BERGISCHE UNIVERSITÄT WUPPERTAL

Spectral Projection

Robustness and Orthogonality Considerations

Martin Galgon

Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

Vorgelegt an und genehmigt von der
Fakultät für Mathematik und Naturwissenschaften
der Bergischen Universität Wuppertal

Gutachter:
Prof. Dr. Bruno Lang
Prof. Dr. Andreas Frommer

Prüfungskommission:
Prof. Dr. Bruno Lang
Prof. Dr. Andreas Frommer
Prof. Dr. Matthias Bolten
Prof. Dr. Norbert Eicker

Wuppertal, 24. Juni 2020
— Datum der mündlichen Prüfung —

DOI: 10.25926/xffw-rb50
[https://doi.org/10.25926/xffw-rb50]

Danksagung

Ich möchte einer Reihe von Personen danken, welche auf die eine oder andere Weise zum Gelingen dieser Dissertation beigetragen haben.

Kathrin Haselmeier
Nils Kintscher
Sonja Sokolović
Matthias Hüsken
Stephanie Friedhoff
Niklas Manderla
Akihiro Ida
Gerhard Wellein
Thomas Manderla
Andreas Alvermann
Sarah Huber
Claudia Schimmel
Masatoshi Kawai
Georg Hager
Toshiyuki Imamura
Peter Feuerstein
Kira Maag
Ludger Humbert
Karsten Kahl
Marcel Schweitzer
Ina Manderla
Manuel Tsolakis
Melven Röhrig-Zöllner
Akira Imakura
Holger Arndt
Cyan, Inc.
Yasunori Futamura
Andreas Frommer
Bruno Lang
Christie Alappat
Dorothee Müller
Hannah Rittich
Jonas Thies
Holger Fehske
Matthias Bolten
Moritz Kreutzer
Faisal Shahzad
Jan Hahne
Sebastian Birk
Björn Leder
Lukas Krämer
Norbert Eicker
Andreas Pieper
Elke Just
Artur Strebel
Achim Basermann
Nemanja Božović
Daniela Ebeling
Brigitte Schultz
Katharina Manderla
Anja Galgon
Peter Langer
Teodor Nikolov
Ian Zwaan
Valeriy Manin
Robin Chan
Aya Motohashi
Dieter Galgon
Kengo Nakajima
Matthias Rottman
Annemarie Galgon
Daniel Losch
Gustavo Ramirez
Tetsuya Sakurai
Paul Zingler
Dominik Ernst

Vielen Dank!

Wuppertal, 11. Mai 2020 Martin Galgon

Abstract

This work deals with a special incarnation of subspace iteration—spectral projection—in order to solve Eigenproblems of standard or generalized form, given by Hermitian matrices or definite matrix pairs. After establishing the general theory, a selection of possible approximations of the ideal filtering function to obtain an approximate projector for a specified spectral target interval is highlighted, and many aspects of the convergence behavior of the resulting methods and its pathological peculiarities are analyzed experimentally. The work touches on implementation aspects and briefly introduces an accompanying software framework for parallel solution of said eigenproblems on large hybrid parallel supercomputers with many cores. In the remainder of the work, the major crucial implication of a parallel implementation that aims to subdivide the target interval for independent computation, the maintenance or reestablishment of orthogonality among the computed eigenvectors of all subintervals, are examined. Many aspects of possible approaches are presented and their feasibility is evaluated.

Zusammenfassung

Diese Arbeit beschäftigt sich mit einer speziellen Inkarnation der Unterraumiterationsmethode—spektraler Projektion—zur Lösung von Eigenproblemen in Standard- oder verallgemeinerter Form, bestehend aus hermiteschen Matrizen oder definiten Matrixpaaren. Die grundlegende Theorie wird etabliert, eine Auswahl möglicher Näherungen der idealen Filterfunktion zur Konstruktion eines approximativen Projektors für ein vorgegebenes spektrales Zielintervall präsentiert, und viele Aspekte des Konvergenzverhaltens der resultierenden Methode sowie seine pathologischen Absonderlichkeiten durch geeignete Experimente analysiert. Die Arbeit beleuchtet Implementierungsaspekte und eine begleitende Software zur Lösung besagter Eigenprobleme auf großen, hybrid-parallelen Supercomputern mit einer großen Anzahl Kerne wird kurz vorgestellt. Nachfolgend wird die wichtigste Implikation einer parallelen Implementierung, welche versucht, das Zielintervall zur unabhängigen Berechnung in Teilintervalle zu unterteilen, nämlich das Aufrechterhalten oder die Wiederherstellung der Orthogonalität zwischen den Eigenvektoren aller berechneter Teilintervalle, untersucht. Viele Aspekte möglicher Vorgehensweisen werden präsentiert und ihre Verwendbarkeit bewertet.

Die Arbeit des Autors wurde unterstützt vom Bundesministerium für Bildung und Forschung innerhalb des Projektes *ELPA - Eigenwert-Löser für Petaflop-Anwendungen*, Förderkennzeichen 01IH08007B, und *ELPA-AEO - Algorithmische Erweiterungen und Optimierungen*, Förderkennzeichen 01IH15001B, sowie von der Deutschen Forschungsgemeinschaft im Rahmen des Schwerpunktprogrammes *Software for Exascale Computing (SPPEXA)*, in den Unterprojekten *ESSEX* und *ESSEX-II* (SPP 1648).

Contents

Nomenclature 0

1 Introduction 1

1.1 Eigenvalue problems . . . 3
1.1.1 Scalar product and orthogonality . . . 4
1.1.2 Norms . . . 5
1.1.3 Hermitian and positive definite matrices . . . 6
1.2 Vector iteration methods . . . 7
1.2.1 The family of power iteration methods . . . 7
1.2.1.1 Definite pairs . . . 8
1.2.2 Eigendecomposition . . . 8
1.2.2.1 Definite pairs . . . 10
1.2.2.2 Power iteration . . . 12
1.3 Subspace iteration . . . 14
1.3.1 Vector projection and orthogonalization . . . 15
1.3.2 Gram-Schmidt orthonormalization . . . 16
1.3.2.1 Definite pairs . . . 18
1.3.3 Invariant subspaces . . . 18
1.3.3.1 Power iteration . . . 19
1.3.3.2 Definite pairs . . . 19
1.3.4 Rayleigh-Ritz . . . 19
1.3.4.1 Power iteration . . . 21
1.3.4.2 Definite pairs . . . 22
1.3.5 Compatible pencils and harmonic Rayleigh-Ritz . . . 22
1.4 Spectral filters . . . 23
1.4.1 Projection and projectors . . . 23
1.4.1.1 Definite pairs . . . 23
1.4.2 Spectral projectors . . . 24
1.4.2.1 Definite pairs . . . 25
1.5 Matrix functions . . . 25
1.5.1 Filtering functions and approximate projectors . . . 26
1.6 Iteration and convergence . . . 27
1.6.1 Convergence . . . 27
1.6.2 Metrics and residual . . . 28
1.6.3 Residual bounds . . . 29
1.7 Digression: Related algorithms . . . 30
1.7.1 Orthogonal iteration . . . 30

1.7.2 The Arnoldi and Lanczos methods . . . 32
1.7.2.1 Definite pairs . . . 33
1.7.2.2 Restarts . . . 33
1.7.3 Other methods . . . 33

2 Spectral projection algorithms **37**
2.1 Polynomial filters . . . 40
2.1.1 Polynomial approximation . . . 40
2.1.2 Chebyshev polynomials . . . 42
2.1.2.1 Window function . . . 43
2.1.2.2 Application . . . 43
2.1.3 Matrix polynomials . . . 44
2.1.4 Gibbs phenomenon and smoothing kernels . . . 45
2.1.5 Restrictions . . . 45
2.1.6 Discrete approximation . . . 48
2.2 Contour integration . . . 49
2.2.1 Representation of the window function . . . 50
2.2.2 Numerical integration . . . 51
2.2.2.1 Trapezoidal rule . . . 51
2.2.2.2 Midpoint rule . . . 52
2.2.2.3 Gauss-Legendre quadrature . . . 52
2.2.3 Computation . . . 54
2.2.4 Selection function . . . 56
2.3 Zolotarev approximation . . . 57
2.3.1 Approximation of the sign function . . . 58
2.3.2 Window function . . . 60
2.3.3 Partial fraction form . . . 63
2.3.4 Computation of Jacobi's elliptic functions . . . 68
2.4 Electronic filter design . . . 69
2.4.1 Butterworth Filter . . . 73
2.4.1.1 Design . . . 74
2.4.2 Chebyshev Type-I Filter . . . 74
2.4.2.1 Design . . . 75
2.4.3 Chebyshev Type-II Filter . . . 76
2.4.3.1 Design . . . 78
2.4.4 Elliptic Filter . . . 78
2.4.4.1 Computation of inverse elliptic functions . . . 82
2.4.4.2 Design . . . 82
2.5 Sakurai-Sugiura-type methods . . . 83
2.5.1 A root finding method for analytic functions . . . 83
2.5.2 An improved root finding method . . . 85
2.5.3 Application to generalized eigenvalue problems . . . 87
2.5.3.1 Blocking and Eigenvectors . . . 87
2.5.3.2 Rayleigh-Ritz . . . 90

2.5.3.3 Filtering functions 91
2.5.3.4 Transformations of the weight function 91
2.5.3.5 Computation 92
2.5.3.6 Relation to spectral projection 92
2.6 Other filtering methods 93
2.6.1 Rational filter types 93
2.6.2 Delta filters 94
2.6.3 Least squares 94
2.6.4 Beyond conventional subspace iteration 94

3 BEAST 95
3.1 The BEAST framework 95
3.1.1 Basics 96
3.1.2 Parallelism 96
3.1.2.1 Process group hierarchy 100
3.1.2.2 Interval communication protocol 101
3.1.3 Orthogonalization layer 103
3.2 Meet the BEAST – short feature overview 104
3.2.1 Related features 104
3.2.2 Linear solvers 104
3.2.2.1 Callback interface 105
3.2.2.2 A (hybrid) parallel direct solver for banded linear systems 105
3.2.3 Unrelated features 107
3.2.4 Layer structure 110
3.3 Feeding the BEAST – Applications 110

4 Taming the BEAST – Quality of results 113
4.1 Experiments 116
4.1.1 Synthesis of non-trivial sparse definite matrix pencils with predefined spectrum 116
4.1.2 Matrix test set 118
4.1.3 Ritz value pairing 120
4.2 Convergence revisited 121
4.2.1 Undersized searchspaces 123
4.2.1.1 On-the-fly increase of searchspace size 124
4.2.1.2 Detection of undersized searchspaces 124
4.2.2 Enlarged searchspaces 125
4.2.2.1 On-the-fly restriction of searchspace size 127
4.2.2.2 Locking 128
4.2.2.3 Rank deficiency 128
4.2.2.4 Further reduction 132
4.2.2.5 Optimization of searchspace size 133
4.2.2.6 Addendum 134

4.2.3 SSM effective filter . . . 135
4.2.4 Ritz phantoms . . . 139
4.2.4.1 Disturbance of convergence . . . 141
4.2.5 Clustered eigenvalues . . . 144
4.2.5.1 Quasi-multiplicity and Quasi-eigenspaces . . . 144
4.2.5.2 Expulsion of clustered values . . . 145
4.2.5.3 Clustered super convergence . . . 148
4.2.5.4 Disturbances through Ritz phantoms . . . 150
4.2.6 Comparison of filter functions . . . 150
4.2.6.1 Definition of transition band and stopband gain . . 150
4.2.6.2 Fair conditions . . . 151
4.2.6.3 Strict searchspace size . . . 153
4.2.6.4 Relaxed searchspace size . . . 154
4.2.6.5 Choice of interval . . . 157
4.2.7 Difficulty of linear systems . . . 157
4.3 Achievable residual . . . 160
4.3.1 Absolute rock bottom . . . 160
4.3.1.1 Mixed precision . . . 164
4.3.1.2 Linear solving effort . . . 165
4.3.2 Kick-off residual . . . 166
4.3.3 Saturation residual . . . 167
4.4 Termination criteria . . . 169
4.5 Achievable orthogonality . . . 173
4.5.1 Theoretical limit . . . 174
4.5.2 Estimated upper bound . . . 176
4.5.3 Directional overlap . . . 177
4.5.3.1 Verification of orthogonality bounds . . . 182
4.6 Evolution of orthogonality . . . 185
4.7 Remarks . . . 188

5 Taming the BEAST – Orthogonalization 189
5.1 The many orthogonalities of BEAST . . . 189
5.2 Establishing intra-orthogonality . . . 190
5.2.1 QR decomposition . . . 191
5.2.2 QR via Givens rotations . . . 191
5.2.3 QR via Householder reflections . . . 192
5.2.4 Gram-Schmidt QR . . . 194
5.2.5 B-orthogonality . . . 194
5.2.6 Cholesky QR . . . 195
5.2.7 Rank revelation . . . 195
5.2.8 Methods based on singular value decomposition . . . 196
5.2.9 SVQB . . . 196
5.2.10 B-orthogonal QR via SVQB . . . 197
5.2.11 Parallel block-orthogonalization . . . 197

5.2.12 TSQR . . . 197
5.2.13 Condition and multi-orthogonalization . . . 199
5.2.14 Weak Gram-Schmidt . . . 200
5.2.14.1 Computational effort . . . 207
5.2.15 Residual . . . 208
5.2.16 Fitness for purpose . . . 210
5.3 Establishing inter-orthogonality . . . 213
5.3.1 Intermediate orthogonalization . . . 214
5.3.2 Post-iteration and retention of orthogonality . . . 219
5.3.3 Overlapping intervals . . . 221
5.3.4 Orthogonalization sequences . . . 221
5.3.5 Block Gram-Schmidt . . . 222
5.3.5.1 Parallel application . . . 222
5.3.6 Residual . . . 224
5.3.7 Orthogonalization order . . . 225
5.3.8 A posteriori orthogonalization . . . 229
5.3.9 Detection and removal of duplicates . . . 234
5.3.10 Butterfly orthogonalization . . . 240
5.3.11 Block weak Gram-Schmidt . . . 242
5.3.12 Butterfly results . . . 243
5.3.13 Regular residual updates . . . 246
5.3.14 Skipping intra-orthogonalization . . . 247
5.3.15 Reduction of interaction distance . . . 250
5.3.16 Large examples . . . 252
5.3.17 Shifting scheme . . . 254
5.3.18 Application-specific requirements . . . 254
5.3.19 Performance . . . 259
5.4 Upshot . . . 261

Conclusion and outlook **263**
Conclusion . . . 263
Outlook . . . 265

A Algorithms for elliptic functions **267**
A.1 Computation of Jacobi's elliptic functions . . . 267
A.1.1 The Algebraic-Geometric Mean . . . 268
A.2 Computation of inverse elliptic functions . . . 271

Index **273**
List of Algorithms **277**
List of Experiments **279**
List of Figures **281**
List of Tables **285**
Bibliography **287**

Nomenclature

This list describes certain symbols with special meaning or properties used throughout this thesis.

A, B	Matrices of the eigenproblem
x, X	Eigenvector, block of eigenvectors (typically of (A, B))
λ, Λ	Eigenvalue, diagonal matrix of eigenvalues (typically of (A, B))
$\boldsymbol{\lambda}$, $\boldsymbol{x}$, $\boldsymbol{\Lambda}$, $\boldsymbol{X}$	Computed quantities (typically Ritz values and -vectors)
P, $\boldsymbol{P}$	Projector, approximate projector
δ, Δ	Eigenvalues of an (approximate) projector (scalar, diagonal matrix)
σ, Σ	Singular values, diagonal matrix of singular values
$[\lambda_{\min}, \lambda^{\max}]$	Target interval
Ω	Target region
$\mathcal{C} = \partial\Omega$	Target contour
χ_Ω, χ_ω	Window function (ideal filter)
$f(\lambda)$	Filtering function, eigenvalues of an approximate projector
U	Searchspace set or searchspace basis
n	System size, dimension of the matrices A and B
m	Number of columns of a matrix or a vector block
m_0	Searchspace size
κ	Modulus of an elliptic function
$\mathbf{i}$, $\Re$, $\Im$	Imaginary unit, real part, imaginary part
ε	Machine precision
I	Unit matrix
$\langle\cdot,\cdot\rangle$, $\lVert\cdot\rVert$	Scalar product, norm of a vector or matrix
$\cdot^H$, $\overline{\cdot}$	Hermitian conjugated vector or vector block, conjugated scalar
$\lfloor\cdot\rfloor$, $\lceil\cdot\rceil$	Rounding downwards or upwards, respectively

Chapter 1
Introduction

The solution of eigensystems is among the most everlasting problems of numerical mathematics, owed to its quasi-omnipresence in applications throughout natural sciences, engineering and other fields. We will exemplarily touch on just a few.

The most prominent field for eigenproblems to play a significant role is likely oscillation and vibration analysis, be it statics in architecture, string vibration in music, signal processing, acoustics in general, or machine engineering. Oscillations are often modeled as constructs of springs, masses and dampers (see any vibration analysis book, e.g., [Tim37]). Assuming an exponential solution for the resulting system of differential equations ultimately leads to a matrix formulation that resembles an eigenequation comprised of symmetric mass and stiffness matrices. Eigenproblems occur frequently in the computation of differential equations [Eri+96].

In the Dirac [Dir58] and von Neumann [Neu71] formulations of quantum mechanics, the state of a system of physical variables, such as position, energy, or momentum, is represented by a vector in a complex Hilbert space, typically a function space, allowing for superposition in terms of linear combination. The dynamic variables are described as self-adjoint linear operators and the measured physical values of observables, real-valued dynamic variables represented by Hermitian operators, are the (then also real) spectral values of the operator. Measuring an observable brings the system (with a certain probability) into one of the eigenstates of this observable, also called wavefunction collapse, such that consecutive measurements of the same variable yield identical results. An example is the Hamiltonian operator corresponding to the total energy of the system as appearing in the Schrödinger equation. With discretization, the operators become Hermitian matrices and the possible outcomes of measurements become its eigenvalues.

In this or similar manner, Eigenproblems also appear in quantum mechanical material science, such as the analysis of the electronic properties of graphene [Net+09] and topological insulators [HK10] where they relate to electric conductivity or edge magnetism, molecular simulations [Blu+09] where they have to be solved in order to compute observables such as total energy, stress, or electrostatic moments, or the Anderson model of localization [Els+99] where they represent quantum mechanical energy levels.

The Google PageRank [Pag+98] is another example that is cited frequently as application for the power method. To quantify the importance of a web page and thus its relevance as result when a search is conducted, a measure based on the connectivity between web pages via hyperlinks is derived that also models the propagation of importance through these connections. The final rankings are then computed as the dominant eigenvector of the adjacency matrix of the directed weighted graph representing the web connectivity.

Due to ever growing problem sizes required in computations and simulations to model or observe the desired effects, the demand for suitable hard- and software creates a continuous inflow of new challenges for researchers in the field. With the foreseeable advent of exascale computers, not only are challenges posed to hardware manufacturers. Also, algorithms have to be restructured or even reinvented to exploit the vast computing potential of modern hybrid parallel high performance computer architectures. With the potential for parallelism not being infinitely extensible, already the choice of algorithm plays an important role, not all being exascale capable or even properly parallelizable at all.

For many, if not most, applications not all eigenvalues and eigenvectors are required. While the computation of the complete eigenproblem is expensive, both in terms of computation time and—maybe more significantly—memory consumption, obtaining just the required portion of the eigenvalues and eigenvectors exhibits potential for optimization of both, tailored to the respective application. Depending on the application, this may translate to one extremal eigenpair, few largest or least magnitude eigenpairs, or sometimes larger portions of the interior of the spectrum. Methods like shift-and-invert power iteration and the Jacobi-Davidson method [SV00], for example, allow to compute a set of eigenpairs close to a specified target somewhere in the spectrum of (A, B).

With many applications and new challenges for eigensolvers, we aim at solving large-scale and sparse standard and generalized eigenproblems. While we ultimately restrict ourselves to Hermitian eigenproblems due to a more complete theory and the widespread applicability, many of the methods introduced here are not inherently restricted to Hermitian problems. Since it can be expected that—with increasing problem size—direct solvers will grow increasingly infeasible in terms of memory consumption and processing time, given that in general the whole spectrum is computed, we will focus on a family of iterative eigensolver algorithms—*spectral projection*—that is able to compute only certain parts of a spectrum and exploit the sparsity of the involved matrices.

This chapter will introduce the problem of finding eigenvalues and eigenvectors of (A, B) alongside fundamental algorithms and mathematical foundations required to solve such *eigenproblems* using subspace iteration and spectral projection. Many of the standard terms, definitions, concepts and methods, which are pragmatically introduced in this and the following chapters, such as scalar products, norms, metrics, matrix pencils, eigenvalue problems, eigendecomposition, Schur form, matrix functions, matrix square roots, matrix similarity, the power method, inverse iteration, subspace iteration, and spectral projection can be found and are discussed more in-depth in the relevant literature, including [Wil65; Par80; SS90; TB97; Ste01; Wat07; Hig08; Saa11; GV13; Krä14], just to name a few. The elements are introduced in the order the discussion of spectral projection requires it.

1.1 Eigenvalue problems

Let $A_1, \ldots, A_k \in \mathbb{C}^{n\times n}$ and $\xi_1, \ldots, \xi_k \in \mathbb{C}$. The linear combination

$$\sum_{i=1}^{k} \xi_i A_i$$

is called *matrix pencil* of degree k. Let in particular the matrix pencil $A - \lambda B$ be denoted by (A, B). We will refer to this particular pencil simply as *the matrix pencil*. Non-zero vectors $x \in \mathbb{C}^n \neq 0$ and scalars $\lambda \in \mathbb{C}$ that fulfill

$$(A - \lambda B)x = 0$$

are called *eigenvectors* and *eigenvalues* of (A, B), respectively. A pair (x, λ) is referred to as *eigenpair*. If x is an eigenvector, so is αx for $\alpha \in \mathbb{C} \setminus \{0\}$. Should $x_1, \ldots, x_k$ be eigenvectors associated with the same eigenvalue λ, any linear combination $\sum_{i=1}^{k} \alpha_i x_i \neq \mathbf{0}$ is an eigenvector as well. For a specific eigenvalue λ, the space containing these vectors, i.e., the null space of $A - \lambda B$, is called the *eigenspace* of the matrix pencil associated with the eigenvalue λ. Eigenvalues are the roots of the *characteristic polynomial*

$$p_{(A,B)}(\lambda) = \det(A - \lambda B).$$

Definition 1.1 *Let matrices $A, B \in \mathbb{C}^{n\times n}$, $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, $\lambda_i \in \mathbb{C}$, and linearly independent vectors $X = [x_1, \ldots, x_n] \in \mathbb{C}^{n\times n}$, $x_i \neq \mathbf{0}$; then the equation*

$$AX = BX\Lambda \tag{1.1}$$

is called generalized eigenproblem *for the matrix pencil (A, B). If $B = I$, Equation (1.1) becomes the* standard eigenproblem $AX = X\Lambda$. *The conglomeration of some or all of the solutions is the pair (X, Λ).*

The set of all eigenvalues of a matrix pencil (A, B) is called its *spectrum*, denoted by $\operatorname{spec}(A, B)$. The eigenvalues are not necessarily distinct, and unless otherwise noted we assume them ordered $|\lambda_1| \geq \ldots \geq |\lambda_n|$. Eigenvalues may appear more than once, with a given *multiplicity*. We differentiate *algebraic multiplicity*—the number of appearances of an eigenvalue and the order of the respective root of the characteristic polynomial—and *geometric multiplicity*—the dimension of the associated eigenspace. Eigenvalues that appear more than once but whose eigenspace is of lower dimension are called *degenerate*. Square matrices have exactly n eigenvalues, due to $B = I$ in the characteristic polynomial. If B is singular, however, the characteristic polynomial can be of degree $< n$ and the matrix pencil has less than n eigenvalues. If even $p_{(A,B)}(\lambda) \equiv 0$, any value in $\mathbb{C}$ satisfies the conditions for being an eigenvalue.

For this reason B will henceforth be assumed to be invertible. Then the transformation of the generalized eigenproblem $AX = BX\Lambda$ to a standard eigenproblem

$$B^{-1}AX = X\Lambda$$

exists and shares its solution (X, Λ). Certain properties of the original problem may be lost, e.g., in that $B^{-1}A$ in general is neither sparse nor Hermitian (see Section 1.1.3), even if A and B are.

While we will try to introduce all concepts in a context-sensitive manner, the following two are too fundamental, even in a pragmatic introduction such as this.

1.1.1 Scalar product and orthogonality

Let $v = [v_1, \dots, v_n]^T \in \mathbb{C}^n$ and $w = [w_1, \dots, w_n]^T \in \mathbb{C}^n$. A scalar product is a mapping $\langle \cdot, \cdot \rangle : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}$ with properties

- $\langle v, v \rangle \geq 0$ (with equality if and only if $v = \mathbf{0}$)
- $\langle v, w \rangle = \overline{\langle w, v \rangle}$
- $\langle \alpha v + \beta w, z \rangle = \alpha \langle v, z \rangle + \beta \langle w, z \rangle$

The standard scalar product of the Cartesian coordinates of the vectors v and w,

$$\langle v, w \rangle = \sum_{i=1}^{n} v_i \overline{w_i} = w^H v,$$

is also known as *Euclidean scalar product*. Two non-zero vectors v and w are called *orthogonal*, if $\langle v, w \rangle = 0$. A set of vectors $Q = [q_1, \dots, q_k]$ is called orthogonal, if all vectors are pairwise orthogonal, i.e., $Q^H Q = \operatorname{diag}\{d_1, \dots, d_k\}$.

It is possible to define different scalar products that fulfill these criteria; but in this context only the standard scalar product, as defined above, will be used, though with a small extension. The scalar product with respect to a Hermitian positive definite (see Section 1.1.3) matrix B, or B-scalar product, with the same properties is given as

$$\langle v, w \rangle_B = \langle Bv, w \rangle = w^H B v.$$

Analogously, two non-zero vectors v and w are called *B-orthogonal* if $\langle v, w \rangle_B = 0$. A set of vectors $Q = [q_1, \dots, q_k]$ is called B-orthogonal if all vectors are pairwise B-orthogonal, i.e., $Q^H B Q = \operatorname{diag}\{d_1, \dots, d_k\}$. Henceforth, the term orthogonal will implicitly refer to B-orthogonal. Should it be necessary to differentiate, we will refer to orthogonality in the sense of $B = I$ as I-orthogonality.

1.1.2 Norms

As quantification of the length of a vector, a map $\|\cdot\| : \mathbb{C}^n \to \mathbb{C}$ with properties

- $\|v\| \geq 0$
- $\|\alpha v\| = |\alpha|\|v\|$
- $\|v + w\| \leq \|v\| + \|w\|$
- $\|v\| = 0 \implies v = \mathbf{0}$

assigns a non-negative scalar value to the vector. Every scalar product induces a norm

$$\|v\| = \sqrt{\langle v, v\rangle}.$$

In the case of the standard scalar product, this is the 2-norm $\|\cdot\|_2$, here sometimes with respect to B, $\|\cdot\|_B$, induced by the B-scalar product. Combined with the norm, the geometric interpretation of the scalar product often serves as a measure of the angle between two vectors,

$$\cos(\theta) = \frac{\langle v, w\rangle}{\|v\|\|w\|}.$$

A set of vectors $Q = [q_1, \ldots, q_k]$ is called *unitary*, if $Q^H BQ = I_k$, i.e., Q is orthogonal and $\|q_i\| = 1\ \forall i = 1, \ldots, k$. The equivalent term for real matrices is *orthonormal*, which we will use here for the real and complex case. We will refer to $Q^H B$ as *left inverse* of Q. In the case of square Q this is the regular inverse $Q^{-1} = Q^H B$.

Given a vector norm on a space $\mathcal{S}$ and letting $v \in \mathcal{S}$, a corresponding matrix norm is defined as

$$\|Z\| = \max_{v \neq \mathbf{0}} \frac{\|Zv\|}{\|v\|} = \max_{\|v\|=1} \|Zv\|.$$

Typically $\mathcal{S} = \mathbb{C}^n$, but if Z is real $\mathcal{S} = \mathbb{R}^n$ [Krä14]. The induced operator norm for a matrix Z is the maximum scaling that occurs when applied to any unit-length vector. For general (possibly non-square) matrices, these factors are given by the singular values of Z (see also Section 5.2.8). With singular values $\sigma_1, \ldots, \sigma_n$ of Z, sorted descendingly,

$$\|Z\| = \sigma_1$$

and, if Z is Hermitian (see Section 1.1.3),

$$\|Z\| = |\lambda_1|.$$

Consider the singular value decomposition[1] $ZW = V\Sigma$. For single vectors w_i and v_i from W and V, respectively, both of unit length, we write $Zw_i = \sigma_i v_i$. The length of the resulting vector then is σ_i. Any non-singular vector may be written as linear combination of the singular vectors W and, after normalization, the length of the resulting vector cannot exceed the maximum singular value. Since $Z^H Z =$

[1] The SVD will be discussed in a little more detail in Section 5.2.8.

$W\Sigma V^H V\Sigma W^H = W\Sigma^2 W^H$, and thus $Z^H Z W = W\Sigma^2$, singular values of Z are the square roots of the eigenvalues of $Z^H Z$. For Hermitian matrices with $Z = X\Lambda X^H$ we have $Z^H Z = X\Lambda X^H X\Lambda X^{-1} = X\Lambda^2 X^H$ and we see that the singular values of Z are the absolute values of the eigenvalues of Z.

1.1.3 Hermitian and positive definite matrices

A matrix is called *Hermitian*, if $M = M^H$ holds. A Hermitian matrix has an all-real spectrum and orthogonal eigenvectors, as we will see in Section 1.2.2. A Hermitian matrix is called *positive definite*, if $v^H M v > 0$ holds for all $v \in \mathbb{C}^n \setminus \{\mathbf{0}\}$, and *positive semi-definite*, if $v^H M v \geq 0$ holds under identical conditions. Note that positive definiteness is exactly the requirement of positivity of the scalar product. A Hermitian matrix is positive definite, if and only if all eigenvalues are greater than zero. This is easy to verify. For eigenvectors x_i, $i = 1, \ldots, n$, of a matrix M, due to the eigenequation, $\lambda x_i^H x_i = x_i^H M x_i$. With the positivity of the scalar product $x_i^H x_i$ follows $\lambda > 0 \iff x_i^H M x_i > 0$. Any other vector v can be written as linear combination of the eigenvectors x_i,

$$v = \sum_{i=1}^{n} \xi_i x_i.$$

Then

$$Mv = \sum_{i=1}^{n} \xi_i M x_i = \sum_{i=1}^{n} \xi_i \lambda_i x_i$$

and

$$v^H M v = \sum_{i=1}^{n} \sum_{j=1}^{n} \xi_i \overline{\xi_j} \lambda_i x_j^H x_i$$

where $x_j^H x_i = 0$ for $i \neq j$, assuming the orthogonality of eigenvectors. Therefore

$$v^H M v = \sum_{i=1}^{n} |\xi_i|^2 \lambda_i x_i^H x_i.$$

With the positivity of the scalar product follows $v^H M v > 0$ if all $\lambda_i > 0$. Since the condition of positivity must also hold for the single summands—$v^H M v > 0$ shall hold for all vectors v, in particular for eigenvectors $v = x_i$—the other direction follows immediately.

The equivalent of Hermitian matrices for standard eigenvalue problems are *definite pairs* (A, B) with Hermitian A and Hermitian positive definite (hpd) B for the generalized eigenproblem. Definite pairs have all-real spectra and their eigenvectors are B-orthogonal (see Section 1.2.2). Then the eigenvector matrix X has the inverse $X^{-1} = X^H B$ and $X^H B X = X X^H B = I$. In the case of a Hermitian matrix or a definite pair we will also denote the associated eigenproblem as *Hermitian eigenproblem*. Hermitian eigenproblems often allow for simplifications, and in some cases certain operations only work for Hermitian eigenproblems. Whenever definite pairs influence the introduced concept, a short paragraph will detail the implications.

