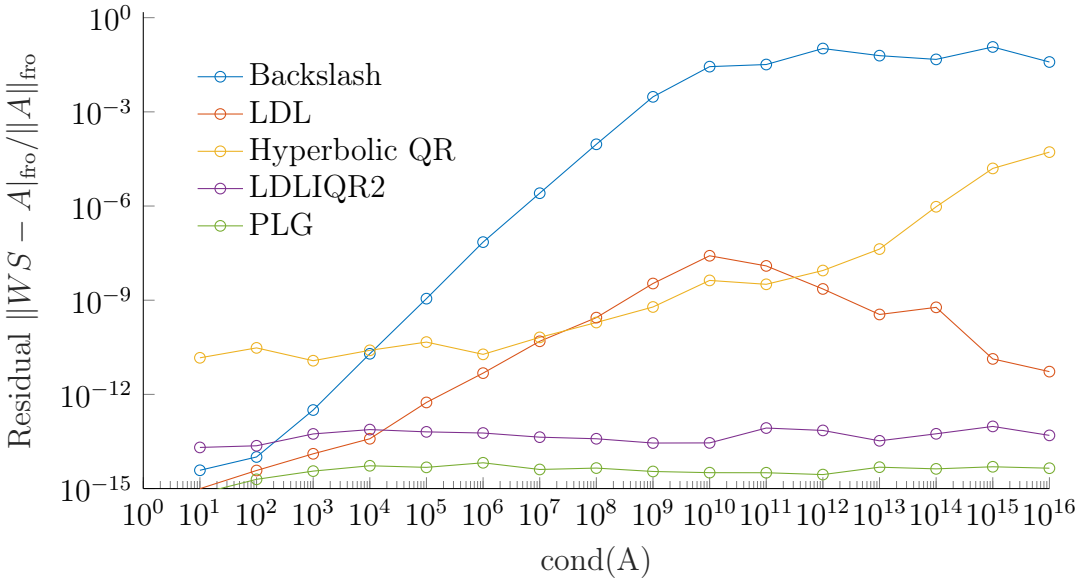
In light of the asymptotic cubic convergence of the iteration (see [95, Sec. 4.9.2]) we use the stopping criterion

$$\|X_{k+1} - X_k\|_F \leq (5\epsilon)^{\frac{1}{3}}, \tag{8.23}$$

where ϵ is the machine precision.

We take the same values for s and ℓ as in the QDWH algorithm [130], which are given in (8.16). As explained there, this makes sense for definite pseudosymmmetric matrices. The resulting convergence behavior is the same as in the standard setting. Further investigation of the convergence behavior is needed to devise sensible values for s and ℓ in the general case. Here the iteration may act on complex values. This consideration goes beyond the scope of this thesis. We use the same values as in the definite case even when they are not completely justified.

Figure 8.1: Residuals for different iterations for computing the generalized polar decomposition of pseudosymmetric matrices $A \in \mathbb{R}^{200\times 200}$ with a certain condition number. "Backslash" refers to the naive implementation, "LDL" refers to iteration (8.18), "Hyperbolic QR" and "LDLIQR2" refer to the variants of iteration (8.19). "PLG" refers to the variant using permuted Lagrangian graph bases described in Algorithm 8.2.



We first compare the algorithms in terms of their achieved residual for badly conditioned matrices. We consider square matrices and their generalized polar decomposition for a given signature matrix $\Sigma := \begin{bmatrix} I_n & \\ & -I_n \end{bmatrix}$ ($M = N = \Sigma$ in Definition 8.2).

Example 1 A real pseudosymmetric matrix with a condition number $\kappa = 10^k$ is generated as $A := \Sigma Q D Q^{\mathsf{T}}$. Q is a random orthogonal matrix (`orth(rand(2*n))`), and D is a diagonal matrix containing equally distributed values between 1 and 10^k, with alternating signs. A polar decomposition $A \approx WS$ is computed and the resulting residual $\|WS - A\|_F/\|A\|_F$ for matrices of size 200×200 ($n = 100$) is given in Figure 8.1. The residuals were averaged over 10 runs with different randomly generated matrices.

We see that a naive implementation of the ΣDWH iteration (8.17) leads to a highly unstable method. The accuracy improves as the iteration is rewritten to employ the LDL^{T} decomposition, see (8.18). This can be interpreted as exploiting structure that is hidden and ignored in the original formulation. As in the naive implementation, the accuracy deteriorates for ill-conditioned matrices. Surprisingly, for matrices with a condition number higher than 10^{11}, this trend is reversed and the method performs quite well for extremely ill-conditioned matrices. A possible explanation is that MAT-

LAB function `ldl` estimates the condition number of the input and acts differently, in our case preferably, for ill-conditioned matrices. The LDL^T-based iteration can be read as an iteration based on the indefinite QR decomposition, see Theorem 7.5 and iteration (8.19), that has been computed via the pivoted LDL^T decomposition. For computing a hyperbolic QR decomposition directly, using a column elimination approach, we used available MATLAB code [99], based on the works [12, 65, 92]. In our setting, this does not perform well. For well-conditioned matrices, this approach delivers the worst accuracy. For ill-conditioned matrices, it yields better results than the naive implementation, but is still highly dependent on the condition number. The two remaining methods use the indefinite QR decomposition via a double LDL^T decomposition (LDLIQR2) and permuted Lagrangian graph bases (PLG). These result in a high accuracy, which is independent of the condition number. For well-conditioned matrices, $LDLIQR2$ does not seem to be preferable, as it yields a higher residual than even the naive implementation. However, the residual stays at a consistently low order of magnitude as the condition number increases. Using PLGs consistently delivers the best results regarding accuracy, in the well-conditioned as well as in the ill-conditioned setting.

The disadvantage of the PLG approach is that it relies on very recently developed, fine-grained algorithms. Therefore, no optimized implementations are available yet and the runtimes resulting from a prototype MATLAB implementation are very high. Formulating the computation of PLGs in a way that exploits current computer architectures is a challenge not yet addressed. This method would need to be block-based in order to exploit the memory hierarchy, be parallelizable and avoid communication. The LDLIQR2 approach on the other side is easily implemented and only relies on the LDL^T factorization for which highly optimized implementations are available. However, both approaches rely on pivoting strategies, implying a considerable cost for communication if they are to be deployed in a massively parallel setup.

In a practical implementation, a combination of the LDL^T, LDLIQR2 and PLG approach should be considered, as it is possible for each iteration step to be performed by a different method. For badly conditioned matrices, the first steps could be performed via PLG. As soon as the condition number of the iterate has improved, another method could be employed, which shows better performance. The computation of PLGs is potentially very expensive, in particular for ill-conditioned matrices. The exact behaviour should be investigated in order to develop this method further.

We now compare the developed algorithms with other available methods, in particular concerning convergence properties. A standard approach for computing (generalized) polar decompositions, and a natural candidate to compare to our developed methods is the scaled Newton iteration (see e.g. [95]). Before moving to further numerical experiments, we describe it briefly in the following.

For a given signature matrix Σ, it is given as

$$X_{k+1} = \frac{1}{2}(\mu_k X_k + \mu_k^{-1}\Sigma X_k^{-*}\Sigma), \qquad X_0 = A. \tag{8.24}$$

It is called the Newton iteration as it represents the Newton method for solving $A^\star A =$

I. See also [94] for details. For the DWH iteration, we have shown in Corollary 8.6 that the iteration acts as a matrix sign function iteration on the self-adjoint factor of the decomposition. This observation also holds for the Newton iteration. Let $X_k = WS_k$ be a generalized polar decomposition of the iterate, then iteration (8.24) is equivalent to

$$X_{k+1} = W\left(\frac{1}{2}(\mu_k S_k + \mu_k^{-1} S_k^{-1})\right), \qquad X_0 = A.$$

The part in large parentheses is the Newton iteration for the matrix sign function acting on S_k. In the standard setting, the self-adjoint factor is Hermitian and its eigenvalues are real. This is exploited to devise scaled iterations which drive these values closer to one and therefore accelerate convergence (see [95, 61, 130]). For the generalized polar decomposition, the values are not necessarily real. In this case, we can fall back on scaling strategies for the matrix sign function which show good convergence properties in practice. In particular, we consider determinantal scaling [60], where

$$\mu_k := |\det S_k|^{-\frac{1}{n}} = |\det X_k|^{-\frac{1}{n}}.$$

The computation via the iterate X_k becomes possible because signature matrices and automorphisms with respect to them have a determinant of ± 1. Its computation is cheap as it can be computed from the diagonal values of the LU factorization, which is used to compute X_k^{-*}.

For the next numerical example, we generate matrices for which the generalized polar decomposition with respect to Σ is guaranteed to exist, but where the eigenvalues of the self-adjoint factor are all complex.

Example 2 For the generalized polar decomposition $A = WS$, we prescribe the self-adjoint factor S with a condition number $\kappa = 10^k$. The absolute values r_j of the eigenvalues $\lambda_j = r_j \exp(\mathrm{i}\phi_j)$ of H are uniformly distributed between $10^{-\lfloor k/2 \rfloor}$ and $10^{\lceil k/2 \rceil}$. ϕ_j is uniformly distributed between $-\pi/2$ and $\pi/2$, i.e. all eigenvalues lie in the right half plane. S is generated using two random orthogonal matrices $Q_1, Q_2 \in \mathbb{R}^{n\times n}$, $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}$,

$$S := Q^{\mathsf{T}} \begin{bmatrix} \mathrm{Re}(\lambda_1) & & & -\mathrm{Im}(\lambda_1) & & \\ & \ddots & & & \ddots & \\ & & \mathrm{Re}(\lambda_n) & & & -\mathrm{Im}(\lambda_n) \\ \mathrm{Im}(\lambda_1) & & & \mathrm{Re}(\lambda_1) & & \\ & \ddots & & & \ddots & \\ & & \mathrm{Im}(\lambda_n) & & & \mathrm{Re}(\lambda_n) \end{bmatrix} Q.$$

The polar factor W is prescribed as

$$W := \begin{bmatrix} Q_3 & \\ & Q_4 \end{bmatrix} \begin{bmatrix} C_W & S_W \\ S_W & C_W \end{bmatrix}.$$

Q_3 and Q_4 are random orthogonal matrices. The matrix $\begin{bmatrix} C_W & S_W \\ S_W & C_W \end{bmatrix}$ describes a series of hyperbolic Givens rotations, i.e.

$$C_W = \operatorname{diag}(\cosh\omega_1, \ldots, \cosh\omega_n)\,, \quad S_W = \operatorname{diag}(\sinh\omega_1, \ldots, \sinh\omega_n),$$

where $\omega_1, \ldots, \omega_n$ are uniformly distributed angles between 0 and $\frac{1}{4}\pi$. Averaged results for 20 matrices of size 200×200 ($n = 100$) are given in Table 8.1.

For the Newton iteration we use the stopping criterion given in [95, Chapter 8],

$$\|X_{k++1} - X_k\|_F \leqq (2\epsilon)^{\frac{1}{2}},$$

where ϵ denotes the machine precision.

For the ΣDWH iteration, we employ permuted graph bases (Algorithm 8.1), available in the `pgdoubling` package associated with [127]. It is compared to the Newton iteration with determinantal scaling (DN) and the Newton iteration with sub-optimal scaling [61] (SON). We generate 20 different random matrices and report the average number of iterations and the resulting residual $\|A - \tilde{W}\tilde{S}\|_F/\|A\|_F$, where $\tilde{W}$ and $\tilde{S}$ are the computed polar factors. We influence the condition number of A indirectly via $\kappa = \operatorname{cond}(S)$. It is about twice as high as κ because of the used hyperbolic Givens rotations.

In the standard setting, DWH and SON converge in 6 [130], respectively 9 [61], steps. Here, the iterations act as scalar iterations on the eigenvalues of the self-adjoint factor, who happen to be real in the standard case, but not in the indefinite setting. Still, we can observe that they converge significantly faster than the Newton iteration with determinantal scaling, in particular for ill-conditioned matrices. ΣDWH generally seems to need about 2/3 as many iteration steps as SON. Whether the cost per iteration is comparable, depends on the chosen implementation method for the DWH iteration. The simplest method is based on one LDL^T decomposition (8.18) and the main cost is a symmetric matrix inversion, just as in the Newton variants. If higher stability is needed in the case of badly conditioned matrices it can be obtained at the expense of a higher costs per iteration. This can be done by employing the LDLIQR2 iteration or by improving the corresponding subspace via Lagrangian graph bases (Algorithm 8.1).

ΣDWH displays the lowest backward error for the Σ-orthogonal factor W, which deteriorates for all methods as matrices become ill-conditioned. All methods yield a factor W that shows a good Σ-orthogonality. SON and ΣDWH both do a much better job than DN at recovering the self-adjoint factor S with backward errors of order 10^{-14} instead of 10^{-2}. DN and SON sometimes fail to converge for badly conditioned matrices.

We see that ΣDWH can compete with standard methods, even if no definite pseudosymmetric structure is given. Note that ΣDWH is the only one of the three methods that can directly be applied to non-square matrices, in order to compute the canonical generalized polar decomposition.

The results of Example 2 should be seen as preliminary, as the scaling factors and the stopping criterion (8.23) are not completely justified in the non-definite case. They

Table 8.1: Convergence behavior for different methods computing the generalized polar decomposition with respect to Σ of a 200×200 matrix (Example 2).

	κ	10	10^5	10^{10}	10^{15}
	cond(A)	2.15e+01	1.98e+05	1.98e+10	2.02e+15
# iterations	ΣDWH	8.70	9.70	10.65	10.60
	DN	12.30	20.00	32.95	44.13 [1]
	SON	14.05	15.45	16.45	16.74 [2]
residual	ΣDWH	5.06e-15	7.68e-15	9.88e-15	3.00e-15
	DN	2.98e-15	2.98e-15	2.93e-15	2.96e-15
	SON	3.00e-15	2.98e-15	2.96e-15	2.89e-15
rel. error W	ΣDWH	1.35e-14	9.45e-12	5.35e-08	8.01e-03
	DN	1.18e-14	3.96e-11	1.53e-06	7.83e-02
	SON	1.32e-14	3.12e-11	2.24e-07	5.22e-03
rel. error S	ΣDWH	1.05e-14	2.76e-14	3.51e-14	4.51e-14
	DN	9.98e-15	9.00e-12	8.52e-07	6.65e-02
	SON	1.43e-14	2.42e-14	2.33e-14	2.83e-14
$\|\Sigma W^{\mathsf{T}}\Sigma W - I\|_F$	ΣDWH	1.16e-15	1.23e-15	1.07e-15	1.25e-15
	DN	3.19e-15	3.19e-15	3.13e-15	3.12e-15
	SON	3.21e-15	3.18e-15	3.16e-15	3.09e-15

[1] 5 out of 20 runs did not converge. [2] 1 out of 20 runs did not converge.

do, however, motivate further research to devise iterations based on rational functions acting on complex values.

Example 3 We generate pseudosymmetric matrices as in Example 1, but additionally ensure the definiteness of ΣA by choosing only positive values for D. We compare the same methods as in Example 2 with respect to convergence properties. 20 matrices were generated and averaged results are reported in Table 8.2.

As expected, we see the convergence of ΣDWH and of the Newton iteration with sub optimal scaling within 6, respectively 9, iterations.

8.6 Using Zolotarev functions to accelerate the iteration

In this section we present an iteration that converges In even less steps than the DWH iteration in the case of definite pseudosymmetric matrices. To this aim, we generalize the ideas from [131], using Zolotarev functions as rational iterations.

Let $A \in \mathbb{K}^{n\times n}$ be a matrix, that is self adjoint with respect to a signature matrix Σ, such that $\Sigma A > 0$. Then A has only real eigenvalues (see Theorem 4.5). Let A be

Table 8.2: Convergence behavior for different methods computing the generalized polar decomposition with respect to Σ of definite pseudosymmetric matrices of size 200×200 (Example 3).

	κ	10	10^5	10^{10}	10^{15}
# iterations	ΣDWH	4.00	5.00	6.00	6.00
	DN	6.00	15.10	30.50	44.50
	SON	6.00	7.00	8.00	9.00
residual	ΣDWH	1.38e-15	4.47e-14	2.34e-14	2.85e-14
	DN	8.11e-16	2.46e-14	5.30e-14	1.05e-14
	SON	8.14e-16	3.20e-14	3.03e-14	1.04e-14
$\|\Sigma W^{\mathsf{T}}\Sigma W - I\|_F$	ΣDWH	1.26e-15	1.95e-13	2.03e-13	6.92e-14
	DN	7.31e-16	6.87e-14	5.66e-14	3.13e-14
	SON	7.16e-16	6.94e-14	5.64e-14	3.09e-14

scaled, such that its eigenvalues lie in $[-1, 1]$, and let $0 < \ell < |\lambda|$ for all $\lambda \in \Lambda(A)$, and let $A = WS$ be a generalized polar decomposition with respect to a signature matrix Σ. Then the eigenvalues of $S = (A^{\star_\Sigma} A)^{\frac{1}{2}}$ lie in $(\ell, 1]$.

According to Corollary 8.6, a rational iteration with starting point A acts as a rational scalar function on the eigenvalues of S, lying in $(\ell, 1]$. If they reach 1, the iteration has converged and arrived at the polar factor W.

The main idea explored in this chapter is the following. If a rational function approximates the scalar function sign(.) well on $(\ell, 1]$, it is a good candidate for the first iteration for computing the polar decomposition, because it drives all the eigenvalues of S closer to 1. The lower bound on the eigenvalues of the self-adjoint factor of the iterate, now called ℓ_1 also got moved closer to 1. The next iteration should be based on a good approximation of sign(.) on the much narrower interval $(\ell_1, 1]$. This way, convergence in just a few steps can be achieved.