⊞

1.2 Vector iteration methods

Due to the fundamental importance of eigenvalue problems, many algorithms to solve them have been proposed over time. One of the most commonly known algorithms is most likely the power iteration. It is also one of the simplest methods to compute a dominant eigenvector both algorithmically and numerically, as it only requires multiplication with a matrix and normalization of a vector to iterate an approximate eigenvector. A dominant eigenvector in this case is an eigenvector associated with an eigenvalue that is larger than all other eigenvalues. In the following, a family of iterative eigensolver algorithms closely related to the power iteration method is introduced. It will serve as vehicle to introduce the underlying concepts, required to understand these kinds of methods as well as possible consequences for solving eigenproblems. The fundamental idea is then extended to subspace iteration and finally to spectral projection. The term *iteration* will refer to the repeated application of such an algorithm to one or several vectors. We will therefore say that we *iterate* these vectors or their associated directions and call distinct instances from this sequence of vectors *iterates*.

1.2.1 The family of power iteration methods

Instead of introducing all variants of power iteration separately, a generalized algorithmic frame is presented in Algorithm 1.1, where specific incarnations are produced by choosing an operator C from Table 1.1.

```
Input: Random vector $\boldsymbol{x}_0 = v_0$
Output: Approximate eigenpair $(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$
1: for $i = 1, \ldots, k$ do
2:     apply $C$ to $\boldsymbol{x}_{i-1}$ and obtain $\hat{\boldsymbol{x}}_i$
3:     $\boldsymbol{x}_i \leftarrow \frac{\hat{\boldsymbol{x}}_i}{\|\hat{\boldsymbol{x}}_i\|}$
4: $\boldsymbol{\lambda}_k \leftarrow \frac{\boldsymbol{x}_k^H A \boldsymbol{x}_k}{\boldsymbol{x}_k^H B \boldsymbol{x}_k}$
```

Algorithm 1.1: *Generic power iteration.*

The different methods, resulting from choosing an operator C and its application from Table 1.1, constitute the family of power iteration methods. They can be used for computing the distinct eigenpair of C with the eigenvalue of largest magnitude. Depending on the operator, the method targets different eigenpairs of (A, B). The *shift-inverse iteration* computes the eigenpair with eigenvalue closest to a given shift $z \in \mathbb{C}$. If the shift z is 0, the method becomes the *inverse iteration*, used to compute the distinct eigenpair with the eigenvalue of smallest magnitude. Without inversion, if the method is applied to $B^{-1}A$ instead, it becomes the well known *power iteration*,

Name	Operator C	Application
Power iteration	$B^{-1}A$	solve $B\hat{\boldsymbol{x}}_i = A\boldsymbol{x}_{i-1}$ for $\hat{\boldsymbol{x}}_i$
Shifted iteration	$B^{-1}A - zI$	solve $B\hat{\boldsymbol{x}}_i = (A - zB)\boldsymbol{x}_{i-1}$ for $\hat{\boldsymbol{x}}_i$
Inverse iteration	$(B^{-1}A)^{-1}$	solve $A\hat{\boldsymbol{x}}_i = B\boldsymbol{x}_{i-1}$ for $\hat{\boldsymbol{x}}_i$
Shift-inverse iteration	$(B^{-1}A - zI)^{-1}$	solve $(A - zB)\hat{\boldsymbol{x}}_i = B\boldsymbol{x}_{i-1}$ for $\hat{\boldsymbol{x}}_i$

Table 1.1: *Power iteration flavors and methods of application.*

here for generalized eigenproblems. It then computes the distinct eigenpair with the largest magnitude eigenvalue. The *shifted iteration* has fewer practical uses but can be employed to compute "the other end of the spectrum", assuming the largest magnitude eigenvalue is known to determine the required shift.

All these methods are only suitable to compute one eigenpair unless additional measures are taken. Note that any norm may be used for normalizing the iterates as eigenvectors are ambiguous with regard to their length. Also note that, following from the eigenequation, the choice of $\boldsymbol{x}^H$ for computing the quotient is arbitrary.

Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of C. The methods converge to the eigenvector associated with λ_1 only if $\lambda_1 > \lambda_2$, e.g., one eigenvalue is larger in magnitude than all others (see Equation (1.2)).

1.2.1.1 Definite pairs

If B is Hermitian and positive definite, the B-norm exists and for a B-normal $\boldsymbol{x}$ then $\boldsymbol{x}^H B\boldsymbol{x} = 1$. Thus, the computation of the quotient simplifies to $\boldsymbol{\lambda} = \boldsymbol{x}^H A\boldsymbol{x}$.

Having A and B Hermitian and/or positive definite may have major impact on the solution of the linear systems, in particular if an iterative solver is to be used.

1.2.2 Eigendecomposition

If the eigenvalue problem $AX = BX\Lambda$ can be written as a decomposition into X and Λ,

$$B^{-1}A = X\Lambda X^{-1},$$

this is called an *eigendecomposition* of (A, B). The effect of the application of a matrix or a matrix pencil to a vector can intuitively be understood by employing their eigendecompositions.

The above operator, applied to a vector v,

$$X\Lambda X^{-1}v = \sum_{i=1}^{n} \lambda_i x_i x_i^{-1} v,$$

where x_i^{-1} denotes the i-th column of X^{-1}, can be interpreted as scaling the projected directions with factors λ_i in the directions x_i. Geometrically, by solving for the coefficients of v in the basis X, the application of X^{-1} transforms v into the coordinates of span(X), where the scaling is applied in the basis directions x_i. Application of X reverses the basis transformation. Representing an operation on a vector with respect to different bases is referred to as *similarity transform* (see, e.g., [Saa11]) and the matrices $B^{-1}A$ and Λ are therefore called *similar*. Similar matrices have the same eigenvalues since for any invertible matrix M and $C = X\Lambda X^{-1}$

$$MCM^{-1} = MX\Lambda(MX)^{-1}.$$

While the eigendecomposition is not unique—both the order of eigenvalues and the bases for all eigenspaces may be chosen arbitrarily—the set of diagonal elements of λ is. Repeated application of the Operator,

$$\left(X\Lambda X^{-1}\right)^k = X\Lambda^k X^{-1},$$

is equivalent to repeatedly scaling v in the directions of the x_i. If $\lambda_i = 0$, the direction x_i is removed from v. If the eigendecomposition exists, the eigenvectors form a basis

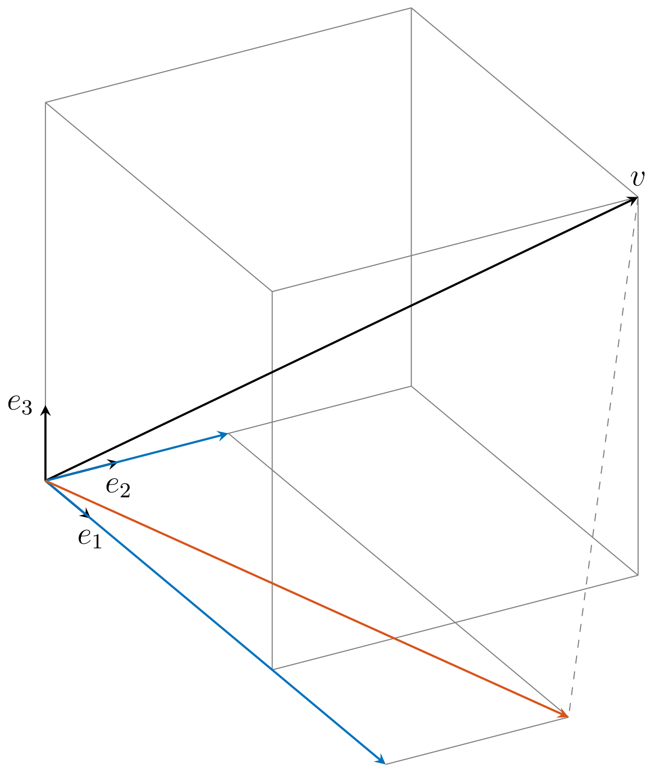


Figure 1.1: *Effect of applying a three-dimensional matrix pencil with eigenvalues* $\left\{1\frac{1}{2}, \frac{1}{2}, 0\right\}$ *and eigenvectors* e_1*,* e_2*, and* e_3 *to* $v = [5,5,5]^T$*.*

of $\mathbb{C}^n$, the so-called *eigenbasis*, and the matrix pencil is called *diagonalizable* due to X being a similarity transform that yields a diagonal matrix Λ.

If the matrix of eigenvectors X is not invertible, i.e., its columns are not linearly independent or there are too few of them, the eigendecomposition does not exist.

There are cases where the dimension of an eigenspace is smaller than the multiplicity d of the associated eigenvalue. Then it is impossible to find a linearly independent basis with dimension d for this eigenspace and there is no basis of linearly independent eigenvectors that spans $\mathbb{C}^n$. Such a matrix pencil is called *defective*.

1.2.2.1 Definite pairs

Definite pairs have an all-real spectrum and a full set of B-orthogonal eigenvectors. To verify that, we first employ the *Schur form* (see, e.g., [Saa11], incl. proof) for Hermitian matrices to verify their real spectra and full set of orthogonal eigenvectors. With an additional result from [SS90] (see also [Krä14]) we can extend the property to definite pairs.

First, verify that a square matrix C is similar to a triangular matrix T via an orthonormal transformation Q,

$$C = QTQ^{-1} = QTQ^H \implies T = Q^H CQ.$$

For a matrix of size one, the above trivially holds. Let x be a normalized eigenvector associated with an eigenvalue λ of C and $X_\perp$ its orthonormal complement such that $H = [x, X_\perp] \in \mathbb{C}^{n\times n}$. Then

$$H^H CH = \begin{pmatrix} x^H Cx & x^H CX_\perp \\ X_\perp^H Cx & X_\perp^H CX_\perp \end{pmatrix} = \begin{pmatrix} \lambda x^H x & * \\ \lambda X_\perp^H x & \widetilde{C} \end{pmatrix} = \begin{pmatrix} \lambda & * \\ 0 & \widetilde{C} \end{pmatrix}.$$

Assume $\widetilde{C}$ has the Schur form $\widetilde{Q}\widetilde{T}\widetilde{Q}^H$ and $\widetilde{T} = \widetilde{Q}^H \widetilde{C}\widetilde{Q}$. Then

$$\begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q}^H \end{pmatrix} H^H CH \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q}^H \end{pmatrix} \begin{pmatrix} \lambda & * \\ 0 & \widetilde{C} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix} = \begin{pmatrix} \lambda & * \\ 0 & \widetilde{T} \end{pmatrix}$$

is triangular and

$$Q = H \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix}$$

is orthonormal if H is chosen orthonormal since

$$H \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix} \left(H \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix} \right)^H = H \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix}^H H^H = HH^H.$$

By induction follows the existence of the Schur form of a matrix C. If C is Hermitian,

$$C^H = \left(QTQ^H\right)^H = QT^H Q^H = QTQ^H = C$$

and therefore $T = T^H$ is diagonal and real. Thus, QTQ^H is an eigendecomposition of C.

To obtain a similar result for definite pairs (A, B), we first require a decomposition of the matrix B of the form $B = KK^H$. With this we have a new way to transform a generalized eigenproblem to standard form at our disposal, albeit with different eigenvectors Y, which can be transformed to eigenvectors X of the original problem by $X = K^{-H}Y$. It is

$$\begin{aligned} AX = BX\Lambda &\iff AX = KK^H X\Lambda \\ &\iff K^{-1}AK^{-H}K^H X = K^H X\Lambda \\ &\iff K^{-1}AK^{-H}Y = Y\Lambda, \end{aligned}$$

where we assume that the inverse of K exists. If A and B are Hermitian, then $K^{-1}AK^{-H}$ is Hermitian, and let

$$K^{-1}AK^{-H} = Y\Lambda Y^H$$

be its eigendecomposition with orthonormal eigenvectors Y and real Λ. The eigenvectors of the original generalized eigenproblem $X = K^{-H}Y$ are B-orthogonal since

$$\left(K^{-H}Y\right)^H BK^{-H}Y = Y^H K^{-1}BK^{-H}Y = Y^H K^{-1}KK^H K^{-H}Y = Y^H Y = I$$

and both eigenproblems have the same spectrum.

An important special case is the concept of a matrix square root. For a non-indefinite diagonal matrix D with real spectrum, the square root can intuitively be defined by applying the scalar square root to the diagonal entries, yielding a matrix $D^{\frac{1}{2}}$ such that $D = D^{\frac{1}{2}}D^{\frac{1}{2}}$. Then, for any positive semi-definite diagonalizable matrix B with real spectrum, the matrix square root can be defined as

$$B = X\Lambda X^{-1} = X\Lambda^{\frac{1}{2}}\Lambda^{\frac{1}{2}}X^{-1} = \underbrace{X\Lambda^{\frac{1}{2}}X^{-1}}_{B^{\frac{1}{2}}}\underbrace{X\Lambda^{\frac{1}{2}}X^{-1}}_{B^{\frac{1}{2}}}.$$

If B is additionally Hermitian, i.e., its spectrum is non-negative and its eigenvectors orthonormal, by the above definition the square root of B is Hermitian as well,

$$\left(X\Lambda^{\frac{1}{2}}X^H\right)^H = X\left(\Lambda^{\frac{1}{2}}\right)^H X^H = X\Lambda^{\frac{1}{2}}X^H.$$

For the inverse of the matrix square root of B to exist, we have to require B to be positive definite. Note that, if a matrix $B = X\Lambda X^H$ is Hermitian, $B^{-1} = X\Lambda^{-1}X^H$ is Hermitian as well. Another instance of this type of decomposition that is more relevant for practical use due to its relatively easy and therefore cheap computation is the Cholesky decomposition (see Section 5.2.6).

Knowing that the eigenvectors are B-orthonormal, if (A, B) is a definite pair, the eigendecomposition becomes

$$B^{-1}A = X\Lambda X^{-1} = X\Lambda X^H B$$

and the operator can be written as

$$X\Lambda X^H B = \sum_{i=1}^{n} \lambda_i x_i x_i^H B.$$

When applied to a vector v, the form is reminiscent of orthogonal vector projection (see Section 1.3.1),

$$X\Lambda X^H Bv = \sum_{j=1}^{n} \lambda_i \langle v, x_j \rangle_B x_j,$$

and the intuitive perception of the eigendecomposition from above becomes even clearer.

1.2.2.2 Power iteration

With the intuitive interpretation of the eigendecomposition in the context of the power iteration method, the iterated vectors are gradually bent in the direction of the eigenvectors associated with the largest magnitude eigenvalues of the applied matrix. The interpretation of the eigendecomposition furthermore hints at an increased convergence of the power method, when the target eigenvalue is well separated from the rest, $|\lambda_1| \gg |\lambda_2|$, seeing that scaling in one direction significantly stronger than others improves the approximation of the associated eigenvector. It can indeed be shown that the power method for one vector converges with ratio $|\lambda_2|/|\lambda_1|$ (see, e.g., [GV13]; with proof).

Write the initial vector v_0 as linear combination of eigenvectors with factors γ_j,

$$v_0 = \sum_{j=1}^{n} \gamma_j x_j,$$

where we, for now, require $\gamma_1 \neq 0$. Then we can write the i-th iterate (including any normalization in the factors γ_j) as

$$A^i v_0 = \sum_{j=1}^{n} \gamma_j A^i x_j = \sum_{j=1}^{n} \gamma_j \lambda_j^i x_j.$$

Factoring out the largest magnitude eigenvalue λ_1,

$$A^i v_0 = \gamma_1 \lambda_1^i x_1 + \sum_{j=2}^{n} \gamma_j \lambda_j^i x_j = \gamma_1 \lambda_1^i x_1 + \lambda_1^i \underbrace{\left[\sum_{j=2}^{n} \gamma_j \left(\frac{\lambda_j}{\lambda_1} \right)^i x_j \right]}_{\xrightarrow[i\to\infty]{} 0}, \tag{1.2}$$

the sequence of iterates, adjusted for repeated scaling by λ_1 as $A^i v_0 / \lambda_1^i$, converges against a multiple (γ_1) of x_1, which is the eigenvector associated with the largest magnitude eigenvalue, with ratio $|\lambda_2|/|\lambda_1|$, it being the factor vanishing the slowest.

The above formulation also suggests that, if the initial guess v_0 does not contain components in the direction of the eigenvector associated with the dominant eigenvalue $\gamma_1 = 0$, the method will instead converge to the eigenvector associated with the second largest magnitude eigenvalue, if v_0 contains components in this direction. In theory, if v_0 is written as linear combination of eigenvectors with $\gamma_j = 0$ for $j < k$, the method should converge to x_k with ratio $|\lambda_{k+1}|/|\lambda_k|$.

Relating the convergence behavior to the spectrum of the different operator matrices from Table 1.1 clarifies the target eigenvalues of the methods described earlier. For example, the matrix

$$(A - zB)^{-1}B = \left(B^{-1}(A - zB)\right)^{-1} = \left(B^{-1}A - zI\right)^{-1}$$

from the first step of the shift-invert flavor of Algorithm 1.1 has the eigendecomposition

$$\begin{aligned}\left(X\Lambda X^{-1} - zI\right)^{-1} &= \left(X\Lambda X^{-1} - zXX^{-1}\right)^{-1}\\ &= \left(X(\Lambda - zI)X^{-1}\right)^{-1} = X(\Lambda - zI)^{-1}X^{-1}\end{aligned}$$

and therefore the same eigenvectors as (A, B) and eigenvalues $1/(\lambda - z)$, where λ is an eigenvalue of (A, B). This is the power method applied to a transformed eigenproblem, using the identity of eigenvectors to find eigenvalues of the original problem. Table 1.2 lists the eigenproblems related to the flavors of the power iteration (as matrix pencils) and the resulting transformed eigenvalues. The shift-and-invert

Name	Matrix pencil	Eigenvalues
Power iteration	(A, B)	λ
Shifted iteration	$(A - zB, B)$	$\lambda - z$
Inverse iteration	(B, A)	$1/\lambda$
Shift-inverse iteration	$(B, A - zB)$	$1/(\lambda - z)$

Table 1.2: *Matrix pencils of power iteration flavors and associated eigenvalues.*

operation first shifts the spectrum such that the target value z is mapped to zero, and then reflects it about the unit circle with factor $1/|z|$, so that eigenvalues closest to zero map to values with maximum absolute value, as illustrated in Figure 1.2. The conjugation that is applied implicitly by the inversion does not have an impact on the result. If z happens to be an eigenvalue, the matrix is not invertible anymore and the method breaks. However, moving z arbitrarily close to an eigenvalue of interest will speed up the convergence of the method since the shift-invert operation will increase the absolute value of the target eigenvalue significantly and separate it well from the rest as long as there is no other eigenvalue with similar absolute value. However, the condition of the matrix $A - zB$ increases accordingly. This is of particular interest, if an iterative linear system solver is to be used because this type of solver typically is sensitive to increased condition (see also Section 4.2.7).

The computation of the associated eigenvalue λ then relies on the fact that the eigenvector associated with the largest magnitude eigenvalue of $(A - zB)^{-1}$ is identical to the eigenvector associated with the eigenvalue closest to z of $B^{-1}A$ and uses the eigenequation of (A, B) to compute an approximation of the associated eigenvalue. A variation of the power method for Hermitian matrices that replaces

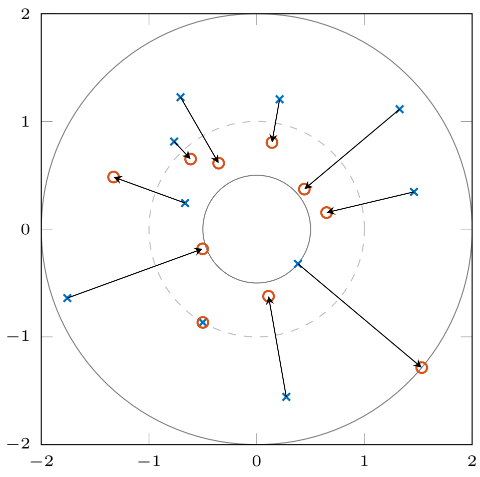


Figure 1.2: *Complex spectral inversion. The original eigenvalues are marked as* $\times$ *and the modified eigenvalues as* $\circ$. *The inversion-induced conjugation is ignored in this representation to emphasize the changes in absolute value.*

z with the approximate eigenvalue (in this context also called *Rayleigh quotient*) in each iteration, is called *Rayleigh quotient iteration*. It applies the above observation for improving performance of the shift-invert power iteration at the cost of increased condition of the linear systems.

1.3 Subspace iteration

Power iteration methods are commonly introduced as single vector iterations, but the concept can be extended to multiple vectors to find more than one eigenpair. Of course, naive simultaneous iteration of blocks of vectors $\boldsymbol{X}_i$ in Algorithm 1.1 ultimately causes all vectors to approximate the same eigenvector associated with the largest magnitude eigenvalue, as is apparent from the proof of convergence of the power method. However, the situation requires a more differentiated analysis. Considering the factors λ_j/λ_1, it is clear that directions with large difference in magnitude compared to λ_1 vanish more quickly than directions related to eigenvalues with similar magnitude. In the case of multiple largest magnitude eigenvalues $\lambda_1 = \ldots = \lambda_k < \lambda_{k+1} \leq \ldots$ the corresponding directions will not vanish at all.

Similar to before, write the i-th iterate as

$$A^i v_0 = \sum_{j=1}^{k} \gamma_j \lambda_j^i x_j + \sum_{j=k+1}^{n} \gamma_j \lambda_j^i x_j = \lambda_k^i \left[\sum_{j=1}^{k} \gamma_j \left(\frac{\lambda_j}{\lambda_k} \right)^i x_j + \sum_{j=k+1}^{n} \gamma_j \left(\frac{\lambda_j}{\lambda_k} \right)^i x_j \right], \tag{1.3}$$

with separation at some index k. Certainly, the factors λ_j/λ_k are larger than λ_j/λ_1 for $k > 1$, but λ_{k+1}/λ_k may still be (much) smaller than λ_2/λ_1, indicating that the iterates may preserve directions of the first k eigenvectors over some iterations, before ultimately converging to x_1. We deliberately have to avoid the term "convergence" for these k directions in this case, though.

Should multiple vectors be iterated simultaneously, the i-th iterate may be written as

$$A^i V_0 = A^i X\Gamma = X\Lambda^i\Gamma,$$

where V_0 is expressed as linear combinations of eigenvectors, each column with different coefficients (contained in Γ), and the above considerations apply to every vector separately. Let $|\lambda_{k+1}|/|\lambda_k|$, $k < n$, be the smallest ratio of—in terms of their absolute magnitude—adjacent eigenvalues. If we also assume that it differs to great enough extent from any other pairing, we expect $V_i = A^i V_0$ to be an approximation to a basis of the space, spanned by the first k eigenvectors at some point during iteration.

There are some implications for a possible block power method based on this observation, though. Let V_i be comprised of m columns. We then want to derive a method that converges to the first m eigenvectors in order to compute the corresponding eigenvalues; to that end, we need the rank of V_i to not decrease. However, we merely can expect a spanning set for the associated vector space. The V_i will become increasingly rank deficient as directions get dampened in each iteration. Orthogonalization is a measure to counteract this effect, but then it is unclear, to what subspace the method will ultimately converge. If we can allow ourselves an early guess, based on the above observations, we may suspect the first m directions to be preserved with convergence rate corresponding to the separation of their eigenvalues. We may also expect, for $k < m$, to observe convergence to the space spanned by the first k directions earlier than to the remaining directions. Analogously for $m < k$, convergence may be slow based on the separation of the first m directions to the rest and also based on the separation among the first m directions. This is, however, pure conjecture at this point.

In the end, even if we can assume span(V_i) to contain the space, spanned by those eigenvectors we want to compute, how this information can be extracted and converted to approximations of eigenpairs is still to be seen, the Rayleigh quotient from the power method being a first hint. This section will introduce subspace iteration as the method of choice for many of the commonly used algorithms to compute subsets of the eigenpairs of a matrix pencil.

1.3.1 Vector projection and orthogonalization

We here use the more general formulation with respect to B, which simplifies to the conventional understanding if $B = I$. Intuitively, the scalar product $\langle v, w\rangle_B$ describes the length of the component of v in direction of a vector w. This component, or the *orthogonal projection* of v onto a normalized vector w, $\|w\|_B = 1$, thus is

$$v_{\|w} = \langle v, w\rangle_B \, w.$$

Consequently, removing the component of v in direction w results in

$$v_{\perp w} = v - v_{\parallel w}$$

such that $\langle v_{\perp w}, w\rangle_B = 0$. Projecting v onto a subspace spanned by multiple normalized and orthogonal vectors $Q = [q_1, \dots, q_k]$,

$$v_{\parallel Q} = \sum_{j=1}^{k} \langle v, q_j\rangle_B q_j = QQ^H Bv$$

removes components in all but the directions of the q_j. This also allows writing a vector in terms of a linear combination of basis vectors, as used in Section 1.1.3.

Removing the directional components of a set of vectors Q from a second set of vectors V such that $\langle v_i, q_j\rangle_B = 0 \ \forall i, j$ is called *(B-)orthogonalization*. Orthogonalizing a set of vectors V in itself via orthogonalizing pairs (v_i, v_j) with $i \neq j$ achieves $\langle v_i, v_j\rangle_B = 0 \ \forall i \neq j$. If furthermore the vectors are normalized such that $\langle v_i, v_i\rangle_B = 1 \ \forall i$, the process is called *orthonormalization*. Again, the terminology used here omits the explicit relation to B.

1.3.2 Gram-Schmidt orthonormalization

```
Input: Vectors $[v_1, \dots, v_k]$
Output: Orthonormal vectors $[q_1, \dots, q_k]$
1: for $i = 1, \dots, k$ do
2:     $\hat{v}_i \leftarrow v_i$
3:     for $j = 1, \dots, i-1$ do
4:         $\hat{v}_i \leftarrow \hat{v}_i - \langle v_i, q_j\rangle\, q_j$
5:     $q_i \leftarrow \frac{\hat{v}_i}{\|\hat{v}_i\|}$
```

```
Input: Vectors $[q_1, \dots, q_k]$
Output: Orthonormal vectors $[q_1, \dots, q_k]$
1: for $i = 1, \dots, k$ do
2:     $\hat{v}_i \leftarrow v_i$
3:     for $j = 1, \dots, i-1$ do
4:         $\hat{v}_i \leftarrow \hat{v}_i - \langle \hat{v}_i, q_j\rangle\, q_j$
5:     $q_i \leftarrow \frac{\hat{v}_i}{\|\hat{v}_i\|}$
```

Algorithm 1.2: *Gram-Schmidt orthonormalization. Left: Classical formulation. Right: Modified formulation.*

Based on the considerations from the previous section, a method to orthonormalize a set of vectors $Q = [q_1, \dots, q_k]$ can be derived where vector i is orthogonalized against vectors 1 to $i-1$ and then normalized, resulting in orthonormal vectors $Q = [q_1, \dots, q_k]$. This process is called *Gram-Schmidt orthonormalization* and is outlined in Algorithm 1.2, on the left. Since every vector has to be orthogonalized against vectors that already are in the orthogonal subset, the process is inherently sequential. Assuming a distribution by vectors such that a process holds at least one vector, all computations for one pass of the i-loop occur on the same process. On the other hand, the i-loop cannot be parallelized since vector i has to be finished, before vector $i+1$ can be processed.

If the algorithm is performed in non-exact, e.g., floating-point arithmetic, the resulting vectors are merely close to being orthogonal. These inaccuracies, caused by accumulated rounding errors of the inexact computations, are not considered inside the j-loop, where v_i is repeatedly projected onto the q_j. A simple modification instead projects the intermediate vector $\hat{v}_i$ onto the q_j. The resulting modified algorithm is referred to as *modified Gram-Schmidt orthonormalization*, outlined in Algorithm 1.2 on the right.

This modified version of the algorithm also allows for a rearrangement of operations to improve parallelization potential. Instead of removing the directions of the q_j from $\hat{v}_i$, it is also possible to remove the directions of q_i from all remaining vectors v_j. This modification also allows to perform the algorithm in situ, without the need for additional memory to store a separate set of result vectors. The *rearranged modified Gram-Schmidt orthonormalization* is outlined in Algorithm 1.3.

```
Input: Vectors $[v_1, \dots, v_k]$
Output: Orthonormal vectors $[q_1, \dots, q_k]$
1: for $i = 1, \dots, k$ do
2:     $q_i \quad \frac{v_i}{\|v_i\|}$
3:     for $j = i + 1, \dots, k$ do
4:         $v_j \quad v_j - \langle v_j, q_i \rangle \, q_i$
```

Algorithm 1.3: *Modified Gram-Schmidt orthonormalization, reordered.*

Technically it is not necessary to produce normalized vectors, merely the orthogonal projection requires them. Since the length of an orthogonalized vector can become very small if the vectors were almost parallel at the beginning, and in order to not being forced to perform the normalization in every pass of the j-loop, normalization of the result vectors is conventionally an integral part of the algorithm. In a numerical context, almost singular sets of input vectors and the resulting small vector norms can lead to major inaccuracies, such that in extreme cases, where the vector size is too small to properly represent its mathematically correct direction in the presence of inexact arithmetic and number representation, the normalized output vector will not be orthogonal to the other vectors at all.

When the process completes, the resulting vectors Q span the same space as the input vectors V. We verify this for the classical Gram-Schmidt method, the modified and rearranged versions are mathematically equivalent. Let R be the triangular matrix

$$R = \begin{pmatrix} \langle v_1, q_1 \rangle & \cdots & \langle v_i, q_1 \rangle & \cdots & \langle v_k, q_1 \rangle \\ & \ddots & \vdots & & \vdots \\ & & \langle v_i, q_i \rangle & \cdots & \langle v_k, q_i \rangle \\ & & & \ddots & \vdots \\ & & & & \langle v_k, q_k \rangle \end{pmatrix}$$

and note that due to trigonometric relations of the orthogonal projection of v_i onto

$$\hat{v}_i = v_i - \sum_{j=1}^{i-1} \langle v_i, q_j \rangle \, q_j,$$

for the angle θ between v_i and $\hat{v}_i$ as well as between v_i and q_i,

$$\cos(\theta) = \frac{\|\hat{v}_i\|}{\|v_i\|}$$

holds. Thus, with the geometric interpretation of the scalar product and $\|q_i\| = 1$,

$$\langle v_i, q_i \rangle = \|v_i\| \|q_i\| \cos(\theta) = \|v_i\| \frac{\|\hat{v}_i\|}{\|v_i\|} = \|\hat{v}_i\|.$$

Then $V = QR$ and V is a linear combination of Q, meaning $\operatorname{span}(V) \subseteq \operatorname{span}(Q)$. V and Q span the same space, if V has full rank. If V does not have full rank, the process will break due to a division by zero, should a vector be reduced to zero length. It is still possible to generate a full set of orthogonal vectors with the same size as V by introducing new random vectors, whenever a zero vector (or, in a numerical process, a vector with sufficiently small norm) occurs.

1.3.2.1 Definite pairs

In case the B-scalar product and the B-norm exist, the Gram-Schmidt methods can be used for B-orthonormalization by letting $\|\cdot\| = \|\cdot\|_B$ and $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_B$.

Gram-Schmidt orthonormalization is not the only way to obtain orthogonal vector sets. Further methods for orthogonalization are discussed in Section 5.2.

1.3.3 Invariant subspaces

A subspace $\mathcal{V}$, containing only vectors whose images under the application of $B^{-1}A$ are again contained in $\mathcal{V}$, $A\mathcal{V} \subseteq B\mathcal{V}$, is called *invariant subspace* of (A, B). In particular any linearly independent subset of k eigenvectors $X \in \mathbb{C}^{n \times k}$ of (A, B) spans an invariant subspace $\mathcal{X}$ with $\dim(\mathcal{X}) = k$. Analogous to [Ste01] and [Krä14], let $U \in \mathbb{C}^{n \times k}$ with $\operatorname{span}(U) = \mathcal{X}$ span such an invariant subspace. Note that this implies full rank of U. Then U has a left inverse $U^{-\ell}$ such that $U^{-\ell}U = I$ (see Section 1.1.2) and there exists a matrix $H \in \mathbb{C}^{k \times k}$ such that

$$AU = BUH \text{ with } H = U^{-\ell}B^{-1}AU, \tag{1.4}$$

meaning that the j-th column of AU is a linear combination of the columns Bu_i with factors h_{ij}. The matrix H is uniquely determined for the given choice of U and $U^{-\ell}$, since for every other matrix G

$$BUH = BUG \implies U^{-\ell}B^{-1}BUH = U^{-\ell}B^{-1}BUG \implies H = G.$$

⊠

Let H have eigenpairs (W, Λ) such that $HW = W\Lambda$. Then (UW, Λ) are eigenpairs of (A, B), since

$$AUW = BUHW = BUW\Lambda.$$

Consequentially, those eigenvalues and eigenvectors of (A, B) associated with $\mathcal{X}$ can be found by means of some basis U spanning $\mathcal{X}$.

1.3.3.1 Power iteration

For a naively blocked power method, all iterated vectors converge to the vector associated with the largest magnitude eigenvalue. The observations regarding invariant subspaces above have shown that, instead, an approximation of an alternative basis U for the associated invariant subspace $\mathcal{X}$ may be iterated. Certainly, any orthogonal basis for $\mathcal{X}$ meets all criteria. As a consequence, orthonormalization with respect to any scalar product will suffice and also prevent the iterated vectors from converging to the same eigenvector. With X being a set of linearly independent vectors, it is possible to find an orthonormal basis Q with $\text{span}(Q) = \mathcal{X}$. Then $Q^{-\ell} = Q^H$ and $H = Q^H B^{-1} AQ$.

For proper convergence speed, we now will have to assume that for a subspace size of k, also k eigenvalues should be well separated from the rest, i.e., $|\lambda_k| \gg |\lambda_{k+1}|$.

1.3.3.2 Definite pairs

Analogously to above, if B is hpd, it is possible to find a B-orthonormal basis Q such that $Q^{-\ell} = Q^H B$ and $H = Q^H AQ$.

When dealing with generalized eigenproblems, the basis for the similarity transform of the application of the operator is B-orthogonal. Should the iterated vectors be B-orthogonal as well, they themselves will be approximates of the eigenvectors.

1.3.4 Rayleigh-Ritz

The choice of the subspace size is related to the number of eigenvalues we identify as "well separated". It is easy to assume that the associated eigenvectors dominate the constructed approximate subspace, while the influence of other directions remains small due to difference in magnitude of the associated eigenvalues; we understand a subspace like that as an approximation to the invariant subspace, spanned by the dominant eigenvectors.

If we assume $\mathcal{X} \supset \text{span}(U)$, e.g., if $k < \dim(\mathcal{X})$, U again spans an invariant subspace, but with the magnitude of the associated eigenvalues not well separated, it will not be a suitable approximation to a basis of $\mathcal{X}$. Should the dominant eigenvalues among themselves be again well separated, with the same train of thought as above we can expect U to approximate a basis for the invariant subspace spanned by the respective eigenvectors.

For $\mathcal{X} \subset \text{span}(U)$ the form in Equation (1.4) does not exist. But also in this case a similarly powerful statement can be made, the Rayleigh-Ritz theorem, based on

[Ray99] and published in a mathematical context in [Rit09]. We will again follow [Krä+13] and [Krä14], based on [Ste01].

Theorem 1.1 (Rayleigh-Ritz) *Let $\mathcal{X} = \operatorname{span}(X)$ be an invariant subspace of the matrix pencil (A, B) spanned by eigenvectors $X = [x_1, \dots, x_m]$. Let $\mathcal{X}$ be contained in a subspace $\mathcal{U} = \operatorname{span}(U)$ spanned by vectors $U = [u_1, \dots, u_k]$, $k \geq m$. Then, with any matrix $V \in \mathbb{C}^{n\times k}$, for the matrix pencil*

$$\left(V^H AU, V^H BU\right) \tag{1.5}$$

there are m eigenpairs (W, Λ) with $W \in \mathbb{C}^{k\times m}$ and $\Lambda \in \mathbb{C}^{m\times m}$ such that (UW, Λ) are eigenpairs of (A, B).

Proof. Let (X, Λ) be m eigenpairs of (A, B). Since $\mathcal{X} \subseteq \mathcal{U}$, every x_j can be expressed as a linear combination of the u_i with factors w_{ij}. With $W = (w_{ij}) \in \mathbb{C}^{k\times m}$,

$$X = UW.$$

Therefore,

$$\begin{aligned} AX &= BX\Lambda \\ AUW &= BUW\Lambda \\ V^H AUW &= V^H BUW\Lambda. \end{aligned}$$

□

The matrices $V^H AU$ and $V^H BU$ are called *Rayleigh quotients* of A and B, respectively. Eigenvalues of $\left(V^H AU, V^H BU\right)$ are called *Ritz values*, Eigenvectors are called *primitive Ritz vectors* and the eigenpairs are called *Ritz pairs*. The eigenvectors of (A, B) constructed via $X = UW$ are called *Ritz vectors*. Obviously, U at least has rank m. Furthermore, since $m \leq k$, not necessarily all Ritz values of Equation (1.5) correspond to eigenvalues of (A, B). These additional Ritz values are called *spurious Ritz values*, *ghost Ritz values* or *Ritz phantoms*. Algorithms that are based on the Rayleigh-Ritz concept, are commonly called *subspace iteration* or *simultaneous iteration* algorithms. If the subspace is acquired by orthogonalization, it also may be called *orthogonal iteration*.

If the reduced-size generalized eigenproblem from Equation (1.5) is to be solved, we again may require $V^H BU$ to be invertible. Should U be of rank $r < k$, then there is a linear combination UH of the columns of U such that

$$UH = \begin{pmatrix} \widetilde{U} & \mathbf{0} \end{pmatrix} \implies BUH = \begin{pmatrix} B\widetilde{U} & \mathbf{0} \end{pmatrix} \implies V^H BUH = \begin{pmatrix} V^H B\widetilde{U} & \mathbf{0} \end{pmatrix},$$

with $V^H B\widetilde{U} \in \mathbb{C}^{k\times r}$. Similarly, should V be of rank $s < k$, there is a linear combination VG of the columns of V such that

$$G^H V^H = \begin{pmatrix} \widetilde{V}^H \\ \mathbf{0} \end{pmatrix} \implies G^H V^H B = \begin{pmatrix} \widetilde{V}^H B \\ \mathbf{0} \end{pmatrix} \implies G^H V^H BU = \begin{pmatrix} \widetilde{V}^H BU \\ \mathbf{0} \end{pmatrix},$$

with $\widetilde{V}^H BU \in \mathbb{C}^{s\times k}$. The combination of both yields a matrix $\widetilde{V}^H B\widetilde{U} \in \mathbb{C}^{s\times r}$,

$$G^H V^H BUH = \begin{pmatrix} \widetilde{V}^H B\widetilde{U} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Therefore the linear combinations H of columns or the linear combination G^H of rows of $V^H BU$ show the rank deficiency and both U and V have to be of full rank for $V^H BU$ to be invertible. As always, B is assumed to be invertible. Should U be chosen to be I-orthonormal and $V^H = U^H B^{-1}$, the reduced-size eigenproblem is of standard form. The Rayleigh quotients then are $U^H B^{-1} AU$ and $U^H B^{-1} BU = I$.

1.3.4.1 Power iteration

If we assume that nothing about the spectrum is known—in particular the number of well separated eigenvalues has to be guessed—it is unrealistic to expect being able to choose the subspace size properly. The Rayleigh-Ritz method demonstrates that it is sufficient to find a basis which is large enough, as it generalizes the considerations made in the context of invariant subspaces to spaces that merely contain the invariant subspaces, and thus is better suited as a general eigensolver.

```
Input: Random vectors $U_0 = V$
Output: Approximate eigenpairs $(\boldsymbol{X}_k, \boldsymbol{\Lambda}_k)$
1: for $i = 1, \dots, k$ do
2:     apply $C$ to $U_{i-1}$ and obtain $U_i$
3:     orthonormalize $U_i$
4: solve $U_k^H B^{-1} AU_k W = W\boldsymbol{\Lambda}_k$ for $(W, \boldsymbol{\Lambda}_k)$
5: $\boldsymbol{X}_k \leftarrow U_k W$
```

Algorithm 1.4: *Power subspace iteration, standard form.*

Before formulating an algorithm based on the Rayleigh-Ritz method, it should be noted that there are now two flavors of the same algorithm, the first of which relies on orthogonalization to solve a reduced-size standard eigenproblem. The Rayleigh-Ritz method furthermore allows to acquire an approximation of the actual eigenvectors as $X = UW$. These vectors might as well serve as the new approximate basis for the invariant subspace and, even if we do not extract spurious Ritz pairs, i.e., $W \in \mathbb{C}^{k\times k}$, still $\mathcal{X} \subseteq \operatorname{span}(UW)$. Since the vectors are approximations of eigenvectors, we no longer have to care about separating the directions explicitly by orthogonalization. The resulting algorithm has the benefit of not requiring an expensive orthogonalization and an additional inversion of B, at the cost of requiring the solution of a reduced-size general eigenproblem in every iteration. The last aspect is not a real disadvantage in comparison to the first algorithm, since a residual-based convergence criterion also requires obtaining approximate eigenvectors in every iteration.

Input: Random vectors $\boldsymbol{X}_0 = V$
Output: Approximate eigenpairs $(\boldsymbol{X}_k, \boldsymbol{\Lambda}_k)$

```
1: for $i = 1, \dots, k$ do
2:     apply $C$ to $\boldsymbol{X}_{i-1}$ and obtain $U_i$
3:     normalize $U_i$
4:     solve $U_i^H A U_i W = U_i^H B U_i W \boldsymbol{\Lambda_i}$ for $(W, \boldsymbol{\Lambda}_i)$
5:     $\boldsymbol{X}_i \quad U_i W$
```

Algorithm 1.5: *Power subspace iteration, generalized form.*

In both cases only certain columns of $\boldsymbol{X}$ span the invariant subspace while others belong to spurious Ritz pairs. If the spurious Ritz values can be identified, it would be possible to reduce the number of columns iterated by removing those directions. For the standard form of the algorithm, this again requires to compute the reduced eigenproblem and $\boldsymbol{X}_i = U_i W$ in every iteration. Steps from Algorithm 1.4 and Algorithm 1.5 can be mixed in general, for example solving a generalized eigenproblem in Algorithm 1.4 to avoid the inversion of B.

1.3.4.2 Definite pairs

If we choose $V^H = U^H$, from A and B being Hermitian follows that $U^H A U$ and $U^H B U$ are Hermitian as well. If furthermore we choose U to be B-orthonormal, $U^H B U = I$ and the reduced-size eigenproblem again is of standard form.

1.3.5 Compatible pencils and harmonic Rayleigh-Ritz

Disregarding how exactly a spanning set U with $\mathcal{X} \subseteq \operatorname{span}(U)$ was obtained, the contained invariant subspace $\mathcal{X}$ of (A, B) can be used to compute any eigenvalues of matrix pencils with identical eigenvectors, e.g., the ones listed in Table 1.2, using the Rayleigh-Ritz method in the same way eigenvalues for (A, B) were computed directly with the shift or shift-invert subspace power iteration method. Or—the other way around—$\mathcal{X}$ can be obtained from these matrix pencils to compute eigenvalues for (A, B).

The particular case, where eigenpairs of the matrix pencil $(A - zB, B)$ are computed, choosing the subspace basis as $(A - zB)U$ such that the Rayleigh quotients become

$$\left(U^H (A - zB)^H (A - zB) U, U^H (A - zB)^H B U\right),$$

results in a subspace iteration method called *harmonic Rayleigh-Ritz*. It is motivated by the observation of eigenpairs with eigenvalues of small magnitude being prone to spawning spurious eigenpairs [Ste01].

To obtain the eigenvalues of (A, B) the shift z has to be reverted manually.

1.4 Spectral filters

At this point the question arises, whether a better approximation of a basis for an invariant subspace of (A, B) can be found, in particular one that does not rely as strongly on the separation of a few extremal eigenvalues of the original eigenproblem.

The behind-the-scenes of the power iteration based methods revealed that everything hinges on differences in magnitude of eigenvalues, i.e., some eigenvectors have to be strongly amplified, while others remain basically untouched or even are dampened when the associated eigenvalue is smaller than one. The amplification made it necessary to regulate the vector length via normalization.

Intuitively, to obtain a basis for a certain subspace, all other directions can be removed from a given set of possibly random vectors v, effectively orthogonalizing it against the unwanted directions.

1.4.1 Projection and projectors

A *projector* is a linear transformation $P \in \mathbb{C}^{n\times n}$ that satisfies $P^2 = P$. Then P is diagonalizable such that

$$P^2 = Q\Delta Q^{-1}Q\Delta Q^{-1} = Q\Delta^2 Q^{-1} = Q\Delta Q^{-1} = P$$

with $\Delta = \operatorname{diag}(\delta_1, \ldots, \delta_n)$, which implies $\delta_i \in \{0, 1\}$ for $i = 1, \ldots, n$. Let again q_i^{-1} denote the i-th column of Q^{-1}. Then

$$P = Q\Delta Q^{-1} = \sum_{i=1}^{n} \delta_i q_i q_i^{-1}.$$

An operator $P = P^2$ is called *idempotent*. Analogously to the interpretation of the eigendecomposition, a projector describes the removal of directions in the eigenbasis Q while other directions are retained without any scaling. A complementary projector is then easily constructed as

$$I - P = Q(I - \Delta)Q^{-1}$$

with eigenvalues $1 - \delta_i$. Let $Q_k = [q_i \,|\, \delta_i = 1] \in \mathbb{C}^{n\times k}$. Then

$$Q\Delta Q^{-1} = \sum_{i=1}^{n} \delta_i q_i q_i^{-1} = \sum_{\{i \,|\, \delta_i = 1\}} \delta_i q_i q_i^{-1} = Q_k Q_k^{-1}$$

may be called the *reduced representation* of a projector. Note that Q_k^{-1} here denotes a matrix comprised of k columns of Q^{-1}.

1.4.1.1 Definite pairs

A projector whose range is orthogonal to its null space is called *orthogonal projector*. This is in particular the case if the eigenvectors of the projector are orthogonal.

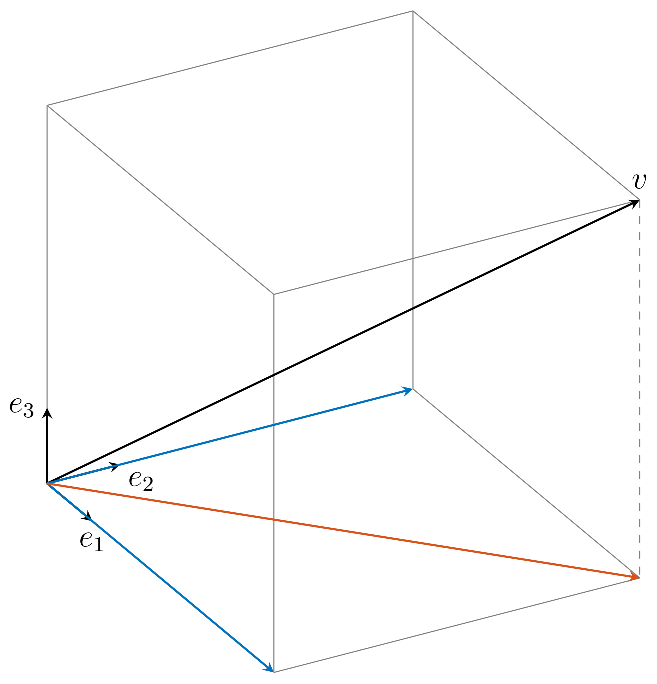


Figure 1.3: *Effect of applying a three-dimensional projector with eigenvalues* $\{1,1,0\}$ *and eigenvectors* e_1*,* e_2*, and* e_3 *to* $v = [5,5,5]^T$.

The operator then has the form $P = Q\Delta Q^H B$ and the application of the reduced representation to a vector v

$$Pv = Q\Delta Q^H Bv = Q_k Q_k^H Bv = \sum_{j=1}^{k} \langle v, q_j \rangle_B q_j$$

matches the form introduced in Section 1.3.1.

1.4.2 Spectral projectors

The approach to dampen certain parts of a spectrum, while retaining the rest, is widely used in electronic filter design (see Section 2.4), e.g., in signal processing.

To obtain a projector onto an invariant subspace $\mathcal{X}$ of (A, B) and effectively remove components in the direction of certain eigenvectors from a set of vectors Y to form a basis of $\mathcal{X}$, a projector of the form

$$P_X = X\Delta X^{-1}$$

acts on the eigenbasis of (A, B) and will be referred to as *spectral projector*. It is idempotent since $X^{-1}X = I$.

Which components are to be removed can be controlled by choosing the diagonal entries of Δ, the eigenvalues of the projector, accordingly. For a set of directions, say $X_\Xi = [x_i \,|\, i \in \Xi]$, where $\Xi \subset \{1, \ldots, n\}$ is the index set of retained directions,

$$\Delta = \operatorname{diag}(\delta_1, \ldots, \delta_n) \quad \text{with } \delta_i = \begin{cases} 1 & \text{if } i \in \Xi \\ 0 & \text{else} \end{cases}$$

constitutes a projector onto $\operatorname{span}(X_\Xi)$. It is identical to the projector $X_\Xi X_\Xi^{-1}$.

If, e.g., eigenpairs associated with eigenvalues λ_i of a matrix pencil (A, B) located inside a closed set $\Omega \subset \mathbb{C}$ with the associated index set $\Xi_\Omega = \{i \,|\, \lambda_i \in \Omega\}$ are to be retained, the corresponding directions are selected by letting $\delta_i = 1 \iff i \in \Xi_\Omega$. With a function

$$\chi_\Omega(\lambda) = \begin{cases} 1 & \text{if } \lambda \in \Omega \\ 0 & \text{else} \end{cases}$$

then $\delta_i = \chi_\Omega(\lambda_i)$.

1.4.2.1 Definite pairs

For eigenproblems with real spectrum, the above reduces to closed intervals on the real axis.

To retain directions associated with eigenvalues λ that are located inside a continuous subsection $\omega = [\lambda_{\min}, \lambda^{\max}]$ of the spectrum of a matrix pencil (A, B) with index set $\Xi_\lambda = \{i \,|\, \lambda_i \in [\lambda_{\min}, \lambda^{\max}]\}$ and $\delta_i = 1 \iff i \in \Xi_\lambda$ analogous to above, the function relating the δ_i to ω is the *window function*

$$\chi_\omega(\lambda) = \begin{cases} 1 & \text{if } \lambda \in [\lambda_{\min}, \lambda^{\max}] \\ 0 & \text{else,} \end{cases}$$

as shown in Figure 1.4, left, and $\delta_i = \chi_\omega(\lambda_i)$.

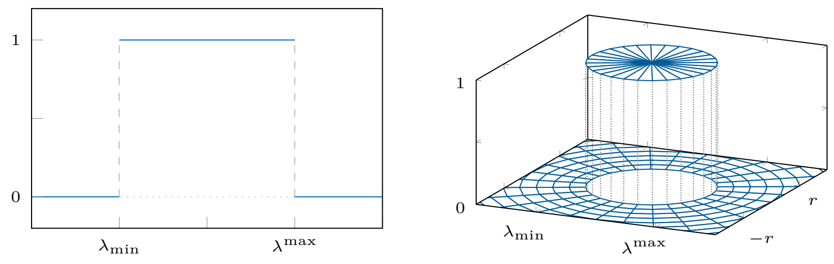


Figure 1.4: *Left: The window function for an interval* $[\lambda_{\min}, \lambda^{\max}]$. *Right: A filter function for a circular region in the complex plane with radius* $r = \frac{1}{2}(\lambda^{\max} - \lambda_{\min})$ *around* $c = \frac{1}{2}(\lambda^{\max} + \lambda_{\min})$.

1.5 Matrix functions

The term *matrix function* denotes mappings $g : \mathbb{C}^{n \times n} \to \mathbb{C}^{n \times n}$ of matrices onto matrices. Given a diagonalizable matrix pencil $B^{-1}A = X \Lambda X^{-1}$, elevating a scalar function $g(x)$ to a matrix function $g(B^{-1}A)$, is generally interpreted as element-wise application to the spectrum of (A, B), retaining the properties of g, see, e.g., [Hig08].

$$g(B^{-1}A) = X^{-1}g(\Lambda)X = X^{-1}\begin{pmatrix} g(\lambda_1) & & \\ & \ddots & \\ & & g(\lambda_n) \end{pmatrix} X$$

The evaluation of a matrix function then boils down to replacing scalar operations in the definition of the function with the corresponding matrix operations. Practically, a spectral projector, as described in the previous section, cannot be constructed without the knowledge of the eigenvectors X_Ξ. If the selection pattern is simple enough to find a scalar function that can be elevated to and evaluated as a matrix function only by the means of operations on A and B, thereby implicitly supplying information of the eigenvectors, spectral parts can be selected without explicit knowledge of the eigenvectors or eigenvalues.

1.5.1 Filtering functions and approximate projectors

Scalar functions $\mathcal{X}_\Omega$ on some closed set Ω with jump discontinuities at the boundary of Ω cannot be represented as matrix functions exclusively operating on A and B since their definition relies on the knowledge of the eigenvalues inside Ω and providing a projector requires the eigenvectors X.

There are, however, approximations to these kinds of functions that fulfill the requirements. In fact, the approximation of a jump discontinuity—or the window function in particular—by a continuous function is a recurring task in other disciplines such as frequency filter design in electronics. The term *spectrum* for the eigenvalues of a matrix is not without reason identical to the term describing the frequency distribution of an oscillation. However, approximations to the window function will generally not be able to remove directions completely, i.e., the function value at the positions of eigenvalues outside of Ω will not be zero, but very small at best, leading to a mere *dampening* of these directions, similar to the effect of very small eigenvalues on the power iteration. We will refer to these functions as *filtering functions*, or f.

The approximate bases for invariant subspaces constructed from these filtering functions are, strictly speaking, not obtained by projection. The matrix representation of the filter,

$$\boldsymbol{P}_f = Xf(\Lambda)X^{-1},$$

at least approximates a projector onto the desired subspace, and we will therefore refer to it as *approximate projector*. Repeated application of the approximate projector will improve the filtering properties since the dampening factors for the eigenvectors x_i, $f(\lambda_i) < 1$, decay exponentially as

$$\boldsymbol{P}_f^k = Xf^k(\Lambda)X^{-1}.$$

Similarly, every factor $f(\lambda_i) > 1$ grows exponentially and will amplify the corresponding directions accordingly, as is the case for the power method. In fact, filtering functions do not necessarily have to obey $f(\lambda_i) \approx 1$ for $\lambda_i \in \Omega$.

Overall, the imperfection of filtering functions makes methods to solve eigenproblems based on them inherently iterative again.

1.6 Iteration and convergence

The family of power iteration methods is inherently iterative. This is because the subspace basis that is constructed can only ever be an approximation to a basis for the actual invariant subspace associated with the dominant eigenvalues. Similarly, for the subspace iteration methods introduced in this chapter, $\mathcal{X}$ is never exactly contained in the space spanned by the set of vectors the method operates on. Should that ever be the case, in exact arithmetic and assuming the reduced eigenproblem can be solved exactly, the Rayleigh-Ritz method would yield the exact result in one step. Rather, we have to expect a set of m linearly independent initial guess vectors to be a linear combination of all eigenvectors and applying an operator will never completely remove directions from them. We therefore have to assume that, due to the behavior of the power iteration, ultimately all vectors will converge to a basis for the m eigenvectors of largest magnitude. The result following below supports this assumption.

Spectral projection methods, in exact arithmetic, would yield exact results, were the filtering function the window function. Since it is not, the filtering properties are successively improved by iteration. Indeed, iteration of the projection process is approximately equivalent to constructing powers $Xf^k(\lambda)X^{-1}$ of the filtering function, improving its filtering properties in every step. Similarly to above, directions are not removed but increasingly dampened such that in this case we interpret the basis to only contain the filtered directions, even though this is only approximately correct. However, continuous iteration likely will reduce the contribution of the dampened directions until they become numerically insignificant and, ultimately, we expect the iterated basis to become rank deficient (also in a numerical sense) because of that.

1.6.1 Convergence

The following result from [Saa11] (with proof) gives a bound for the convergence rate of the subspace iteration method.

Theorem 1.2 (Convergence of subspace iteration) *Let P be the exact projector onto the invariant subspace associated with the eigenpairs $([x_1,\dots,x_m],\operatorname{diag}(\lambda_1,\dots,\lambda_m))$ of (A,B). Let $\boldsymbol{X}_k$ be the approximate eigenvectors in the k-th iteration and P_k the orthogonal projector onto $\operatorname{span}(\boldsymbol{X}_k)$. Let in particular $\boldsymbol{X}_0$ be the initial guess and assume that exact projection $P\boldsymbol{X}_0$ does not reduce the rank, i.e., the columns of $P\boldsymbol{X}_0$ are linearly independent.*

Then, for each eigenvector x_i, $i=1,\dots,m$, of (A,B), there is a uniquely deter-

mined vector $s_i \in \operatorname{span}(\boldsymbol{X}_0)$ such that s_i is projected onto x_i, $Ps_i = x_i$, and

$$\|x_i - P_k x_i\| \leq \|x_i - s_i\| \left(\left| \frac{\lambda_{m+1}}{\lambda_i} \right| + \varepsilon_k \right)^k,$$

where $\varepsilon_k \longrightarrow 0$ for $k \longrightarrow \infty$.

The convergence speed of the subspace iteration method is therefore determined by the distance between an eigenvector x_i and its associated direction in the space of the initial guess, s_i, and, more importantly, by the separation of λ_i from λ_{m+1}.

Of course, spectral projection algorithms should be understood as performing subspace iteration on the (approximate) projector matrix, given again that the eigenvectors of the projector are identical to those of (A, B), and the computed approximate eigenspace can be used to find eigenvalues of (A, B) (or other compatible pencils). As such, the above inequality relates to the separation of eigenvalues, this time of the projector, which are given by the filtering function at the positions of the eigenvalues of (A, B), and thus convergence is now bound to the dampening properties of the filter function or its ability to separate unwanted from wanted eigenvalues. This simple and straight forward adaption can also be found in [Via12] and [Krä14]. The dominance of an eigenvector of (A, B) then also refers to the magnitude of the associated eigenvalue of the approximate projector $\boldsymbol{P}$ and its separation to the other eigenvalues of $\boldsymbol{P}$. Should the filter approximate the window function, the separation of eigenvalues at the boundaries of the region of interest or, more directly, the dampening effect of the filter on those eigenvalues dictates convergence speed and as a direct consequence, dense spectra will require filtering functions with steeper flanks to achieve comparable convergence rates.

In [Krä14] the convergence behavior is analyzed to greater extent, providing error bounds for Ritz vectors, Ritz values and subspaces.

1.6.2 Metrics and residual

Every norm induces a metric, a quantification of the distance of two vectors. A mapping $d(v, w) : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{R}$ is formally called *metric* if it has the properties

- $d(v, w) \geq 0$,
- $d(v, w) = d(w, v)$,
- $d(v, w) = 0 \iff v = w$,
- $d(v, w) \leq d(v, z) + d(z, w)$.

A standard choice is the Euclidean metric

$$d(v, w) = \|v - w\|.$$

Let $(\boldsymbol{X}, \boldsymbol{\Lambda})$ denote a computed approximation of the exact solution (X, Λ). Without knowing the exact solution (X, Λ) of the eigenvalue problem, the distance between two supposedly equal quantities might serve as a measure for the accuracy of $(\boldsymbol{X}, \boldsymbol{\Lambda})$. The *residual* of an approximate solution to the eigenvalue problem of (A, B) is

$$\|A\boldsymbol{X} - B\boldsymbol{X}\boldsymbol{\Lambda}\|.$$

Oftentimes, it makes sense to track the quality of separate approximate eigenpairs $(\boldsymbol{x}, \boldsymbol{\lambda})$. Let the residual matrix be

$$R = A\boldsymbol{X} - B\boldsymbol{X}\boldsymbol{\Lambda}.$$

The column-wise norms of R then give an eigenpair-wise impression of the quality of the solution.

Since $r_i = \|A\boldsymbol{x}_i - \boldsymbol{\lambda}_i B\boldsymbol{x}_i\|$, the norm of $\boldsymbol{x}_i$ plays an important role for the residual as it directly influences its magnitude. To obtain a thoroughly consistent image of the quality of a solution, it is advisable to require the observed eigenvectors to remain at unit length. Similarly, smaller eigenvalues $\boldsymbol{\lambda}_i$ will show smaller residuals.

1.6.3 Residual bounds

If (A, B) is a definite pair and using the concept of a matrix square root (cf. Section 1.2.2, definite pairs), the following result for bounding the location of an Eigenvalue in relation to a Ritz value can be found [Krä14] (standard version originally from [Par80]).

Theorem 1.3 (Ritz value pairing) *Let $R = A\boldsymbol{X} - B\boldsymbol{X}\boldsymbol{\Lambda}$ with $\boldsymbol{X} \in \mathbb{C}^{n\times m}$ B-orthonormal. Then there are m pairs $(\boldsymbol{\lambda}_i, \lambda_j)$, $1 \le i \le m$ and $1 \le j \le n$ such that*

$$|\boldsymbol{\lambda}_i - \lambda_j| \le \left\|B^{-\frac{1}{2}}R\right\| \le \frac{1}{\sqrt{\lambda_{\min}(B)}}\|R\|.$$

Here, $\lambda_{\min}(B)$ refers to the smallest magnitude eigenvalue of B. This result states that for a computed Ritz value there is at least one eigenvalue in the surrounding disk defined by the residual of the approximate solution. The application of this result can be difficult in practice if the presence of spurious Ritz values with large residual r_i has to be assumed. In this case, the residual R is large as well. However, a similar result (also [Krä14], with proof, standard version originally from [Par80]) can be obtained per Ritz pair residual.

From [Par80; Krä14] we know that, for Hermitian matrices M, unit vectors y, and scalars $\alpha \in \mathbb{C}$, there is an eigenvalue λ of M such that

$$|\lambda - \alpha| \le \|My - y\alpha\|.$$

Analogously to [Krä14], we may reformulate this statement for definite pairs by letting

$$M = B^{-\frac{1}{2}}AB^{-\frac{1}{2}} \quad \text{and} \quad y = B^{\frac{1}{2}}v,$$

where v is chosen B-normal such that $B^{\frac{1}{2}}v$ is a unit vector. Then

$$|\lambda - \alpha| \leq \left\| B^{-\frac{1}{2}}(Av - Bv\alpha) \right\|.$$

This is true in particular for approximate Ritz pairs $(\boldsymbol{x}, \boldsymbol{\lambda})$.

Theorem 1.4 (Ritz value distance) *For any computed Ritz pair* $(\boldsymbol{x}, \boldsymbol{\lambda})$ *with residual* r *there is an eigenvalue* λ *of* (A, B) *such that*

$$|\lambda - \boldsymbol{\lambda}| \leq \left\| B^{-\frac{1}{2}}(A\boldsymbol{x} - B\boldsymbol{x}\boldsymbol{\lambda}) \right\| = \left\| B^{-\frac{1}{2}} r \right\|.$$

The statement is roughly the same as Theorem 1.3 but without any means of pairing Ritz values and eigenvalues. For multiple computed pairs, this means that correct association is impossible should any of the disks defined by the above relation overlap. Explicit pairing is often not necessary in practice.

1.7 Digression: Related algorithms

Several other methods for solving eigenproblems exist that do not precisely fit the frame of, but rely on, concepts closely related to subspace iteration. This section will give an overview without delving too deeply into the workings of those methods.