Luckily, explicit formulas for rational best-approximations of the sign function on an interval $(\ell, 1]$ with the necessary form (8.11) where found by Zolotarev in 1877 [188]. In [131] Zolotarev functions are used to devise an iteration which computes the standard polar decomposition in just 2 steps. The algorithmic cost of the steps is increased compared to other iterative techniques, but the additional computations can be performed completely in parallel. We extend this approach for computing the polar decomposition of definite pseudosymmetric matrices.

We call the unique rational function of degree $(2r + 1, 2r)$ solving

$$\min_{R \in \mathcal{R}_{2r+1,2r}} \max_{x \in [-1,-\ell] \cup [\ell,1]} |\operatorname{sign}(x) - R(x)|$$

for a given $0 < \ell < 1$ and an integer r, *type* $(2r + 1, 2r)$ *Zolotarev function*. They are

given explicitly in form of

$$Z_{2r+1}(x;\ell) := Cx \prod_{j=1}^{r} \frac{x^2 + c_{2j}}{x^2 + c_{2j-1}}. \tag{8.25}$$

The coefficients $c_1, \ldots, c_{2r}$ are determined via the Jacobi elliptic functions $\mathrm{sn}(u;\ell)$ and $\mathrm{cn}(u;\ell)$ as

$$c_i = \ell^2 \frac{\mathrm{sn}^2(\frac{iK'}{2r+1};\ell')}{\mathrm{cn}^2(\frac{iK'}{2r+1};\ell')}, \qquad i = 1, \ldots 2r, \tag{8.26}$$

where $\ell' = \sqrt{1-\ell^2}$ and $K' = \int_0^{\pi/2} (1-(\ell')^2 \sin^2(\theta))^{-1/2} d\theta$ are familiar quantities in the context of Jacobi elliptic functions (see e.g. [5, Chapter 17], [7, Chapter 5]). Details on the computation of the coefficients can be found in [131]. $C > 0$ is a uniquely determined constant, which will later be substituted with a normalization constant $\hat{C}$. The authors of [131] provide MATLAB functions to compute the parameters in a stable fashion, which are used in our implementation.

Zolotarev also showed (see [8, Chapter 9], [146, Chapter 4]), that $Z_{2r+1}(x;\ell)$ solves

$$\max_{P,Q \in \mathcal{P}_r} \min_{\ell \le x \le 1} x \frac{P(x^2)}{Q(x^2)}.$$

For $r = 1$, this optimization problem was solved in [130], leading to the dynamically weighted Halley (DWH) iteration (see Section 8.2). This iteration was used in Section 8.3 to compute the generalized polar decomposition. An iteration based on Zolotarev functions therefore generalizes the DWH approach in terms of rational functions of higher degrees.

A key observation in [131] is that the composition of Zolotarev functions is again a Zolotarev function. More precisely, it holds

$$\hat{Z}_{2r+1}(\hat{Z}_{2r+1}(x;\ell);\ell_1) = \hat{Z}_{(2r+1)^2}(x;\ell),$$

where

$$\hat{Z}_{2r+1}(x;\ell) = \frac{Z_{2r+1}(x;\ell)}{Z_{2r+1}(1;\ell)} = \hat{C}x \prod_{j=1}^{r} \frac{x^2 + c_{2j}}{x^2 + c_{2j-1}}, \text{ with } \hat{C} = \prod_{j=1}^{r} \frac{1 + c_{2j-1}}{1 + c_{2j}}, \tag{8.27}$$

is a scaled Zolotarev function and $\ell_1 = \hat{Z}_{2r+1}(\ell;\ell)$. It can be verified that with $r := 8$ and $\ell \ge 10^{-16}$, it holds $Z_{(2r+1)^2}([\ell, 1], \ell]) \subseteq [1 - 10^{-15}, 1]$. Employing Corollary 8.6 for two steps on a matrix $A = WS$ with the rational iteration

$$g_1(x) = \hat{Z}_{2r+1}(x;\ell),$$
$$g_2(x) = \hat{Z}_{2r+1}(x;\ell_1),$$

we see that the eigenvalues of $g_2(g_1(S))$ are in the interval $[1 - 10^{-15}, 1]$, under the condition that all eigenvalues of S are smaller than 1 and larger than $\ell \ge 10^{-16}$. In

this sense, $g_2(g_1(A)) \approx W$ has converged to the polar factor W, after two iteration steps. Choosing a higher r, algorithms can be devised, that converge in just one step. It was argued in [131] that a 2-step approach is a sensible choice to acquire a robust algorithm. This way, potential instabilities, e.g. in the computation of the Zolotarev coefficients c_i, are forgivable.

Remark 8.15:
In the implementation of the Zolotarev iteration for a matrix A, first A is scaled, such that all its eigenvalues lie in $(-1, 1)$. Then ℓ should be a lower bound on the minimum absolute value in the spectrum of the scaled A, i.e.

$$\ell \lesssim \kappa_{\text{abs}}^{-1} := \frac{\min\{|\lambda| : \lambda \in \Lambda(A)\}}{\max\{|\lambda| : \lambda \in \Lambda(A)\}}.$$

Choosing $r := 8$ guarantees convergence in 2 steps, if $\kappa_{\text{abs}}^{-1} > \ell > 10^{-16}$, which can often be assumed. If κ_{abs} is known to be smaller, a larger ℓ and a smaller r can be chosen, while convergence is still guaranteed in two steps. Details are given in Table 1 in [131], where now κ_2 needs to be substituted with κ_{abs}. ◊

The scaled Zolotarev function can be represented in a partial fraction decomposition

$$\hat{Z}_{2r+1}(x;\ell) = \hat{C}x\left(1 + \sum_{j=1}^{r} \frac{a_j}{x^2 + c_{2j-1}}\right), \tag{8.28}$$

$$a_j = -\left(\prod_{k=1}^{r}(c_{2j-1} - c_{2k})\right) \cdot \left(\prod_{k=1,k\neq j}^{r}(c_{2j-1} - c_{2k-1})\right). \tag{8.29}$$

An iteration (8.11) derived from (8.28) takes the form

$$X_{k+1} = \hat{C}(X_k + \sum_{j=1}^{r} a_{j,k} X_k (X_k^{\star_M} X_k + c_{2j-1,k} I)^{-1}). \tag{8.30}$$

With $M = \Sigma$ as a signature matrix, iteration (8.30) becomes

$$X_{k+1} = \hat{C}(X_k + \sum_{j=1}^{r} a_{j,k} X_k (X_k^* \Sigma X + c_{2j-1,k}\Sigma)^{-1}\Sigma). \tag{8.31}$$

Computing the inverse via an LDL^{T} decomposition leads to a first practical iteration.

$$\begin{cases} Z_{2j-1,k} = (X_k^* \Sigma X_k + c_{2j-1,k}\Sigma), \quad [L_j, D_j, P_j] = \text{ldl}(Z_{2j-1,k}), \\ X_{k+1} = \hat{C}(X_k + \sum_{j=1}^{r} a_j X_k P_j L_j^{-*} D_j^{-1} L_j^{-1} P_j^{\mathsf{T}} \Sigma) \end{cases} \tag{8.32}$$

The first line of iteration (8.32) means that in iteration k, the LDL^T decomposition

$$Z_{2j-1,k} = P_j L_j D_j L_j^* P_j^\mathsf{T}$$

is computed for each $Z_{2j-1,k}$, $j = 1, \ldots, r$. P_j is a permutation matrix, L_j is lower triangular, and D_j is block-diagonal with 1×1 or 2×2 blocks.

In Section 8.3, the special case for $r = 1$ is derived. There, the iteration is rewritten in a way, such that it becomes inverse-free and a hyperbolic QR decomposition is employed instead. A special case of Lemma 8.8, in conjunction with Lemma 8.9 is given in the following lemma and can be used to rewrite iteration (8.31).

Lemma 8.16:
Let Σ be a signature matrix, $\eta \in \mathbb{R}$. For $X \in \mathbb{R}^{n\times n}$, let $\begin{bmatrix} \eta X \\ I \end{bmatrix} = HR$, $H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \in \mathbb{R}^{2n\times n}$, $R \in \mathbb{R}^{n\times n}$ be a decomposition, such that $H^* \begin{bmatrix} \Sigma & \\ & \Sigma \end{bmatrix} H = \hat{\Sigma}$, where $\hat{\Sigma} \in \mathbb{R}^{n\times n}$ is another signature matrix. Then

$$\eta X (I + |\eta|^2 X^{\star_\Sigma} X)^{-1} = H_1 \hat{\Sigma} H_2^* \Sigma. \qquad \lozenge$$

Using Lemma 8.16 with $\eta = \frac{1}{\sqrt{c_{2j-1,k}}}$, (8.31) can be rewritten as

$$\begin{cases} \begin{bmatrix} X_k \\ \sqrt{c_{2j-1,k}} I \end{bmatrix} = \begin{bmatrix} H_{1,j} \\ H_{2,j} \end{bmatrix} R_j, \text{ where } \begin{bmatrix} H_{1,j} \\ H_{2,j} \end{bmatrix}^* \begin{bmatrix} \Sigma & \\ & \Sigma \end{bmatrix} \begin{bmatrix} H_{1,j} \\ H_{2,j} \end{bmatrix} = \hat{\Sigma} \\ X_{k+1} = \hat{C}(X_k + \sum_{j=1}^r \frac{a_j}{\sqrt{c_{2j-1}}} H_{1,j} \hat{\Sigma} H_{2,j}^* \Sigma). \end{cases} \tag{8.33}$$

As in iteration (8.32), the first line refers to the computation of a total of r independent decompositions $\begin{bmatrix} X_k \\ \sqrt{c_{2j-1,k}} I \end{bmatrix} = H_j R_j$ for $j = 1, \ldots, r$, per iteration step. One way of acquiring the needed matrix H is the hyperbolic QR decomposition, which we introduced in Theorem 7.1. As in the ΣDWH algorithm presented in the previous sections, another possibility is the `LDLIQR2` algorithm presented in Section 7.2.4.

Algorithm 8.3 presents the pseudocode of a Zolotarev-based computation of the generalized polar factor, employing `LDLIQR2`. We assume that convergence is reached after just two steps, which are explicitly written in the algorithm. For the computation of iterate X_1, iteration (8.33) is employed. For the computation of the H matrices, we use Algorithm 7.1. We assume that this approach shows a better numerical stability than column-elimination based approaches, which was shown empirically in Section 8.5. Under this assumption, the second iterate X_2 can safely be computed using the LDL^T-based iteration (8.32). A detailed reasoning for the standard case is found in [131].

Algorithm 8.3 converges even for badly conditioned matrices. As explained in [131], for well-conditioned A, it is possible to skip the first iteration or choose a lower Zolotarev rank $r < 8$. In exact arithmetic the algorithm converges after 2 steps, in

Algorithm 8.3: Hyperbolic Zolo-PD for definite pseudosymmetric matrices.

Data: Signature matrix Σ $A \in \mathbb{C}^{n\times n}$ such that ΣA is Hermitian positive definite,
Zolotarev rank r according to Remark 8.15

Result: $S = \operatorname{sign}(A)$.

Estimate $\alpha \gtrsim \max\{|\lambda| : \lambda \in \Lambda(A)\}$, $\beta \lesssim \min\{|\lambda| : \lambda \in \Lambda(A)\}$.

$X_0 \leftarrow \frac{1}{\alpha}A$, $\ell \leftarrow \frac{\beta}{\alpha}$

`// First iteration:`

for $j = 1, \dots, 2r$ **do**

  $c_j \leftarrow \ell^2 \mathrm{sn}^2(\frac{\mathrm{i}K'}{2r+1}; \ell')/\mathrm{cn}^2(\frac{\mathrm{i}K'}{2r+1}; \ell')$ `// See` (8.26)

for $j = 1, \dots, r$ **do**

  $a_j = -\left(\prod_{k=1}^{r}(c_{2j-1} - c_{2k})\right) \cdot \left(\prod_{k=1,k\neq j}^{r}(c_{2j-1} - c_{2k-1})\right)$ `// See` (8.29)

$\hat{C} \leftarrow \prod_j^r \frac{1+c_{2j-1}}{1+c_{2j}}$ `// See` (8.27)

Compute X_1 according to (8.33), using Algorithm 7.1:

$$\begin{cases} \begin{bmatrix} X_0 \\ \sqrt{c_{2j-1}}I \end{bmatrix} = \begin{bmatrix} H_{1,j} \\ H_{2,j} \end{bmatrix} R_j, \text{ where } \begin{bmatrix} H_{1,j} \\ H_{2,j} \end{bmatrix}^* \begin{bmatrix} \Sigma & \\ & \Sigma \end{bmatrix} \begin{bmatrix} H_{1,j} \\ H_{2,j} \end{bmatrix} = \hat{\Sigma} \quad (j = 1, \dots, r) \\ X_1 \leftarrow \hat{C}(X_k + \sum_{j=1}^{r} \frac{a_j}{\sqrt{c_{2j-1}}} H_{1,j}\hat{\Sigma}H_{2,j}^*\Sigma) \end{cases}$$

$\ell \leftarrow \hat{C}\ell \prod_{j=1}^{r}(\ell^2 + c_{2j})/(\ell^2 + c_{2j-1})$

Repeat Step 3 to Step 7 to update c_j for $j = 1, \dots, 2r$, a_j for $j = 1, \dots, r$ and $\hat{C}$.

`// Second iteration:`

Compute X_2 according to (8.32):

$$\begin{cases} Z_{2j-1,1} = (X_1^*\Sigma X_1 + c_{2j-1,1}\Sigma), \quad [L_j, D_j, P_j] = \mathrm{ldl}(Z_{2j-1,1}), \quad (j = 1, \dots, r) \\ X_2 = \hat{C}(X_1 + \sum_{j=1}^{r} a_j X_1 P_j L_j^{-*} D_j^{-1} L_j^{-1} P_j^{\mathsf{T}}\Sigma) \end{cases}$$

if $\|X_2 - X_1\|_F / \|X_2\|_F \leq u^{1/(2r+1)}$ **then**

  $S \leftarrow X_2$

else

  $A \leftarrow X_2$, return to Step 1.

the sense presented in this section. As a safeguard for numerical errors we adopt the stopping criterion from [131], to guarantee convergence, using the known convergence rate of $2r + 1$.

Another possibility for computing an iterant X_k in a more stable fashion than the LDL^T-based iteration (8.32) was explored in Section 8.4 in the context of the ΣDWH iteration. Lemma 8.14 can be used to rewrite iteration (8.31), such that the inverse of a better conditioned matrix is computed. The main idea is to compute a well conditioned basis of the subspace $\begin{bmatrix} X_k \\ \sqrt{c_{2j-1}} I \end{bmatrix}$. This can be done by employing permuted Lagrangian graph bases described in [127]. This is an interesting path to explore, because the observed accuracy in Section 8.4 is even better than for the `LDLIQR2` algorithm. `LDLIQR2` yielded satisfactory results as well, i.e. the accuracy did not derail for badly conditioned matrices. The implementation uses available routines (i.e. the LDL^T factorization) and is more straight-forward. An implementation of the approach using permuted Lagrangian graph bases is left as future work.

Numerical results regarding the iteration based on Zolotarev function can be found in Chapter 9, where it is used in a structure-preserving divide-and-conquer method for definite pseudosymmetric eigenvalue problems.

8.7 Conclusions

In this chapter, we have presented a generalization of the QDWH method to compute the canonical generalized polar decomposition of a matrix with respect to a signature matrix Σ. One variant employs the hyperbolic QR decomposition as a tool. If Σ is chosen as the identity, this decomposition becomes the standard QR decomposition and can safely be computed with a column elimination approach, based on Householder transformations. This yields the well-known QDWH iteration.

Several options were provided on how to realize the iterations. Employing a computation of the hyperbolic QR decomposition based on column elimination forms the most natural generalization of QDWH. Other approaches, however, yield better results regarding stability. Methods based on LDL^T factorizations (LDLIQR2, Section 7.2.4) or on permuted (Lagrangian) graph bases (Algorithm 8.1 and 8.2) perform better in this regard.

Using these variants, a stability similar to Newton methods can be observed, but fewer iterations are needed. For the important class of definite pseudosymmetric matrices, the convergence behavior corresponds to the standard QDWH method. Convergence up to machine precision can be guaranteed in 6 steps for reasonably conditioned matrices.

The theoretical results we gave, in particular Lemma 8.8, provide a greater flexibility in the algorithmic design for DWH-based iterations, which might be utilized further than the scope of this thesis permits. Other methods for computing well-conditioned bases could also yield good results. Being more flexible in algorithmic design becomes increasingly important in view of modern computer architectures. In general, these

become more heterogeneous. They employ different levels of parallelism on various scales, have restrictions on available memory or use numerous accelerators and GPUs. Our framework provides the flexibility to find solutions, which could exploit the architecture at hand to its full potential.

Our main motivation came from computing the matrix sign function of large definite pseudosymmetric matrices. Here, the iteration acts as a rational function on what can be understood as generalized singular values. This allows the use of Zolotarev functions as best-approximations to the sign function of higher degree, yielding an iteration that converges in two steps. The individual steps take more work but are embarrassingly parallel and well-suited for large-scale high performance computations. In the field of computational quantum physics this is exactly what is needed, making this research direction promising.

In Section 8.6 we commented on the possibility to use permuted Lagrangian graph bases to improve the accuracy of the Zolotarev iteration for computing the matrix sign function. This direction could be explored further in the future. Further investigation should include a well-founded analysis on the stability of the proposed methods.

CHAPTER 9

A STRUCTURE-PRESERVING SPECTRAL DIVIDE-AND-CONQUER METHOD

Contents

9.1 Introduction

In this final chapter, we introduce a promising new class of algorithms for the structure-preserving solution of pseudosymmetric eigenvalue problems. The Bethe-Salpeter eigenvalue problems introduced Chapters 3 belong to this class of matrices. We have seen in Chapter 4 that one form of the BSE eigenvalue problem can be reduced to a product eigenvalue problem of half the original size. Algorithms based on this finding were developed in Chapter 6. If the definiteness property 6.2 holds, one further transformation yields a symmetric eigenvalue problem. If the definiteness property does not hold, one ends up with a pseudosymmetric problem, see Theorem 6.5. The algorithm presented in this chapter can therefore be a valuable tool applied directly on the BSE matrices, or on the reduced form, even if definiteness can not be guaranteed.