1.7.1 Orthogonal iteration

For any matrix $Z \in \mathbb{C}^{n\times k}$ there exists a decomposition $Z = QR$ into a unitary matrix $Q \in \mathbb{C}^{n\times k}$ and a triangular matrix $R \in \mathbb{C}^{k\times k}$. Such a decomposition is called *thin* [GV13] or *reduced* [TB97] *QR decomposition*. Then, if Z has full rank, Q is an orthonormal basis for $\operatorname{span}(Z)$ and if $\mathcal{X} \subseteq \operatorname{span}(Z)$ then $\mathcal{X} \subseteq \operatorname{span}(Q)$. One way to obtain such a decomposition is the Gram-Schmidt orthonormalization introduced in Section 1.3.2. Other methods are discussed in Chapter 5.

Not to be confused with the *QR factorization* itself, which is commonly used for orthogonalization of a set of vectors, the *QR algorithm* is predicated on the QR factorization to solve eigenproblems in a similar way the power subspace iteration method does (see, e.g., [Saa11]).

For iterates $Q_kR_k = CQ_{k-1}$ of the power method, the Rayleigh quotients become

$$\left(Q_k^H C Q_k, I\right) = \left(Q_k^H Q_{k+1} R_{k+1}, I\right).$$

According to [Saa11], Q_k converges essentially[2] to some orthogonal matrix Q. We can therefore assume that $Q_{k+1} \approx Q_k S$ as k increases, where $S = Q_k^H Q_{k+1}$ is the (approximately) diagonal sign matrix that holds the (possibly complex) signs on the diagonal, always assuming that Q_k and Q_{k+1} are normalized. The reduced eigenproblem then becomes

$$SR_{k+1}W \approx W\Lambda_z.$$

2) This term describes a convergence in terms of pure direction, ignoring the sign.

Since R is triangular, its eigenvalues appear as the diagonal entries (remember the definition of the characteristic polynomial and the computation of determinants) and the explicit solution of the reduced eigenproblem is unnecessary if eigenvectors can be disregarded and if the sign matrix S of two consecutive iterations is tracked. To this end, only a pair-wise multiplication of the corresponding vectors in Q_k and Q_{k+1} is required and no full inner product has to be computed. Then the eigenvalues of SR_{k+1} are obtained by element-wise multiplication of $\operatorname{diag}(S)$ and $\operatorname{diag}(R_{k+1})$, avoiding the solution of a reduced eigenproblem.

To obtain the corresponding eigenvalues of (A, B), as opposed to the more explicitly Rayleigh-Ritz based methods before, an explicit transformation may be necessary, e.g., $\lambda = 1/\lambda_z + z$ for the shift-invert operator. The resulting algorithm is commonly called *orthogonal iteration* Algorithm 1.6. This algorithm is remarkably similar to

```
Input: Random vectors $Q_0$
Output: Approximate eigenpairs $(\boldsymbol{X}_k, \boldsymbol{\Lambda}_k)$
1: for $i = 1, \dots, k$ do
2:     $T_i \leftarrow Q_{i-1}$
3:     apply $C$ to $Q_{i-1}$ and obtain $Y_i$
4:     $Q_i R_i = Y_i$
5: $S = T_k^H Q_k$
6: solve $SR_k W = W\boldsymbol{\Lambda}_k$ for $(W, \boldsymbol{\Lambda}_k)$    ▷ optional
7: $\boldsymbol{X}_k \leftarrow Q_k W$                                     ▷ optional
```

Algorithm 1.6: *Orthogonal iteration, exploiting the factor R as described in this section. The step in line 5 can be simplified since only the diagonal of S is required. The step in line 6 can be significantly simplified by computing the eigenvalues of SR_k directly via* $\operatorname{diag}(S)$ *and* $\operatorname{diag}(R_k)$.

Algorithm 1.4. The only difference is the use of SR and the form typically found in the literature indeed computes the reduced eigenvalue problem for the Rayleigh quotient $Q_k^H C Q_k$. If this is replaced with $Q_k^H B^{-1} A Q_k$, Algorithm 1.6 and Algorithm 1.4 are identical. In [Saa11], the term subspace iteration also refers to orthogonal iteration. The formulation presented here does require additional storage for T_i (only one vector block needs to be stored, of course) but omits the additional multiplication with C and the inner product $Q_k^H C Q_k$. To reduce the additionally required memory, the signs can be updated in each iteration such that only m values have to be stored, if m is the number of columns of the Q_i. This comes at the price of having to compute the scalar products of the m column pairs of T_i and Q_i in every iteration.

Algorithm 1.6 is not to be confused with the QR algorithm (or QR iteration), which is based on orthogonal iteration but computes a complete Schur decomposition of a matrix.

As before, a convergence criterion based on the residual of the approximate eigenpairs requires the solution of the reduced eigenproblem and the computation of the

Ritz vectors in every iteration. The QR decomposition and its use for orthogonalization will be discussed in larger detail in Section 5.2.1.

1.7.2 The Arnoldi and Lanczos methods

A method to compute eigenpairs of a matrix A that shares many of the theoretical aspects introduced in this chapter is the *Arnoldi method* [Arn51]. See also, e.g., [Saa11] or [GV13]. Consider a sequence of iterates v_i produced by the power method applied to a single vector v_0 as application of powers of the matrix operator C, $K = \left[v_0, Cv_0, \dots, C^k v_0\right]$. The space

$$\mathcal{K}_k = \operatorname{span}\left(v_0, Cv_0, \dots, C^{k-1} v_0\right)$$

is called the *Krylov subspace* [Kry31] of order k generated by C and v_0. With considerations similar to the approximation of an invariant subspace $\mathcal{X}$ for the k dominant eigenpairs of (A, B), it is reasonable to assume that $\mathcal{X} \subseteq \mathcal{K}$ and the vectors K can be used in a Rayleigh-Ritz process. In contrast to subspace iteration, the subspace basis K is built successively by application of the operator to only one vector in each step. Consequentially, since orthogonalization is required for the same reasons it was required in the power subspace methods, the process is intertwined with a vector-wise orthonormalization algorithm, e.g., the Gram-Schmidt orthonormalization introduced in Section 1.3.2. Analogous to the Gram-Schmidt

```
Input: Random vector $v_0$, size of Krylov subspace $k$
Output: $(Q, H)$ such that $CQ \approx QH$, approximate
        eigenpairs $(\boldsymbol{X}, \boldsymbol{\Lambda})$
1: $q_1 = \frac{v_0}{\|v_0\|}$
2: for $i = 2, \dots, k$ do
3:     $v_i \leftarrow$ apply $C$ to $q_{i-1}$
4:     $q_i \leftarrow$ orthonormalize $v_i$ against $[q_1, \dots, q_{i-1}]$
5:         with $h_{j,i-1} \leftarrow \langle v_i, q_j \rangle$ for $j = 1, \dots, i-1$
6:     $h_{i,i-1} \leftarrow \|q_i\|$
7: solve $HW = W\boldsymbol{\Lambda}$ for $(W, \boldsymbol{\Lambda})$ with $H = (h_{i,j}) \in \mathbb{C}^{k-1,k-1}$
8: $\boldsymbol{X} \leftarrow QW$ with $Q = [q_1, \dots, q_{k-1}]$
```

Algorithm 1.7: *The Arnoldi algorithm.*

procedure $v_i = Cq_{i-1}$ is the i-th vector to be orthonormalized against $q_1, \dots, q_{i-1}$,

$$\hat{v}_i = Cq_{i-1} - \sum_{j=1}^{i-1} \langle Cq_{i-1}, q_j \rangle \, q_j$$

is the i-th non-normalized orthogonal result vector, and $q_i = \hat{v}_i/\|\hat{v}_i\|$ the final i-th orthonormal result vector. In particular

$$Cq_{i-1} = \|\hat{v}_i\| q_i + \sum_{j=1}^{i-1} \langle Cq_{i-1}, q_j\rangle\, q_j$$

is a linear combination of $[q_1, \dots, q_i]$. Then, with $q_1 = v_0/\|v_0\|$, [Saa11]

$$\operatorname{span}\Big(v_0, Cv_0, \dots, C^{k-1}v_0\Big) = \operatorname{span}\Big(q_1, Cq_1, \dots, C^{k-1}q_1\Big) = \operatorname{span}(q_1, q_2, \dots, q_k)$$

and the upper Hessenberg matrix H constructed by Algorithm 1.7 satisfies $Q^H C Q = H$ since

$$h_{\ell,i} := q_\ell^H C q_i = \|\hat{v}_{i+1}\| q_\ell^H q_{i+1} + \sum_{j=1}^{i} \langle Cq_i, q_j\rangle\, q_\ell^H q_j = \begin{cases} \|\hat{v}_{i+1}\| & \text{if } \ell = i+1 \\ \langle Cq_i, q_\ell\rangle & \text{if } \ell < i+1 \\ 0 & \text{otherwise} \end{cases}$$

for $\ell, i = 1, \dots, k-1$. If the stabilized version of Gram-Schmidt is used, the factors $\langle v_i, q_j\rangle$ change to $\langle \hat{v}_i, q_j\rangle$, where the $\hat{v}_i$ denote the intermediate vectors generated during orthogonalization.

The matrix H then gives the Rayleigh quotient for the Rayleigh-Ritz procedure of standard form, see Section 1.3.4.

1.7.2.1 Definite pairs

In the case of a Hermitian operator, the Arnoldi method simplifies to the *Lanczos method* [Lan50]. The matrix H is then tridiagonal due to $q_\ell^H C q_i = q_\ell^H C^H q_i = \overline{q_i^H C q_\ell}$, i.e., H is Hermitian if the operator C is Hermitian. This is the case for operators of the form $B^{-1}A$, $B^{-1}(A - zB)$ and $(A - zB)^{-1}B$ only if z is real and $B = I$.

1.7.2.2 Restarts

As an intuitive extension to the above algorithm, if the Krylov subspace can be assumed to be large enough to contain the invariant subspace $\mathcal{X}$ of size $m \le k$, instead of increasing the subspace size further to generate larger powers of C, restarting the whole process with q_k as starting vector to generate powers up to C^{2k-2}, with the j-th restart generating powers up to C^{jk-j}, improves the approximation of the subspace.

1.7.3 Other methods

What follows is a short overview over other important methods and their relation to subspace iteration and spectral projection, as illustrated in Figure 1.5.

Acceleration by modification of the spectrum using a suitable function that can be applied as matrix function to emphasize certain (often extremal) eigenvalues

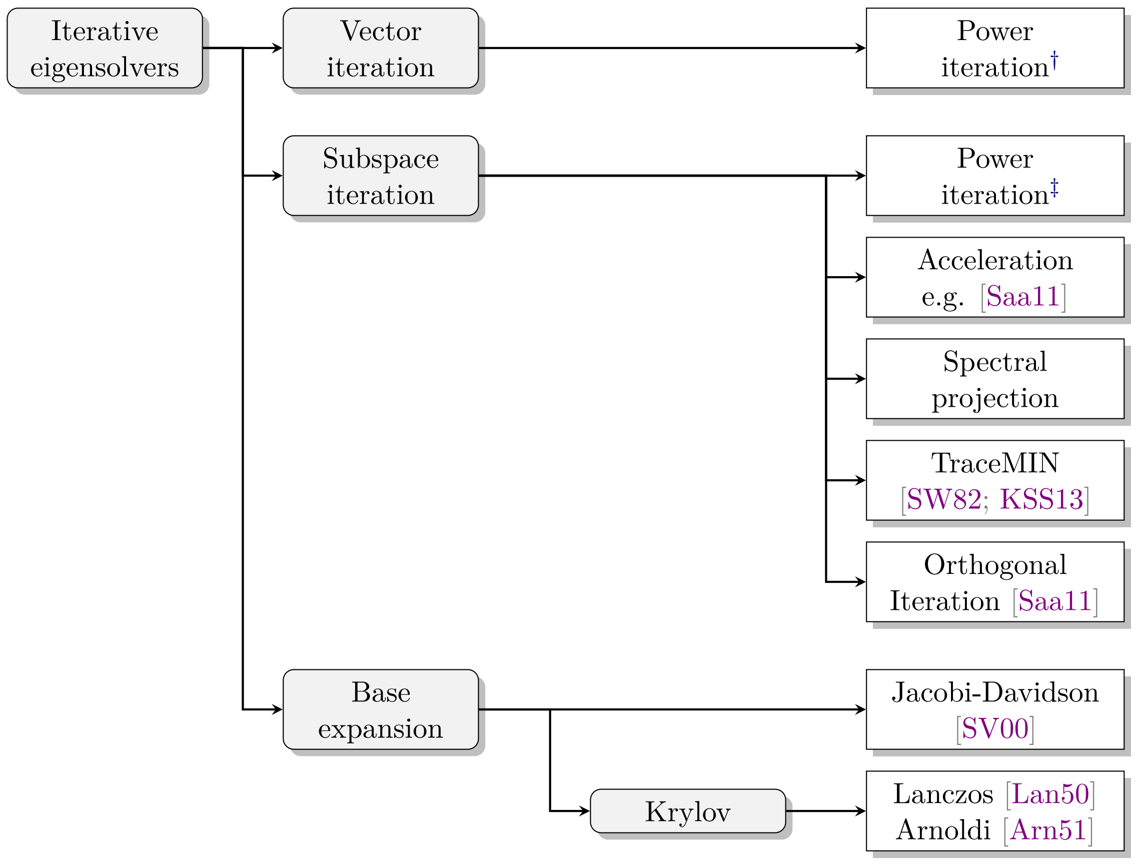


Figure 1.5: *Non-exhaustive classification hierarchy of iterative eigenvalue algorithms. The power iteration algorithms include all the variations listed in Table 1.1 and Table 1.2.*

follows, in principle, the same idea as spectral projection. Instead of dampening of certain directions and approximation of the window function, the focus is the amplification of directions to increase separation and improve convergence speed, as seen in Equation (1.2), Equation (1.3), and Theorem 1.2. It can be used to accelerate the single vector power iteration or multi vector Rayleigh-Ritz subspace iteration. An example is the acceleration by Chebyshev polynomials [Wil65; Saa11], using their property to grow large fast outside of $[-1, 1]$. See also Section 2.1.5 and Section 2.6. The elliptic rational function from Section 2.4.4 could be used in the same way.

The TraceMIN algorithm [SW82; KSS13] finds successively improved approximations to the eigenvectors associated with the smallest eigenvalues by solving an optimization problem to minimize the trace of a reduced eigenproblem, followed by the Rayleigh-Ritz procedure to extract eigenpairs. Due to this improvement of basis vectors, we classify TraceMIN as subspace iteration algorithm.

†) This includes all possible spectral modifications in order to accelerate convergence and, with this, all filters used in subspace iteration.
‡) This includes all operators from Table 1.1 and Table 1.2.

The Davidson method [Dav75] improves the approximation of eigenpairs by expanding the subspace used in the Rayleigh-Ritz procedure by a vector being orthogonal to the original basis vectors. If the suggested choice of Davidson for this vector is replaced with Jacobi's orthogonal correction vector (and the occurrence of the eigenvalue is replaced by the associated Ritz value) [Jac46; SV00], the result is the Jacobi-Davidson method [SV00].

The Arnoldi and Lanczos methods rely on the Rayleigh-Ritz procedure for the extraction of eigenpairs and, similarly to the Davidson and Jacobi-Davidson methods, extend the subspace basis vector by vector using orthogonalization. They have a strong relation to the power iteration method since they build a Krylov subspace, which is not the case for the Davidson and related methods. Note that, theoretically, filters from subspace iteration or spectral projection can also be used in the Arnoldi process.

In Figure 1.5, power iteration is used as umbrella term for the complete family of power iteration algorithms, see Table 1.1 and Table 1.2. Of course, every algorithm that manipulates the spectrum of the operator to improve convergence and extracts the eigenpairs of the original matrix pencil via Rayleigh-Ritz, can be executed as single vector version. These are not included in Figure 1.5.

Chapter 2
Spectral projection algorithms

The previous chapter introduced the use of approximate spectral projectors in a Rayleigh-Ritz subspace iteration algorithm as a promising method for computing arbitrary portions of the spectrum of a matrix pencil (A, B). For many applications, it is to be expected that the spectral region of interest will either be a closed, simply connected domain containing the eigenvalues of the least magnitude or two separate domains containing the eigenvalues of the largest magnitude. In the latter case, assuming a filtering function based on the window function is all that is available, the two domains would have to be computed separately. Of course, it would be possible to approximate a filtering function that filters exactly the regions of interest. Though, the more complex these functions become, the harder it typically is to acquire good approximations that can be applied without too much computational effort. A frequently employed approach is to find approximations of a target function as superposition of members of a simpler family of functions.

A prominent incarnation is the Fourier series which approximates a periodic function by a summation of weighted sine functions of different frequencies, which ultimately leads to the Fourier transform. Since the computation of sine matrix functions via their Taylor series to sufficient accuracy involves high orders of matrix powers and the computation of the approximate filter again requires the evaluation of multiple matrix sine functions, the required amount of matrix multiplications seems to make this method infeasible for spectral filtering.

Other well-known methods rely on polynomial or rational approximation. One example is the approximation by Chebyshev polynomials. Due to the possibility of recursive definition, evaluating these polynomials can be achieved with three-term recurrence. This enables its application to a set of vectors with as many matrix multiplications as the degree of the polynomial, in conjunction with computing the required Chebyshev polynomials of increasing degree, provided two vector sets holding previous values can be stored alongside the right-hand sides.

In 2009, a method based on the Cauchy integral theorem for finding a matrix representation of the window function as projector to be used in a Rayleigh-Ritz process was introduced in [Pol09]. The method becomes a rational approximation by use of a numerical approximate integration scheme such as Gauss-Legendre quadrature. While it is difficult to employ polynomial approximations for the computation of generalized eigenproblems due to many linear system solves involving B, rational approximations that have a partial fraction form without or only with trivial polynomial remainder are well suited for generalized eigenproblems but require a number of shifted linear system solves equal to the number of poles of the rational function. Since good approximations with steep filter flanks require the poles to be located very close to the interval boundary, the linear systems can become arbitrarily ill-conditioned, depending, beyond the spectral distribution of the matrix, also on the spectral density which is often related to matrix size and on the distance of the interval boundaries to eigenvalues of the matrix pencil. While the systems typically will never become singular—the shifts are non-real for almost all cases—they can come very close to being considered singular. Additionally, due to the non-real shift, the systems are never Hermitian, even if A and B both are. These conditions make the linear systems challenging for any iterative linear system solver.

Another rational approximation of the window function can be derived from Zolotarev's approximation of the sign function [Zol77] using appropriate Möbius transformations. Filters of this type have been used for spectral projection in [Güt+15] and [LY17].

In electronic filter design, filters that go by the name of Chebyshev and Zolotarev are common tools of the trade. Theory and design processes are versatile and well explored. By specifying certain constraints on the desired properties of a filter, the frequency response of the filter is shaped and the minimum required degree necessary to satisfy the initially given requirements can be inferred. It seems natural to employ an existing extensive framework for designing filters for our spectral filtering approaches, given that the described filters again are rational functions, can be expressed in partial fraction form, and eventually are evaluated in the same way as contour integration and Zolotarev approximation. We will therefore dedicate a section to these filters, noting that they relate to the filters described earlier almost exclusively in name.

We will also adopt some jargon used in the design of electronic filters for use in discussions of the several other filtering methods described in this chapter.

Gain The magnitude of the frequency response of the filter (often in dB). This is what the filtering curve describes and thus directly relates to the dampening properties of the filter as a function of frequency, or, in our case, the real axis.

Passband Spectral intervals of little or no dampening. In electronic filter design, this describes frequencies that pass the filter (almost) without being affected. In the context of spectral filtering, eigendirections related to eigenvalues inside this interval are retained when applying the filter.

Stopband Spectral intervals of high dampening. In electronic filter design, this describes frequencies that are dampened significantly when passing the filter. In the context of spectral filtering, eigendirections related to eigenvalues inside this interval are strongly suppressed when applying the filter. In the case of approximated window functions, the stopband can be considered split in half.

Transition band The region between passband and stopband where the transition from high gain to low gain results in a flank with a certain steepness. The width of this region is an important criterion as it is directly related to the convergence speed of the method (in combination with spectral density inside the transition band) and its ability to separate wanted from unwanted eigenpairs.

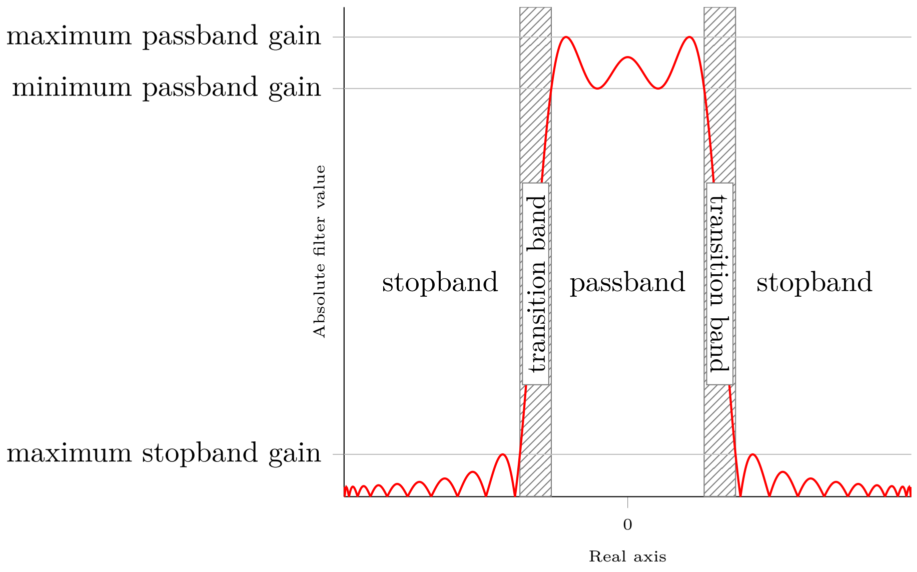


Figure 2.1: *Filter properties: passband, stopbands, transition bands, and important gain characteristics. Absolute filter values are shown.*

Figure 2.1 illustrates these terms. While the term *band* is used for ranges of frequencies, it is worth to remind ourselves that spectral filtering of matrix pencils only applies to a finite set of discrete "frequencies", the eigenvalues of the matrix pencil.

For our purpose it will be sufficient to consider low-pass filters, since the transfer function is reflected for negative frequencies and the spectrum can be transformed such that the target interval fits the filtered region. Linear transformation in terms of translating and scaling a filtering function can always easily be incorporated in terms of matrix operations since they correspond to matrix shifts and scalar scaling, respectively. It is therefore easy to transform the spectrum of a matrix pencil to the desired interval, whenever it is required, possibly at the cost of additional but cheap operations when applying the filter.

This chapter will treat polynomial approximations using Chebyshev polynomials first, given that this is the only filter on the list that is a matrix polynomial and thus can be applied by matrix multiplication alone. All other filters presented here are rational filters, requiring the solution of linear systems. The first two rational filters that will be covered are the Cauchy filter and its variations as well as the Zolotarev filter. Both have been used in spectral filtering for eigenpair computation before, the Cauchy filter preceding the Zolotarev filter. They are introduced in this order as well. The last four conventional filters introduced here will be the filter types from electronic filter design, roughly in the order of their complexity. The Butterworth filter, for example, is perfectly smooth; type I and II Chebyshev filters introduce oscillations in the pass- or stopband, respectively; elliptic filters combine oscillations in both pass- and stopband. Further, the Butterworth filter is equivalent to a Cauchy filter using the midpoint integration rule and elliptic filters can be understood as a generalization of the Zolotarev filter, allowing more parameters to be tweaked. To the best of our knowledge, these filters have, despite their flexibility, not been used in the spectral filtering context, or at least not prominently, which is why they are covered last here.

Finally, an extension to the concept of Cauchy integral based filtering originating from a root finding method for analytic functions, the family of Sakurai-Sugiura methods (SSM), is introduced to highlight differences and similarities compared to spectral filtering.

Prerequisite for a good and correct implementation often is at least a basic understanding of the mathematical concepts and foundations a numerical method is built upon. Beyond providing all information necessary for an implementation we therefore also include basic derivations for most of the aspects of the methods introduced here. Also, while the next chapters will cover details of implementation and effects of algorithmic choices, some consequences for a possible implementation of certain approaches will be mentioned in this chapter already.

2.1 Polynomial filters

Polynomials are among the simplest classes of functions as they only require multiplication and addition for evaluation. This is particularly important if a matrix function is to be evaluated.

Often, the selected basis polynomials of ascending degree form orthogonal bases of the inner product space of polynomials. The approximated function is then evaluated as weighted sum of these basis polynomials. The recursive definition of the Chebyshev polynomials (see, e.g., [Saa11]) makes it possible to intertwine computation of the next required Chebyshev polynomial and the computation of the approximated function with little additional effort in terms of memory consumption. This approach, however, is limited to real functions.

Depending on the function to be approximated, the computation of the weights can be difficult and may only be possible approximately. For simple functions like the window function, it is possible to compute the coefficients analytically.

2.1.1 Polynomial approximation

Analogous to the vector scalar product from Section 1.1.1, the concept of an inner product can be extended to function spaces (see, e.g., [Akh56]),

$$\langle g, h \rangle = \int g(z)\overline{h(z)}\, dz.$$

A more general formulation includes a weight function $w(z)$ that does not influence its properties as inner product, but careful choice of w will simplify the computations of these integrals later on (see, e.g., [AS74]),

$$\langle g, h \rangle_w = \int w(z) g(z) \overline{h(z)} \, dz.$$

From there, similarly to expressing a vector as linear combination of a set of basis vectors (Section 1.3.1), it may be possible to represent functions as linear combinations of different functions. While we casually use the term representation, it is not yet clear whether such a representation can exist, given arbitrary basis functions p_n—in order to find such a representation of a function $g(z)$, the basis functions $p_n(z)$ are required to span the function space $h(z)$ is defined on. Since the respective function space is potentially infinite dimensional, the number of basis functions required to represent the target function may be infinite as well,

$$g(z) = \sum_{i=0}^{\infty} \underbrace{\frac{\langle g, p_i \rangle_w}{\langle p_i, p_i \rangle_w}}_{=: \, c_i} p_i(z), \tag{2.1}$$

where the weight c_i combines the "component" of g in "direction" p_i and the "normalization" factor of p_i, to use common vector nomenclature. A simple example is the space of polynomials. With no fixed maximum degree, it is easy to see that an infinite sequence $(1, z, z^2, \ldots)$ is required to form a basis for all polynomials in z.

Assuming the c_i to be descendingly ordered by magnitude, discarding all summands of Equation (2.1) above a certain index d yields an approximation of g in terms of the p_i. Restricting the interval of approximation simplifies the approximation and possibly reduces the number of summands necessary to find a good approximation. The interval of approximation is dictated by the integration region of $\langle g, p_n \rangle_w$ and $\langle p_n, p_n \rangle_w$.

Computing the coefficients c_i becomes possible by exploiting that the basis functions are orthogonal in the above sense and if the target function is sufficiently simple. Should the function be more complex, approximations of the coefficients can be obtained instead.

The use of trigonometric functions (and an obviously necessary constant term) for example leads to the family of trigonometric series whose most well-known member is the Fourier series. It is used to approximate periodic functions or the periodic continuation of function segments. Their use as matrix functions is very limited by the complexity of the matrix sine and cosine functions.

The family of polynomials on the other hand is computationally simple; they can be evaluated using just the simplest arithmetic operations and thus are more feasible for our inherent goal to identify spectral projectors whose application can be expressed in terms of simple matrix operations.

2.1.2 Chebyshev polynomials

The polynomials defined by

$$T_n(z) = \cos\Big(n\cos^{-1}(z)\Big)$$

on $|z| \leq 1$ are called *Chebyshev polynomials of the first kind* (see, e.g., [Wil65; AS74; Par80; Ste98; Saa11; Wei+06]). The terms $\langle T_i, T_j\rangle_w$ are computed by substitution with $\theta = \cos^{-1}(z)$ as

$$\int_{-1}^{1} w(z)\cos(i\cos^{-1}(z))\cos(j\cos^{-1}(z))\,dz = \int_0^\pi \cos(i\theta)\cos(j\theta)\,d\theta, \qquad (2.2)$$

provided the weight function $w(z)$ is chosen as

$$w(z) = -\frac{d}{dz}\cos^{-1}(z) = \frac{1}{\sqrt{1-z^2}}.$$

Here, the sign resulting from reversing the direction of integration after substituting is included in the weight function. Repeated partial integration or the use of the trigonometrical identity [AS74]

$$\cos(\theta)\cos(\phi) = \tfrac{1}{2}\cos(\theta-\phi) + \tfrac{1}{2}\cos(\theta+\phi),$$

which we will use here, for $i \neq j$ then gives

$$\int_0^\pi \cos(i\theta)\cos(j\theta)\,d\theta = \frac{1}{2}\left[\frac{\sin((i-j)\theta)}{i-j} + \frac{\sin((i+j)\theta)}{i+j}\right]_0^\pi = 0.$$

This demonstrates the orthogonality of the Chebyshev polynomials with respect to the weight function $w(z)$ as above, making them a basis for the space of polynomials of unbounded degree. As such, they can be used to approximate functions by some polynomial, the degree determining the quality of the approximation and also the cost of evaluation. For $i = j \neq 0$, as required for the coefficients c_i,

$$\int_0^\pi \cos^2(i\theta)\,d\theta = \frac{1}{2}\left[\theta + \frac{\sin(2i\theta)}{2i}\right]_0^\pi = \frac{1}{2}\pi$$

and further for $i = j = 0$

$$\int_0^\pi 1\,d\theta = [\theta]_0^\pi = \pi.$$

It should be noted that without reversing the sign, the weight function would violate the requirement of positive definiteness for the inner product $\langle T_i, T_i\rangle_w$, as can be seen from the above. With this, the coefficients c_i can be computed as

$$c_i = \begin{cases} \frac{1}{\pi}\langle T_i, g\rangle_w & \text{if } i = 0 \\ \frac{2}{\pi}\langle T_i, g\rangle_w & \text{else.} \end{cases}$$

2.1.2.1 Window function

With $g(z) = \chi_\omega(z)$, the window function for the interval $\omega = [\lambda_{\min}, \lambda^{\max}] \subset [-1, 1]$ on the real axis, the coefficients c_i can be computed analytically. Using the same substitution as in Equation (2.2),

$$\int_{-1}^{1} w(z)\chi_\omega(z)T_i(z)\,dz = \int_{\lambda_{\min}}^{\lambda^{\max}} w(z)T_i(z)\,dz = \int_{\cos^{-1}(\lambda^{\max})}^{\cos^{-1}(\lambda_{\min})} \cos(i\theta)\,d\theta.$$

Note that, to compensate for the sign from the weight function, the direction of integration is reversed again. For $i > 0$

$$\int_{\cos^{-1}(\lambda^{\max})}^{\cos^{-1}(\lambda_{\min})} \cos(i\theta)\,d\theta = \left[\frac{\sin(i\theta)}{i}\right]_{\cos^{-1}(\lambda^{\max})}^{\cos^{-1}(\lambda_{\min})} = \frac{\sin(i\cos^{-1}(\lambda_{\min})) - \sin(i\cos^{-1}(\lambda^{\max}))}{i}$$

and

$$\int_{\cos^{-1}(\lambda^{\max})}^{\cos^{-1}(\lambda_{\min})} 1\,d\theta = \cos^{-1}(\lambda_{\min}) - \cos^{-1}(\lambda^{\max})$$

if $i = 0$. For the coefficients c_i of the polynomial approximating $\chi_{[\lambda_{\min},\lambda^{\max}]}(z)$ the final explicit form is then

$$c_i = \begin{cases} \frac{1}{\pi}\Big(\cos^{-1}(\lambda_{\min}) - \cos^{-1}(\lambda^{\max})\Big) & \text{if } i = 0 \\ \frac{2}{i\pi}\Big(\sin\Big(i\cos^{-1}(\lambda_{\min})\Big) - \sin\Big(i\cos^{-1}(\lambda^{\max})\Big)\Big) & \text{else.} \end{cases}$$

2.1.2.2 Application

For the Chebyshev polynomials of the first kind there exists a simple three term recurrence relation that allows for the computation without explicit use of trigonometric functions, which is a critical condition if the polynomial is to be elevated to a matrix function.

$$\begin{aligned} T_0(z) &= 1 \\ T_1(z) &= z \\ T_n(z) &= 2zT_{n-1}(z) - T_{n-2}(z) \end{aligned}$$

T_0 and T_1 are trivially confirmed and the validity of the recursive relation is easily confirmed by using simple trigonometric identities. Write the relation in angular form by substituting $z = \cos(\theta)$,

$$T_n(\cos(\theta)) = 2\cos(\theta)\cos(n\theta - \theta) - \cos(n\theta - 2\theta).$$

Since $2\cos(a)\cos(b) = \cos(a - b) + \cos(a + b)$ and $\cos(-a) = \cos(a)$,

$$T_n(\cos(\theta)) = \cos(2\theta - n\theta) + \cos(n\theta) - \cos(n\theta - 2\theta) = \cos(n\theta).$$

An approximation of the window function (or any function, given the proper coefficients c_i) is then found as per Equation (2.1),

$$g(z) \approx f(z) = \sum_{i=0}^{d} c_i T_i(z),$$

where d is the degree of the resulting polynomial.