The necessary tools were developed in previous chapters. A central role is played by the generalized polar decomposition with respect to a signature matrix, which was explored in depth in Chapter 8. In the computation of the generalized polar decompositions, the GR decompositions studied in Chapter 7 played a vital role. In this chapter, they are used as tools once more in order to compute structure-preserving subspace bases.

The main idea of this chapter is to exploit the ability of the matrix sign function to find invariant subspaces, see Theorem 2.22. In the case of self-adjoint matrices, the matrix sign function coincides with the generalized orthogonal factor of a generalized polar decomposition, compare Theorem 8.4. The eigenvalue problem can be projected onto a subspace, leading to eigenvalue problems of smaller sizes. The method of doing this successively, until full diagonalization is achieved, is called *spectral divide-and-conquer* [17, 18, 36, 106]. The main challenge in this chapter is to devise such a method, which respects and preserves the given pseudosymmetric structure. This means that the smaller, projected eigenvalue problem should again be pseudosymmetric.

Given a diagonalizable matrix $A \in \mathbb{K}^{n\times n}$, $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$, we are interested in full diagonalization, i.e. finding $V \in \mathbb{K}^{n\times n}$, such that

$$V^{-1}AV = D. \tag{9.1}$$

For $\mathbb{K} = \mathbb{C}$, the matrix D is diagonal and contains the eigenvalues of A as diagonal values. For $\mathbb{K} = \mathbb{R}$, the matrix D is block diagonal with blocks of size 1×1, corresponding to real eigenvalues, or of size 2×2, corresponding to complex eigenvalues. The well-established standard approach for computing an eigenvalue decomposition (9.1) starts by computing the Schur decomposition of A

$$Q^*AQ = T,$$

where Q is orthogonal (or unitary) T is (block) upper triangular, via the QR algorithm [83]. The eigenvectors of T are computed via backward substitution or the eigenvectors of A are recovered via inverse iteration [13]. The QR algorithm, however, has proven difficult to parallelize and is not well-suited for computing only parts of the eigenvalue spectrum [17]. This is why spectral divide-and-conquer algorithms were explored as an alternative [17, 18, 19, 118]. They are based on the idea of spectral division. A matrix V is found such that

$$V^{-1}AV = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}. \tag{9.2}$$

This is achieved when the first columns of V form a basis of an invariant subspace of A and the remaining columns complement them to form a basis of $\mathbb{K}^n$. Now, the eigenvalue problems of the smaller matrices A_{11} and A_{22} are considered. Repeating this method recursively leads to a spectral divide-and-conquer scheme for the triangularization of a matrix.

The required subspace bases are acquired by employing the matrix sign function, which is computed via an iterative scheme [36, 106]. In general, the computational

effort of spectral divide-and-conquer methods is higher than that of QR based algorithms. This is why optimized implementations that exploit available parallelism, are needed, representing a high implementation effort.

For symmetric matrices, it is clear that symmetry must be exploited in order to formulate efficient eigenvalue algorithms. For spectral division, an orthogonal matrix V is sought that realizes block diagonalization instead of the block-triangularization in (9.2).

This is done in the spectral divide-and-conquer approach, as presented in [133]. For symmetric matrices, the computation of the matrix sign function can be parallelized particularly well [131], making it competitive with standard approaches in a high performance setting [116]. An important aspect is that spectral divide-and-conquer methods require less communication than QR based approaches. In recent years, efforts were increasingly directed towards finding communication-avoiding implementations of essential tools in numerical linear algebra [26]. Spectral divide-and-conquer methods can be implemented using these available building blocks [25]. On more advanced architectures, avoiding communication is more important than avoiding flops in order to minimize the runtime.

In this chapter, we show how the spectral divide-and-conquer approach is extended to solve eigenvalue problems of matrices with pseudosymmetric structure. In the following, statements are formulated for (pseudo-)symmetric matrices, but also hold to (pseudo)Hermitian matrices.

Efforts to exploit pseudosymmetric structure led to the development of the HR algorithm [55, 56, 49], which was already mentioned as a possibility to compute the structured decomposition in Theorem 6.4. It generalizes the symmetric QR algorithm and is motivated by the following observation. A generalized eigenvalue problem with symmetric matrices

$$Ax = \lambda Bx, \qquad A = A^{\mathsf{T}}, \quad B = B^{\mathsf{T}}, \tag{9.3}$$

where B is nonsingular, can be cast into a pseudosymmetric standard eigenvalue problem. Neither A nor B need to be positive definite. B has a decomposition $B = R^{\mathsf{T}}\Sigma R$, where Σ is a signature matrix, see Lemma 2.17. Then (9.3) is equivalent to

$$\Sigma R^{-\mathsf{T}} A R^{-1} y = \lambda y, \qquad y = Rx.$$

$\Sigma R^{-\mathsf{T}} A R^{-1}$ is clearly a pseudosymmetric matrix.

The QR algorithm computes its results with high accuracy because only (implicit) orthogonal transformations are involved. This is not true for the HR algorithm, which uses $(\Sigma, \hat{\Sigma})$-orthogonal matrices instead (also called pseudoorthogonal) [182]. A $(\Sigma, \hat{\Sigma})$-orthogonal matrix H, where Σ and $\hat{\Sigma}$ are two signature matrices, fulfills $H^{\mathsf{T}}\Sigma H = \hat{\Sigma}$. They are used to transform a matrix to upper triangular form, similar to the QR algorithm.

On top of that, the HR algorithm suffers from the same drawbacks as the QR algorithm in a high-performance environment: It is hard to parallelize and not communication-avoiding, as explained above.

The spectral divide-and-conquer method developed in this chapter presents a promising alternative. It can be parallelized and only relies on building blocks, for which communication-avoiding and well performing implementations exist. It can be used to compute only parts of the spectrum with reduced computational effort. Stability concerns are addressed by employing alternatives to or variants of the HR decomposition in the computation of the matrix sign function, presented in Chapter 8.

As stated before, the Bethe-Salpeter eigenvalue problems are the main motivation for developing the algorithm presented in this chapter. The definiteness property (4.3) plays a central role in most methods available in the literature, for example in the algorithm which was improved in Chapter 5. The new spectral divide-and-conquer approach in this chapter does not rely on this assumption. However, if the property holds, it can be exploited algorithmically. For these matrices, we showed in Chapter 8 that the used iterations have the same favorable convergence properties as in the symmetric setting and that an acceleration using Zolotarev function is possible. Furthermore, the first round of spectral division will be proven to decouple the problem into a positive and a negative definite symmetric matrix.

The remaining chapter is structured as follows. Section 9.2 explains the idea of spectral divide-and-conquer methods and presents a generalization of this approach for pseudosymmetric matrices. The acquisition of $(\Sigma, \hat{\Sigma})$-orthogonal representations of invariant subspaces is essential for structure preservation in the spectral division. In Section 9.3, we point out a link between QR decompositions of symmetric projection matrices (describing orthogonal projections) and Cholesky factorizations. This link exists analogously for pseudosymmetric projection matrices and the LDL^T factorization. We use this insight to compute required basis representations via the LDL^T factorization. Section 9.4 shows how the definiteness property

$$\Sigma A = (\Sigma A)^\mathsf{T} > 0 \tag{9.4}$$

is exploited in the presented algorithms. The computation of proper basis representations simplifies to a partial Cholesky factorization (Section 9.4.2). Section 9.5 presents the results of numerical experiments regarding the new method. Conclusions and further research directions are given in Section 9.6.

9.2 Structure-preserving divide-and-conquer methods

The property of the matrix sign function to acquire invariant subspaces was originally used to solve algebraic Riccati equations [149]. Later, it was used as a building block to devise parallelizable methods for eigenvalue computations of nonsymmetric matrices [17, 169]. In [133] a spectral divide-and-conquer algorithm for symmetric matrices is formulated, based on the relation between the matrix sign function and the polar decomposition. In this section, we generalize this approach to pseudosymmetric matrices, see Definition 2.5.

The definition is equivalent to ΣA (or $A\Sigma$) being symmetric. Essentially, a pseudosymmetric matrix is symmetric up to sign changes of certain rows (or columns). This definition is slightly different than the one given, e.g., in [117], as we allow any signature matrix and not just $\Sigma_{p,q} = \begin{bmatrix} I_p & \\ & -I_q \end{bmatrix}$.

In Section 9.2.1 we outline the general idea of spectral division, which reduces a large eigenvalue problem to two smaller ones. Recursively applying this technique yields parallelizable methods for acquiring all eigenvalues and eigenvectors. Section 9.2.2 recounts how a symmetric structure can be preserved in this context. The same line of thinking is applied to pseudosymmetric matrices in Section 9.2.3.

9.2.1 General spectral divide-and-conquer

It is a well-known concept to use invariant subspaces of a matrix to block-triangularize it with a similarity transformation. In the following, we focus on real matrices, but everything extends to complex matrices. For real matrices we end up with 2×2 matrix blocks on the diagonal for complex eigenvalues, whereas for complex matrices, 1×1 blocks suffice.

Theorem 9.1:
Let $A \in \mathbb{R}^{n \times n}$and $V_1 \in \mathbb{R}^{n \times k}$ be a basis for an invariant subspace of A and

$$V = \begin{bmatrix} V1 & V2 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

have full rank. Then

$$V^{-1}AV = \begin{bmatrix} A_{11} & A_{21} \\ 0 & A_{22} \end{bmatrix}, \qquad A_{11} \in \mathbb{R}^{k \times k},\ A_{22} \in \mathbb{R}^{(n-k)\times(n-k)}. \qquad \lozenge$$

Recursively applying the idea of Theorem 9.1 with shifts leads to a divide-and-conquer scheme, given in Algorithm 9.1.

This algorithm serves as a prototype for structure-preserving methods developed in the next subsections. The key idea is to choose the subspace basis in Step 4 in a way that preserves the structure in the spectral division.

9.2.2 Symmetric spectral divide-and-conquer

In this section we consider the symmetric eigenvalue problem, i.e. $A = A^\mathsf{T}$. A structure-preserving method requires the spectral division $V^{-1}AV$ to be symmetric. This is exactly fulfilled by orthogonal matrices, i.e. for matrices fulfilling $V^{-1} = V^\mathsf{T}$. A structure-preserving variant of Theorem 9.1 for symmetric matrices is given in the following.

Theorem 9.2:
Let $A = A^\mathsf{T} \in \mathbb{R}^{n \times n}$ and $V_1 \in \mathbb{R}^{n \times k}$ be a basis of an invariant subspace of A and

$$V = \begin{bmatrix} V1 & V2 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Algorithm 9.1: Unstructured spectral divide-and-conquer for block triangularization.

Data: $A \in \mathbb{R}^{n\times n}$.

Result: V, T such that $V^{-1}AV = T$ is block upper triangular.

1 Stop if A is of size 1×1 or 2×2 with a complex conjugate pair of eigenvalues.

2 Find shift σ such that $A - \sigma I$ has eigenvalues with positive and negative real part and no eigenvalues with zero real part.

3 Compute $S = \operatorname{sign}(A - \sigma I)$ via an iteration.

4 Get a basis V_+ of $\operatorname{range}(S + I)$ and V_- such that $V_0 = \begin{bmatrix} V_+ & V_- \end{bmatrix}$ has full rank. Then

$$V_0^{-1} A V_0 = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}.$$

5 Repeat spectral divide-and-conquer for A_{11}, i.e. find V_1 such that $V_1^{-1} A_{11} V_1 = T_{11}$ is block upper triangular.

6 Repeat spectral divide-and-conquer for A_{22}, i.e. find V_2 such that $V_2^{-1} A_{22} V_2 = T_{22}$ is block upper triangular.

7 $V \leftarrow V \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}$, $T \leftarrow \begin{bmatrix} T_{11} & V_1^{-1} A_{12} V_2 \\ 0 & T_{22} \end{bmatrix}$.

be orthogonal. Then

$$V^{-1}AV = V^{\mathsf{T}}AV = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix},$$
$$A_{11} = A_{11}^{\mathsf{T}} \in \mathbb{R}^{k\times k},\ A_{22} = A_{22}^{\mathsf{T}} \in \mathbb{R}^{(n-k)\times(n-k)}. \qquad \lozenge$$

The symmetric version of Algorithm 9.1 follows immediately as Algorithm 9.2. Due to the symmetry of A and by restricting the subspace basis to be orthogonal, this method can become highly viable. For symmetric A, $\operatorname{sign}(A)$ can be computed in a stable way via the QDWH iteration [130, 132] or the Zolotarev iteration [131]. The basis extraction can be done by performing a rank-revealing QR decomposition or a subspace iteration [133] if pivoting is considered too expensive.

9.2.3 Pseudosymmetric spectral divide-and-conquer

We now apply the line of thinking presented in Section 9.2.2 to pseudosymmetric matrices. The role of structure-preserving similarity transformations was played by orthogonal matrices in Section 9.2.2. For pseudosymmetric matrices this role is played by $(\Sigma, \hat{\Sigma})$-orthogonal matrices. A $(\Sigma, \hat{\Sigma})$-orthogonal matrix V is defined by

$$V^{\mathsf{T}}\Sigma V = \hat{\Sigma},$$

where Σ and $\hat{\Sigma}$ denote signature matrices in our setup. What $(\Sigma, \hat{\Sigma})$-orthogonal matrices have in common with orthogonal matrices is that their (pseudo-)inverses are

Algorithm 9.2: Symmetric spectral divide-and-conquer for block diagonalization.

Data: $A = A^{\mathsf{T}} \in \mathbb{R}^{n\times n}$.

Result: Orthogonal V, diagonal D such that $V^{\mathsf{T}}AV = D$.

1 Stop if A is of size 1×1.

2 Find shift σ such that $A - \sigma I$ has positive and negative eigenvalues and no zero eigenvalues.

3 Compute $S = \operatorname{sign}(A - \sigma I)$ via an iteration.

4 Get a basis V_+ of range$(S + I)$ and V_- such that $V_0 = \begin{bmatrix} V_+ & V_- \end{bmatrix}$ is orthogonal. Then

$$V_0^{\mathsf{T}} A V_0 = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}, \quad A_{11} = A_{11}^{\mathsf{T}},\ A_{22} = A_{22}^{\mathsf{T}}.$$

5 Repeat spectral divide-and-conquer for A_{11}, i.e. find V_1 such that $V_1^{\mathsf{T}} A_{11} V_1 = D_{11}$ is diagonal.

6 Repeat spectral divide-and-conquer for A_{22}, i.e. find V_2 such that $V_2^{\mathsf{T}} A_{22} V_2 = D_{22}$ is diagonal.

7 $V \leftarrow V_0 \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}$, $D \leftarrow \begin{bmatrix} D_{11} & 0 \\ 0 & D_{22} \end{bmatrix}$.

easily computed in form of

$$V^\dagger = \hat{\Sigma} V^{\mathsf{T}} \Sigma.$$

Lemma 9.3:
If $V \in \mathbb{R}^{m\times n}$ is a $(\Sigma, \hat{\Sigma})$-orthogonal matrix and $A \in \mathbb{R}^{m\times m}$ is pseudosymmetric with respect to Σ, i.e. $\Sigma A = A^T \Sigma$, then $\hat{A} = V^\dagger A V$ is pseudosymmetric with respect to $\hat{\Sigma}$, i.e. $\hat{\Sigma}\hat{A} = \hat{A}^T \hat{\Sigma}$. ◊

Proof. With $V^\dagger = \hat{\Sigma} V^T \Sigma$, $\Sigma V = (V^\dagger)^T \hat{\Sigma}$, $\Sigma^2 = I_m$ and $\hat{\Sigma}^2 = I_n$ we have

$$\hat{\Sigma}(V^\dagger A V) = V^{\mathsf{T}} \Sigma A V = V^{\mathsf{T}} A^{\mathsf{T}} \Sigma V = V^{\mathsf{T}} A^{\mathsf{T}} (V^\dagger)^{\mathsf{T}} \hat{\Sigma} = (V^\dagger A V)^{\mathsf{T}} \hat{\Sigma}. \qquad \square$$

$(\Sigma, \hat{\Sigma})$-orthogonal matrices already played a central role in the computation of the generalized polar decomposition, described in Chapter 8. In the resulting algorithms their purpose is therefore twofold: First they are needed in the iteration computing the matrix sign function, then they are needed to extract subspace bases that guarantee structure preservation.

Methods for computing these matrices include the HR decomposition [54] and methods described in Chapter 7. They prescribe Σ and yield $\hat{\Sigma}$ and the $(\Sigma, \hat{\Sigma})$-orthogonal matrix. We do not actually care about how $\hat{\Sigma}$ looks exactly, as long as it is a signature matrix. This way, pseudosymmetry as we defined it in Definition 2.5, not being bound

Algorithm 9.3: Pseudosymmetric spectral divide-and-conquer for block diagonalization.

Data: Signature matrix Σ, pseudosymmetric A with respect to Σ, i.e. $\Sigma A = (\Sigma A)^\mathsf{T}$.

Result: Signature matrix $\hat{\Sigma}$, and $(\Sigma, \hat{\Sigma})$-orthogonal V such that $V^\dagger A V = D$ is block diagonal with blocks no larger than 2×2.

1 Stop if A is of size 1×1 or 2×2 with a complex conjugate pair of eigenvalues.

2 Find shift σ such that $A - \sigma I$ has eigenvalues with positive and negative real part and no eigenvalues with zero real part.

3 Compute $S = \operatorname{sign}(A - \sigma I)$ via an iteration.

4 Get a basis V_+ of range$(S + I)$ and V_- such that $V_0 = \begin{bmatrix} V_+ & V_- \end{bmatrix}$ is (Σ, Σ_0)-orthogonal with $\Sigma_0 = \begin{bmatrix} \Sigma_+ & \\ & \Sigma_- \end{bmatrix}$. Then

$$V_0^\dagger A V_0 = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}, \quad \Sigma_+ A_{11} = (\Sigma_+ A_{11})^\mathsf{T}, \ \Sigma_- A_{22} = (\Sigma_- A_{22})^\mathsf{T}.$$

5 Repeat spectral divide-and-conquer for A_{11} with $\Sigma := \Sigma_+$, i.e. find (Σ_+, Σ_1)-orthogonal V_1 such that $V_1^\dagger A_{11} V_1 = D_{11}$ is diagonal.

6 Repeat spectral divide-and-conquer for A_{22} with $\Sigma := \Sigma_-$, i.e. find (Σ_-, Σ_2)-orthogonal V_2 such that $V_2^\dagger A_{22} V_2 = D_{22}$ is diagonal.

7 $V \leftarrow V_0 \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}$, $\hat{\Sigma} \leftarrow \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix}$, $D \leftarrow \begin{bmatrix} D_{11} & 0 \\ 0 & D_{22} \end{bmatrix}$.

to a specific Σ, is preserved. These kind of matrices, i.e. $(\Sigma, \hat{\Sigma})$-orthogonal matrices, where $\hat{\Sigma}$ does not matter, are sometimes called "hyperexchange" (e.g. in [179, 180]).

These observations can be used to formulate a pseudosymmetric variant of Algorithm 9.1, given in Algorithm 9.3. In this algorithm, the property preserved in the spectral division is the pseudosymmetry. This means, that Σ does not stay fixed, but is permuted and truncated in each division step.

9.3 Computing $(\Sigma, \hat{\Sigma})$-orthogonal representations of subspaces

Symmetric spectral divide-and-conquer methods rely on variants of the QR decomposition. The natural generalization in the indefinite context is the hyperbolic QR decomposition (see Theorem 7.1). Similar to the orthogonal QR decomposition, it can be computed by applying transformations that introduce zeros below the diagonal, column by column. Details are found in Section 7.2.2. In [163], the indefinite QR decomposition, see Theorem 7.5, is presented, which improves stability by allowing 2×2 blocks on the diagonal of R and additional pivoting. This variant can also be

computed via the LDL^T decomposition of $A^\mathsf{T}\Sigma A$; a link which was exploited in Chapter 7 and Chapter 8. There, the stability of computations is improved by applying this method twice.

In the context of this chapter we aim to compute the indefinite QR decomposition of a pseudosymmetric projection matrix. We will see that in this special case, an indefinite QR decomposition can be computed via the LDL^T decomposition without the need to form $A^\mathsf{T}\Sigma A$. The deterioration of accuracy associated with forming $A^\mathsf{T}\Sigma A$ is avoided and no "reothogonalization", as presented in Chapter 7, is needed.

We start with an observation regarding the symmetric divide-and-conquer method. Here, the matrix sign function computes a symmetric projection matrix, representing an orthogonal projection.

Lemma 9.4:
$P \in \mathbb{R}^{n\times n}$ is an orthogonal projection matrix, i.e. $P^2 = P$ and $P = P^\mathsf{T}$, with rank r. Let $R^T R = P$, $R \in \mathbb{R}^{r\times n}$ be a low-rank Cholesky factorization, where R has full row rank. Then R^T has orthogonal columns, i.e. $RR^\mathsf{T} = I_r$, and $R^T R = P$ is a thin QR decomposition of P. ◊

Proof. Because P is positive semi-definite, the low-rank Cholesky factorization exists [88]. From $P = P^2$ follows $R^\mathsf{T}R = R^\mathsf{T}RR^\mathsf{T}R$ and therefore $RR^\mathsf{T} = I_r$. □

Lemma 9.4 states that for projection matrices attained via the matrix sign function, the low-rank Cholesky and the thin QR decomposition are equivalent.

Let $P_+ = \frac{1}{2}(I_n + \operatorname{sign}(A))$ be the projection on the subspace of A associated with positive eigenvalues. The advantage of computing the (full) QR decomposition $\begin{bmatrix} Q_+ & Q_- \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}$ is that you immediately get a basis Q_- for the complementing subspace, associated with negative eigenvalues. The Cholesky factorization applied in the sense of Lemma 9.4 can only yield a thin QR decomposition. However, the same procedure can be applied on $P_- = \frac{1}{2}(I_n - \operatorname{sign}(A))$. The two thin QR decompositions can be combined to form a full one. Indeed, let Q_+ and Q_- be acquired from P_+ and P_- via Lemma 9.4. $Q_+^\mathsf{T}Q_+ = I$ and $Q_-^\mathsf{T}Q_- = I$ follow immediately from the orthogonality proven in the lemma. From $P_+ = Q_+Q_+^\mathsf{T}$ follows ${Q_+}^\mathsf{T} = Q_+^\mathsf{T}P_+$ and from $P_- = Q_-Q_-^\mathsf{T}$ follows $Q_- = P_-Q_-$. From the definition of the projectors in Lemma 2.22 we have $P_+P_- = 0$ and therefore $Q_+^\mathsf{T}Q_- = Q_+P_+^\mathsf{T}P_-Q_- = 0$.

The details are given in Algorithm 9.4. In step 3 we use the trace of a projection matrix to determine its rank.

In the symmetric context, computing the QR decomposition like this does not have an obvious benefit over computing a QR decomposition the standard way. It is even pointed out in [168], that this way of computing the relevant subspaces may cause numerical errors. With this notion in our minds, we hold back on fully recommending this method in the general case. However, we use it to investigate a new pathway in the indefinite context, to which it can be generalized. Here, an LDL^T decomposition can be used instead of a hyperbolic QR decomposition, which is a computational routine much more widely used. Mature algorithms and well-maintained implementations are

Algorithm 9.4: Compute orthogonal invariant subspace representations of a symmetric matrix via Cholesky.

Data: $A = A^{\mathsf{T}} \in \mathbb{R}^{n\times n}$ nonsingular.
Result: An orthogonal basis $Q = \begin{bmatrix} Q_+ & Q_- \end{bmatrix}$, where Q_+ is a basis of the invariant subspace of A associated with positive eigenvalues, Q_- is a basis of the invariant subspace of A associated with negative eigenvalues.

1 $S \leftarrow \operatorname{sign}(A)$.
2 $P_+ \leftarrow \frac{1}{2}(I_n + S)$.
3 Compute rank of P_+: $r_+ \leftarrow \operatorname{tr}(P_+)$.
4 $Q_{1,+} \leftarrow \texttt{chol}(P_+(1:r_+, 1:r_+))$.
5 $Q_+ \leftarrow \begin{bmatrix} Q_{1,+} \\ P_+(r_+ + 1:n, 1:r_+)Q_{1,+}^{-\mathsf{T}} \end{bmatrix}$.
6 $P_- \leftarrow \frac{1}{2}(I_n - S)$.
7 Compute rank of P_-: $r_- \leftarrow n - r_+$.
8 $Q_{1,-} \leftarrow \texttt{chol}(P_-(1:r_-, 1:r_-))$.
9 $Q_- \leftarrow \begin{bmatrix} Q_{1,-} \\ P_+(r_- + 1:n, 1:r_-)Q_{1,-}^{-\mathsf{T}} \end{bmatrix}$.

available and ready to use, e.g. in MATLAB as the command `ldl`. Details are given in the following lemma.

Lemma 9.5:
Σ is a given signature matrix, $P \in \mathbb{R}^{n\times n}$ is a projection matrix and pseudosymmetric with respect to Σ, i.e. $P^2 = P$ and $\Sigma P \Sigma = P^{\mathsf{T}}$, with rank r. Let $R^T\hat{\Sigma}R = \Sigma P$, $R \in \mathbb{R}^{r\times n}$ be a scaled low-rank LDL^{T} factorization, where R has full row rank and $\hat{\Sigma} \in \mathbb{R}^{r\times r}$ is another signature matrix. Then R^T is $(\Sigma, \hat{\Sigma})$- orthogonal,, i.e. $R\Sigma R^{\mathsf{T}} = \hat{\Sigma}$, and $HR = P$ with $H = \Sigma R^{\mathsf{T}}\hat{\Sigma}$ is a decomposition of P, where H is $(\Sigma, \hat{\Sigma})$-orthogonal.$\Diamond$

Proof. With $P = \Sigma R^{\mathsf{T}}\hat{\Sigma}R$ and $P = P^2$ it follows $\Sigma R^{\mathsf{T}}\hat{\Sigma}R = \Sigma R^{\mathsf{T}}\hat{\Sigma}R\Sigma R^{\mathsf{T}}\hat{\Sigma}R$ and therefore

$$\hat{\Sigma} = \hat{\Sigma}R\Sigma R^{\mathsf{T}}\hat{\Sigma}. \tag{9.5}$$

or equivalently

$$\hat{\Sigma} = R\Sigma R^{\mathsf{T}}.$$

We used $\hat{\Sigma}^2 = I_r$. Equation (9.5) is equivalent to $H := R^{\dagger} = \Sigma R^{\mathsf{T}}\hat{\Sigma}$ being $(\Sigma, \hat{\Sigma})$-orthogonal: $H^{\mathsf{T}}\Sigma H = \hat{\Sigma}$. So we have a decomposition $P = \Sigma R^{\mathsf{T}}\hat{\Sigma}R = HR$. □

If R in Lemma 9.5 is computed with the Bunch-Kaufman algorithm [51] (e.g. MATLAB `ldl`), it can be a permuted block-triangular matrix. Then $P = HR$ is an indefinite QR decomposition given in Theorem 7.5. The shape of R is not important in the given context as we are only interested in the subspace representation given by H.

Algorithm 9.5: Compute hyperbolic invariant subspace representations of a pseudosymmetric projection matrix via LDL^T.

Data: Signature matrix Σ, $A = \Sigma A^\mathsf{T} \Sigma \in \mathbb{R}^n$ nonsingular.
Result: A signature matrix $\hat{\Sigma}$, which is a permuted variant of Σ, a $(\Sigma, \hat{\Sigma})$-orthogonal basis $Q = \begin{bmatrix} Q_+ & Q_- \end{bmatrix}$, i.e. $Q^\mathsf{T} \Sigma Q = \hat{\Sigma}$, where Q_+ is a basis of the invariant subspace of A associated with positive eigenvalues, Q_- is a basis of the invariant subspace of A associated with negative eigenvalues.

1 $S \leftarrow \operatorname{sign}(A)$.
2 $P_+ \leftarrow \frac{1}{2}(I_n + S)$.
3 Compute rank of P_+, $r_+ \leftarrow \operatorname{tr}(P_+)$.
4 $[L_+, D_+] \leftarrow \texttt{ldl}(\Sigma P_+)$.
5 Diagonalize D_+ if it has blocks on the diagonal: $[V_+, D_+] \leftarrow \texttt{eig}(D_+)$, such that $D_+(1 : r_+, 1 : r_+)$ contains the nonzero diagonal values of D_+.
6 $R_+ \leftarrow (L_+ V_+(:, 1 : r_+) D_+(1 : r_+, 1 : r_+)^{\frac{1}{2}})^\mathsf{T}$, $\quad \hat{\Sigma}_+ \leftarrow \operatorname{sign}(D_+(1 : r_+, 1 : r_+))$.
7 $P_- \leftarrow \frac{1}{2}(I_n - S)$.
8 Compute rank of P_-, $r_- \leftarrow n - r_p$.
9 $[L_-, D_-] \leftarrow \texttt{ldl}(\Sigma P_-)$.
10 Diagonalize D_- if it has blocks on the diagonal: $[V_-, D_-] \leftarrow \texttt{eig}(D_-)$, such that $D_-(1 : r_-, 1 : r_-)$ contains the nonzero diagonal values of D_-.
11 $R_- \leftarrow (L_- V_-(:, 1 : r_-) D_-(1 : r_-, 1 : r_-)^{\frac{1}{2}})^\mathsf{T}$, $\quad \hat{\Sigma}_- \leftarrow \operatorname{sign}(D_-(1 : r_-, 1 : r_-))$.
12 $\hat{\Sigma} \leftarrow \operatorname{diag}(\Sigma_+, \Sigma_-)$
13 $Q_+ \leftarrow \Sigma R_+^\mathsf{T} \hat{\Sigma}$, $\quad Q_- \leftarrow \Sigma R_-^\mathsf{T} \hat{\Sigma}$.

The indefinite variant of Algorithm 9.4 is given in Algorithm 9.5. In contrast to the MATLAB `chol` command, the `ldl` command is not bothered by singular matrices, such as the given projectors. This is why steps 5 and 9 in Algorithm 9.4 do not have a correspondence in Algorithm 9.5. The Cholesky-based algorithm (Algorithm 9.4) computes the Cholesky factorization of the upper left block and expands it in order to get a low-rank version. The LDL^T-based algorithm (Algorithm 9.5) on the other hand computes an LDL^T decomposition of the whole matrix and truncates it in Steps 6 and 11. In a performance-aware implementation this approach may be reconsidered to decrease the number of necessary operations. Within this thesis we are mainly interested in a prototypical implementation. This way, we see if the approach is promising, possibly directing further research endeavors.

9.4 Definite pseudosymmetric matrices

In this section, we consider pseudosymmetric matrices with an additional property. We call a pseudosymmetric matrix A with respect to a signature matrix Σ *definite* if ΣA is positive definite. The matrices resulting from the Bethe-Salpeter equation

(BSE) approach in computational quantum physics often fulfill this property, specified in property (4.3).

9.4.1 Decoupling the indefinite eigenvalue problem into two symmetric definite problems

In the following, we explain how the spectral divide-and-conquer algorithm described in Section 9.2.3 simplifies for definite pseudosymmetric matrices. The problem is reduced to two Hermitian positive definite eigenvalue problems after just one spectral division step.

As a first result, we present the following theorem, clarifying the spectral structure of definite pseudosymmetric matrices. It is an extension of Theorem 4.5, additionally clarifying the structure of the eigenvectors, and a more general variant of Theorem 3 in [159]. Our version is independent of the additional structure of Bethe-Salpeter matrices given in (4.1) and (4.2). It can be proven in a similar fashion relying on the simultaneous diagonalization of ΣA and Σ.

Theorem 9.6:
Let $A \in \mathbb{K}^{n\times n}$ be a definite pseudosymmetric matrix with respect to Σ, where Σ has p positive and $n-p$ negative diagonal entries. Then A has only real, nonzero eigenvalues, of which p are positive and $n-p$ are negative. There is an eigenvalue decomposition

$$AV = V\Lambda, \qquad \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)\,,$$

where $\lambda_1, \ldots, \lambda_p > 0$, and $\lambda_{p+1}, \ldots, \lambda_n < 0$, such that

$$V^*\Sigma V = \begin{bmatrix} I_p & \\ & -I_{n-p} \end{bmatrix}. \tag{9.6}$$

♢

Proof. The first part of the proof is given in the proof of Theorem 4.5. We see, that there are eigenvectors of A, given as a matrix X, that fulfill

$$X^{\mathsf{H}}\Sigma X = \Lambda, \tag{9.7}$$

where Λ contains the eigenvalues of A. They are all real and nonzero. The columns of X can be arranged such that the positive eigenvalues are given in the upper left part of Λ and the negative ones are given in the lower right part. It is left to observe that due to (9.7), X can be scaled in form of $V := X|\Lambda|^{-\frac{1}{2}}$, where $|.|$ denotes entry-wise absolute values, such that (9.6) holds. □

For pseudosymmetric matrices that are definite, the structure-preserving spectral divide-and-conquer algorithm (Algorithm 9.3) shows a special behaviour we exploit. Generally, after one step of spectral division, we get two smaller matrices that are pseudosymmetric with respect to two submatrices of the original signature matrix Σ, denoted Σ_+ and Σ_- in Algorithm 9.3. The p positive and the $n-p$ negative

values on the diagonal of Σ split up in an unpredictable way. For definite matrices, they split up neatly: The positive values gather in $\Sigma_+ = I_p$ and the negative values gather in $\Sigma_- = -I_{n-p}$. After spectral division, the upper left block A_{11} is definite pseudosymmetric with respect to I_p, i.e. symmetric positive definite. The lower right block A_{22} is definite pseudosymmetric with respect to $-I_{n-p}$, i.e. symmetric negative definite. The theory explaining this behavior is given in the following theorem.

Theorem 9.7:
Let $A \in \mathbb{K}^{n\times n}$ be a definite pseudosymmetric matrix with respect to Σ with p positive and $n-p$ negative diagonal values. Let H be a basis of the invariant subspace of A associated with the p positive (respectively $n-p$ negative) eigenvalues, such that $H^*\Sigma H = \hat{\Sigma}$, where $\hat{\Sigma}$ is another signature matrix. Then $H^\dagger AH$ is Hermitian positive (respectively negative) definite and $\hat{\Sigma} = I_p$ (respectively $\hat{\Sigma} = -I_{n-p}$). ◊

Proof. We first show that $\hat{\Sigma} = I_p$. Let $AV = V\Lambda$ be the eigenvalue decomposition given in Theorem 9.6. Let $V_p = \begin{bmatrix} v_1 & \dots v_p \end{bmatrix}$ denote the first p columns of V, associated with the positive eigenvalues $\Lambda_+ = \operatorname{diag}(\lambda_1, \dots, \lambda_p)$. Then $AV_+ = V_+\Lambda_+$ and

$$V_+^*\Sigma V_+ = I_p. \tag{9.8}$$

As H spans the same subspace as V_+ there must be $X \in \mathbb{K}^{p\times p}$ such that $H = V_+X$. Then $H^*\Sigma H = X^*V_+^*\Sigma V_+X = X^*X \geq 0$. The only signature matrix with this property is the identity. Then it holds $H^\dagger = H^*\Sigma$ and therefore

$$H^\dagger AH = H^*\Sigma AH$$

is Hermitian positive definite, as ΣA is Hermitian positive definite. Concerning the negative eigenvalues it can be shown, that $\hat{\Sigma} = -I_{n-p}$ and therefore

$$H^\dagger AH = -H^*\Sigma AH$$

is Hermitian negative definite. □

Theorem 9.7 greatly simplifies the divide-and-conquer method for definite pseudosymmetric matrices (Algorithm 9.3). We only need one spectral division step and can then fall back on existing algorithms for symmetric positive matrices. They can be of the divide-and-conquer variety, e.g. developed in [133], but do not have to be. In a high-performance setting, parallelized algorithms implemented in libraries such as ELPA [119] can be used.