2.1.3 Matrix polynomials

Since Chebyshev polynomials are defined on the real axis only, we have to restrict ourselves to definite matrix pencils. Let the approximate filtering function χ be the polynomial $f(\lambda) = \sum_{i=0}^{d} c_i T_i(\lambda)$ applied to the spectrum of $B^{-1}A$. The filtering matrix then is

$$\Delta_f = \operatorname{diag}(f(\lambda_1), \ldots, f(\lambda_k))$$

and the approximate projector is

$$\boldsymbol{P}_f = X \Delta_f X^H B. \tag{2.3}$$

The polynomial $f(\lambda)$ is defined on $\operatorname{spec}(A, B)$ and the matrix has the eigendecomposition $B^{-1}A = X \Lambda X^H B$ with $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_k)$. It is thus possible to elevate the projector formulation to matrix form,

$$\begin{aligned}
\boldsymbol{P}_f &= X \operatorname{diag}(f(\lambda_1), \ldots, f(\lambda_k)) X^H B \\
&= X \sum_{i=0}^{d} c_i T_i(\operatorname{diag}(\lambda_1, \ldots, \lambda_k)) X^H B \\
&= \sum_{i=0}^{d} c_i T_i\Big(X \operatorname{diag}(\lambda_1, \ldots, \lambda_k) X^H B\Big) \\
&= \sum_{i=0}^{d} c_i T_i\Big(B^{-1}A\Big) \approx P_\chi,
\end{aligned}$$

where the T_i are again polynomials which may be elevated to matrix polynomials in identical manner. The Chebyshev recurrence relation from above becomes

$$\begin{aligned}
T_0(B^{-1}A) &= I \\
T_1(B^{-1}A) &= B^{-1}A \\
T_k(B^{-1}A) &= 2B^{-1}A T_{k-1}(B^{-1}A) - T_{k-2}(B^{-1}A)
\end{aligned}$$

or, applied to vectors V,

$$\begin{aligned}
T_0 &= V \\
T_1 &= B^{-1}AV \\
T_k &= 2B^{-1}A T_{k-1} - T_{k-2}
\end{aligned}$$

where T_i are intermediate vectors for which two buffers may be used in alternating fashion. However, the d inversions of B required for its evaluation make this method infeasible for cases where $B \neq I$.

2.1.4 Gibbs phenomenon and smoothing kernels

Attempting to approximate functions with jump discontinuities using only continuous and inherently smooth functions like polynomials, compulsorily leads to oscillations in the vicinity of the discontinuities [Wei+06; LTE01]. This effect is known as *Gibbs phenomenon*. To counteract the oscillatory nature of the resulting functions, modifications of the coefficients can be employed to smooth the approximation of the window function at the cost of decreasing the steepness of its flanks. A set of modifiers for the coefficients of the approximation for the sake of reducing oscillations is called *smoothing kernel*.

A number of appropriate kernels are detailed and analyzed in [Wei+06]; their definition is listed in Table 2.1. The modifiers are applied to the coefficients as $c_i' = c_i \varrho_i$. Figure 2.2 shows examples for the various forms the filtering function can assume. Filtering functions using kernels with parameters can vary greatly depending on the choice of parameters and only few examples are shown.

Name	Coefficient multipliers
Dirichlet	$\varrho_k \equiv 1$
Fejér	$\varrho_k = 1 - \frac{k}{d+1}$
Jackson	$\varrho_k = \frac{1}{d+2}\Big((d-k+2)\cos\Big(\frac{k\pi}{d+2}\Big) + \sin\Big(\frac{k\pi}{d+2}\Big)\cot\Big(\frac{\pi}{d+2}\Big)\Big)$
Lorentz	$\varrho_k(\mu) = \frac{1}{\sinh(\mu)}\sinh\Big(\mu - \frac{pk}{d+1}\Big)$
Lanczos	$\varrho_k(\mu) = \Big(\frac{d+1}{k\pi}\sin\Big(\frac{k\pi}{d+1}\Big)\Big)^{\mu}$
Wang-Zunger	$\varrho_k(\mu,\nu) = \exp\Big(-\Big(\frac{\mu k}{d+1}\Big)^{\nu}\Big)$

Table 2.1: *A selection of smoothing kernels.*

2.1.5 Restrictions

As has been pointed out before, the application of Chebyshev polynomial based filters to generalized eigenvalue problems is often infeasible if linear solves with B are not very inexpensive. Figure 2.3 shows an example of a Chebyshev polynomial filter. Due to the fact that the function is growing rapidly outside the region of interest, the contours have been capped for emphasis (colored in yellow). Since the polynomial approximation is only valid on the real axis, application to non-Hermitian problems is thus impossible as confirmed by the contour plot. It should be noted that Chebyshev polynomials have been used in the solution of non-Hermitian eigenproblems [Saa11]. The method is, however, fundamentally different from the approach here, even if it is used in subspace iteration as well. The foundation of the method described in [Wil65; Saa11] exploits the property of the Chebyshev polynomials to grow large quickly

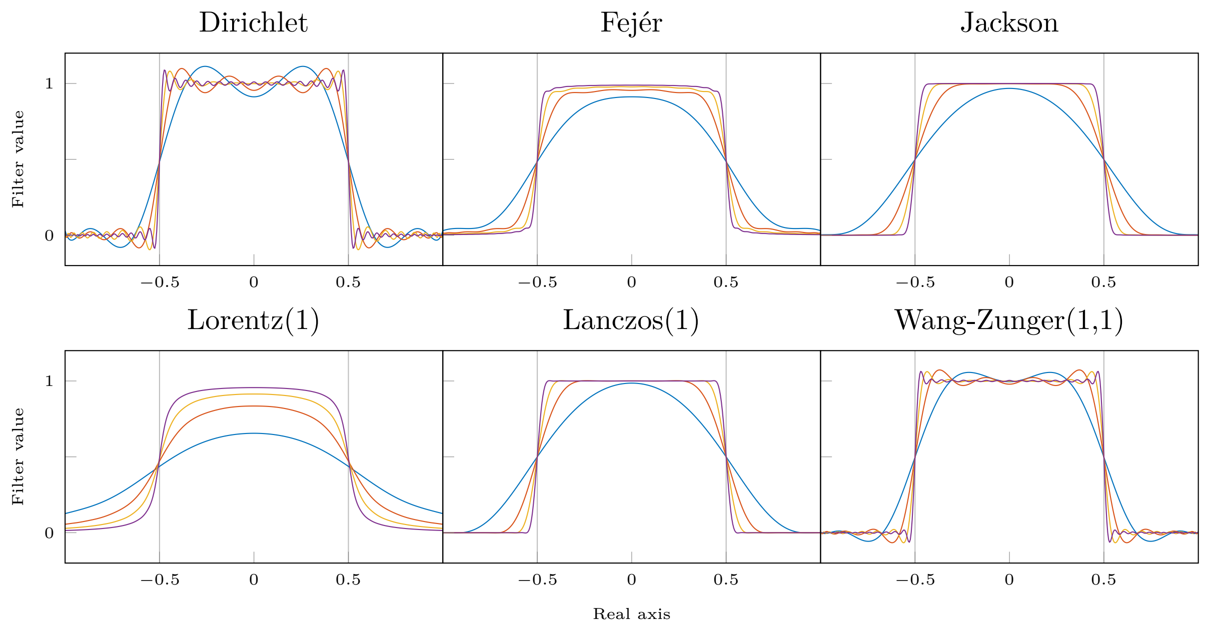


Figure 2.2: *Chebyshev filtering functions for $d = 10, 25, 50, 100$ (blue, red, yellow, purple, respectively; higher degrees yield better approximations) using various smoothing kernels.*

outside of $[-1, 1]$ in order to amplify directions associated with extremal eigenvalues, an idea very similar to power iteration in the context of subspace iteration. In the generalization of the Chebyshev polynomials to complex arguments using their hyperbolic definition [Saa11], this behavior is expanded to ellipses in the complex plane. Values outside the corresponding ellipse are amplified and thus dominate the iterated subspace. This method can also be translated to the approximation of a filtering function, see Section 2.4.2.

The Chebyshev approximation requires the spectrum of the matrix pencil to be mapped into $[-1, 1]$; this has two major implications. First, smaller target intervals require larger degrees to obtain the same steepness in the filter flanks at the interval boundaries. Since the total approximation range is fixed, this comes as no surprise. Second, since the Chebyshev polynomials are symmetric inside $[-1, 1]$, a target interval that does not have the form $[-a, a]$, i.e., is off-center, will result in an asymmetric filtering function since a more sudden change in amplitude is required on the side closest to -1 or 1. In particular, convergence will be limited by the flank with lesser steepness, possibly requiring larger degrees. An additional complication is the differing value of the filtering function at the interval boundaries. This will be important later, when methods for detecting the number of non-spurious Ritz values inside the interval are required. This last point can be remedied by choosing a transformation of the spectrum of the matrix pencil such that the transformed target interval is again centered, which comes at the cost of an even smaller transformed interval.

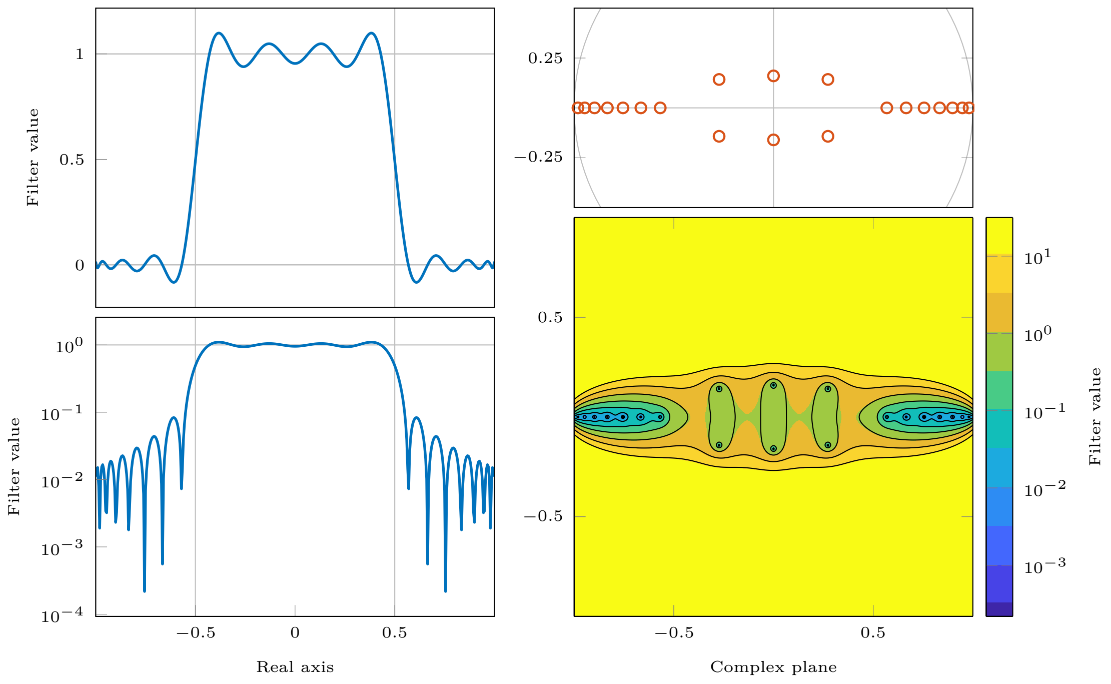


Figure 2.3: *Example of a Chebyshev polynomial based filter of degree* $d = 25$ *for the interval* $[-0.5, 0.5]$. *Left: non-logarithmic and logarithmic plots of the filter function. Top right: root map. Bottom right: logarithmic contour plot of the filter function in the complex plane.*

After all, the mapping of the (real) spectrum of a definite matrix pencil into $[-1, 1]$ requires the knowledge of at least its largest magnitude eigenvalue, or better its left- and rightmost eigenvalues—the *spectral range* of the matrix pencil. Using an algorithm such as Arnoldi or Lanczos (see Section 1.7.2) yields an approximation of the largest magnitude eigenvalue (positive or negative), say, λ^*. Performing the algorithm again with a shifted matrix pencil (see also Section 1.2.1)

$$(A - \lambda^* B, B)$$

yields λ', the other end λ_* of the spectral range, shifted by $-\lambda^*$,

$$\lambda_* = \lambda' + \lambda^*.$$

Assuming the spectral range is contained in $[\lambda^- = \lambda_* - \varepsilon, \lambda^+ = \lambda^* + \varepsilon]$ and the interval is valid (otherwise switch λ_* and λ^*) with some $\varepsilon > 0$ to account for potential inaccuracies, the matrix pencil mapped into $[-1, 1]$ is

$$\frac{2}{\lambda^+ - \lambda^-}\left(B^{-1}A - \frac{\lambda^+ + \lambda^-}{2} I\right) = \frac{2}{\lambda^+ - \lambda^-} B^{-1}A - \frac{\lambda^+ + \lambda^-}{\lambda^+ - \lambda^-} I$$

and the transformed target interval is

$$[\lambda_{\min}^*, \lambda_*^{\max}] = \left[\frac{2}{\lambda^+ - \lambda^-}\left(\lambda_{\min} - \frac{\lambda^+ + \lambda^-}{2}\right), \frac{2}{\lambda^+ - \lambda^-}\left(\lambda^{\max} - \frac{\lambda^+ + \lambda^-}{2}\right)\right].$$

2.1.6 Discrete approximation

A related method of approximation using Chebyshev polynomials often found in the literature is based on a restricted orthogonality relation that is limited to a certain set of discrete points and can be derived from the continuous relation above by employing a numerical integration scheme (see, e.g., [Pre+92]). It can be used to find approximations of the coefficients should they not be computable analytically. The Gaussian quadrature rule of order η for a weight function $w(z)$ as above gives the Chebyshev-Gauss quadrature method [AS74]

$$\int_{-1}^{1} \frac{g(z)}{\sqrt{1-z^2}}\,dz = \sum_{k=1}^{\eta} w_k g(z_k)$$

with abscissae

$$z_k = \cos\left(\frac{\pi(2k-1)}{2\eta}\right)$$

and weights

$$w_k = \frac{\pi}{\eta}.$$

The quadrature rule is exact for polynomials up to a degree of $2\eta - 1$. Therefore, with the continuous orthogonality relation and $T_i(z)T_j(z)$ being a polynomial of degree $i+j$, following [Pre+92], if $i+j < 2\eta$

$$\int_{-1}^{1} w(z)T_i(z)T_j(z)\,dz = \frac{\pi}{\eta}\sum_{k=1}^{\eta} T_i(z_k)T_j(z_k) = \begin{cases} 0 & \text{if } i \neq j \\ \frac{1}{2}\pi & \text{if } i = j \neq 0 \\ \pi & \text{if } i = j = 0 \end{cases}$$

and finally

$$\sum_{k=1}^{\eta} T_i(z_k)T_j(z_k) = \begin{cases} 0 & \text{if } i \neq j \\ \frac{1}{2}\eta & \text{if } i = j \neq 0 \\ \eta & \text{if } i = j = 0. \end{cases}$$

The integration nodes z_k are the η roots of $T_\eta(z)$,

$$T_\eta(z_k) = \cos\left(\eta \cos^{-1}\cos\left(\frac{\pi(2k-1)}{2\eta}\right)\right) = \cos\left(\frac{\pi(2k-1)}{2}\right) = 0.$$

Computing $\langle T_i, g\rangle_w$ in similar fashion yields the approximates

$$\int_{-1}^{1} w(z)g(z)T_i(z)\,dz \approx \frac{\pi}{\eta}\sum_{k=1}^{\eta} g(z_k)T_i(z_k) = \frac{\pi}{\eta}\sum_{k=1}^{\eta} g(z_k)\cos\left(\frac{i\pi(2k-1)}{2\eta}\right)$$

and, with this,

$$c_i \approx \begin{cases} \dfrac{1}{\eta}\displaystyle\sum_{k=1}^{\eta} g(z_k) & \text{if } i = 0 \\ \dfrac{2}{\eta}\displaystyle\sum_{k=1}^{\eta} g(z_k)\cos\left(\dfrac{i\pi(2k-1)}{2\eta}\right) & \text{else.} \end{cases}$$

This allows for approximations of g of degree $d < \eta$ so as not to violate the orthogonality relation. For the η points $g(z_k)$, there is a polynomial of at most degree $\eta - 1$, $p_{\eta-1}$, that passes through them, $p_{\eta-1}(z_k) = g(z_k)$. For this polynomial, the Chebyshev-Gauss quadrature of $\langle T_i, p_{\eta-1}\rangle_w$ is exact, $T_i\, p_{\eta-1}$ being a polynomial of at most degree $2\eta - 1$. Since there is no difference between the quadratures $\langle T_i, g\rangle_w$ and $\langle T_i, p_{\eta-1}\rangle_w$ in this case and a polynomial of degree d can be represented exactly as linear combination of orthogonal polynomials of degrees $0, \dots, d$ (such a linear combination is orthogonal to basis polynomials of higher degree and thus the higher order coefficients are zero), here the Chebyshev polynomials $T_0, \dots, T_d$, the approximating polynomial of degree $\eta - 1$ constructed above is $p_{\eta-1}$ and the approximation of g is therefore exact on the η roots z_k of T_η. Computing the coefficients c_i for the window function χ of an interval $[\lambda_{\min}, \lambda^{\max}] \subset [-1, 1]$ is trivial.

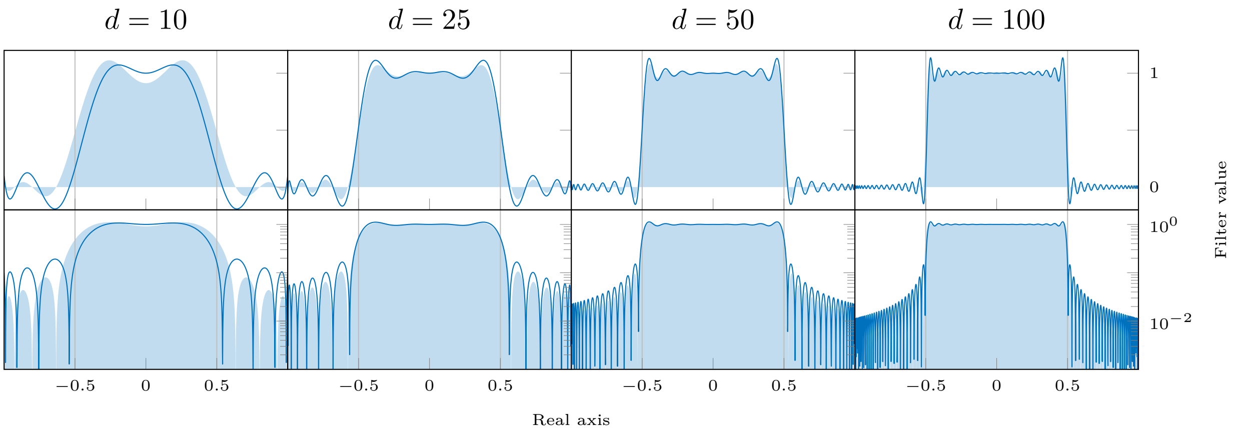


Figure 2.4: *Discrete Chebyshev filtering functions for $d = \eta - 1 = 10, 20, 50, 100$. The continuous Chebyshev filter of identical degree is shown as filled area for comparison.*

Figure 2.4 shows some examples for resulting filtering functions. The position and form of the filter flanks is determined by the location of the interval boundaries in between two roots of T_η; the discrepancy vanishes with growing degree and increased root frequency. Due to this circumstance and the additional layer of approximation during the computation of $\langle T_i, g\rangle_w$, this method is inherently inferior to the method introduced before, and is only included for the sake of clarification and completeness.

2.2 Contour integration

An alternative way to construct a filtering function, that ultimately yields a rational approximation and is better suited for generalized eigenvalue problems, is based on contour integration. Instead of approximating the window function directly, we find an exact integral representation of it by chasing a pole along the real axis and using a pole-counting integral to determine whether the pole is inside an integration contour or not. The final filtering function is produced by the numerical integration method of choice and again an approximation of the window function. We will refer to these filters as *contour integration filters* or *Cauchy filters*.

2.2.1 Representation of the window function

Define a function $\phi(z) = \frac{1}{z-\lambda}$, $z \in \mathbb{C}$ with a pole at $z = \lambda$. By varying λ, the pole is moved around on the complex plane. With a contour $\mathcal{C} = \partial\Omega$, if it is possible to detect whether the pole is inside that contour, an exact filtering function can be constructed. Note that the behavior on $\partial\Omega$ is undefined.

Theorem 2.1 (Cauchy's theorem [Ahl79]) *Let g be a function that is holomorphic on $\mathcal{U} \subseteq \mathbb{C}$ and $\mathcal{C}$ a simple closed curve in $\mathcal{U}$. Then*

$$\oint_{\mathcal{C}} g(z)\, dz = 0.$$

With this, letting the integration contour around an isolated singularity collapse to a single point yields the following (here slightly specialized) statement.

Corollary 2.1 (Cauchy's integral formula [Ahl79]) *Let h be a holomorphic function and $\mathcal{C}$ be a simple closed curve in $\mathbb{C}$. Then, for any point a in the interior of $\mathcal{C}$,*

$$h(a) = \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} \frac{h(z)}{z-a} dz.$$

Therefore, we can define the filtering function

$$\chi_\Omega(\lambda) = \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} \frac{1}{z-\lambda} = \begin{cases} 1 & \text{if } \lambda \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

as perfect representation of χ_Ω. As before, a projector using this filtering function may be written as

$$\begin{aligned} X \operatorname{diag}(\chi_\Omega(\lambda_1), \dots, \chi_\Omega(\lambda_N))X^{-1} &= \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} X \operatorname{diag}\Big(\frac{1}{z-\lambda_1}, \dots, \frac{1}{z-\lambda_N}\Big) X^{-1}\, dz \\ &= \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} X(zI-\Lambda)^{-1} X^{-1}\, dz \\ &= \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} (zB-A)^{-1} B\, dz. \end{aligned}$$

The matrix $(zB-A)^{-1}B$ is conventionally called *resolvent* of (A, B). Functionally, there is no difference to choosing $(A-zB)^{-1}B$ which we have come to know as the compatible matrix pencil of a shift-and-invert transformation of (A, B). It merely switches the sign of the integral and the direction of computed eigenvectors with it.

2.2.2 Numerical integration

The application of the previously constructed projector matrix to a set of right-hand side column vectors Y,

$$\frac{1}{2\pi\mathbf{i}} \oint_C (zB - A)^{-1} BY \, dz,$$

can now be computed by means of numerical integration, or *quadrature*. We would like to be able to compute the result as a matrix function, therefore we require the formula to only consist of matrix additions, multiplications, and the solution of linear systems, which limits the scalar form to rational functions.

The most commonly known quadrature schemes that fit these criteria are approximations that take the form

$$\int_a^b g(z) \, dz \approx \sum_{i=0}^{k} w_i g(z_i),$$

basically the partial fraction form of a rational function with fewer roots than (simple) poles. Examples of numerical integration schemes that take this form are the *closed Newton-Cotes formulas* (for example the trapezoidal rule), *open Newton-Cotes formulas* (for example the midpoint rule), and *Gaussian quadrature* with weight function $\omega(z) \equiv 1$, the *Gauss-Legendre quadrature* rule. An overview can also be found in [Krä14] and a more in-depth reading is provided by the sources cited therein [DR84; Kre98; KU98; Wei02; Mul06]. A condensation of integration rules that is as comprehensive as it is pragmatic can be found in [AS74] or [Olv+10]. We will shortly introduce the examples mentioned above.

2.2.2.1 Trapezoidal rule

Newton-Cotes formulas are approximations of the integral of a function obtained by piecewise approximation of the function using simple polynomials based on the function's value at equidistant subdivisions (abscissae) of the integration interval. In closed Newton-Cotes formulas, the boundaries of the integration interval serve as first and last abscissae. If the degree of the approximating polynomial is one, i.e., the function is approximated by piecewise linear segments, the integral is approximated by a sequence of trapezoids which are defined by the values of the target function at the abscissae, the outermost points of the trapezoids. The result is known as the *trapezoidal rule*.

The integral is approximated as the sum of the areas of $k-1$ trapezoids, resulting in k abscissae with distance $\rho = (b-a)/(k-1)$,

$$\begin{aligned}
\int_a^b g(z) \, dz &\approx \sum_{i=1}^{k-1} \frac{\rho}{2}(g(z_i) + g(z_{i+1})) \quad \text{with} \quad z_i = a + (i-1)\rho \\
&= \sum_{i=1}^{k} w_i g(z_i) \quad \text{with} \quad w_i = \begin{cases} \frac{\rho}{2} & \text{if } i = 1 \vee i = k \\ \rho & \text{else.} \end{cases}
\end{aligned}$$

Here, the final form results from the duplicate appearance of all abscissae but the very first and very last one. For closed contours it is $z_1 = z_k$ and the rule becomes

$$\int_a^b g(z)\,dz \approx \sum_{i=1}^{k-1} \rho g(z_i).$$

2.2.2.2 Midpoint rule

In open Newton-Cotes formulas, the boundaries of the integration interval do not serve as abscissae. The abscissae are spaced such that the distance to the interval boundaries is half the distance between two abscissae. If the degree of the approximating polynomial is zero, i.e., the function is approximated by piecewise constant segments, the integral is approximated by a sequence of rectangles whose heights are defined by the value of the target function at their midpoint position. The result is known as the *midpoint rule*.

The integral is approximated as the sum of the areas of k rectangles, resulting in k abscissae with distance $\rho = (b-a)/k$,

$$\int_a^b g(z)\,dz \approx \sum_{i=1}^{k} \rho g(z_i) \quad \text{with} \quad z_i = a + \left(i - \tfrac{1}{2}\right)\rho.$$

The weights are constant, $w_i = \rho$.

2.2.2.3 Gauss-Legendre quadrature

Gaussian quadrature rules of order k are defined by requiring exactness for polynomials of degree $d \leq 2k-1$. Different quadrature rules are associated with certain families of polynomials whose zeros determine the abscissae, and their value at these abscissae relate to the weights of the quadrature rule. In particular, abscissae are not necessarily positioned equidistantly. For Gauss-Legendre quadrature, these polynomials are the Legendre polynomials on $[-1,1]$. The computation of the abscissae and weights is described in [GW69], see also [Krä14]. We will reiterate the process here in condensed form for the sake of completeness.

Every sequence of orthogonal polynomials is generated by a three term recurrence of the form

$$p_j(z) = (a_j z + b_j)p_{j-1}(z) - c_j p_{j-2}(z) \quad \text{for} \quad j = 1, \ldots, k, \tag{2.5}$$

which may be rewritten in matrix from as

$$z \begin{pmatrix} p_0(z) \\ p_1(z) \\ \vdots \\ p_{k-1}(z) \end{pmatrix} = \begin{pmatrix} -\frac{b_1}{a_1} & \frac{1}{a_1} & & & \\ \frac{c_2}{a_2} & -\frac{b_2}{a_2} & \frac{1}{a_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{c_{k-1}}{a_{k-1}} & -\frac{b_{k-1}}{a_{k-1}} & \frac{1}{a_{k-1}} \\ & & & \frac{c_k}{a_k} & -\frac{b_k}{a_k} \end{pmatrix} \begin{pmatrix} p_0(z) \\ p_1(z) \\ \vdots \\ p_{k-1}(z) \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \frac{1}{a_k} p_k(z) \end{pmatrix}$$

or, in short,

$$zp(z) = Tp(z) + \frac{1}{a_k} p_k(z) e_k,$$

with T being the tridiagonal matrix. Then $p_k(\mu_j) = 0$ only if $\mu_j p(\mu_j) = Tp(\mu_j)$, i.e., μ_j is an eigenvalue of T and $p(\mu_j)$ is the corresponding eigenvector. Therefore, the eigenvalues μ_j of T are the abscissae of the Gauss quadrature rule. A diagonal similarity transform D of T yields a symmetric tridiagonal matrix

$$J = DTD^{-1} = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{pmatrix}$$

where

$$\alpha_i = -\frac{b_i}{a_i} \quad \text{and} \quad \beta_i = \sqrt{\frac{c_{i+1}}{a_i a_{i+1}}}$$

and it can be shown that, if q_j are unit-length eigenvectors of J with

$$Jq_j = \mu_j q_j \quad \text{for} \quad j = 1, \ldots, k,$$

the weights are given as

$$\eta_j = q_{j,1}^2 \int_{-1}^{1} \omega(z)\, dz$$

where $q_{j,1}$ is the first component of eigenvector q_j. In the case of Gauss-Legendre quadrature $\omega(z) \equiv 1$ and hence

$$\eta_j = 2q_{j,1}^2.$$

For Legendre Polynomials[1] $P_j(z)$ the recurrence relation Equation (2.5) is given as [AS74]

$$jP_j(z) = (2j-1)zP_{j-1}(z) - (j-1)P_{j-2}(z)$$

and, with this, the coefficients are

$$a_j = \frac{2j-1}{j}, \quad b_j \equiv 0, \quad c_j = \frac{j-1}{j}.$$

Therefore

$$\alpha_j \equiv 0, \quad \beta_j = \sqrt{\frac{j}{(j+1)\left(\frac{2j-1}{j}\right)\left(\frac{2j+1}{j+1}\right)}} = \sqrt{\frac{j}{j+1}\frac{j}{j^2}\frac{j+1}{4-\frac{1}{j^2}}} = \frac{1}{\sqrt{4-\frac{1}{j^2}}}.$$

The computation of the abscissae and weights now requires the solution of a very small standard eigenproblem and any tridiagonal direct eigensolver will suffice for this task.

1) Legendre functions of order zero.