9.4.2 Computing $(\Sigma, \hat{\Sigma})$-orthogonal representations

The computation of pseudoorthogonal subspace representations described in Section 9.3 also simplifies. In step 6 of Algorithm 9.5 the smaller signature matrix Σ_+ related to the subspace associated with positive eigenvalues, is computed by taking the signs of the diagonal matrix D of the previously computed LDL^T decomposition. Because

Algorithm 9.6: Compute $(\Sigma, \hat{\Sigma})$-orthogonal invariant subspace representations of a definite pseudosymmetric projection matrix via Cholesky.

Data: Signature matrix Σ with r_+ positive and r_- negative diagonal values, $A \in \mathbb{R}^n$, such that ΣA is symmetric positive definite.

Result: A $(\Sigma, \hat{\Sigma})$-orthogonal basis $Q = \begin{bmatrix} Q_+ & Q_- \end{bmatrix}$, where $\hat{\Sigma} = \operatorname{diag}(I_{r_+}, -I_{r_-})$, i.e. $Q^\mathsf{T} \Sigma Q = \hat{\Sigma}$, where Q_+ is a basis of the invariant subspace of A associated with positive eigenvalues, Q_- is a basis of the invariant subspace of A associated with negative eigenvalues.

1 $S \leftarrow \operatorname{sign}(A)$.
2 $P_+ \leftarrow \frac{1}{2}(I_n + S)$.
3 $Q_{1,+} \leftarrow \texttt{chol}(\Sigma(1:r_+, 1:r_+) P_+(1:r_+, 1:r_+))$.
4 $Q_+ \leftarrow \begin{bmatrix} \Sigma(1:r_+, 1:r_+) Q_{1,+}^\mathsf{H} \\ P_+(r_+ + 1:n, 1:r_+) Q_{1,+}^{-1} \end{bmatrix}$.
5 $P_- \leftarrow \frac{1}{2}(I_n - S)$.
6 $Q_{1,-} \leftarrow \texttt{chol}(-\Sigma(1:r_-, 1:r_-) P_-(1:r_-, 1:r_-))$.
7 $Q_- \leftarrow \begin{bmatrix} \Sigma(1:r_-, 1:r_-) Q_{1,-}^\mathsf{H} \\ P_-(r_- + 1:n, 1:r_-) Q_{1,-}^{-1} \end{bmatrix}$.

of Theorem 9.7 we know, that $\Sigma_+ = I_p$. The LDL^T decomposition was taken of ΣP_+, which hence must be positive semidefinite. Therefore, the LDL^T decomposition can be substituted by a low-rank Cholesky factorization, similar to the symmetric case described in Algorithm 9.4. The computation of the rank (Step 3 in Algorithm 9.5) is omitted because we know that A has as many positive eigenvalues as Σ has positive diagonal values according to Theorem 9.6. The operations are summarized in Algorithm 9.6.

Numerical experiments (in particular examples from electronic structure theory, presented in Section 9.5.2) show that Algorithm 9.6 can break down due to roundoff errors in floating point arithmetic. This happens, when numerical errors lead to ΣP_+ having negative eigenvalues or ΣP_- having positive eigenvalues, such that the Cholesky decomposition breaks down. In order to intercept this case, we implement a more robust variant based on a truncated LDL^T decompositions given in Algorithm 9.7.

9.5 Numerical experiments

In this section we apply one step of spectral divide-and-conquer (Algorithm 9.3) on definite pseudosymmetric matrices. The matrix sign function is computed by the hyperbolic Zolo-PD algorithm (Algorithm 8.3), or algorithms based on the ΣDWH iteration presented in Chapter 8 or a Newton iteration with suboptimal scaling presented in [61]. We expect these algorithms to show the same convergence properties as in the symmetric case, due to Corollary 8.6. Zolo-PD should converge in 2 steps, ΣDWH in 6 steps and Newton in 9 steps. We compute $(\Sigma, \hat{\Sigma})$-orthogonal subspace

Algorithm 9.7: Robust computation of $(\Sigma, \hat{\Sigma})$-orthogonal invariant subspace representations of a definite pseudosymmetric projection matrix via LDL^{T}.

Data: Signature matrix Σ with r_+ positive and r_- negative diagonal values, $A \in \mathbb{R}^n$, such that ΣA is symmetric positive definite.

Result: A$(\Sigma, \hat{\Sigma})$-orthogonal basis $Q = \begin{bmatrix} Q_+ & Q_- \end{bmatrix}$, where $\hat{\Sigma} = \operatorname{diag}\left(I_{r_+}, -I_{r_-}\right)$, i.e. $Q^{\mathsf{T}}\Sigma Q = \hat{\Sigma}$, where Q_+ is a basis of the invariant subspace of A associated with positive eigenvalues, Q_- is a basis of the invariant subspace of A associated with negative eigenvalues.

1 $S \leftarrow \operatorname{sign}(A)$.
2 $P_+ \leftarrow \frac{1}{2}(I_n + S)$.
3 $[L_+, D_+] \leftarrow \texttt{ldl}(\Sigma P_+)$.
4 Diagonalize D_+ if it has blocks on the diagonal: $[V_+, D_+] \leftarrow \texttt{eig}(D_+)$, such that the diagonal entries of D_+ are given in descending order.
5 $Q_+ \leftarrow \Sigma L_+ V_+(:, 1 : r_+) D_+^{\frac{1}{2}}(1 : r_+, 1 : r_+)$.
6 $P_- \leftarrow \frac{1}{2}(I_n - S)$.
7 $[L_-, D_-] \leftarrow \texttt{ldl}(-\Sigma P_-)$.
8 Diagonalize D_- if it has blocks on the diagonal: $[V_-, D_-] \leftarrow \texttt{eig}(D_-)$, such that the diagonal entries of D_- are given in descending order.
9 $Q_- \leftarrow \Sigma L_- V_-(:, 1 : r_-) D_-^{\frac{1}{2}}(1 : r_-, 1 : r_-)$.

representations used in the spectral division, We use Algorithm 9.6 in Examples 1 and 2 and compare Algorithm 9.6 and 9.7 in Examples 3 and 4. The experiments were performed on a laptop with an Intel® Core™ i7-8550U processor, running with 1.8 GHz on 4 cores, using MATLAB R2018a.

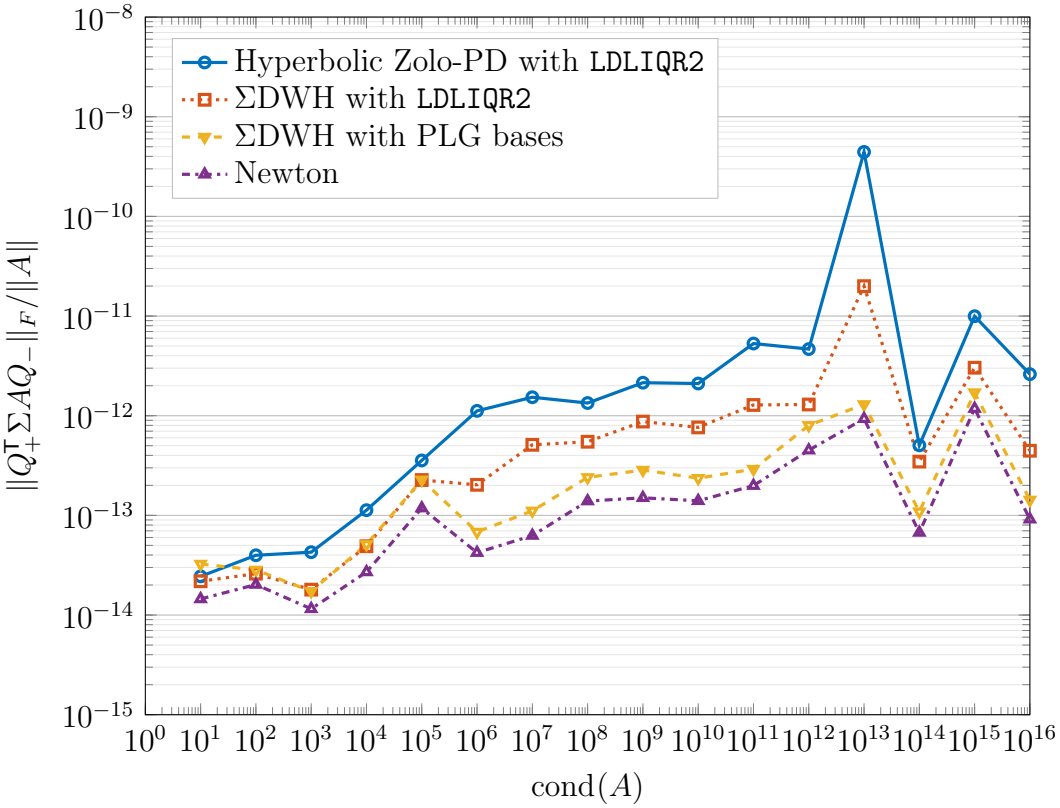
9.5.1 Random pseudosymmetric matrices

The goal of our first numerical experiment is to determine the achieved accuracy for different methods computing the matrix sign function.

Example 1 Σ is a signature matrix, where the diagonal values are chosen to be 1 or -1 with equal probability. Given a number $\kappa = \operatorname{cond}(A)$, we generate real 250×250 matrices as $A = \Sigma Q D Q^{\mathsf{T}}$. D is a diagonal matrix containing equally spaced values between 1 and κ. Q is a random orthogonal matrix (`Q=orth(rand(n,n))` in MATLAB). We perform 10 runs for different randomly generated matrices and compare the backward error represented by $\|Q_+^{\mathsf{T}}\Sigma A Q_-\|_{\mathrm{F}}/\|A\|_{\mathrm{F}}$, that is achieved by the different methods described in Chapter 8.

The averaged results are given in Figure 9.1. All methods yield backward errors smaller than 10^{-9}, even for badly conditioned matrices. All show a similar behavior. Hyperbolic Zolo-PD shows the highest backward error. Compared to the DWH-based iteration, this is expected, because Zolotarev functions of higher order are used. The

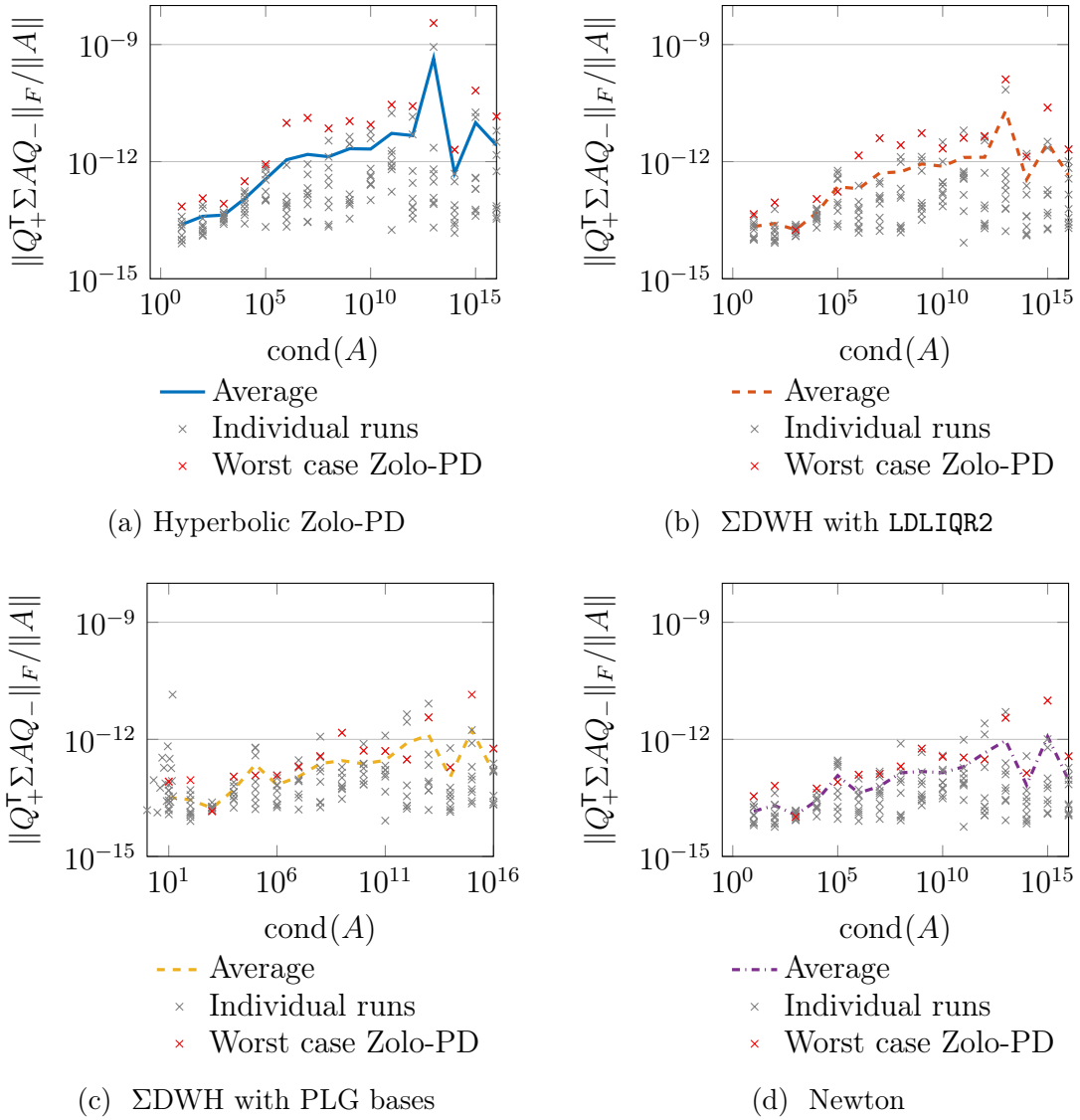
Figure 9.1: *Example 1*: Average residual after one spectral divide-and-conquer step, for 10 random matrices of size 250×250 with certain condition numbers. Different methods are used for computing the matrix sign function.



direct application of Zolotarev functions of high degree is known to be unstable [131]. In the indefinite setting, this phenomenon seems to appear sooner than in the standard setting described in [131]. The accuracy of DWH can be improved by employing permuted Lagrangian graph bases. This way, the accuracy is comparable to a Newton approach [61]. Permuted Lagrangian graph bases can also be employed for Zolotarev iterations of higher order, which may be explored in future work. Figure 9.2 displays the data of the individual runs of the same experiment. Here we see, that even badly conditioned matrices often achieve a backward-error of 10^{-14}, but some outliers increase the average. Further investigations are required in order to answer the question of what backward error can be achieved for a given matrix, The red crosses denote the matrices of a given κ for which hyperbolic Zolo-PD performed worst. We see, that for the same matrices, ΣDWH with `LDLIQR2` and the Newton iteration also perform worse than on other matrices generated in the same way. The quality therefore seems innate to the considered matrix. When PLG bases are employed, this relation can not be observed as clearly but is still noticeable.

The second example provides first insights on the performance which can be expected by using different methods.

Figure 9.2: *Example 1*: Residuals after one step of spectral divide-and-conquer for 10 runs with randomly generated matrices of size 250×250 with certain condition numbers.



(a) Hyperbolic Zolo-PD

(b) ΣDWH with `LDLIQR2`

(c) ΣDWH with PLG bases

(d) Newton

Example 2 A random matrix of size $5\,000 \times 5\,000$ is generated as in Example 1. We measure the number of iterations and the runtime using different methods to compute the matrix sign function. We measure the runtime of the sequential implementation of Zolo-PD, as well as the runtime resulting from its critical path. This means that we only take the runtime of one of the r independent steps in each iteration, i.e. the first lines in iterations (8.32) and (8.33), into account. The measured runtime reflects a performance which can be achieved when these independent computations are implemented in parallel. We compare it to runtimes achieved by ΣDWH based on $LDLIQR2$ (Algorithm 7.1) and ΣDWH based on LDL^T factorizations (iteration

Table 9.1: *Example 2:* Number of iterations, runtimes and error for different methods of spectral division for a matrix of size $5\,000 \times 5\,000$.

	κ	10^2	10^8	10^{12}
# iterations	Hyperbolic Zolo-PD	2	2	2
	ΣDWH with `LDLIQR2`	5	6	6
	ΣDWH with LDL^T	5	6	not converged
	Newton	7	9	9
runtime is s	Hyperbolic Zolo-PD (critical path)	941.70 (298.66)	1136.91 (255.86)	1240.86 (257.70)
	ΣDWH with `LDLIQR2`	883.79	988.39	1067.43
	ΣDWH with LDL^T	281.95	304.27	not converged
	Newton	355.05	379.38	416.19
backward error $\frac{\|Q_+^\mathsf{T}\Sigma A Q_-\|_F}{\|A\|_F}$	Hyperbolic Zolo-PD	7.42e-14	1.05e-11	1.78e-13
	ΣDWH with `LDLIQR2`	7.88e-14	2.81e-12	3.29e-13
	ΣDWH with LDL^T	8.50e-14	1.38e-11	not converged
	Newton	1.70e-13	6.66e-13	1.01e-13

(8.18)), and the Newton iteration [61]. This iteration was also presented in Section 8.5. The computation of PLG bases is not yet suited for large-scale performance-critical algorithms, which is why it is not included in the comparison. The results are found in Table 9.1.