Since the quadrature rule is restricted to the interval $[-1,1]$, a generalization to arbitrary intervals is required. Let μ_i and η_i be the Gauss-Legendre nodes and weights on $[-1,1]$ from above. The generalized nodes and weights on $[a,b]$, z_i and w_i, are then given by the transformation to arbitrary intervals via substitution,

$$\begin{aligned}\int_a^b g(z)dz &= \int_{-1}^1 \frac{b-a}{2} g\left(\frac{b-a}{2}z + \frac{b+a}{2}\right)dz \\ &\approx \sum_{i=1}^k \frac{b-a}{2}\eta_i g\left(\frac{b-a}{2}\mu_i + \frac{b+a}{2}\right) = \sum_{i=1}^k w_i g(z_i).\end{aligned}$$

2.2.3 Computation

First and foremost, the application of one of the integration schemes introduced above along a contour demands the parametrization of said contour. This is easily accomplished in the case $\mathcal{C}$ is a circle around c with radius r (for definite pairs with target interval $[\lambda_{\min}, \lambda^{\max}]$ this amounts to $c = (\lambda^{\max} + \lambda_{\min})/2$ and $r = (\lambda^{\max} - \lambda_{\min})/2$). The parametrization $\varphi_c(t)$ is the transformation to polar coordinates with an additional shift,

$$z = \varphi_c(t) = c + re^{\mathbf{i}t} \quad \text{for} \quad t \in [0, 2\pi),$$

and integration by substitution requires the knowledge of the first derivative of the parametrization,

$$\varphi_c'(t) = \mathbf{i}re^{\mathbf{i}t}.$$

Another easily parameterized contour is the ellipse around c with semi-major axis of length r along the real axis and eccentricity e. The length of the semi-minor axis then is

$$\ell = \sqrt{(1-e^2)r^2}$$

and the parametrization $\varphi_e(t)$ can be written as

$$z = \varphi_e(t) = c + r\cos t + \mathbf{i}\ell \sin t \quad \text{for} \quad t \in [0, 2\pi).$$

Its first derivative is

$$\varphi_e'(t) = -r\sin t + \mathbf{i}\ell\cos t.$$

Other shapes of contours are possible, e.g., a piecewise linear curve. Now, let $\mathcal{R}(z) = (zB - A)^{-1}BY$ be the resolvent. To arrive at the formulation originally given in [Pol09], that also allows to halve the number of integration nodes and linear systems under certain circumstances, we lay out the derivation in more detail. Given a parametrization $z = \varphi(t)$ which transforms the integral to circular or elliptic contours gives

$$\frac{1}{2\pi\mathbf{i}}\int_{\mathcal{C}} \mathcal{R}(z)\,dz = \frac{1}{2\pi\mathbf{i}}\int_0^{2\pi} \varphi'(t)\mathcal{R}(\varphi(t))\,dt.$$

If A and B are Hermitian and the contour hence is symmetrical to the real axis,

$$\varphi(-t) = \overline{\varphi(t)} \quad \text{and} \quad \varphi'(-t) = -\overline{\varphi'(t)},$$

the integral can be split into two half-contours [Pol09] and recombined as

$$\begin{aligned}
\frac{1}{2\pi\mathbf{i}}\int_0^{2\pi}\varphi'(t)\mathcal{R}(\varphi(t))dt &= \frac{1}{2\pi\mathbf{i}}\left[\int_0^{\pi}\varphi'(t)\mathcal{R}(\varphi(t))dt + \int_\pi^{2\pi}\varphi'(t)\mathcal{R}(\varphi(t))dt\right]\\
&= \frac{1}{2\pi\mathbf{i}}\left[\int_0^{\pi}\varphi'(t)\mathcal{R}(\varphi(t))dt + \int_0^{\pi}\varphi'(-t)\mathcal{R}(\varphi(-t))dt\right]\\
&= \frac{1}{2\pi\mathbf{i}}\left[\int_0^{\pi}\varphi'(t)\mathcal{R}(\varphi(t))dt - \int_0^{\pi}\overline{\varphi'(t)}\mathcal{R}\left(\overline{\varphi(t)}\right)dt\right]\\
&= \frac{1}{2\pi\mathbf{i}}\int_0^{\pi}\varphi'(t)\mathcal{R}(\varphi(t)) - \overline{\varphi'(t)}\mathcal{R}\left(\overline{\varphi(t)}\right)dt.
\end{aligned}$$

If furthermore A, B, and Y are real, then

$$(\overline{z}B - A)^{-1}BY = \left(\overline{zB - A}\right)^{-1}BY = \overline{(zB - A)^{-1}BY}$$

and

$$\begin{aligned}
\frac{1}{2\pi\mathbf{i}}\int_0^{\pi}\varphi'(t)\mathcal{R}(\varphi(t)) - \overline{\varphi'(t)}\mathcal{R}\left(\overline{\varphi(t)}\right)dt &= \frac{1}{2\pi\mathbf{i}}\int_0^{\pi}\varphi'(t)\mathcal{R}(\varphi(t)) - \overline{\varphi'(t)\mathcal{R}(\varphi(t))}\,dt\\
&= \frac{1}{2\pi\mathbf{i}}\int_0^{\pi}2\mathbf{i}\Im\{\varphi'(t)\mathcal{R}(\varphi(t))\}\,dt\\
&= \frac{1}{\pi}\int_0^{\pi}\Im\{\varphi'(t)\mathcal{R}(\varphi(t))\}\,dt.
\end{aligned}$$

Further,

$$\begin{aligned}
\frac{1}{2\pi\mathbf{i}}\int_0^{\pi}\varphi'(t)\mathcal{R}(\varphi(t)) - \overline{\varphi'(t)\mathcal{R}(\varphi(t))}\,dt &= -\frac{1}{2\pi}\int_0^{\pi}\mathbf{i}\varphi'(t)\mathcal{R}(\varphi(t)) + \overline{\mathbf{i}\varphi'(t)\mathcal{R}(\varphi(t))}\,dt\\
&= -\frac{1}{2\pi}\int_0^{\pi}2\Re\{\mathbf{i}\varphi'(t)\mathcal{R}(\varphi(t))\}\,dt\\
&= -\frac{1}{\pi}\int_0^{\pi}\Re\{\mathbf{i}\varphi'(t)\mathcal{R}(\varphi(t))\}\,dt,
\end{aligned}$$

which might be useful for example if $\varphi'(t) \propto \mathbf{i}$. These are the final forms as found in [Pol09]. They allow to cut the number of linear solves in half, should the eigenproblem be symmetric and real. This optimization can also be applied to all filters that are still to follow in this chapter. Note that the sign of the integral merely flips the resulting eigenvectors. It is equivalent to inverting the integration direction of $\mathcal{C}$. Combining weights, parametrization, and numerical integration,

$$\frac{1}{2\pi\mathbf{i}}\int_{\mathcal{C}}\mathcal{R}(z)dz \approx \sum_{i=1}^{k}\varpi_i\mathcal{R}(z_i) + \overline{\varpi_i}\mathcal{R}(\overline{z_i})$$

with

$$\varpi_i = \frac{1}{2\pi\mathbf{i}}w_i\varphi'(t_i) \quad \text{and} \quad z_i = \varphi(t_i).$$

The degree of this filter is $d = 2k$.

2.2.4 Selection function

The effective filtering function for a given quadrature scheme is sometimes referred to as *selection function* in the literature [Krä14; Lau12]. As every other filter, it specifies the dampening of an eigendirection based on the location of its associated eigenvalue when applied via numerical integration. This function can simply be retrieved by applying the quadrature scheme to the ideal representation of the window function derived in Equation (2.4),

$$f(\lambda) = \sum_{i=1}^{k} \frac{\varpi_i}{z_i - 1} + \frac{\overline{\varpi_i}}{\overline{z_i} - 1},$$

and, if the function is reduced to the real axis using half the contour,

$$f(\lambda) = 2\sum_{i=1}^{k} \Re\left\{\frac{\varpi_i}{z_i - \lambda}\right\}.$$

Figure 2.5 shows an example of a Cauchy integral based filter function using Gauss-Legendre quadrature. Only the six smallest of the overall ten roots are shown. The

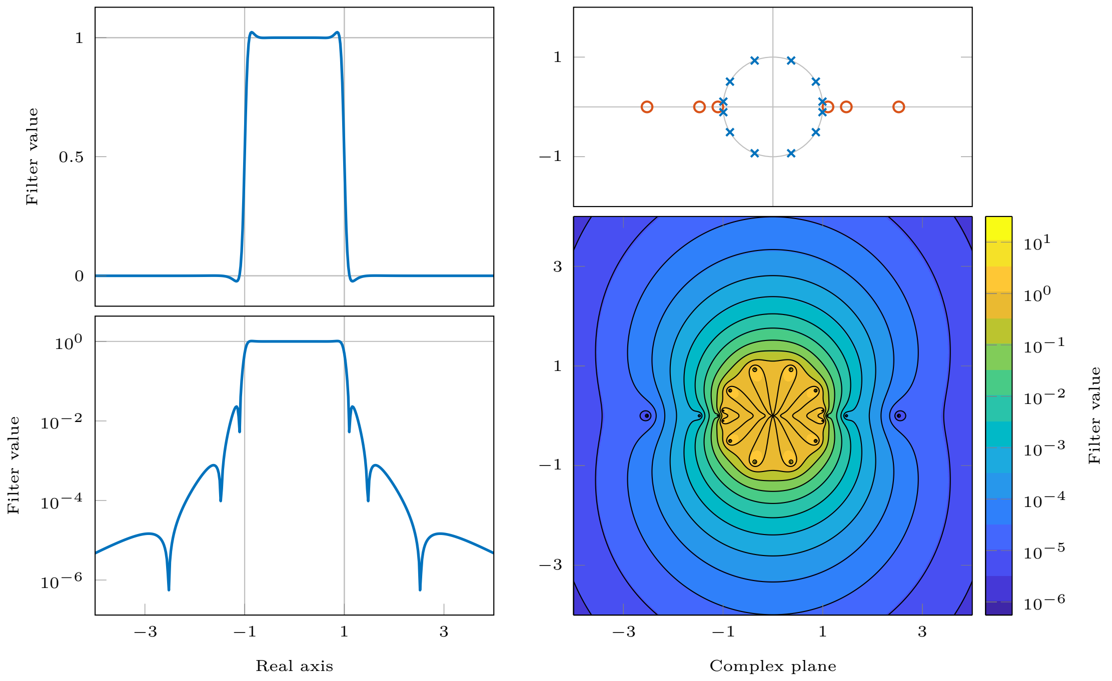


Figure 2.5: *Example of a Cauchy filter of degree* $d = 12$ *using Gauss-Legendre quadrature on a circular contour. Left: non-logarithmic and logarithmic plots of the filter function. Top right: pole-root map. Roots are marked as circles, poles as crosses. Bottom right: logarithmic contour plot of the filter function in the complex plane.*

contour plot reveals that the filtering flank in the complex plane is only sharp at the interval boundaries on the real axis. It is thus to be expected that the convergence speed diminishes proportionally to the least steep section of the filter along the contour (assuming eigenvalues are present in this region). The midpoint quadrature rule results in a more even distribution of poles, i.e., in less variation of flank steepness around the contour and, as such, is better suited for solving non-Hermitian eigenproblems.

Figure 2.6 shows examples of filtering functions produced by different integration schemes. Note in particular that the trapezoidal rule produces discontinuities at the

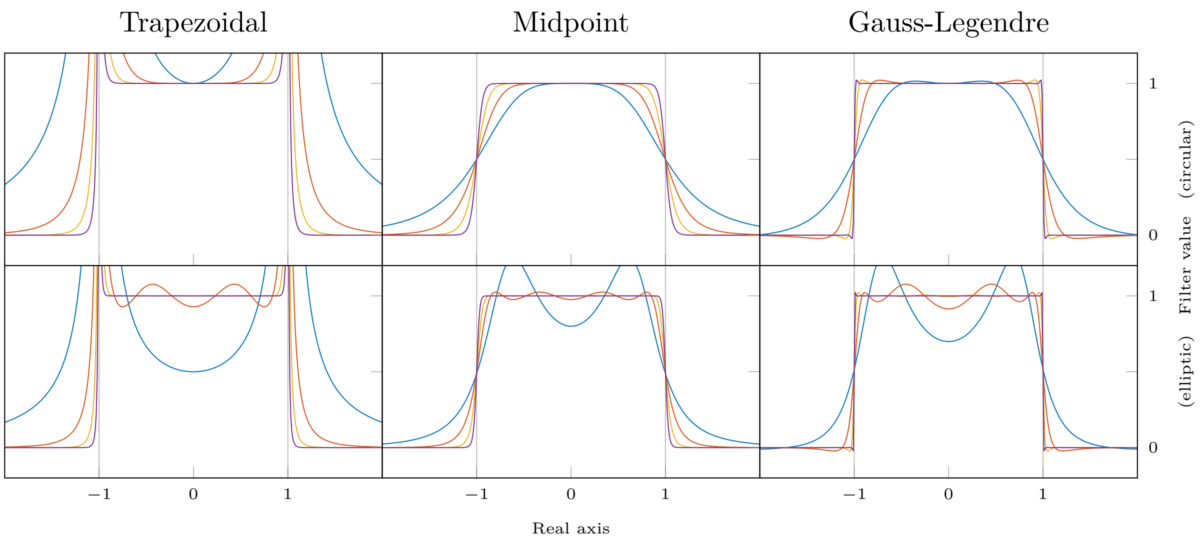


Figure 2.6: *Cauchy filtering functions for* $k = 2, 4, 8, 16$ *(blue, red, yellow, purple, respectively; higher orders yield better approximations) and different integration schemes.*
Top: circular contour, bottom: elliptic contour with $\ell = 1/2$.

interval boundaries and is therefore not well suited for spectral filtering. Deviating from the circular contour and bringing the poles closer to the real axis additionally introduces oscillations in the passband. While the oscillations vanish again with growing degree, increasing the eccentricity of the ellipse amplifies the oscillations. Moving the poles away from the real axis smooths the filter and reduces the steepness of the flanks. It can be shown [Krä14; TP] that for the midpoint and Gauss-Legendre rules the filtering function passes 1/2 at the interval boundaries.

2.3 Zolotarev approximation

In function approximation theory, the solution to Zolotarev's fourth problem [Zol77] finds the best approximation of the sign function

$$\operatorname{sgn}(z) = \frac{z}{\sqrt{z^2}} = \begin{cases} -1 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \end{cases}$$

on the intervals $[-1, -\kappa] \cup [\kappa, 1]$, $0 < \kappa < 1$, by a rational function of order m, in the sense that its maximum deviation from unity on the above intervals is the smallest among all rational functions of identical order. Filtering functions of the Zolotarev type were used in spectral projection algorithms in [Güt+15] and [LY17]. Zolotarev's solution to the problem above is based on elliptic functions.

The complete derivation of the solution exceeds the scope of this work and the reader is therefore referred to [Akh90; PP87; Akh56; Ken05] for an in-depth analysis of Zolotarev's problems, their solution, and an introduction to rational approximation and elliptic functions in general. We will, however, detail all formulas and algorithms required to implement filtering functions based on Zolotarev's approximation as well as the transformation from sign function to window function following the references mentioned above.

2.3.1 Approximation of the sign function

The *Jacobi elliptic functions* are defined as inverses of the *incomplete elliptic integral of the first kind* for the *modulus* κ,

$$\nu = \int_0^z \frac{1}{\sqrt{(1-t^2)(1-\kappa^2 t^2)}}\, dt, \tag{2.6}$$

via

$$\begin{aligned} \operatorname{sn}(\nu, \kappa) &= z \\ \operatorname{cn}(\nu, \kappa) &= \sqrt{1 - \operatorname{sn}^2(\nu, \kappa)}\ ^{\dagger} \\ \operatorname{dn}(\nu, \kappa) &= \sqrt{1 - \kappa^2 \operatorname{sn}^2(\nu, \kappa)} \end{aligned}$$

and

$$K(\kappa) = \int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1-\kappa^2 \sin^2\theta}}\, d\theta = \int_0^1 \frac{1}{\sqrt{(1-t^2)(1-\kappa^2 t^2)}}\, dt$$

is the *complete elliptic integral of the first kind* for modulus κ.

Throughout the literature, solutions to Zolotarev's problem of approximating the sign function might be presented in varying form. In particular, the approximation problem of the sign function on the intervals

$$[-1, -\kappa] \cup [\kappa, 1] \quad \text{(henceforth identified by \textcircled{I})}$$

is equivalent to approximating the sign function on intervals

$$\left[-\frac{1}{\kappa}, -1\right] \cup \left[1, \frac{1}{\kappa}\right] \quad \text{(henceforth identified by \textcircled{II})}.$$

†) The validity of this definition is tied to the fact that we do not have to care about the sign of $\operatorname{cn}(\nu, \kappa)$ since it only ever appears squared and $\nu \leq K(\kappa)$. In [Ken05] it is pointed out that the sign is -1 for $(4\eta + 1)K(\kappa) < \nu < (4\eta + 3)K(\kappa)$, $\eta \in \mathbb{N}_0$.

Both are related through division by κ. The associated solutions take the form

$$\textcircled{I}\quad \frac{2\zeta}{1+\zeta}\frac{z}{\mathcal{M}\kappa}\prod_{j=1}^{\eta}\frac{\kappa^2+c_{2j}z^2}{\kappa^2+c_{2j-1}z^2} \quad\text{and}\quad \textcircled{II}\quad \frac{2\zeta}{1+\zeta}\frac{z}{\mathcal{M}}\prod_{j=1}^{\eta}\frac{1+c_{2j}z^2}{1+c_{2j-1}z^2}.$$

For the definition of the coefficients c_i, we mix the forms found in [Akh90] and [Ken05] to circumvent divisions by zero which otherwise would have to be interpreted as infinity,

$$c_j = \frac{\operatorname{cn}^2\!\left(K'\frac{j}{2\eta},\kappa'\right)}{\operatorname{sn}^2\!\left(K'\frac{j}{2\eta},\kappa'\right)} = \frac{1-\operatorname{sn}^2\!\left(K'\frac{j}{2\eta},\kappa'\right)}{\operatorname{sn}^2\!\left(K'\frac{j}{2\eta},\kappa'\right)}.$$

Here, $\kappa' = \sqrt{1-\kappa^2}$ is the complementary modulus for κ and $K' = K(\kappa')$. It is $\operatorname{sn}(K',\kappa') = 1$ and thus $c_{2\eta} = 0$. We may therefore instead write

$$\textcircled{I}\quad \frac{2\zeta}{1+\zeta}\frac{z\kappa}{\mathcal{M}}\frac{\prod_{j=1}^{\eta-1}\kappa^2+c_{2j}z^2}{\prod_{j=1}^{\eta}\kappa^2+c_{2j-1}z^2} \quad\text{and}\quad \textcircled{II}\quad \frac{2\zeta}{1+\zeta}\frac{z}{\mathcal{M}}\frac{\prod_{j=1}^{\eta-1}1+c_{2j}z^2}{\prod_{j=1}^{\eta}1+c_{2j-1}z^2}.$$

Finally, the scaling factors are given as

$$\mathcal{M} = \prod_{j=1}^{\eta}\frac{1+c_{2j}}{1+c_{2j-1}} = \prod_{j=1}^{\eta}\frac{\operatorname{sn}^2\!\left(K'\frac{2j-1}{2\eta},\kappa'\right)}{\operatorname{sn}^2\!\left(K'\frac{2j}{2\eta},\kappa'\right)}$$

and

$$\frac{1}{\zeta} = \frac{\xi}{\mathcal{M}}\prod_{j=1}^{\eta}\frac{1+c_{2j}\xi^2}{1+c_{2j-1}\xi^2}$$

with

$$\xi = \frac{1}{\operatorname{dn}\!\left(\frac{K'}{2\eta},\kappa'\right)}.$$

See in particular [Ken05] for a derivation of $1/\zeta$ and ξ. Let—for convenience of notation—$c_j' = 1/c_j$. Often the solution is reformulated as

$$\textcircled{I}\quad \frac{2\zeta}{1+\zeta}\frac{z\kappa}{\mathcal{M}}\frac{\prod_{j=1}^{\eta-1}c_{2j}}{\prod_{j=1}^{\eta}c_{2j-1}}\frac{\prod_{j=1}^{\eta-1}\kappa^2c_{2j}'+z^2}{\prod_{j=1}^{\eta}\kappa^2c_{2j-1}'+z^2} \qquad \textcircled{II}\quad \frac{2\zeta}{1+\zeta}\frac{z}{\mathcal{M}}\frac{\prod_{j=1}^{\eta-1}c_{2j}}{\prod_{j=1}^{\eta}c_{2j-1}}\frac{\prod_{j=1}^{\eta-1}c_{2j}'+z^2}{\prod_{j=1}^{\eta}c_{2j-1}'+z^2}$$

and the several constants are collected in

$$D = \frac{2\zeta}{1+\zeta}\frac{1}{\mathcal{M}}\frac{\prod_{j=1}^{\eta-1}c_{2j}}{\prod_{j=1}^{\eta}c_{2j-1}}.$$

The maximum deviation of the approximation from -1 or 1 on the corresponding intervals is

$$\frac{1-\zeta}{1+\zeta}.$$

2.3.2 Window function

So far, the result above approximates the sign function on two approximation intervals with parameter κ. The approximation equioscillates about -1 or 1 on the corresponding intervals for $\textcircled{I}$ and $\textcircled{II}$. Constructing an approximation of the window function based on this approximation of the sign function requires a mapping of arguments of the window filter function to arguments of the sign function. In [Güt+15], a Möbius transformation is employed to achieve this transformation.

A Möbius transformation is a function $T(z) : \mathbb{C}_\infty \longrightarrow \mathbb{C}_\infty$ on the Riemann sphere of the form

$$T(z) = \frac{\alpha + \beta z}{\gamma + \delta z}$$

which is obviously invariant w.r.t. multiplying its four coefficients with a constant and w.l.o.g. we therefore may choose one of the non-zero coefficients equal to one. This also shows that three point pairs are necessary to uniquely determine the associated Möbius transformation that maps the pairs onto each other. If a fourth constraint is to be enforced, here due to requirements of symmetry, an additional variable has to be introduced. The transformation described in [Güt+15] is derived from the four constraints

$$\begin{array}{lllll}
\textcircled{I} & T(-1) = 0 & T(-G) = \kappa & T(G) = 1 & T(1) = \infty \\
\textcircled{II} & T(-1) = 0 & T(-G) = 1 & T(G) = \dfrac{1}{\kappa} & T(1) = \infty
\end{array}$$

which finally eliminates the initial difference between $\textcircled{I}$ and $\textcircled{II}$. The additional variable G represents the transition band of the resulting filter. The transformation resulting from the above constraints is equivalent to

$$\begin{array}{lllll}
\textcircled{I} & T\left(-\dfrac{1}{G}\right) = -\kappa & T(-G) = \kappa & T(G) = 1 & T\left(\dfrac{1}{G}\right) = -1 \\
\textcircled{II} & T\left(-\dfrac{1}{G}\right) = -1 & T(-G) = 1 & T(G) = \dfrac{1}{\kappa} & T\left(\dfrac{1}{G}\right) = -\dfrac{1}{\kappa}
\end{array}$$

but the first formulation makes solving the system of equations somewhat simpler. The visualization in Figure 2.7 is meant to clarify the choice of transformation for the case $\textcircled{I}$ and the second formulation of the Möbius transformation since it relates to passband, stopband, and transition band. Let $s_\eta(x)$ be the approximated sign function. The second illustration of s_η is warped to improve visibility and to emphasize its behavior outside $[-1, 1]$. It is worth noting that $[G, 1/G]$ (third image) is (reversely) mapped to $(-\infty, -1] \cup [1, \infty)$ (second and first image). Equivalently, $(-\infty, -1/G] \cup [1/G, \infty)$ (third image) is mapped to $[-1, -\kappa]$ (second and first image), bounding the filter function outside of $[-1/G, 1/G]$ (third image) accordingly. The warped representation of the sign function hints at why the resulting window function can be symmetric at all. In the representations of the approximated sign function of the second image, the behavior outside $[-1, 1]$ is in principle identical to the behavior inside $[-\kappa, \kappa]$, the limit for $z \longrightarrow \pm\infty$ of $s_\eta(z)$ being equivalent to $s_\eta(0)$ in this sense.

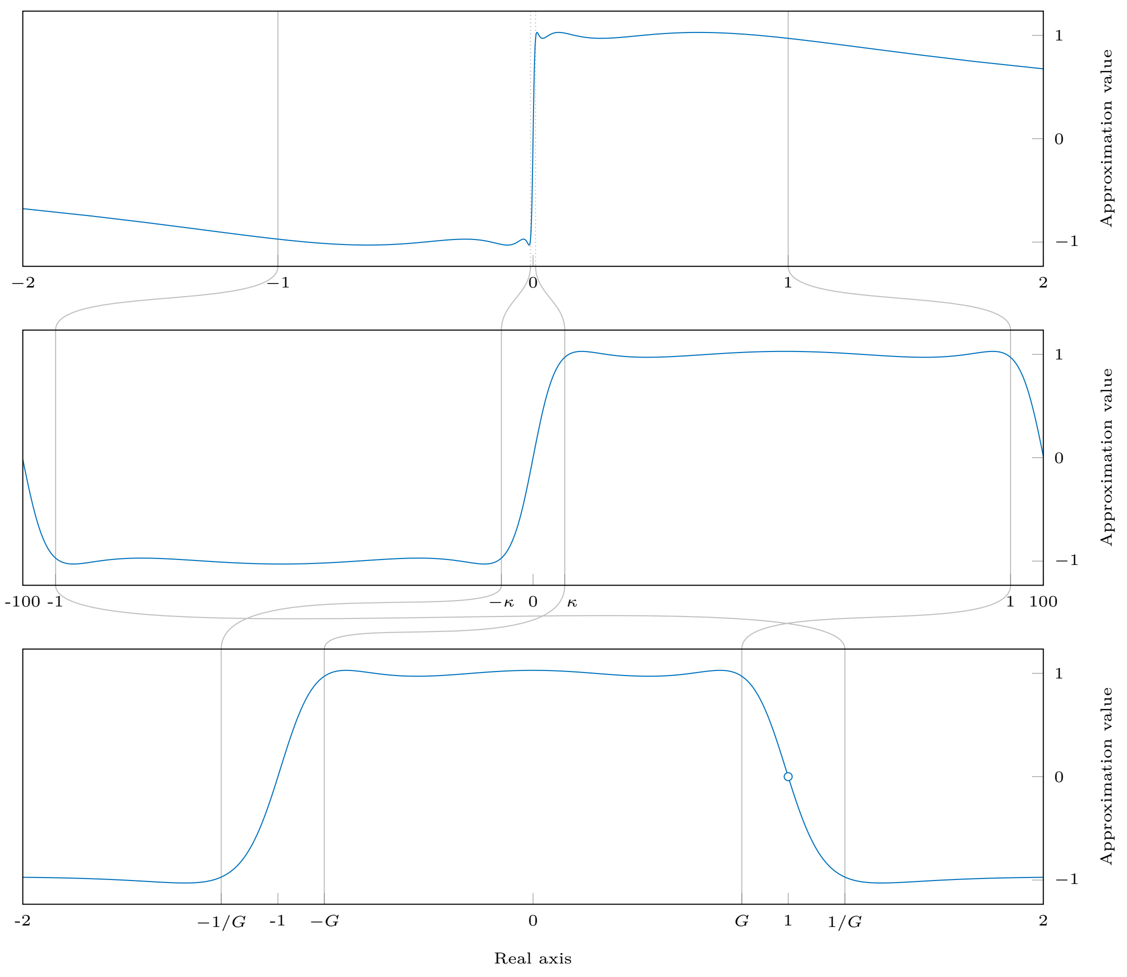


Figure 2.7: *Visualization of the Möbius transformation for $\eta = 3$ and $\kappa = 10^{-2}$.*
Top: Approximation of the sign function $s_\eta(z)$ on $[-1, -\kappa] \cup [\kappa, 1]$.
Middle: Distorted view of the sign function for emphasis.
Bottom: Resulting approximation of the window function $s_\eta(T(z))$.

The corresponding point on the window function approximation is marked by a small circle in the third image.

Solving the system of equations for the parameters of the Möbius transformation with $\alpha = \beta$ and $\gamma = -\delta$ from the first and fourth constraint and choosing $\gamma = 1$ yields

$$\kappa = \frac{(G-1)^2}{(G+1)^2} \qquad \Longrightarrow \qquad G = \frac{1+\sqrt{\kappa}}{1-\sqrt{\kappa}}$$

for both Ⓘ and Ⓘⓘ as well as

$$\text{Ⓘ} \quad \alpha = \sqrt{\kappa} \qquad \text{ⓘⓘ} \quad \alpha = \frac{1}{\sqrt{\kappa}}.$$

In [LY17], a Möbius transformation is proposed that produces a filtering function tailored at the eigengaps at the interval boundaries by requiring

$$T\left(\lambda_{\min}^{-}\right) = -1 \qquad T\left(\lambda_{\min}^{+}\right) = 1 \qquad T\left(\lambda_{-}^{\max}\right) = \kappa \qquad T\left(\lambda_{+}^{\max}\right) = -\kappa$$

for the known eigengaps

$$\left[\lambda_{\min}^{-}, \lambda_{\min}^{+}\right] \quad \text{and} \quad \left[\lambda_{-}^{\max}, \lambda_{+}^{\max}\right]$$

around $\lambda_{\min}$ and $\lambda^{\max}$, respectively. In particular, the filter will not necessarily be symmetrical anymore. Here, κ is to be interpreted as a new variable to be solved for from the system of equations. Hence, the modulus for the elliptic functions is defined by the Möbius transformation. With $\lambda_{\min}^{-} = -\frac{1}{G}$, $\lambda_{\min}^{+} = -G$, $\lambda_{-}^{\max} = G$, and $\lambda_{+}^{\max} = \frac{1}{G}$ we arrive at the Möbius transformation described before.

Since we generally assume that information such as eigengaps in the vicinity of interval boundaries is not readily available, we stick to the transformation introduced first, extended by a shift and a scaling factor to include the mapping of the spectrum of the matrix into the filtering interval of the Zolotarev filter.

We additionally require the approximation of the window function to equioscillate about one in the passband and about zero in the stopband, which can be handled by simply shifting and scaling the sign function accordingly. We obtain the sifted sign function as

$$\widetilde{s}_\eta(z) = \frac{s_\eta(z) + 1}{2}.$$

Then the mapping $T(-1) = 0$ accounts for having the final filter function pass $1/2$ at -1 and 1.[3] The deviation in pass- and stopband accordingly becomes

$$\frac{1-\zeta}{2+2\zeta}.$$

Finally, note that for Hermitian problems targeting extremal eigenpairs, the approximated (and shifted) sign function $\widetilde{s}_\eta(z)$ itself is a good candidate for a filtering function that exhibits significantly better separation properties than an appropriately shifted window function. It does, however, require the spectrum of a matrix pencil to be mapped into the region of approximation given by κ, similarly to the case of polynomials in Section 2.1.5. The properties do not translate well to complex arguments, as is the case with the approximated window function, and usage for non-Hermitian cases is not recommended.

[3] $s_\eta(z) \longrightarrow 0$ for $z \longrightarrow \pm\infty$.

2.3.3 Partial fraction form

While it is possible to elevate the form given above to a matrix function by inserting $X^{-1}X = I$ in between any two linear factors in numerator and denominator, e.g.,

$$\begin{aligned}
s_\eta\Big(B^{-1}A\Big) &= X \operatorname{diag}(s_\eta(\lambda_1), \ldots, s_\eta(\lambda_k))X^{-1} \\
&= DX\Lambda X^{-1} \prod_{j=1}^{\eta-1} X\Big(c'_{2j}I + \Lambda^2\Big)X^{-1} \prod_{j=1}^{\eta} X\Big(c'_{2j-1}I + \Lambda^2\Big)^{-1} X^{-1} \\
&= DX\Lambda X^{-1} \prod_{j=1}^{\eta-1} X\Big(\mathbf{i}\sqrt{c'_{2j}}I + \Lambda\Big)X^{-1}X\Big(-\mathbf{i}\sqrt{c'_{2j}}I + \Lambda\Big)X^{-1} \\
&\qquad \prod_{j=1}^{\eta} X\Big(\mathbf{i}\sqrt{c'_{2j-1}}I + \Lambda\Big)^{-1}X^{-1}X\Big(-\mathbf{i}\sqrt{c'_{2j-1}}I + \Lambda\Big)^{-1}X^{-1} \\
&= DX\Lambda X^{-1} \prod_{j=1}^{\eta-1} \Big(\mathbf{i}\sqrt{c'_{2j}}I + X\Lambda X^{-1}\Big)\Big(-\mathbf{i}\sqrt{c'_{2j}}I + X\Lambda X^{-1}\Big) \\
&\qquad \prod_{j=1}^{\eta} \Big(\mathbf{i}\sqrt{c'_{2j-1}}I + X\Lambda X^{-1}\Big)^{-1}\Big(-\mathbf{i}\sqrt{c'_{2j-1}}I + X\Lambda X^{-1}\Big)^{-1} \\
&= DB^{-1}A \prod_{j=1}^{\eta-1} \Big(\mathbf{i}\sqrt{c'_{2j}}I + B^{-1}A\Big)\Big(-\mathbf{i}\sqrt{c'_{2j}}I + B^{-1}A\Big) \\
&\qquad \prod_{j=1}^{\eta} \Big(\mathbf{i}\sqrt{c'_{2j-1}}I + B^{-1}A\Big)^{-1}\Big(-\mathbf{i}\sqrt{c'_{2j-1}}I + B^{-1}A\Big)^{-1} \\
&= DB^{-1}A \prod_{j=1}^{\eta-1} B^{-1}\Big(\mathbf{i}\sqrt{c'_{2j}}B + A\Big)B^{-1}\Big(-\mathbf{i}\sqrt{c'_{2j}}B + A\Big) \\
&\qquad \prod_{j=1}^{\eta} \Big(\mathbf{i}\sqrt{c'_{2j-1}}B + A\Big)^{-1}B\Big(-\mathbf{i}\sqrt{c'_{2j-1}}B + A\Big)^{-1}B \\
&= D\Big(\mathbf{i}\sqrt{c'_{2\eta-1}}B + A\Big)^{-1}B\Big(-\mathbf{i}\sqrt{c'_{2\eta-1}}B + A\Big)^{-1}A \\
&\quad \prod_{j=1}^{\eta-1} \Big[\Big(\mathbf{i}\sqrt{c'_{2j-1}}B + A\Big)^{-1}\Big(-\mathbf{i}\sqrt{c'_{2j}}B + A\Big) \\
&\qquad \Big(-\mathbf{i}\sqrt{c'_{2j-1}}B + A\Big)^{-1}\Big(\mathbf{i}\sqrt{c'_{2j}}B + A\Big)\Big],
\end{aligned}$$

where the last step is possible by reordering the linear factors of the scalar form of s_η, it is not the most convenient form for numerical computation, even if the inversion of B can be eliminated. Another aspect of this formulation is its infeasibility for parallel computation (other than parallelization over right-hand sides or parallelization of basic arithmetic routines) in a possible implementation, since the application to a set of right-hand side vectors implies sequentiality if no matrix-matrix products can be computed, which typically is the case. However, being a rational function, it is possible to find a partial fraction form of the approximated sign function as well as the approximated window function. Neglecting any scaling, a rational function may be represented in terms of their poles $\times_i$ and roots $\bigcirc_i$,

$$\frac{p(z)}{q(z)} = \frac{\prod_{i=1}^{\deg(p)}(z - \bigcirc_i)}{\prod_{i=1}^{\deg(q)}(z - \times_i)}.$$

If $\deg(p) < \deg(q) = d$ and all roots of q are simple, a partial fraction representation is easily obtained as [Olv+10]

$$\frac{p(z)}{q(z)} = \sum_{i=1}^{d} \frac{a_i}{z - \times_i}$$

where the coefficients a_i are given by[4]

$$a_i = (z - \times_i)\frac{p(z)}{q(z)}\bigg|_{z\,=\,\times_i} = \prod_{k=1}^{\deg(p)} (\times_i - \bigcirc_k) \prod_{\substack{k=1 \\ k\neq i}}^{\deg(q)} \left(\frac{1}{\times_i - \times_k}\right). \tag{2.7}$$

Should $\deg(p) = \deg(q)$, a step of polynomial long division is required to first bring the function to the form

$$\frac{p(z)}{q(z)} = 1 + \frac{o(z)}{q(z)}$$

and then find a partial fraction representation of $o(z)/q(z)$. In [LY17], an explicit formulation for a partial fraction decomposition with equal numbers of roots and poles,

$$\prod_{j=1}^{d} \frac{z + \bigcirc_j}{z + \times_j} = 1 + \sum_{j=1}^{d} \frac{b_j^{(d)}}{z + \times_j} \quad \text{with} \quad b_j^{(d)} = (\bigcirc_j - \times_j) \prod_{\substack{k=1 \\ k\neq j}}^{d} \frac{\bigcirc_k - \times_j}{\times_k - \times_j}, \tag{2.8}$$

is given in a form varied to match the Zolotarev problem. Since a proof is only sketched very roughly, we will derive it here in detail by induction over d. For $d = 1$ the above holds trivially via polynomial long division,

$$\begin{aligned} & b_1^{(1)} = \bigcirc_1 - \times_1 \\ \iff & z + \bigcirc_1 = z + \times_1 + b_1^{(1)} \\ \iff & \frac{z + \bigcirc_1}{z + \times_1} = 1 + \frac{b_1^{(1)}}{z + \times_1}. \end{aligned}$$

Now assume Equation (2.8) holds for $d - 1$. Then

$$\frac{z + \bigcirc_d}{z + \times_d} \prod_{j=1}^{d-1} \frac{z + \bigcirc_j}{z + \times_j} \overset{\text{i.h.}}{=} \frac{z + \bigcirc_d}{z + \times_d}\left(1 + \sum_{j=1}^{d-1} \frac{b_j^{(d-1)}}{z + \times_j}\right) = \frac{z + \bigcirc_d}{z + \times_d} + \sum_{j=1}^{d-1} \frac{z + \bigcirc_d}{z + \times_d}\,\frac{b_j^{(d-1)}}{z + \times_j}. \tag{2.9}$$

[4] See also Theorem 2.2 in Section 2.5.1.

Each term in the summation above is a rational function with fewer roots than poles after appending the additional root/pole pair. Then their sum is a rational function with fewer roots than poles and we can find a partial fraction decomposition of the summation term element-wise for the terms 1 to $d-1$ (all other terms of the summation vanish by multiplication with the linear factor of a root that does not cancel with the respective pole) with coefficients

$$(z+\times_j)\frac{z+\bigcirc_d}{z+\times_d}\,\frac{b_j^{(d-1)}}{z+\times_j}\bigg|_{z\,=\,-\times_j} = \frac{\bigcirc_d-\times_j}{\times_d-\times_j}\,b_j^{(d-1)} = b_j^{(d)} \quad \text{for} \quad j=1,\dots,d-1$$

and for the d-th term with the coefficient (here the summation terms do not vanish)

$$\sum_{j=1}^{d-1}(z+\times_d)\frac{z+\bigcirc_d}{z+\times_d}\,\frac{b_j^{(d-1)}}{z+\times_j}\bigg|_{z\,=\,-\times_d} = \sum_{j=1}^{d-1}\frac{\bigcirc_d-\times_d}{\times_j-\times_d}\,b_j^{(d-1)}.$$

The additional term in Equation (2.9) is reformulated using polynomial long division as

$$\frac{z+\bigcirc_d}{z+\times_d} = 1+\frac{\bigcirc_d-\times_d}{z+\times_d}$$

and its rational part is added to the d-th term and its coefficient,

$$\begin{aligned}(\bigcirc_d-\times_d)+\sum_{j=1}^{d-1}\frac{\bigcirc_d-\times_d}{p_j-\times_d}\,b_j^{(d-1)} &= (\bigcirc_d-\times_d)\left(1+\sum_{j=1}^{d-1}\frac{b_j^{(d-1)}}{\times_j-\times_d}\right)\\ &\overset{\text{i.h.}}{=} (\bigcirc_d-\times_d)\left(\prod_{j=1}^{d-1}\frac{\bigcirc_j-\times_d}{\times_j-\times_d}\right) = b_d^{(d)},\end{aligned}$$

where we let $z=-\times_d$ for the application of the induction hypothesis in the last step. With this, the proof is complete. □

Following [LY17] further, application of the above to Zolotarev's approximation gives

$$\begin{aligned}z\frac{\prod_{j=1}^{\eta-1}c'_{2j}+z^2}{\prod_{j=1}^{\eta}c'_{2j-1}+z^2} &= \frac{z}{c'_{2\eta-1}+z^2}\left(1+\sum_{j=1}^{\eta-1}\frac{b_j}{c'_{2j-1}+z^2}\right)\\ \text{with}\quad b_j &= \left(c'_{2j}-c'_{2j-1}\right)\prod_{\substack{k=1\\k\neq j}}^{\eta-1}\frac{c'_{2\kappa}-c'_{2j-1}}{c'_{2\kappa-1}-c'_{2j-1}}.\end{aligned}$$

Factoring in the additional pole yields a term that is again a rational function of fewer roots than poles, and we find a partial fraction representation

$$\frac{1}{c'_{2\eta-1}+z^2}+\frac{1}{c'_{2\eta-1}+z^2}\sum_{j=1}^{\eta-1}\frac{b_j}{c'_{2j-1}+z^2} = \sum_{j=1}^{\eta}\frac{a_j}{c'_{2j-1}+z^2}$$

with (most terms again vanish by multiplication with the linear factor of a root that does not cancel with the respective pole)

$$a_j = \left(c'_{2j-1} + z^2\right) \frac{1}{c'_{2\eta-1} + z^2} \frac{b_j}{c'_{2j-1} + z^2}\Bigg|_{z^2 = -c'_{2j-1}} = \frac{b_j}{c'_{2\eta-1} - c'_{2j-1}}$$

for $j = 1, \ldots, \eta - 1$ and (terms do not vanish for the η-th coefficient and the separate term is included)

$$a_\eta = \frac{c'_{2\eta-1} + z^2}{c'_{2\eta-1} + z^2} + \frac{c'_{2\eta-1} + z^2}{c'_{2\eta-1} + z^2} \sum_{j=1}^{\eta-1} \frac{b_j}{c'_{2j-1} + z^2}\Bigg|_{z^2 = -c'_{2\eta-1}} = 1 + \sum_{j=1}^{\eta-1} \frac{b_j}{c'_{2j-1} - c'_{2\eta-1}}.$$

Resuming the transformation of Zolotarev's approximation,

$$\begin{aligned}
z\frac{\prod_{j=1}^{\eta-1} c'_{2j} + z^2}{\prod_{j=1}^{\eta} c'_{2j-1} + z^2} &= z\sum_{j=1}^{\eta} \frac{a_j}{c'_{2j-1} + z^2} = \frac{1}{2}\sum_{j=1}^{\eta} \frac{2za_j}{c'_{2j-1} + z^2} \\
&= \frac{1}{2}\sum_{j=1}^{\eta} \frac{a_j\left(z - \mathbf{i}\sqrt{c'_{2j-1}}\right) + a_j\left(z + \mathbf{i}\sqrt{c'_{2j-1}}\right)}{\left(z + \mathbf{i}\sqrt{c'_{2j-1}}\right)\left(z - \mathbf{i}\sqrt{c'_{2j-1}}\right)} \\
&= \frac{1}{2}\sum_{j=1}^{\eta} \left(\frac{a_j}{\mathbf{i}\sqrt{c'_{2j-1}} + z} + \frac{a_j}{-\mathbf{i}\sqrt{c'_{2j-1}} + z} \right).
\end{aligned}$$

Substituting a Möbius transformation $T(z)$ finally gives

$$\begin{aligned}
&\frac{1}{2}\sum_{j=1}^{\eta} \left(\frac{a_j}{\mathbf{i}\sqrt{c'_{2j-1}} + T(z)} + \frac{a_j}{-\mathbf{i}\sqrt{c'_{2j-1}} + T(z)} \right) \\
=&\frac{1}{2}\sum_{j=1}^{\eta} \left(\frac{a_j(\gamma + \delta z)}{\mathbf{i}\sqrt{c'_{2j-1}}(\gamma + \delta z) + \alpha + \beta z} + \frac{a_j(\gamma + \delta z)}{-\mathbf{i}\sqrt{c'_{2j-1}}(\gamma + \delta z) + \alpha + \beta z} \right) \\
=&\frac{1}{2}\sum_{j=1}^{\eta} (\gamma + \delta z) \left(\frac{a_j\left(\beta + \mathbf{i}\delta\sqrt{c'_{2j-1}}\right)^{-1}}{\frac{\alpha + \mathbf{i}\gamma\sqrt{c'_{2j-1}}}{\beta + \mathbf{i}\delta\sqrt{c'_{2j-1}}} + z} + \frac{a_j\left(\beta - \mathbf{i}\delta\sqrt{c'_{2j-1}}\right)^{-1}}{\frac{\alpha - \mathbf{i}\gamma\sqrt{c'_{2j-1}}}{\beta - \mathbf{i}\delta\sqrt{c'_{2j-1}}} + z} \right).
\end{aligned}$$

We call the transformed poles

$$\times_j = \frac{\alpha + \mathbf{i}\gamma\sqrt{c'_{2j-1}}}{\beta + \mathbf{i}\delta\sqrt{c'_{2j-1}}}$$

and the poles of the conjugated terms $\overline{\times_j}$. Note that, due to $\alpha = \beta$ and $\gamma = -\delta$ for the Möbius transformation from above, $|\times_j| = 1$, i.e., all poles are located on the unit circle. Finding again a partial fraction form using Equation (2.8) (note that the coefficients are computed the same way as in Equation (2.7) by eliminating a pole) yields

$$w_j = \frac{1}{2}(\gamma + \delta z)\frac{a_j}{\beta + \mathbf{i}\delta\sqrt{c'_{2j-1}}}\Bigg|_{z\,=\,-\times_j} = \frac{a_j(\gamma - \delta \times_j)}{2\left(\beta + \mathbf{i}\delta\sqrt{c'_{2j-1}}\right)}$$

as well as $\overline{w_j}$ as the coefficients of the conjugated terms. The Möbius transformation has introduced a scaling while Equation (2.8) does not consider any scaling factor. Multiplying both sides in Equation (2.8) with a constant scaling factor $\mathcal{D}$ shows that using the method above in contrast to the method shown in Equation (2.8) for computing the coefficients actually computes coefficients that are scaled by $\mathcal{D}$ as well. Since $\mathcal{D}$ also applies to the constant term that, in Equation (2.8), is simply unity, the scaling factor is necessary to formulate the correct partial fraction form. The factor $\mathcal{D}$ is nothing more than the quotient of the highest power of z in the numerator and denominator of the rational function, and we may make the following simple deduction. For the simple partial fraction form

$$\sum_{j=1}^{d} \frac{a_i z + b_i}{\times_i + z} = \sum_{j=1}^{d} \frac{a_i z \prod_{\substack{k=1\\k\neq j}}^{d}(\times_i + z) + b_i \prod_{\substack{k=1\\k\neq j}}^{d}(\times_i + z)}{\prod_{k=1}^{d}(\times_i + z)}$$

that resembles our case from above, considering only terms of the highest power yields

$$\mathcal{D} = \sum_{j=1}^{d} a_i.$$

Applied to the transformed Zolotarev approximation, this now gives

$$\begin{aligned}\mathcal{D} &= \sum_{j=1}^{\eta}\left(\frac{a_i\delta}{2\left(\beta + \mathbf{i}\delta\sqrt{c'_{2j-1}}\right)} + \frac{a_i\delta}{2\left(\beta - \mathbf{i}\delta\sqrt{c'_{2j-1}}\right)}\right)\\ &= \sum_{j=1}^{\eta} \frac{a_i\delta\left(\beta + \mathbf{i}\delta\sqrt{c'_{2j-1}} + \beta - \mathbf{i}\delta\sqrt{c'_{2j-1}}\right)}{2\left(\beta^2 + \delta^2 c'_{2j-1}\right)} = \sum_{j=1}^{\eta} \frac{a_i\beta\delta}{\beta^2 + \delta^2 c'_{2j-1}}\end{aligned}$$

and we can write the final composition as

$$\widetilde{s}_\eta(T(z)) = \frac{s_\eta(T(z)) + 1}{2} = \frac{1}{2} + \frac{\mathcal{D}}{2}\sum_{j=1}^{\eta}\frac{a_i\beta\delta}{\beta^2 + \delta^2 c'_{2j-1}} + \frac{\mathcal{D}}{2}\sum_{j=1}^{\eta}\left(\frac{w_j}{z + \times_j} + \frac{\overline{w_j}}{z + \overline{\times_j}}\right),$$

which is a rational function of degree $d = 2\eta$. A rational function

$$\rho_d(z) = \tau + \sum_{j=1}^{d} \frac{w_j}{z + \times_j}$$

of this form is easily written as matrix function and applied to the real axis,

$$\begin{aligned}\rho_d\Big(B^{-1}A\Big) &= X\operatorname{diag}(\rho_d(\lambda_1),\dots,\rho_d(\lambda_k))X^{-1} = X\left(\tau I + \sum_{j=1}^{d} w_j(\Lambda + \times_j I)^{-1}\right)X^{-1}\\ &= \tau I + \sum_{j=1}^{d} w_j X(\Lambda + \times_j I)^{-1}X^{-1} = \tau I + \sum_{j=1}^{d} w_j\Big(X\Lambda X^{-1} + \times_j I\Big)^{-1}\\ &= \tau I + \sum_{j=1}^{d} w_j\Big(B^{-1}A + \times_j I\Big)^{-1} = \tau I + \sum_{j=1}^{d} w_j(A + \times_j B)^{-1}B.\end{aligned}$$

To place the filtering region arbitrarily in the complex plane, a shift c and a scaling factor r are required. For definite pairs with target interval $[\lambda_{\min}, \lambda^{\max}]$, it is again $c = (\lambda^{\max} + \lambda_{\min})/2$ and $r = (\lambda^{\max} - \lambda_{\min})/2$, see Section 2.2.3. These parameters can be integrated into a given Möbius transformation as

$$T(z) = \frac{r\alpha - c\beta + \beta z}{r\gamma - c\delta + \delta z}.$$

Since poles come in complex conjugated pairs, the optimizations mentioned in Section 2.2.3, including the possibility to halve the number of linear system solves in case of a real matrix pencil, are applicable.

Figure 2.8 shows an example of a Zolotarev filter. Only ten of the overall twelve roots are shown. Pass- and stopband oscillations are imperceptible in the non-logarithmic plot already at low degrees. For non-Hermitian problems, the filtering region would be defined as the unit circle, but the spacing of the poles in the imaginary directions causes the separation ability of the filter to diminish heavily. The effect intensifies when the poles move towards -1 or 1 by decreasing κ. Overall, the filter is not well suited for use in the solution of non-Hermitian problems.

2.3.4 Computation of Jacobi's elliptic functions

While conventional trigonometric functions are generally available for every programming language directly via the accompanying runtime libraries, the evaluation of Jacobi's elliptic functions may require the use of additional third-party libraries. A non-exhaustive list of libraries that include implementations of Jacobi's elliptic functions includes the GNU Scientific Library [Gal+09; GNU], the Math Toolkit of the Boost C++ libraries [Agr+], the DSP System Toolbox™ for MATLAB® and Simulink® [MLB] (formerly also via the High-Order Digital Parametric Equalizer Design Toolbox [Orf05]), and the SymPy Python library's Mpmath module [Meu+17], to name just a few.

Algorithms for the computation of these functions are relatively simple. Therefore, we are able to include them in Section A.1 of the appendix, in case a suitable library is not available, is missing certain functions or does not behave as expected.

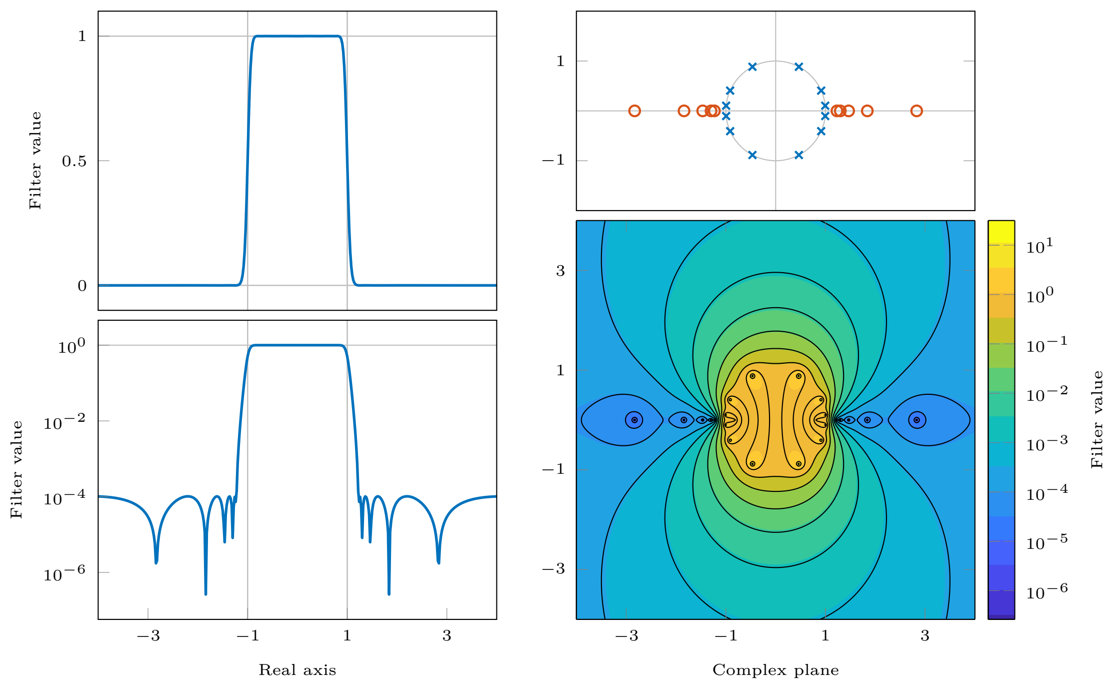


Figure 2.8: *Example of a Zolotarev filter of degree* 12 *with* $\kappa = 10^{-3}$. *Left: non-logarithmic and logarithmic plots of the filter function. Top right: pole-root map. Roots are marked as circles, poles as crosses. Bottom right: logarithmic contour plot of the filter function in the complex plane.*

2.4 Electronic filter design

The symbol $H(s)$ is commonly used for a filter's transfer function in the complex frequency domain of the Laplace transform. Applied to only the (purely imaginary) frequency part ω, the graph of $|H(\mathbf{i}\omega)|^2$ constitutes the filter's frequency response, the component we are interested in. Similar to our definition of the window function, an ideal low-pass filter would be given as

$$|H(\mathbf{i}\omega)|^2 = \begin{cases} 1 & \text{if } \omega \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

This formulation can intuitively be approximated in slightly rewritten form as

$$|H(\mathbf{i}\omega)|^2 = \frac{1}{1 + \varepsilon^2 g^2(\omega)},$$

where $g(\omega) \leq \zeta$ if $\omega \leq 1$ and $g(\omega) \longrightarrow \infty$ for $\omega \longrightarrow \infty$ or at least $g(\omega) \geq \xi$ for $\omega \geq 1 + \rho$. The function g is called the *characteristic function* of the filter and the factor ε, sometimes called *ripple factor*, is included for normalization and scaling. The characteristic function g is squared to keep its value positive and the gain of the filter smaller or equal to one. Squaring the characteristic function also keeps the transfer function symmetrical and the overall degree even, ensuring that no pole will be purely real as to not create an essential discontinuity in the filtering graph in the case of some filter types.

The quicker $g^2(\omega)$ tends towards infinity or the larger ξ is for frequencies greater than $1+\rho$ and the smaller the ζ, the better an approximation we obtain. Since the behavior of g for $|\omega| < 1$ translates to the behavior of the filter in the passband, the parameter ε can be used to scale potential oscillations within the passband. To keep the effects of the parameter transparent, it is reasonable to require $g(1) = 1$.

This approach is fundamentally different from the ones described before as it uses the property of a function g^2 to be bounded from above inside $[0,1]$ and either quickly tend towards infinity or be bounded from below for $\omega \geq 1+\rho$ in contrast to finding an approximate representation for the ideal low-pass filter by combining certain basis functions or by contour integration. The algorithmic result is, however, very similar.

The typical design concept is visualized in Figure 2.9. It specifies four quantities

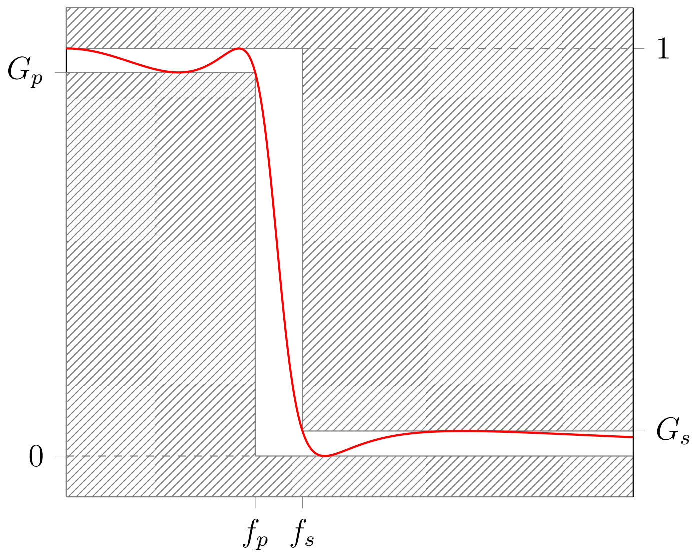


Figure 2.9: *Filter design parameters: cutoff frequency f_p, end of transition band f_s, minimum passband gain G_p, and maximum stopband gain G_s.*

as constraints for the filter gain in pass- and stopband and the width and position of the transition band, as outlined in Table 2.2. Given these four parameters, other parameters—in particular the degree $2d$—can be computed easily due to the simple

Symbol	Description
G_p	Minimum allowed gain in the passband of the filter.
G_s	Maximum allowed gain in the stopband of the filter.
f_p	*Cutoff frequency*, end of the passband and start of the transition band.
f_s	End of the transition band and start of the stopband.

Table 2.2: *Filter design parameters.*

form of the filtering function. Note that we use $2d$ as degree here for reasons of recognition, somewhat similar to $d = 2\nu$ for the Zolotarev filter or $d = 2k$ for Cauchy filters. Since the half-degree d is required to be an integer, based on the computed result δ, the next larger integer $d = \lceil \delta \rceil$ is chosen. Conventionally, the filter is scaled to not exceed one but given G_p, the scaling might be chosen such that the gain in the passband section is contained in $[1 - G_p/2, 1 + G_p/2]$ instead to minimize the deviation from one in the passband. Of course, G_s has to be scaled accordingly in this case. Similarly, the filter may be shifted by $-G_s/2$ such that the absolute value of the oscillations in the stopband are reduced to one half of their original amplitude. In combination with appropriate scaling, oscillations in pass- and stopband can be minimized. This has, however, only minuscule effects on filter quality.

The filters most frequently covered in the literature are the *Butterworth* filter, the *Chebyshev* filter (type I and II), and the *Elliptic* filter (also *Cauer* filter or *Zolotarev* filter), see, e.g., [Dan74; Ant79], whom we will follow here. Other filters exist, but the general procedure does not differ from what will be described below.

The filtering functions derived for use in electronics act on the complex frequency $\mathbf{i}\omega$ and thus on the imaginary axis. Swapping real and imaginary parts of the poles (and roots), transforms the filter to act on the real axis. Furthermore, only half of the poles are used due to stability concerns, a constraint that bears no importance to us. Using only the poles in the left half of the complex plane (potential roots are purely imaginary) for the transfer function effectively describes

$$G(\omega) = |H(\mathbf{i}\omega)| = \frac{1}{\sqrt{1 + \varepsilon^2 g^2(\omega)}}$$

and offloads part of the frequency response function into an imaginary part, such that only its complex absolute value constitutes the desired filtering function (with diminished filtering properties). A complex-valued filter becomes problematic when elevated to a matrix function since the complex matrix square root to compute the absolute value is not easily obtainable. A real-valued filter can trivially be obtained by using the full set of poles (and possibly duplicated roots) and may be used instead, resolving this issue.

In order for the filter to be applicable in the context of a spectral filtering framework, its numerical application has to be reduced to the available matrix and vector operations. To achieve this—given that all filters described herein are rational functions—the filtering functions have to be decomposed into partial fraction form. The reader is referred to Section 2.3.3 for information on how to achieve this. The representation of the rational function in terms of its poles and roots is correct except for scaling. Since the characteristic function is scaled to pass one at the cutoff frequency, the gain of the filter at the cutoff frequency is $1/(1 + \varepsilon^2)$. With this, the scaling factor can be computed as

$$\beta = \frac{1}{1 + \varepsilon^2} \prod_{j=1}^{\eta_r} \left(\frac{1}{1 - \bigcirc_j} \right) \prod_{j=1}^{\eta_p} (1 - \times_j),$$

where $\bigcirc_j$ are the η_r roots and $\times_j$ are the $\eta_p = 2d$ poles of the rational function. Alternatively, for some filters $f(0) = 1$ either for all or odd half-degrees and $f(0) = 1/(1+\varepsilon^2)$ for even half-degrees.

In terms of a matrix function for the matrix pencil (A, B), the rational function for the filter of degree $2d$, applied to vectors Y, takes the form

$$\alpha\beta Y + \beta \sum_{j=1}^{2d} w_j (A - \times_j B)^{-1} BY,$$

where α is either 1, if the number of poles and roots is identical, or 0, if the number of roots is smaller than the number of poles. The case of $\eta_r > \eta_p$ shall not be considered here.

Since poles come in complex conjugated pairs, all optimizations mentioned in Section 2.2.3, in particular the possibility to halve the number of linear system solves in case of a real matrix pencil, are applicable.

Finally, clearly specifying a filtering region for a filter in the complex plane is not easily done in the non-Hermitian case. An elliptic contour defined by the poles of the filter is a possible candidate, if applicable. For the Hermitian case, however, the question arises where to map the interval boundaries to, given only a transition interval. Information on eigengaps is typically not available, so possible choices boil down to the center of the transition band or the point where the filtering function passes 1/2, the *half-point* $f_{1/2}$. The latter can easily be computed by requiring $\varepsilon^2 g^2(\omega) = 1$ or $g(\omega) = \pm 1/\varepsilon$. For arbitrary placement of the filter in the complex plane, a shift c and a scaling factor r are required, such that the desired filtering region is mapped onto the respective region of the filter. Similar to before, if in the Hermitian case the target interval $[\lambda_{\min}, \lambda^{\max}]$ is to be mapped into an interval $[\omega_{\min}, \omega^{\max}]$,

$$r = \frac{\lambda^{\max} - \lambda_{\min}}{\omega^{\max} - \omega_{\min}} \quad \text{and} \quad c = \lambda_{\min} - r\omega_{\min}$$

and the transformation of the filter argument is

$$z \to t(z) = \frac{1}{r}(z - c).$$

For a partial fraction representation as depicted above, this is equivalent to a modification of the poles $\times_i$ and coefficients w_i,

$$\frac{w_j}{t(z) - \times_j} = \frac{w_j}{\frac{1}{r}(z-c) - \times_j} = \frac{rw_j}{z - (c + r\times_j)}.$$

The positions f_p, f_s, and $f_{1/2}$ can be transformed to relate to the target interval accordingly by inverting the transformation above or vice versa.

In the following, the common filter types will briefly be described in the form mentioned above to yield real-valued filtering functions acting on the real axis. All filters have in common that the desired cutoff frequency f_p is introduced by transformation of the argument,[5] $\omega = z/f_p$.

Chebyshev type-II filters are a special case among the filters introduced below. The necessary adaptions to the general statements made above will be outlined in the corresponding section.