The methods converge as expected and all except ΣDWH with LDL^T show good accuracy. We saw in Chapter 8 that ΣDWH with LDL^T is unstable for badly conditioned matrices. However, if it converges, it is the fastest among the measured methods. The computational effort of one ΣDWH iteration based on LDL^T is comparable to the effort of one Newton iteration, that is also based on an LDL^T factorization. ΣDWH converges in up to 6 steps, and Newton uses up to 9 steps. If `LDLIQR2` is employed instead of LDL^T in the ΣDWH, the computational effort doubles, as a second LDL^T decomposition is used for "reorthogonalization". This makes it slower than the Newton iteration. If the critical path of the hyperbolic Zolo-PD is followed, an even lower runtime can be achieved. It could be accelerated at the cost of stability, when LDL^T

Table 9.2: *Example 3:* Results for Bethe-Salpeter matrix computed for Lithium Fluoride.

	Hyperbolic Zolo-PD	ΣDWH with `LDLIQR2`	ΣDWH with LDL^{T}	Newton
# iterations	2	5	5	7
Zolotarev rank	4	1	1	not applicable
backward error (Chol, Alg. 9.6)	1.02e-10	7.42e-11	9.62e-11	2.48e-10
backward error (LDL, Alg. 9.7)	6.93e-18	7.26e-18	6.99e-18	1.48e-17

decompositions are used instead of `LDLIQR2`.

9.5.2 Applications in electronic structure computations

We now apply the developed method on two motivating examples concerning electronic excitations in solids and molecules.

Example 3 The `exciting` package [86, 178] implements various ab initio methods for computing excited states of solids or molecules, based on (linearized) augmented planewave + local orbital ((L)APW+lo) methods. It can be used to compute the optical scattering spectrum of Lithium Fluoride, based on the Bethe-Salpeter equation. The main computational effort in this example is to compute eigenvalues and eigenvectors of a matrix of the form

$$H_{LF} = \begin{bmatrix} A_{LF} & B_{LF} \\ -B_{LF} & -A_{LF} \end{bmatrix} \in \mathbb{C}^{2560\times 2560},\ A_{LF} = A_{LF}^{\mathsf{H}},\ B_{LF} = B_{LF}^{\mathsf{H}}.$$

H_{LF} is a Bethe-Salpeter matrix of form II (see (4.2)). It is pseudohermitian with respect to $\Sigma = \mathrm{diag}(I_n, -I_n)$ and additionally fulfills the definiteness property (4.3), i.e. $\Sigma H_{LF} > 0$. As pointed out in Lemma 4.6, the eigenvalues come in pairs of $\pm\lambda$. One step of spectral division results in a positive definite matrix, from which all eigenvalues and eigenvectors can be reconstructed. We extracted the matrix from the FORTRAN-based `exciting` code as a test example for our MATLAB-based prototype.

The results in Table 9.2 show that convergence is achieved in a limited number of iterations for all methods as expected. The reported backward error $\frac{\|Q_+^{\mathsf{T}}\Sigma A Q_-\|_F}{\|A\|_F}$ depends largely on the chosen method for computing a hyperbolic subspace representation. The Cholesky-based method does not work well. The eigenvalues smallest in modulus easily "pass over", such that the computed quantities ΣP_+ or $-\Sigma P_-$ have negative eigenvalues. The Cholesky-based method in Algorithm 9.6 does not accurately

Table 9.3: *Example 4:* Results for Bethe-Salpeter matrix computed for N_2H_4.

	Hyperbolic Zolo-PD	ΣDWH with `LDLIQR2`	ΣDWH with LDL^T	Newton
# iterations	2	5	5	7
Zolotarev rank	5	1	1	not applicable
backward error (Chol, Alg. 9.6)	1.19e-18	9.23e-19	1.46e-17	2.04e-18
backward error (LDL, Alg. 9.7)	1.22e-18	9.62e-19	1.46e-17	2.13e-18

capture this behavior, while the LDL^T-based method in Algorithm 9.7 alleviates the effect through pivoting.

All methods for computing the matrix sign function work equally well concerning accuracy because H_{LF} is well conditioned ($\mathrm{cond}(H_{LF}) \approx 10$).

Example 4 In [35, 32] a Bethe-Salpeter approach is explored in the context of tensor-structured Hartree-Fock theory for molecules [148]. We consider the N_2H_4 example in [35]. With real-valued orbitals the derivation arrives at a structured eigenvalue problem similar to Example 3, but with real values.

$$H_{N_2H_4} = \begin{bmatrix} A_{N_2H_4} & B_{N_2H_4} \\ -B_{N_2H_4}^\mathsf{T} & -A_{N_2H_4}^\mathsf{T} \end{bmatrix} \in \mathbb{R}^{1314\times 1314},\ A_{N_2H_4} = A_{N_2H_4}^\mathsf{T},\ B_{N_2H_4} \approx B_{N_2H_4}^\mathsf{T}.$$

While the original derivation in [148] yields a symmetric off-diagonal block B, in the construction in [35], this property is lost. The property of pseudosymmetry, however, is not affected, making our developed method applicable.

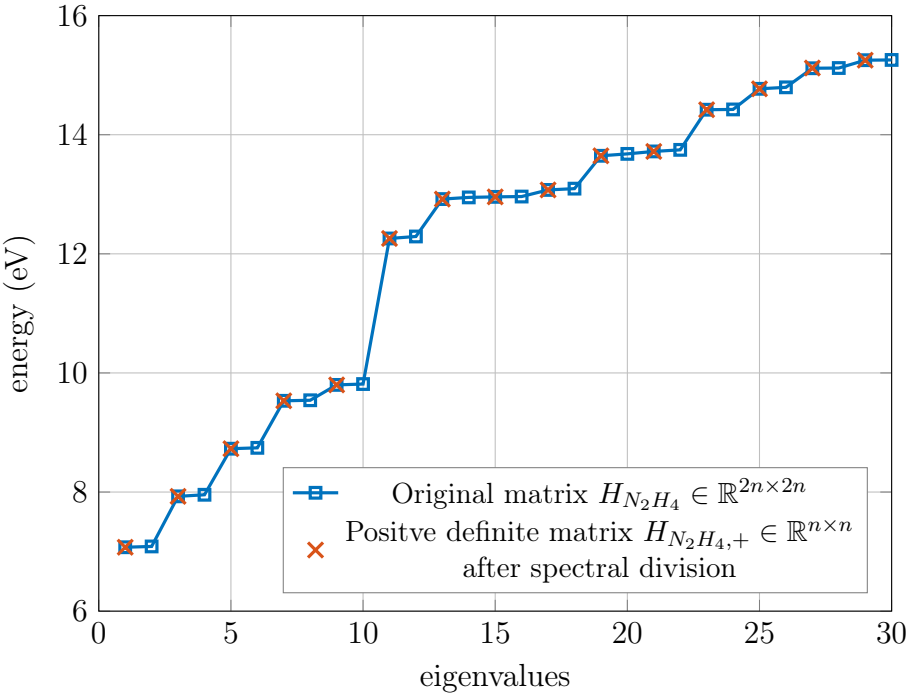
Numerical results of the spectral division are found in Table 9.3. All methods yield good results due to $\mathrm{cond}(H_{N_2H_4}) \approx 5$. In contrast to Example 3, no problem occurs, when the Cholesky decomposition is used for computing hyperbolic subspace representations. An explanation is probably linked to the fact that real matrices instead of complex ones are considered but requires further investigation.

Figure 9.4 corresponds to Figure 2 in [35] and displays absolute values of the eigenvalues of $H_{N_2H_4}$. The red crosses denote the eigenvalues of the positive definite matrix resulting after one step of spectral division (A_{11} in Algorithm 9.3). The remaining eigenvalues have (approximately) equal modulus, but opposite sign and are found as the eigenvalues of the negative definite matrix (A_{22} in Algorithm 9.3).

9.6 Conclusions

We presented a generalization of the well-known spectral divide-and-conquer approach for the computation of eigenvalues and eigenvectors of pseudosymmetric matrices.

Figure 9.4: *Example 4*: Absolute values of eigenvalues corresponding to N_2H_4.



In particular, when matrices with additional definiteness properties are considered, many parallels to the symmetric divide-and-conquer method become apparent. These parallels allow a computation of the matrix sign function, the key element for spectral division approaches, in just two iteration, using Zolotarev functions. Furthermore, the eigenvalue problem is decoupled into two smaller symmetric eigenvalue problems that can be solved with existing techniques. The presented algorithm is a promising new approach in the field of computing electronic excitations.

As we presented a completely new approach for structured eigenvalue computations, naturally, many possible future research directions open up as a consequence of this work. The numerical behavior of the subspace computations (Algorithms 9.6 and 9.7) is not yet fully understood, as the examples presented in Section 9.5.2 show. Regarding the applications concerning electron excitation, the matrices (4.1) and (4.2) show even more structure than has been exploited in the presented methods. Making the proposed iterations aware of these structures, such that they operate directly on the matrix blocks A and B, is a direction towards more efficient methods.

CHAPTER 10

CONCLUSIONS

Contents

10.1 Summary

In this thesis, various algorithms related to structured matrices have been developed. Figure 10.1 gives an overview of its contents in form of a flow chart. It asks questions about the nature and form of the computational problem at hand and guides the reader to the section of this thesis where related material is found. Furthermore, it shows the interdependencies of the chapters. Naturally, the chart is limited to the scope of this thesis. It is not meant to imply, that the given algorithms are the best or only possible solution strategies but represents the contents of this thesis in structured form.

As pointed out at numerous points in this work, the main motivation is drawn from finding solutions to structured eigenvalue problems. Three different structures are considered: skew-symmetric, Bethe-Salpeter and pseudosymmetric matrices. A Bethe-Salpeter matrix can come up in two forms, which we called form I and form II. The connections between the three structures are the following. A Bethe-Salpeter eigenvalue problem of form I (4.1), as long as the matrix fulfills the definiteness condition (4.3), can be transformed to a skew-symmetric eigenvalue problem. An HPC implementation of a suitable algorithm was developed in Chapter 5. Bethe-Salpeter matrices, both form I and form II, are members of the more general class of pseudosymmetric matrices. For this class of matrices, a structure-preserving divide-and-conquer method has been developed in Chapter 9. While the efforts to solve skew-symmetric eigenvalue problems resulted in a production-level code implemented in the HPC library `ELPA`, this is not yet the case for the proposed algorithm for pseudosymmetric matrices. However, there is a justified hope, that it could show good performance in an HPC implementation due to its inherent parallelism and the use of communication-

avoiding elements.

The generalized polar decomposition (Chapter 8) is used as a tool in the structured divide-and-conquer method. The generalized QR decomposition (Chapter 7), more specifically the hyperbolic (or indefinite) QR decomposition, is used as a tool for computing the generalized polar decomposition. Both may be studied in their own right and have potential other areas of applications.

Chapters 2, 3 and 4 do not appear in Figure 10.1 because they do not provide algorithms. They provide foundations for the presented chapters. The mathematical preliminaries (Chapter 2) are referred by all chapters. The chapters dealing exclusively with Bethe-Salpeter eigenvalue problems (Chapters 5 and 6) refer to the physical foundations (Chapter 3) and to mathematical results regarding the matrix structure (Chapter 4). Chapters 2 and 3 compile known theory, while Chapter 4 provides new results.

10.2 Future research

As becomes apparent in Figure 10.1, this thesis developed various algorithms related to the solution of structured eigenvalue problems. Apart from the HPC implementation presented in Chapter 5, all algorithms exist as prototypical MATLAB implementations. They do not yet exploit the available parallelism and are not optimized to run on high performance architectures. The additional implementation effort is significant and may be tackled in future work.

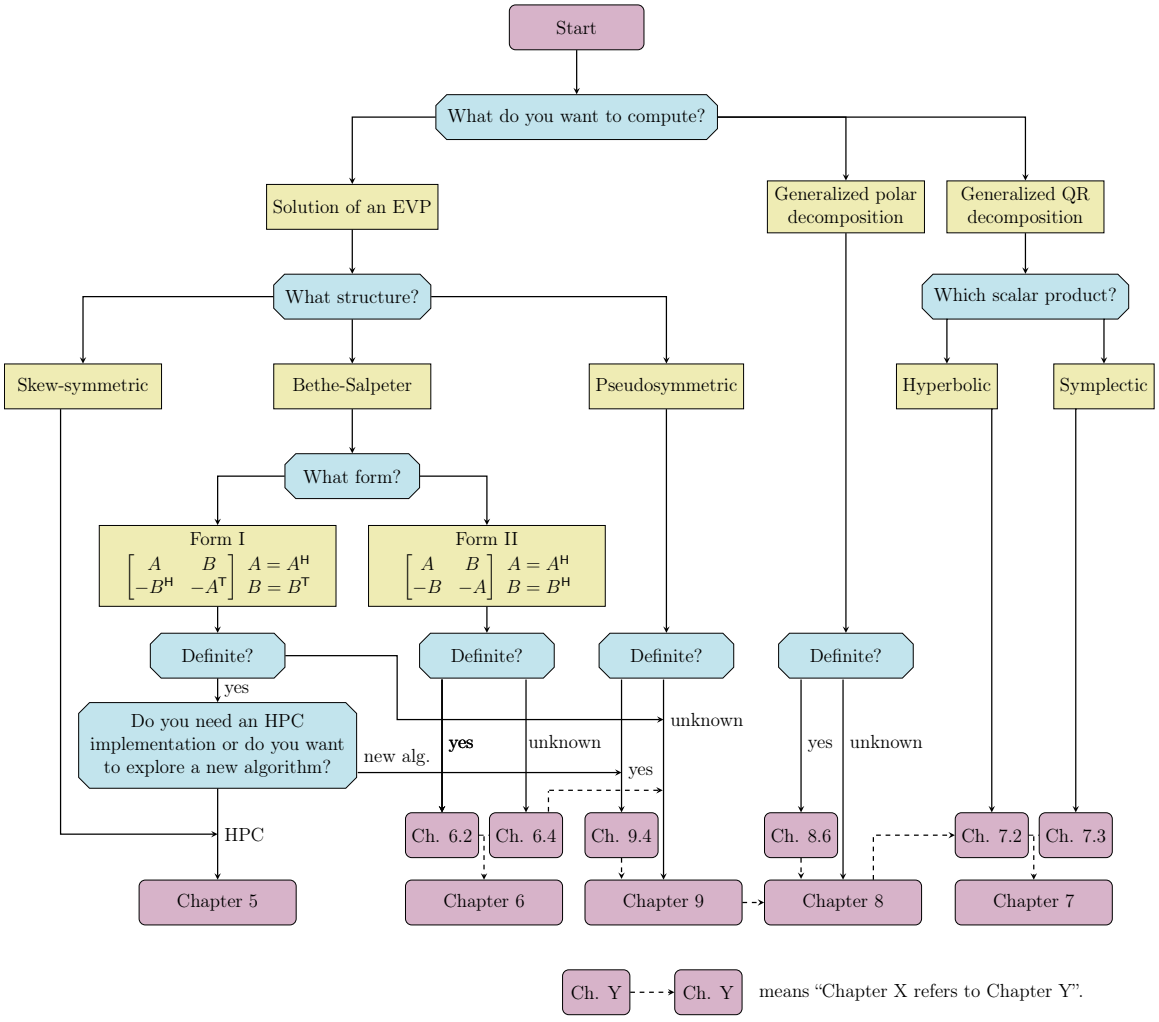
In this process, further algorithmic developments may be developed, as pointed out in the final sections of each chapter. One example is a mix-and-match approach in the iteration for computing the generalized polar decomposition (see Chapter 8). We considered various ways on how to realize the proposed iterations. Future efforts can be directed towards finding the optimal realization for different iteration steps. This combination will depend on the circumstances and may combine the benefits of the different realizations.

Another way towards further improvements in the structured divide-and-conquer method applied on Bethe-Salpeter matrices, is to exploit the additional structure during the iterations for computing the generalized polar decomposition.

The starting points of this thesis were structured matrices that have been acquired in electronic structure calculations as presented in Chapter 3. Here, we have seen, that the formation of these matrices follows many steps of physical and mathematical considerations. The access point to apply concepts from applied mathematics, and numerical linear algebra in particular does not have to be located after the formation of extremely large matrices. Instead, methods for dimensionality reduction should be considered in order to deduce the resulting computational load. If smaller matrices of a similar structure result from this process, the algorithms proposed in this thesis can be applied as a final step.

Any developments in this direction, working at the interconnections of physics and applied mathematics, require strong interdisciplinary collaborative efforts.

Figure 10.1: Summary of this thesis as a flow chart.



BIBLIOGRAPHY

[1] *Exc webpage*. www.bethe-salpeter.org. Accessed: 2019-11-26. 74

[2] *IBM Engineering and Scientific Subroutine Library for Linux on POWER and OpenPOWER*. https://www.ibm.com/docs/en/essl. 20

[3] *Intel® oneAPI Math Kernel Library*. https://software.intel.com/onemkl. 20

[4] *OpenBLAS*. http://www.openblas.net. 20

[5] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions, with formulas, graphs, and mathematical tables*, vol. 55 of National Bureau of Standards Applied Mathematics Series, U.S. Government Printing Office, Washington, D.C., 1972. 140

[6] W. Aggoune, C. Cocchi, D. Nabok, K. Rezouali, M. A. Belkhir, and C. Draxl, *Dimensionality of excitons in stacked van der Waals materials: The example of hexagonal boron nitride*, Phys. Rev. B, 97 (2018), p. 241114. 73

[7] N. I. Akhiezer, *Elements of the Theory of Elliptic Functions*, vol. 79 of Translations of Mathematical Monographs, American Mathematical Society, Providence, RI, 1990. Translated from the second Russian edition by H. H. McFaden. 140

[8] N. I. Akhiezer, *Theory of Approximation*, Dover, New York, 1992. 140

[9] A. Al-Ashouri, E. Köhnen, B. Li, A. Magomedov, H. Hempel, P. Caprioglio, J. A. Márquez, A. B. Morales Vilches, E. Kasparavicius, J. A. Smith, N. Phung, D. Menzel, M. Grischek, L. Kegelmann, D. Skroblin, C. Gollwitzer, T. Malinauskas, M. Jošt, G. Matič, B. Rech, R. Schlatmann, M. Topič, L. Korte, A. Abate, B. Stannowski, D. Neher, M. Stolterfoht, T. Unold, V. Getautis, and S. Albrecht, *Monolithic perovskite/silicon tandem solar cell with >29% efficiency by enhanced hole extraction*, Science, 370 (2020), pp. 1300–1309. 2

[10] S. Albrecht, *Optical absorption spectra of semiconductors and insulators: ab initio calculation of many-body effects*, PhD thesis, EECS Department, University of California, Berkeley, Jan 1999. 39

[11] A. Alvermann, A. Basermann, H. Bungartz, C. Carbogno, D. Ernst, H. Fehske, Y. Futamura, M. Galgon, G. Hager, S. Huber, T. Huckle, A. Ida, A. Imakura, M. Kawai, S. Köcher, M. Kreutzer, P. Kus, B. Lang, H. Lederer, V. Manin, A. Marek, K. Nakajima, L. Nemec, K. Reuter, M. Rippl, M. Röhrig-Zöllner, T. Sakurai, M. Scheffler, C. Scheurer, F. Shahzad, D. S. Brambila, J. Thies, and G. Wellein, *Benefits from using mixed precision computations in the ELPA-AEO and ESSEX-II eigensolver projects*, Jpn. J. Ind. Appl. Math., 36 (2019), pp. 699–717. 62

[12] E. Anderson, *Discontinuous plane rotations and the symmetric eigenvalue problem*, LAPACK Working Note 150, 2000. 112, 135

[13] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, third ed., 1999. 20, 60, 148

[14] C. Ashcraft, R. G. Grimes, and J. G. Lewis, *Accurate symmetric indefinite linear equation solvers*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 513–561. 15, 102

[15] T. Auckenthaler, V. Blum, H.-J. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, and P. Willems, *Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations*, Parallel Comput., 37 (2011), pp. 783 – 794. 62, 65, 68, 96

[16] T. Auckenthaler, H.-J. Bungartz, T. Huckle, L. Krämer, B. Lang, and P. Willems, *Developing algorithms and software for the parallel solution of the symmetric eigenvalue problem*, J. Comput. Sci., 2 (2011), pp. 272 – 278. 68

[17] Z. Bai and J. Demmel, *Design of a parallel nonsymmetric eigenroutine toolbox, Part I*, in Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, R. F. S. et al., ed., Philadelphia, 1993, SIAM, pp. 391–398. 118, 148, 150

[18] ——, *Design of a parallel nonsymmetric eigenroutine toolbox, Part II*, tech. rep., Computer Science Division, University of California, Berkeley, CA 94720, 1994. 148

[19] Z. Bai, J. Demmel, and M. Gu, *An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems*, Numer. Math., 76 (1997), pp. 279–308. 19, 148

[20] Z. Bai, R. Li, and W. Lin, *Linear response eigenvalue problem solved by extended locally optimal preconditioned conjugate gradient methods*, Science China. Mathematics, 59 (2016), pp. 1443–1460. 47

[21] Z. Bai and R.-C. Li, *Minimization principles for the linear response eigenvalue problem I: Theory*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 1075–1100. 47

[22] ——, *Minimization principles for the linear response eigenvalue problem II: Computation*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 392–416. 47, 78, 87, 95

[23] ——, *Minimization principles and computation for the generalized linear response eigenvalue problem*, BIT Numer. Math., 54 (2014), pp. 31–54. 47

[24] Z. Bai and R.-C. Li, *Recent progress in linear response eigenvalue problems*, in Eigenvalue problems: algorithms, software and applications in petascale computing, vol. 117 of Lect. Notes Comput. Sci. Eng., Springer International Publishing, Cham, 2017, pp. 287–304. 47

[25] G. Ballard, J. Demmel, and I. Dumitriu, *Minimizing communication for eigenproblems and the singular value decomposition*, ArXiv, abs/1011.3077 (2010). 149

[26] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, *Minimizing communication in numerical linear algebra*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 866–901. 149

[27] U. Baur, P. Benner, and L. Feng, *Model order reduction for linear and nonlinear systems: A system-theoretic perspective*, Arch. Comput. Methods Eng., 21 (2014), pp. 331–358. 17

[28] P. Benner, *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*, Dissertation, Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz (Germany), Feb. 1997. 10

[29] ——, *Symplectic balancing of Hamiltonian matrices*, SIAM J. Sci. Comput., 22 (2001), pp. 1885–1904. 130

[30] P. Benner, R. Byers, and E. Barth, *Algorithm 800. Fortran 77 subroutines for computing the eigenvalues of Hamiltonian matrices I: The square-reduced method*, ACM Trans. Math. Software, 26 (2000), pp. 49–77. 81

[31] P. Benner, R. Byers, H. Fassbender, V. Mehrmann, and D. Watkins, *Cholesky-like factorizations of skew-symmetric matrices*, Electron. Trans. Numer. Anal., 11 (2000), pp. 85–93. 108, 110, 111, 112

[32] P. Benner, S. Dolgov, V. Khoromskaia, and B. N. Khoromskij, *Fast iterative solution of the Bethe-Salpeter eigenvalue problem using low-rank and QTT tensor approximation*, J. Comput. Phys., 334 (2017), pp. 221–239. 166

[33] P. Benner, H. Fassbender, and D. S. Watkins, *Two connections between the SR and HR eigenvalue algorithms*, Linear Algebra Appl., 272 (1997), pp. 17–32. 89

[34] P. Benner, H. Fassbender, and C. Yang, *Some remarks on the complex J-symmetric eigenproblem*, Linear Algebra Appl., 544 (2018), pp. 407 – 442. 10, 49, 51, 104, 106

[35] P. Benner, V. Khoromskaia, and B. N. Khoromskij, *A reduced basis approach for calculation of the Bethe-Salpeter excitation energies using low-rank tensor factorizations*, Mol. Phys., 114 (2016), pp. 1148–1161. 120, 166

[36] P. Benner, M. Köhler, and J. Saak, *Fast approximate solution of the non-symmetric generalized eigenvalue problem on multicore architectures*, in Parallel Computing: Accelerating Computational Science and Engineering (CSE), M. Bader, A. Bodeand, H.-J. Bungartz, M. Gerndt, G. R. Joubert, and F. Peters, eds., vol. 25 of Advances in Parallel Computing, IOS Press, 2014, pp. 143–152. 148

[37] P. Benner, D. Kressner, and V. Mehrmann, *Skew-Hamiltonian and Hamiltonian eigenvalue problems: Theory, algorithms and applications*, in Proc. Conf. Appl Math. Scientific Comp., Z. Drmač, M. Marusic, and Z. Tutek, eds., Springer-Verlag, Dordrecht, 2005, pp. 3–39. 10, 49, 60, 104

[38] P. Benner, D. Kressner, V. Sima, and A. Varga, *Die SLICOT-Toolboxen für MATLAB*, at-Automatisierungstechnik, 58 (2010), pp. 15–25. 85

[39] P. Benner, V. Mehrmann, and H. Xu, *A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils*, Numer. Math., 78 (1998), pp. 329–358. 10, 78

[40] ——, *A note on the numerical solution of complex Hamiltonian and skew-Hamiltonian eigenvalue problems*, Electron. Trans. Numer. Anal., 8 (1999), pp. 115–126. 85

[41] P. Benner and E. S. Quintana-Ortí, *Solving stable generalized Lyapunov equations with the matrix sign function*, Numer. Algorithms, 20 (1999), pp. 75–100. 118

[42] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*, vol. 4 of Software, Environments and Tools, SIAM Publications, Philadelphia, PA, USA, 1997. 20, 60, 62

[43] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley, *An updated set of basic linear algebra subprograms (BLAS)*, ACM Transactions on Mathematical Software, 28 (2002), pp. 135–151. 19

[44] X. Blase, A. Rubio, S. G. Louie, and M. L. Cohen, *Quasiparticle band structure of bulk hexagonal boron nitride and related systems*, Phys. Rev. B, 51 (1995), pp. 6868–6875. 73

[45] A. Bojanczyk, N. J. Higham, and H. Patel, *Solving the indefinite least squares problem by hyperbolic QR factorization*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 914–931. 99, 115

[46] Y. Bolshakov and B. Reichstein, *Unitary equivalence in an indefinite scalar product: an analogue of singular-value decomposition*, Linear Algebra Appl., 222 (1995), pp. 155–226. 7

[47] Y. Bolshakov, C. V. M. van der Mee, A. C. M. Ran, B. Reichstein, and L. Rodman, *Polar decompositions in finite-dimensional indefinite scalar product spaces: general theory*, Linear Algebra Appl., 261 (1997), pp. 91–141. 7

[48] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer-Verlag, Berlin Heidelberg, 2005. 120

[49] M. Brebner and J. Grad, *Eigenvalues of $Ax = \lambda Bx$ for real symmetric matrices A and B computed by reduction to a pseudo symmetric form and the HR–process*, Linear Algebra Appl., 43 (1982), pp. 99–118. 89, 149

[50] J. R. Bunch, *A note on the stable decompostion of skew-symmetric matrices*, Math. Comp., 38 (1982), pp. 475–479. 108, 110

[51] J. R. Bunch and L. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comp., 31 (1977), pp. 163–179. 15, 156

[52] J. R. Bunch, L. Kaufman, and B. Parlett, *Decomposition of a symmetric matrix*, Numer. Math., 27 (1976), pp. 95–109. 15

[53] J. R. Bunch and B. N. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8 (1971), pp. 639–655. 15

[54] W. Bunse and A. Bunse-Gerstner, *Numerische Lineare Algebra*, Teubner, Stuttgart, 1985. 100, 153

[55] A. Bunse-Gerstner, *Berechnung der Eigenwerte einer Matrix mit dem HR-Verfahren*, in Numerische Behandlung von Eigenwertaufgaben, Band 2 (Tagung, Tech. Univ. Clausthal, Clausthal, 1978), vol. 43 of Internat. Schriftenreihe Numer. Math., Birkhäuser, Basel-Boston, Mass., 1979, pp. 26–39. 149

[56] A. Bunse-Gerstner, *An analysis of the HR algorithm for computing the eigenvalues of a matrix*, Linear Algebra Appl., 35 (1981), pp. 155–173. 89, 149

[57] ——, *An algorithm for the symmetric generalized eigenvalue problem*, Linear Algebra Appl., 58 (1984), pp. 43–68. 89

[58] A. Bunse-Gerstner, *Matrix factorizations for symplectic QR-like methods*, Linear Algebra Appl., 83 (1986), pp. 49–77. 104, 105

[59] A. Bunse-Gerstner and V. Mehrmann, *A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation*, IEEE Trans. Autom. Control, 31 (1986), pp. 1104–1113. 105, 107

[60] R. Byers, *Solving the algebraic Riccati equation with the matrix sign function*, Linear Algebra Appl., 85 (1987), pp. 267–279. 136

[61] R. Byers and H. Xu, *A new scaling for Newton's iteration for the polar decomposition and its backward stability*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 822–843. 136, 137, 160, 162, 164

[62] C. Campos and J. E. Roman, *Restarted Q-Arnoldi-type methods exploiting symmetry in quadratic eigenvalue problems*, BIT Numer. Math., 56 (2016), pp. 1213–1236. 96

[63] G. Cappellini, G. Satta, M. Palummo, and G. Onida, *Optical properties of BN in cubic and layered hexagonal phases*, Phys. Rev. B, 64 (2001), p. 035104. 73

[64] M. Casida, *Time-dependent density functional response theory for molecules*, in Recent Advances in Density Functional Methods, World Scientific, 1995, pp. 155–192. 35, 44, 47, 78, 79

[65] S. Chandrasekaran and A. H. Sayed, *Stabilizing the generalized Schur algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 950–983. 112, 135

[66] D. Child, *The Essentials of Factor Analysis*, Bloomsbury Academic, 2006. 120

[67] P. Cudazzo, L. Sponza, C. Giorgetti, L. Reining, F. Sottile, and M. Gatti, *Exciton band structure in two-dimensional materials*, Phys. Rev. Lett. , 116 (2016), p. 066803. 73

[68] D. Day, *An efficient implementation of the nonsymmetric Lanczos algorithm*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 566–589. 96

[69] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM J. Sci. Comput., 34 (2012), pp. A206–A239. 99

[70] J. Deslippe, G. Samsonidze, D. A. Strubbe, M. Jain, M. L. Cohen, and S. G. Louie, *BerkeleyGW: A massively parallel computer package for the calculation of the quasiparticle and optical properties of materials and nanostructures*, Comput. Phys. Commun., 183 (2012), pp. 1269 – 1289. 82

[71] P. A. M. Dirac, *A new notation for quantum mechanics*, Mathematical Proceedings of the Cambridge Philosophical Society, 35 (1939), p. 416–418. 25

[72] J. J. Dongarra, J. R. Gabriel, D. D. Koelling, and J. H. Wilkinson, *The eigenvalue problem for Hermitian matrices with time reversal symmetry*, Linear Algebra Appl., 60 (1984), pp. 27–42. 120

[73] I. S. Duff, *MA57—a code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Softw., 30 (2004), pp. 118–144. 15

[74] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218. 17

[75] A. Edelman and S. Jeong, *Fifty three matrix factorizations: A systematic approach*, 2021. 7

[76] H. Fassbender and M. Rozložník, *On the conditioning of factors in the SR decomposition*, Linear Algebra Appl., 505 (2016), pp. 224–244. 104, 108

[77] A. L. Fetter and J. D. Walecka, *Quantum Theory of Many-Particle Systems*, Dover Publications, 2003. 36, 37

[78] C. Fiolhais, F. Nogueira, and M. Marques, *A Primer in Density Functional Theory*, Springer, 2003. 29

[79] G. Fugallo, M. Aramini, J. Koskelo, K. Watanabe, T. Taniguchi, M. Hakala, S. Huotari, M. Gatti, and F. Sottile, *Exciton energy-momentum map of hexagonal boron nitride*, Phys. Rev. B, 92 (2015), p. 165122. 73

[80] T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa, *Shifted Cholesky QR for computing the QR factorization of ill-conditioned matrices*, SIAM J. Sci. Comput., 42 (2020), pp. A477–A503. 98, 116

[81] T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, and Y. Yamamoto, *CholeskyQR2: A simple and communication-avoiding algorithm for computing a tall-skinny qr factorization on a large-scale parallel system*, in 2014 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, 2014, pp. 31–38. 99

[82] S. Galambosi, L. Wirtz, J. A. Soininen, J. Serrano, A. Marini, K. Watanabe, T. Taniguchi, S. Huotari, A. Rubio, and K. Hämäläinen, *Anisotropic excitonic effects in the energy loss function of hexagonal boron nitride*, Phys. Rev. B, 83 (2011), p. 081413. 73

[83] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, fourth ed., 2013. 3, 7, 9, 13, 15, 16, 19, 51, 60, 82, 83, 148

[84] X. Gonze, F. Jollet, F. Abreu Araujo, D. Adams, B. Amadon, T. Applencourt, C. Audouze, J. M. Beuken, J. Bieder, A. Bokhanchuk, E. Bousquet, F. Bruneval, D. Caliste, M. Côté, F. Dahm, F. Da Pieve, M. Delaveau, M. Di Gennaro, B. Dorado, C. Espejo, G. Geneste, L. Genovese, A. Gerossier, M. Giantomassi, Y. Gillet, D. R. Hamann, L. He, G. Jomard, J. Laflamme Janssen, S. Le Roux, A. Levitt, A. Lherbier, F. Liu, I. Lukačević, A. Martin, C. Martins, M. J. T. Oliveira, S. Poncé, Y. Pouillon, T. Rangel, G. M. Rignanese, A. H. Romero, B. Rousseau, O. Rubel, A. A. Shukri, M. Stankovski, M. Torrent, M. J. Van Setten, B. Van Troeye, M. J. Verstraete, D. Waroquiers, J. Wiktor, B. Xu, A. Zhou, and J. W. Zwanziger, *Recent developments in the ABINIT software package*, Comput. Phys. Commun., 205 (2016), pp. 106–131. 74

[85] D. J. Griffiths and D. F. Schroeter, *Introduction to Quantum Mechanics*, Cambridge University Press, third ed., 2018. 24, 25

[86] A. Gulans, S. Kontur, C. Meisenbichler, D. Nabok, P. Pavone, S. Rigamonti, S. Sagmeister, U. Werner, and C. Draxl, *Exciting: A full-potential all-electron package implementing density-functional theory and many-body perturbation theory*, J. Phys. Condens. Matter , 26 (2014), p. 363202. 74, 82, 86, 165

[87] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*, CRC Press, Inc., Boca Raton, FL, USA, first ed., 2010. 20

[88] H. Harbrecht, M. Peters, and R. Schneider, *On the low-rank approximation by the pivoted Cholesky decomposition*, Applied Numerical Mathematics, 62 (2012), pp. 428–440. Third Chilean Workshop on Numerical Analysis of Partial Differential Equations (WONAPDE 2010). 155

[89] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer-Verlag New York, second ed., 2009. 17

[90] L. Hedin, *New method for calculating the one-particle Green's function with application to the electron gas problem*, Phys. Rev. , 139 (1965), pp. A796–A823. 36, 37

[91] F. Henneke, L. Lin, C. Vorwerk, C. Draxl, R. Klein, and C. Yang, *Fast optical absorption spectra calculations for periodic solid state systems*, Commun. Appl. Math. Comput. Sci., 15 (2020), pp. 89–113. 78

[92] D. Henrion and P. Hippe, *Hyperbolic QR factorization for J-spectral factorization of polynomial matrices*, in 42nd IEEE International Conference on Decision and Control, vol. 4, 2003, pp. 3479.–3484. 112, 135

[93] N. J. Higham, *The matrix sign decomposition and its relation to the polar decomposition*, Linear Algebra Appl., 212/213 (1994), pp. 3–20. 18

[94] N. J. Higham, *J-orthogonal matrices: properties and generation*, SIAM Rev., 45 (2003), pp. 504–519. 99, 136

[95] ——, *Functions of Matrices: Theory and Computation*, Applied Mathematics, SIAM Publications, Philadelphia, 2008. 17, 18, 19, 79, 118, 119, 120, 123, 133, 135, 136, 137

[96] N. J. Higham, D. Mackey, N. Mackey, and F. Tisseur, *Functions preserving matrix groups and iterations for the matrix square root*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 849–877. 7, 18, 123

[97] N. J. Higham, C. Mehl, and F. Tisseur, *The canonical generalized polar decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2163–2180. 7, 11, 18, 118, 119, 124

[98] P. Hohenberg and W. Kohn, *Inhomogeneous electron gas*, Phys. Rev., 136 (1964), pp. B864–B871. 30

[99] I. Houtzager, *JQR/JRQ/JQL/JLQ factorizations*, MATLAB Central File Exchange, (2015). Retrieved February 12, 2020. 112, 135

[100] Y. H. J. W. Demmel, M. Hoemmen and E. J. Riedy, *Non-negative diagonals and high performance on low-profile matrices from Householder QR*, LAPACK Working Note 203, 2008. 13

[101] Z. Jin, X. Du, Y. Xu, Y. Deng, M. Liu, Y. Zhao, B. Zhang, X. Li, L. Zhang, C. Peng, Y. Duan, J. Yu, L. Wang, K. Yang, F. Liu, R. Jiang, X. Yang, T. You, X. Liu, X. Yang, F. Bai, H. Liu, X. Liu, L. W. Guddat, W. Xu, G. Xiao, C. Qin, Z. Shi, H. Jiang, Z. Rao, and H. Yang, *Structure of M^{pro} from SARS-CoV-2 and discovery of its inhibitors*, Nat., 582 (2020), pp. 289–293. 2

[102] C. Kenney and A. J. Laub, *Rational iterative methods for the matrix sign function*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 273–291. 19, 123

[103] ——, *On scaling Newton's method for polar decomposition and the matrix sign function*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 688–706. 121

[104] V. Khoromskaia and B. N. Khoromskij, *Tensor Numerical Methods in Quantum Chemistry*, De Gruyter, Berlin, Boston, 2018. 30

[105] U. Kintzel, *Procrustes problems in finite dimensional indefinite scalar product spaces*, Linear Algebra Appl., 402 (2005), pp. 1–28. 117, 120

[106] M. Köhler, *Approximate Solution of Non-Symmetric Generalized Eigenvalue Problems and Linear Matrix Equation on HPC Platforms*, Dissertation, Department of Mathematics, Otto-von-Guericke University, Magdeburg, Germany, 2021. Submitted. 148

[107] W. Kohn and L. J. Sham, *Self-consistent equations including exchange and correlation effects*, Phys. Rev., 140 (1965), pp. A1133–A1138. 32

[108] J. Koskelo, G. Fugallo, M. Hakala, M. Gatti, F. Sottile, and P. Cudazzo, *Excitons in van der Waals materials: From monolayer to bulk hexagonal boron nitride*, Phys. Rev. B, 95 (2017), p. 035125. 73

[109] D. Kressner, *Numerical Methods for General and Structured Eigenvalue Problems*, vol. 46 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin Heidelberg, 2005. 7, 78, 104, 106

[110] P. Kůs, A. Marek, S. Köcher, H.-H. Kowalski, C. Carbogno, C. Scheurer, K. Reuter, M. Scheffler, and H. Lederer, *Optimizations of the eigensolvers in the ELPA library*, Parallel Comput., 85 (2019), pp. 167 – 177. 62

[111] P. Kůs, H. Lederer, and A. Marek, *GPU optimization of large-scale eigenvalue solver*, in Numerical Mathematics and Advanced Applications ENUMATH 2017, F. A. Radu, K. Kumar, I. Berre, J. M. Nordbotten, and I. S. Pop, eds., Cham, 2019, Springer International Publishing, pp. 123–131. 73

[112] M. Levy, *Universal variational functionals of electron densities, first-order density matrices, and natural spin-orbitals and solution of the v-representability problem*, Proc. Natl. Acad. Sci., 76 (1979), pp. 6062–6065. 31

[113] H. Li, H. Yang, and H. Shao, *Perturbation analysis for the hyperbolic QR factorization*, Comput. Math. Appl., 63 (2012), pp. 1607–1620. 102

[114] E. H. Lieb, *Density functionals for Coulomb systems*, Int. J. Quantum Chem., 24 (1983), pp. 243–277. 31

[115] L. Lin and J. Lu, *A Mathematical Introduction to Electronic Structure Theory*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2019. 25, 30, 42

[116] H. Ltaief, D. Sukkari, A. Esposito, Y. Nakatsukasa, and D. Keyes, *Massively parallel polar decomposition on distributed-memory systems*, ACM Trans. Parallel Comput., 6 (2019). 118, 149

[117] D. S. Mackey, N. Mackey, and F. Tisseur, *Structured factorizations in scalar product spaces*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 821–850. 7, 49, 151

[118] A. N. Malyshev, *Parallel algorithm for solving some spectral problems of linear algebra*, Linear Algebra Appl., 188/189 (1993), pp. 489–520. 148

[119] A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.-J. Bungartz, and H. Lederer, *The ELPA library: scalable parallel eigenvalue solutions for electronic structure theory and computational science*, J. Phys. Condens. Matter, 26 (2014), p. 213201. 21, 60, 62, 159

[120] R. M. Martin, *Electronic Structure: Basic Theory and Practical Methods*, Cambridge University Press, second ed., 2020. 29

[121] R. S. Martin, C. Reinsch, and J. H. Wilkinson, *Householder's tridiagonalization of a symmetric matrix*, Numer. Math., 11 (1968), pp. 181–195. 67

[122] C. Mehl, *Finite-Dimensional Indefinite Inner Product Spaces and Applications in Numerical Analysis*, Springer Basel, Basel, 2015, pp. 1–17. 88

[123] C. Mehl, V. Mehrmann, and H. Xu, *On doubly structured matrices and pencils that arise in linear response theory*, Linear Algebra Appl., 380 (2004), pp. 3–51. 120

[124] ——, *Structured decompositions for matrix triples: SVD-like concepts for structured matrices*, Oper. Matrices, 3 (2009), pp. 303–356. 92

[125] C. Mehl, A. C. M. Ran, and L. Rodman, *Polar decompositions of normal operators in indefinite inner product spaces*, in Operator theory in Krein spaces and nonlinear eigenvalue problems, vol. 162 of Oper. Theory Adv. Appl., Birkhäuser, Basel, 2006, pp. 277–292. 7

[126] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*, no. 163 in Lecture Notes in Control and Information Sciences, Springer-Verlag, Heidelberg, July 1991. 10

[127] V. Mehrmann and F. Poloni, *Doubling algorithms with permuted Lagrangian graph bases*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 780–805. 128, 130, 132, 137, 144

[128] Message Passing Interface Forum, *MPI: A Message-passing Interface Standard, Version 3.1*, 2015. 20, 62

[129] L. Mirsky, *Symmetric gauge functions and unitarily invariant norms*, Q. J. Math., 11 (1960), pp. 50–59. 17

[130] Y. Nakatsukasa, Z. Bai, and F. Gygi, *Optimizing Halley's iteration for computing the matrix polar decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2700–2720. 118, 121, 122, 126, 133, 136, 137, 140, 152

[131] Y. Nakatsukasa and R. W. Freund, *Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: the power of Zolotarev's functions*, SIAM Rev., 58 (2016), pp. 461–493. 118, 122, 138, 139, 140, 141, 142, 144, 149, 152, 162

[132] Y. Nakatsukasa and N. J. Higham, *Backward stability of iterations for computing the polar decomposition*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 460–479. 118, 120, 122, 152

[133] ——, *Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD*, SIAM J. Sci. Comput., 35 (2013), pp. A1325–A1349. 96, 149, 150, 152, 159

[134] Netlib, *BLAS (Basic Linear Algebra Subprograms)*. http://www.netlib.org/blas/, 2017. 19, 20

[135] ——, *LAPACK (Linear Algebra PACKage)*. http://www.netlib.org/lapack/, 2021. 20

[136] J. Nickolls, I. Buck, M. Garland, and K. Skadron, *Scalable parallel programming with CUDA: Is CUDA the parallel programming model that application developers have been waiting for?*, Queue, 6 (2008), p. 40–53. 22

[137] NVIDIA Corporation, *cuBLAS Library User Guide*, Santa Clara, CA, 2021. version DU-06702-001_v11.4. 22

[138] ——, *CUDA*. https://developer.nvidia.com/cuda-zone, 2021. 22

[139] G. Onida, L. Reining, and A. Rubio, *Electronic excitations: density-functional versus many-body Green's-function approaches*, Rev. Mod. Phys., 74 (2002), pp. 601–659. 32, 33, 35, 38, 41, 44, 45, 48

[140] *OpenMP*. http://openmp.org, 1997–2011. 21

[141] P. J. O'Donnell, *Essential Dynamics and Relativity*, Taylor & Francis Group, 2015. 24

[142] P. Papakonstantinou, *Reduction of the RPA eigenvalue problem and a generalized Cholesky decomposition for real-symmetric matrices*, EPL, 78 (2007), p. 12001. 47, 87, 88

[143] B. N. Parlett and H. C. Chen, *Use of indefinite pencils for computing damped natural modes*, Linear Algebra Appl., 140 (1990), pp. 53–88. 96

[144] R. G. Parr and W. Yang, *Density-Functional Theory of Atoms and Molecules*, Oxford University Press, USA, 1994. 29

[145] C. Penke, *GPU-accelerated computation of the matrix disc function and its applications*, Master's thesis, Otto-von-Guericke-Universität, Magdeburg, Germany, Jan. 2017. 14

[146] P. P. Petrushev and V. A. Popov, *Rational approximation of real functions*, vol. 28 of Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge, 1987. 140

[147] F. Poloni, *Permuted graph matrices and their applications*, in Numerical Algebra, Matrix Theory, Differential-Algebraic Equations and Control Theory: Festschrift in Honor of Volker Mehrmann, Springer International Publishing, Cham, 2015, pp. 107–129. 128

[148] E. Rebolini, J. Toulouse, and A. Savin, *Electronic excitation energies of molecular systems from the Bethe-Salpeter equation: Example of the H2 molecule*, in Concepts and Methods in Modern Theoretical Chemistry, S. Ghosh and P. K. Chattaraj, eds., CRC Press, Boca Raton, 2013, ch. 18, pp. 367–389. 166

[149] J. D. Roberts, *Linear model reduction and solution of the algebraic Riccati equation by use of the sign function*, Int. J. Control, 32 (1980), pp. 677–687. 18, 118, 150

[150] E. Runge and E. K. U. Gross, *Density-functional theory for time-dependent systems*, Phys. Rev. Lett., 52 (1984), pp. 997–1000. 35

[151] D. M. S., G. Dresselhaus, and A. Jorio, *Group Theory: Application to the Physics of Condensed Matter*, Springer-Verlag, Berlin Heidelberg, 2008. 42

[152] S. Sagmeister and C. Ambrosch-Draxl, *Time-dependent density functional theory versus Bethe-Salpeter equation: an all-electron study*, Phys. Chem. Chem. Phys., 11 (2009), pp. 4451–4457. 45, 85

[153] J. J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*, Cambridge University Press, Cambridge, third ed., 2020. 25

[154] E. E. Salpeter and H. A. Bethe, *A relativistic equation for bound-state problems*, Phys. Rev., 84 (1951), pp. 1232–1242. 36

[155] T. Sander, E. Maggio, and G. Kresse, *Beyond the Tamm-Dancoff approximation for extended systems using exact diagonalization*, Phys. Rev. B, 92 (2015), p. 045209. 33, 34, 37, 42, 48, 78, 79, 82

[156] D. Sangalli, A. Ferretti, H. Miranda, C. Attaccalite, I. Marri, E. Cannuccia, P. Melo, M. Marsili, F. Paleari, A. Marrazzo, G. Prandini, P. Bonfà, M. O. Atambo, F. Affinito, M. Palummo, A. Molina-Sánchez, C. Hogan, M. Grüning, D. Varsano, and A. Marini, *Many-body perturbation theory calculations using the yambo code*, J. Phys. Condens. Matter, 31 (2019), p. 325902. 82

[157] R. S. Schreiber and C. Van Loan, *A storage-efficient WY representation for products of Householder transformations*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 53–57. 14, 67

[158] M. Shao, F. H. da Jornada, L. Lin, C. Yang, J. Deslippe, and S. G. Louie, *A structure preserving Lanczos algorithm for computing the optical absorption spectrum*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 683–711. 78, 95

[159] M. Shao, F. H. da Jornada, C. Yang, J. Deslippe, and S. G. Louie, *Structure preserving parallel algorithms for solving the Bethe-Salpeter eigenvalue problem*, Linear Algebra Appl., 488 (2016), pp. 148–167. xiii, 52, 61, 62, 70, 71, 73, 81, 158

[160] H. Shull and G. G. Hall, *Atomic units*, Nature, 184 (1959), pp. 1559–1560. 27

[161] S. H. Simon, *The Oxford Solid State Basics*, Oxford Univ. Press, Oxford, UK, 2013. 41

[162] S. Singer, *Indefinite QR factorization*, BIT Numer. Math., 46 (2006), pp. 141–161. 102

[163] S. Singer and S. Singer, *Rounding-error and perturbation bounds for the indefinite QR factorization*, in Proceedings of the International Workshop on Accurate Solution of Eigenvalue Problems (University Park, PA, 1998), vol. 309, 2000, pp. 103–119. 100, 102, 154

[164] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*, Academic Press, New York, 1990. 82

[165] G. Strang, *Introduction to Linear Algebra*, Wellesley-Cambridge Press, Wellesley, MA, fifth ed., 2016. 7, 12

[166] G. Strinati, *Application of the Green's functions method to the study of the optical properties of semiconductors*, Riv. Nuovo Cimento , 11 (1988), pp. 1–86. 36

[167] D. Sukkari, H. Ltaief, A. Esposito, and D. Keyes, *A QDWH-based SVD software framework on distributed-memory manycore systems*, ACM Trans. Math. Softw., 45 (2019), pp. Art. 18, 21. 118

[168] X. Sun and E. S. Quintana-Ortí, *The generalized Newton iteration for the matrix sign function*, 24 (2002), pp. 669–683. 155

[169] X. Sun and E. S. Quintana-Ortí, *Spectral division methods for block generalized Schur decompositions*, Math. Comp., 73 (2004), pp. 1827–1847. 118, 150

[170] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*, Dover Publications, Inc., first ed., 1996. 29, 30

[171] L. Talirz, L. M. Ghiringhelli, and B. Smit, *Trends in atomistic simulation software usage [article v1.0]*, Living Journal of Computational Molecular Science, 3 (2021), p. 1483. 82

[172] *The Top500 list*. http://www.top500.org. 2, 20

[173] L. N. Trefethen and D. Bau, III, *Numerical linear algebra*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997. 7, 11

[174] C. F. Van Loan, *A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix*, Linear Algebra Appl., 61 (1984), pp. 233–251. 10, 80, 81

[175] F. G. Van Zee and R. A. van de Geijn, *BLIS: A framework for rapidly instantiating BLAS functionality*, ACM Trans. Math. Software, 41 (2015), pp. 14:1–14:33. 20

[176] L. Varrassi, P. Liu, Z. E. Yavas, M. Bokdam, G. Kresse, and C. Franchini, *Optical and excitonic properties of transition metal oxide perovskites by the Bethe-Salpeter equation*, Phys. Rev. Materials, 5 (2021), p. 074601. 2

[177] K. Veselić, *Damped oscillations of linear systems*, vol. 2023 of Lecture Notes in Math., Springer-Verlag, 2011. 120

[178] C. Vorwerk, B. Aurich, C. Cocchi, and C. Draxl, *Bethe–Salpeter equation for absorption and scattering spectroscopy: implementation in the exciting code*, Electronic Structure, 1 (2019), p. 037001. 74, 78, 79, 165

[179] V. Šego, *Two-sided hyperbolic SVD*, Linear Algebra Appl., 433 (2010), pp. 1265–1275. 154

[180] ——, *The hyperbolic Schur decomposition*, Linear Algebra Appl., 440 (2014), pp. 90–110. 154

[181] R. C. Ward and L. J. Gray, *Eigensystem computation for skew-symmetric and a class of symmetric matrices*, ACM Trans. Math. Softw., 4 (1978), pp. 278–285. 60, 64

[182] D. Watkins, *The Matrix Eigenvalue Problem*, Society for Industrial and Applied Mathematics, 2007. 89, 100, 149

[183] E. P. Wigner, *The unreasonable effectiveness of mathematics in the natural sciences. Richard Courant lecture in mathematical sciences delivered at New York University, May 11, 1959*, Comm. Pure Appl. Math., 13 (1960), pp. 1–14. 1

[184] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965. 81

[185] S. Williams, A. Waterman, and D. Patterson, *Roofline: An insightful visual performance model for multicore architectures*, Commun. ACM, 52 (2009), p. 65–76. 20

[186] H.-G. Xu, *A backward stable hyperbolic QR factorization method for solving indefinite least squares problem*, J. Shanghai Univ., 8 (2004), pp. 391–396. 115

[187] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya, *Roundoff error analysis of the Cholesky QR2 algorithm*, Electron. Trans. Numer. Anal., 44 (2015), pp. 306–326. 98

[188] I. Zolotarev, *Application of elliptic functions to questions of functions deviating least and most from zero*, Zap. Imp. Akad. Nauk. St. Petersburg, 30 (1877), pp. 1–59. Reprinted in his Collected Works. 139