2.4.1 Butterworth Filter

Given the conditions for a suitable function g from above, simple power functions of the form $g = \omega^d$ first come to mind. They also satisfy $g(1) = 1$. Poles of the resulting transfer function occur whenever the denominator becomes zero,

$$\begin{aligned} 1 + \varepsilon^2\omega^{2d} = 0 &\iff \omega = \left(-\frac{1}{\varepsilon^2}\right)^{\frac{1}{2d}} \\ &\iff \times_j = \left(\frac{1}{\varepsilon^2}e^{\mathbf{i}(2j-1)\pi}\right)^{\frac{1}{2d}} = \varepsilon^{-\frac{1}{d}}e^{\mathbf{i}\frac{(2j-1)}{2d}\pi} \quad \text{for } j = 1, \dots, 2d, \end{aligned}$$

since the poles repeat for $j > 2d$. We see that variation of ε is nothing more than a shift of the cutoff frequency. Due to symmetry, it is possible to only compute poles of one quadrant and derive the remaining poles by reflecting them about the axes. Pay special attention to the purely imaginary poles in the case of odd half-degrees that appear for $j = (d+1)/2$ and $j = (3d+1)/2$ in this case. Obviously, the poles come in complex conjugate pairs on the unit circle (for $f_p = 1$ and $\varepsilon = 1$). The Butterworth filter is a pole-only filter without roots. The scaling factor can be computed from the poles as the inverse of the product of the poles, which is the rational form evaluated at position zero and here has to evaluate to one. Therefore

$$\beta = \prod_{j=1}^{2d} \omega_j = \prod_{j=1}^{d} \varepsilon^{-\frac{2}{d}} = \frac{1}{\varepsilon^2}.$$

In the literature it is often $\varepsilon = 1$ such that the filter function passes 1/2 at the cutoff frequency. Otherwise, the filter passes 1/2 at

$$z = \pm f_p\,\varepsilon^{-\frac{1}{d}}.$$

Figure 2.10 shows an example of a Butterworth filter. The even distribution of poles results in flanks of similar steepness around a circular region in the complex plane which makes it useful also for the non-Hermitian case. A Butterworth filter with $\varepsilon = 1$ is equivalent to the Cauchy filter using the midpoint rule on a unit-circular contour. This is apparent from the position of poles alone, but as is pointed out in [Krä14], the midpoint quadrature rule with $2d$ nodes gives the selection function

$$f(\lambda) = \frac{1}{1 + \lambda^{2d}},$$

which is nothing more than the definition of the Butterworth filter with $\varepsilon = 1$.

[5] We may as well just choose $f_p = 1$ universally and let the argument transformation handle scaling altogether.

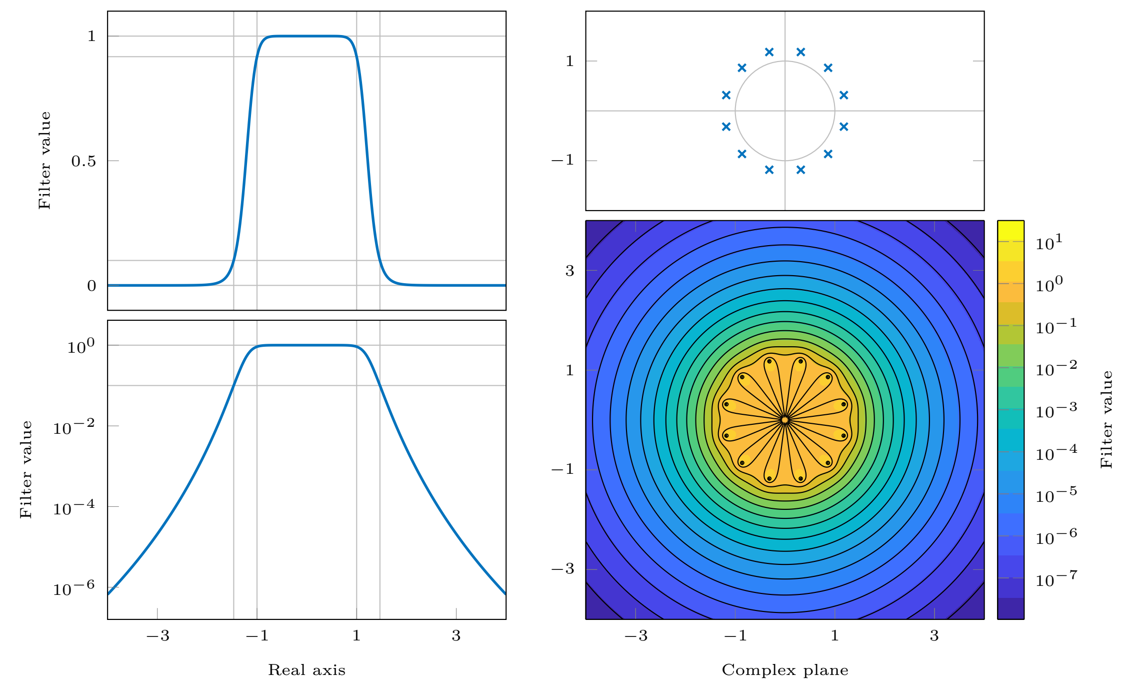


Figure 2.10: *Example of a Butterworth filter of degree* 12 *with* $\varepsilon = 0.3$. *Left: non-logarithmic and logarithmic plots of the filter function. Top right: pole map. Bottom right: logarithmic contour plot of the filter function in the complex plane.*

2.4.1.1 Design

Given the gain G_p at the cutoff frequency f_p and the gain G_s at the end of the transition band f_s, it is not difficult to compute the parameters ε and d. We already know that the characteristic function passes one at f_p and thus

$$G_p = \frac{1}{1+\varepsilon^2} \implies \varepsilon = \sqrt{\frac{1}{G_p} - 1}.$$

Plugging the remaining information into the filter formula gives

$$G_s = \frac{1}{1+\varepsilon^2\left(\frac{f_s}{f_p}\right)^{2\delta}} \implies \delta = \frac{\log\left(\frac{1}{\varepsilon}\sqrt{\frac{1}{G_s}-1}\right)}{\log\left(\frac{f_s}{f_p}\right)} = \frac{\log\left(\frac{1}{\varepsilon^2}\left(\frac{1}{G_s}-1\right)\right)}{2\log\left(\frac{f_s}{f_p}\right)}.$$

Remember that the cutoff frequency is usually implemented by letting $\omega = z/f_p$ and $d = \lceil\delta\rceil$. Since the characteristic function is monotonous, separately in the positive and negative regions, this is enough to conform to the design specifications.

2.4.2 Chebyshev Type-I Filter

Due to the property of Chebyshev polynomials to equioscillate between -1 and 1 inside the interval $[-1, 1]$ and quickly tend towards infinity outside of it, they fulfill the conditions imposed on characteristic functions g, and we thus let $g(\omega) = T_d(\omega)$. Similarly to the roots of Chebyshev polynomials, see Section 2.1.6, we find

$$\begin{aligned} 1 + \varepsilon^2 T_d^2(\omega) = 0 &\iff \cos^2(d\theta) = -\frac{1}{\varepsilon^2} \\ &\iff \cos(d\theta + j\pi) = \pm\frac{\mathbf{i}}{\varepsilon} && \text{for } j = 0, 1, \ldots \\ &\iff \theta_j = \frac{1}{d}\left(\cos^{-1}\left(\pm\frac{\mathbf{i}}{\varepsilon}\right) + j\pi\right) && \text{for } j = 0, 1, \ldots \qquad (2.10) \end{aligned}$$

using $\omega = \cos(\theta)$, $\cos(z + \pi) = -\cos(z)$, and $\cos(z - \pi) = \cos(z + \pi)$. Note that the half-period π of $\cos(z)$ can be used here only due to the choice of sign given by the square root operation in the first line. The choice of sign for $\mathbf{i}/\varepsilon$ is arbitrary since it merely affects the order of the computed poles. The poles repeat for $(j\pi)/d \geq 2\pi$, thus $j < 2d$ and

$$\times_j = \cos(\theta_j) \quad \text{for } j = 0, \ldots, 2d - 1.$$

Due to symmetry, it is sufficient to only compute one quadrant, with special care taken for purely imaginary poles in the case of odd half-degrees which appear for $j = (d-1)/2$ and $j = (3d-1)/2$. The filtering function passes 1/2 at

$$z = \pm f_p \cos\left(\frac{1}{d}\cos^{-1}\left(\frac{1}{\varepsilon}\right)\right).$$

Figure 2.11 shows an example of a type-I Chebyshev filter. The uneven distribution of poles causes flank steepness to decline in increasingly imaginary directions, similarly to the Cauchy filter using Gauss-Legendre integration on an elliptic contour. Usage for non-Hermitian problems is not impossible, but the varying separation properties will reduce convergence speed.

2.4.2.1 Design

As before, the characteristic function passes one at f_p and

$$G_p = \frac{1}{1 + \varepsilon^2} \implies \varepsilon = \sqrt{\frac{1}{G_p} - 1}.$$

The preliminary half-degree δ is again obtained by simply substituting the remaining parameters,

$$\begin{aligned} G_s = \frac{1}{1 + \varepsilon^2 T_\delta^2\left(\frac{f_s}{f_p}\right)} &\implies \cos\left(\delta \cos^{-1}\left(\frac{f_s}{f_p}\right)\right) = \frac{1}{\varepsilon}\sqrt{\frac{1}{G_s} - 1} \\ &\implies \delta = \frac{\cos^{-1}\left(\frac{1}{\varepsilon}\sqrt{\frac{1}{G_s} - 1}\right)}{\cos^{-1}\left(\frac{f_s}{f_p}\right)}. \end{aligned}$$

The characteristic function is equioscillating inside $[-f_p, f_p]$ and monotonous outside. The filter thereby conforms to the specification.

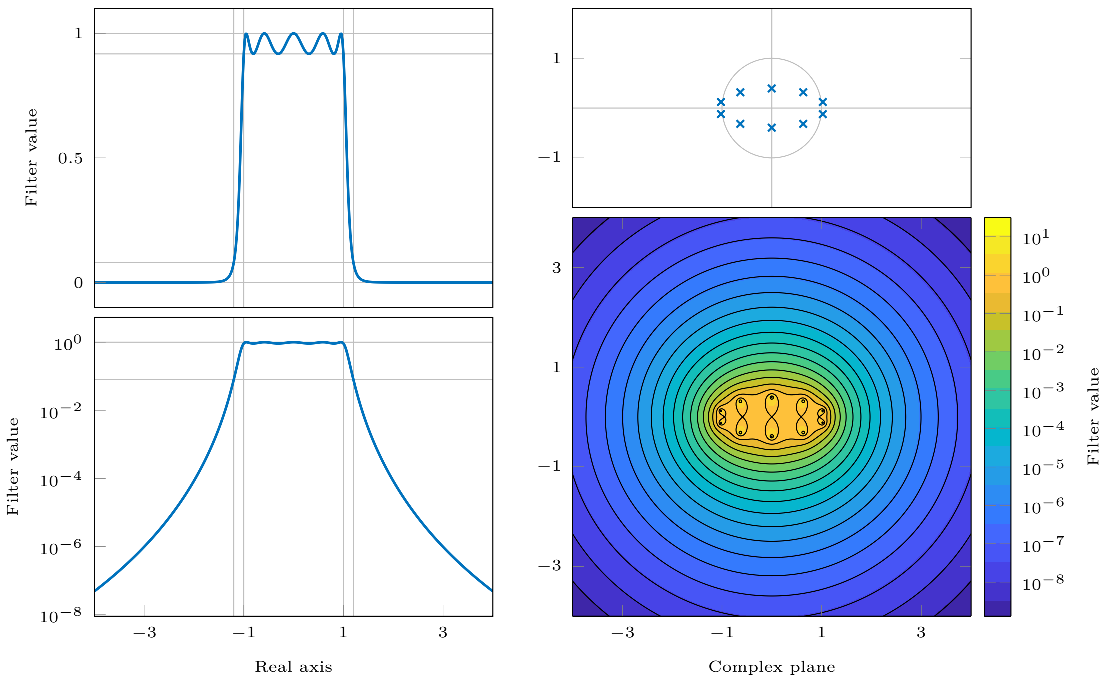


Figure 2.11: *Example of a type-I Chebyshev filter of degree* 10 *with* $\varepsilon = 0.3$. *Left: non-logarithmic and logarithmic plots of the filter function. Top right: pole map. Bottom right: logarithmic contour plot of the filter function in the complex plane.*

2.4.3 Chebyshev Type-II Filter

In contrast to the other filters discussed here, let $\omega = f_s/z$. The inverse Chebyshev filter translates the oscillations in the passband of the Chebyshev filter to the stop-band by choosing the characteristic function as $f(\omega) = 1/T_d^2(\omega)$ with ripple factor $1/\varepsilon^2$. Rewriting the transfer function as

$$\frac{1}{1+\frac{1}{\varepsilon^2 T_d^2(\omega)}} = \frac{\varepsilon^2 T_d^2(\omega)}{1+\varepsilon^2 T_d^2(\omega)}$$

shows that filters of this type have duplicate roots. The poles are obtained as

$$\frac{1}{\times_j} = \cos(\theta_j) \implies \times_j = \frac{1}{\cos(\theta_j)}$$

where θ_j is the same as in Equation (2.10). The roots of the filter are the inverses of the roots of the Chebyshev polynomial of degree d from Section 2.1.6,

$$\frac{1}{\bigcirc_k} = \cos\left(\frac{\pi(2k-1)}{2d}\right) \quad \text{for } j = 1, \ldots, 2d.$$

The roots repeat for $j \geq 2d + 1/2$. For odd half-degrees, there is a pair of purely imaginary poles. The associated roots are formally infinite and need to be skipped in computations. It is therefore advisable to only compute one quadrant, $j = 1, \dots, \lfloor d/2 \rfloor$, and exploit symmetries to obtain the remaining roots. In the case of odd half-degrees, the rational filter function therefore has fewer roots than poles while for even half-degrees the number of roots and poles is equal.

The scaling of the partial fraction form has to be adapted to account for the inversion. Assume

$$f(f_s) = \frac{1}{1 + \frac{1}{\varepsilon^2}} \quad \text{or} \quad f(0) = 1.$$

Then, with $\eta_p = 2d$ being the number of poles and η_r the number of roots,

$$\beta = \frac{1}{1+\frac{1}{\varepsilon^2}} \prod_{j=1}^{\eta_r} \left(\frac{1}{1 - \bigcirc_j} \right) \prod_{j=1}^{\eta_p} (1 - \times_j) = \prod_{j=1}^{\eta_p} \left(\frac{1}{\bigcirc_j} \right) \prod_{j=1}^{\eta_r} (\times_j).$$

The filtering function passes $1/2$ at

$$z = \pm \frac{f_s}{\cos\left(\frac{1}{d} \cos^{-1}\left(\frac{1}{\varepsilon}\right)\right)}.$$

Figure 2.12 shows an example of a type-II Chebyshev filter. All roots are duplicate

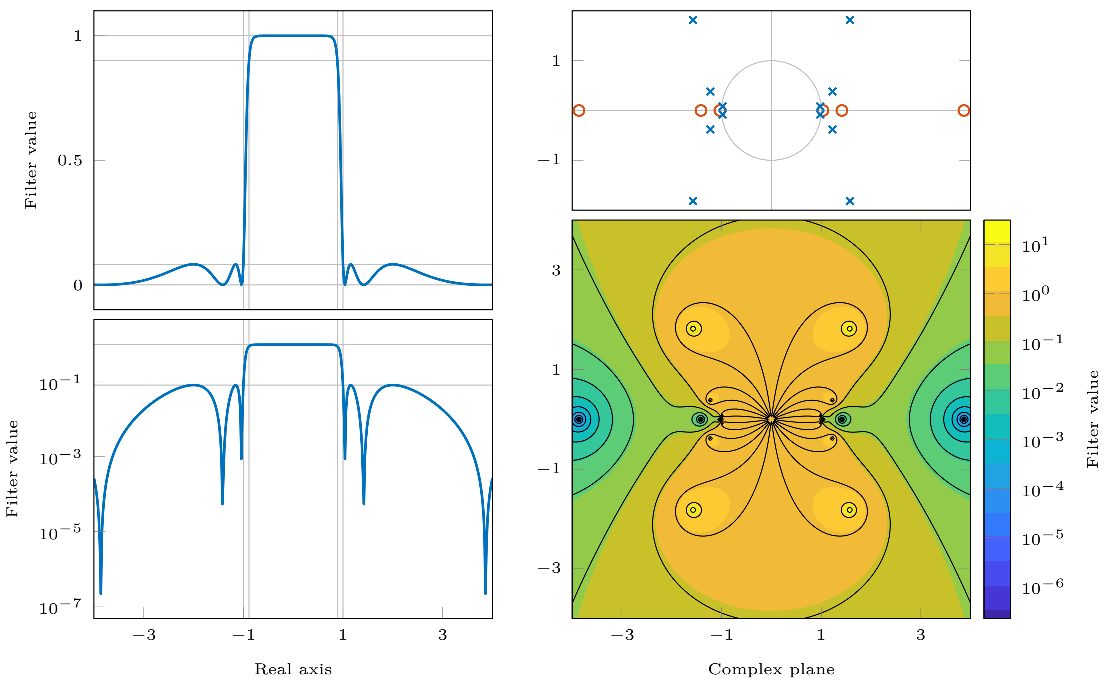


Figure 2.12: *Example of a type-II Chebyshev filter of degree* 12 *with* $\varepsilon = 0.3$. *Left: non-logarithmic and logarithmic plots of the filter function. Top right: pole-root map. Roots are marked as circles, poles as crosses. Bottom right: logarithmic contour plot of the filter function in the complex plane.*

roots. The contour plot reveals that the filter only works well along the real axis and is unusable for non-Hermitian problems.

2.4.3.1 Design

Due to the inversion, the characteristic function passes one at the end of the transition band f_s, and hence ε relates to G_s as

$$G_s = \frac{\varepsilon^2}{1+\varepsilon^2} = \frac{1}{1+\frac{1}{\varepsilon^2}} \implies \varepsilon = \frac{1}{\sqrt{\frac{1}{G_s}-1}}.$$

By substituting the remaining parameters, the preliminary half-degree is obtained as

$$G_p = \frac{1}{1+\frac{1}{\varepsilon^2 T_\delta^2\left(\frac{f_s}{f_p}\right)}} \implies \cos\left(\delta\cos^{-1}\left(\frac{f_s}{f_p}\right)\right) = \frac{1}{\varepsilon\sqrt{\frac{1}{G_p}-1}}$$
$$\implies \delta = \cos^{-1}\left(\frac{1}{\varepsilon\sqrt{\frac{1}{G_p}-1}}\right)\frac{1}{\cos^{-1}\left(\frac{f_s}{f_p}\right)}.$$

The half-degree computed here is, with the definition of ε above, identical to the half-degree computed for the type-I Chebyshev filter. The characteristic function is monotonic, separately for positive and negative values, inside $[-f_s, f_s]$ and larger than or equal to one outside. The filter thereby conforms to the specification.

2.4.4 Elliptic Filter

While an in-depth introduction to the theory of elliptic functions is beyond the scope of this work, fixing just a few elements of the related terminology and notation simplifies the discussion of this filter type. Some of these terms were already introduced in Section 2.3.1.

A Jacobi elliptic function of *modulus* κ is periodic with (at most) periods $4K$ and $4iK'$, therefore called the *quarter periods*, given by the complete elliptic integrals of the first kind of moduli κ and κ', $K = K(\kappa)$ and $K' = K(\kappa')$, with $\kappa' = \sqrt{1-\kappa^2}$.

The *elliptic rational function* is given as [Dan74; Ant79]

$$\mathcal{R}_d(\omega) = \begin{cases} \operatorname{sn}\left(d\frac{K_1}{K}\operatorname{sn}^{-1}(\omega,\kappa),\kappa_1\right) & \text{if } d \text{ odd} \\ \operatorname{sn}\left(d\frac{K_1}{K}\operatorname{sn}^{-1}(\omega,\kappa) + (-1)^{\frac{d}{2}}K_1,\kappa_1\right) & \text{if } d \text{ even} \end{cases}$$

in terms of two moduli κ and κ_1. The sign of the additive term in case of even half-degrees merely changes the sign of the rational function, which is of no further concern in our case. An equivalent formulation is given [LTE01; LT05] via the elliptic function $\operatorname{cd}(\nu,\kappa) = \operatorname{cn}(\nu,\kappa)/\operatorname{dn}(\nu,\kappa)$ as

$$\mathcal{R}_d(\omega) = \operatorname{cd}\left(d\frac{K_1}{K}\operatorname{cd}^{-1}(\omega,\kappa),\kappa_1\right).$$

The structural similarity to the definition of the Chebyshev polynomials is conspicuous. In the design process of elliptic filters, the periods, represented by the two moduli κ and κ_1 are used to define pass- and stopband gain and the width of the transition band. The remaining factor, the degree $2d$ of the filter, is determined to at least conform to these constraints. They are related by the degree equation

$$d\frac{K'}{K} = \frac{K'_1}{K_1}. \tag{2.11}$$

Otherwise, κ_1 can be found as $\kappa_1 = 1/R_d(1/\kappa)$ [LTE01] from the roots and poles of the elliptic rational function, given that they can be computed without the knowledge of κ_1, as is outlined in the following. The elliptic function $\operatorname{cn}(\nu, \kappa)$ is quasi-periodic[6] in $2K$ and, with $\operatorname{cn}(K, \kappa) = 0$, the roots of $\mathcal{R}_d(\omega)$ are given by

$$\begin{aligned} \operatorname{cn}\left(d\frac{K_1}{K}\nu_j, \kappa_1\right) = 0 \iff \nu_j &= \frac{K(2j-1)}{d} && \text{for } j = 1, 2, \ldots \\ \iff \bigcirc_j &= \operatorname{cd}\left(K\frac{2j-1}{d}, k\right) && \text{for } j = 1, 2, \ldots \end{aligned}$$

and $\bigcirc_j = \operatorname{cd}(\nu_j, \kappa)$. Since $\operatorname{cn}(\nu, \kappa)$ and $\operatorname{sn}(\nu, \kappa)$ are $2K$-quasi-periodic and

$$\operatorname{dn}(\nu, \kappa) = \sqrt{1 - \kappa^2 \operatorname{sn}^2(\nu, \kappa)}$$

is thus $2K$-full-periodic, $\operatorname{cd}(\nu, \kappa)$ has again a full period of $4K$. Hence, since the roots appear in pairs which differ in sign,[7] they repeat for $j > d$. To find the poles, we make use of the relation [Dan74]

$$\mathcal{R}_d(\omega) = \frac{1}{\kappa_1 \mathcal{R}_d\left(\frac{1}{\kappa\omega}\right)}.$$

Therefore the poles are located where

$$\mathcal{R}_d\left(\frac{1}{\kappa\omega}\right) = 0 \iff \frac{1}{\kappa \times_j} = \bigcirc_j.$$

In case d is odd, $\mathcal{R}_d(\omega)$ has a root at $\omega = 0$ for index $j_0 = (d+1)/2$. The associated pole is formally at infinity and has to be skipped for computation,

$$\mathcal{R}_d(\omega) = \frac{1}{\varrho} \begin{cases} \displaystyle\prod_{j=1}^{d} \frac{\omega - \bigcirc_j}{\omega - \times_j} & \text{if } d \text{ even} \\ \displaystyle\prod_{j=1}^{d} (\omega - \bigcirc_j) \prod_{\substack{j=1 \\ j \neq j_0}}^{d} \frac{1}{\omega - \times_j} & \text{if } d \text{ odd.} \end{cases}$$

[6] $\operatorname{cn}(\nu + 2K, \kappa) = -\operatorname{cn}(\nu, \kappa)$.

[7] $\operatorname{cn}(K, \kappa) = 0$, $\operatorname{dn}(\nu, \kappa) \geq 0$.

The elliptic rational function is scaled such that it passes one at the cutoff frequency, $\mathcal{R}_d(1) = 1$,

$$\varrho = \begin{cases} \displaystyle\prod_{j=1}^{d} \frac{1-\times_j}{1-\bigcirc_j} & \text{if } d \text{ even} \\ \displaystyle\prod_{\substack{j=1 \\ j\neq j_0}}^{d-1} \frac{1-\times_j}{1-\bigcirc_j} & \text{if } d \text{ odd.} \end{cases}$$

The elliptic rational function again has the property of growing quickly outside of $[-1,1]$, and we let $f(\omega) = \mathcal{R}_d(\omega)$. Rewriting the filter function in terms of the numerator $p_d(\omega)$ and denominator $q_d(\omega)$ of the elliptic rational function $\mathcal{R}_d(\omega)$,

$$\frac{1}{1+\varepsilon^2\mathcal{R}_d^2(\omega)} = \frac{q_d^2(\omega)}{q_d^2(\omega)+\varepsilon^2 p_d^2(\omega)},$$

shows that the poles of $\mathcal{R}_d(\omega)$ are the roots of the filter and have to be duplicated. The poles are found similarly to the poles of the Chebyshev filter from Equation (2.10),

$$\begin{aligned} 1+\varepsilon^2\mathcal{R}_d^2(\omega) = 0 &\iff \operatorname{cd}\Big(d\frac{K_1}{K}\nu_j + 2jK_1, \kappa_1\Big) = \pm\frac{\mathbf{i}}{\varepsilon} && \text{for } j = 1,2,\dots \\ &\iff d\frac{K_1}{K}\nu_j = \operatorname{cd}^{-1}\Big(\pm\frac{\mathbf{i}}{\varepsilon},\kappa_1\Big) - 2jK_1 && \text{for } j = 1,2,\dots \\ &\iff \nu_j = \frac{K}{dK_1}\Big(\operatorname{cd}^{-1}\Big(\pm\frac{\mathbf{i}}{\varepsilon},\kappa_1\Big) - 2jK_1\Big) && \text{for } j = 1,2,\dots \end{aligned}$$

with $\omega = \operatorname{cd}(\nu_j,\kappa)$. Let

$$\vartheta = \frac{K}{dK_1}\operatorname{cd}^{-1}\Big(\pm\frac{\mathbf{i}}{\varepsilon},\kappa_1\Big),$$

such that ν_j can be written

$$\nu_j = \vartheta - \frac{2jK}{d} \quad \text{for } j = 1,\dots,2d.$$

Again, the choice of sign for the square root on the left side allows us to use the half-period $2K_1$ instead of the full period. Since $-\operatorname{cd}(\nu,\kappa) = \operatorname{cd}(\nu+2K,\kappa)$, the choice of the sign of $\mathbf{i}/\varepsilon$ simply shifts the index range and is therefore arbitrary. Due to $\operatorname{cd}(\nu,\kappa)$ being $4K$-periodic, the poles repeat after an index range of length $2d$. Purely imaginary poles appear in case of odd half-degrees for $j = (d+1)/2$ and $j = (3d+1)/2$. The rational filter then has fewer roots than poles. Otherwise, the number of roots and poles is equal. Exploiting symmetry and computing only one quadrant of poles is possible if caution is exercised when purely imaginary poles are involved.

The computation of the poles requires the computation of the inverse of the Jacobi elliptic function $\operatorname{cd}(\nu,\kappa)$. Since [AS74]

$$\operatorname{cd}(\nu,\kappa) = \operatorname{sn}(\nu+K,\kappa) = \operatorname{sn}(K-\nu,\kappa),$$

inverses are found as $\operatorname{cd}^{-1}(\omega,\kappa) = K - \operatorname{sn}^{-1}(\omega,\kappa)$ or $\operatorname{cd}^{-1}(\omega,\kappa) = \operatorname{sn}^{-1}(\omega,\kappa) - K$, depending on which value range is to be considered the principal branch of the inverse. Here, the choice is not of particular importance since it only changes the order in which the poles are computed. In accordance with the typical principal branch of $\cos^{-1}(z)$, the principal values for $\operatorname{cd}^{-1}(\omega,\kappa)$ may be chosen as $[0, 2K]$, in which case the option mentioned first defines the inverse. It is also possible to find exact formulas for the roots and poles without the need for computing elliptic functions [LTE01]. The filtering function passes 1/2 at

$$z = \pm f_p \operatorname{cd}\left(\frac{K}{dK_1} \operatorname{cd}^{-1}\left(\frac{1}{\varepsilon}, \kappa_1\right), \kappa\right).$$

Figure 2.13 shows an example of an elliptic filter. All roots are duplicate roots. The

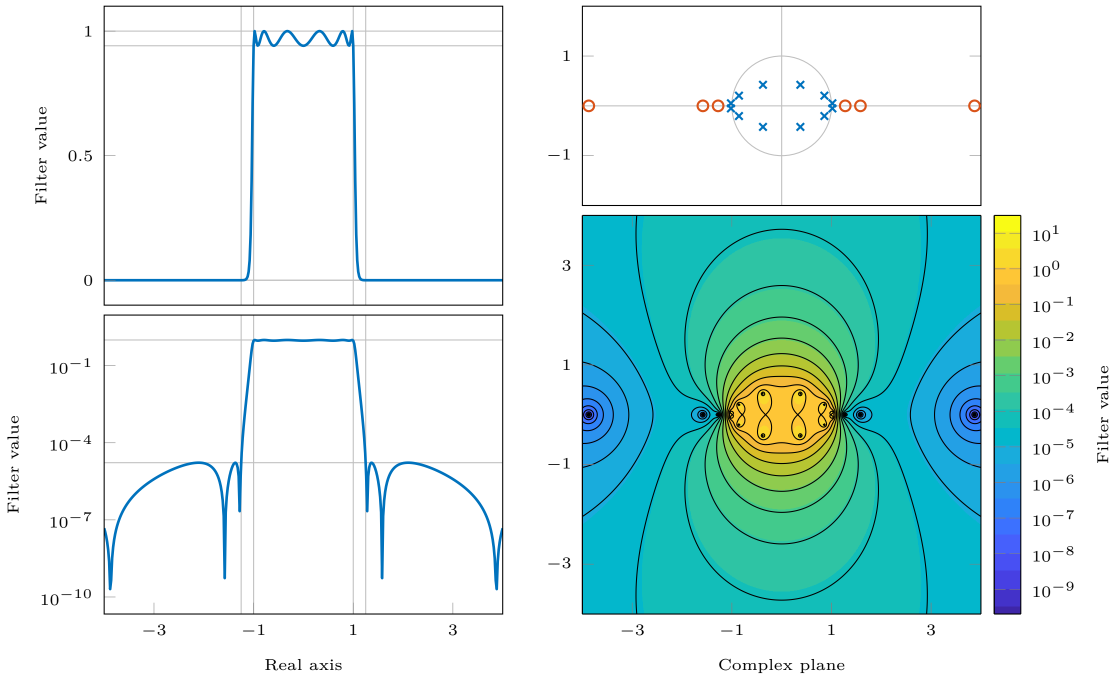


Figure 2.13: *Example of an elliptic filter of degree $d = 12$ with $\varepsilon = 0.25$ and $k = 0.8$. Left: non-logarithmic and logarithmic plots of the filter function. Top right: pole-root map. Roots are marked as circles, poles as crosses. Bottom right: logarithmic contour plot of the filter function in the complex plane.*

stopband ripple is only visible in the logarithmic plot. Due to the pole positions, the flank steepness decreases more rapidly compared to type-I Chebyshev filters, making elliptic filters less suited for non-Hermitian problems. Given their similarities, the astute observer might suspect the Zolotarev filter introduced earlier to be some sort of special case of elliptic filter. This seems impossible at first glance since the Zolotarev filter has only simple roots, while here all roots are duplicate and no choice of parameters can change this. Adjusting an elliptic filter to match a Zolotarev filter, however, requires the filter values to be shifted and scaled such that the filter oscillates about zero in the stopband and about one in the passband. The shift separates the roots and indeed, choosing the same degree, transition band, and passband oscillation yields identical filter functions.

2.4.4.1 Computation of inverse elliptic functions

The elliptic function $\mathrm{cd}(\nu, \kappa)$ can be computed in the same way as $\mathrm{sn}(\nu, \kappa)$ by substituting $\nu + K$ for ν as described before. Suitable libraries for the computation of elliptic functions have been listed in Section 2.3.4.

Again, an algorithm for the computation of inverse elliptic functions is included in Section A.2 of the appendix.

2.4.4.2 Design

As was the case before, the characteristic function is scaled to pass one at the cutoff frequency and

$$\varepsilon = \sqrt{\frac{1}{G_p} - 1}.$$

The two moduli for the elliptic integrals, κ and κ_1, are given by the remaining design parameters,

$$\kappa = \frac{f_p}{f_s}, \qquad \kappa_1 = \frac{\varepsilon}{\sqrt{\frac{1}{G_s} - 1}}.$$

The choice of κ determines the width of the transition band. Using the half-degree computed from κ and κ_1 with Equation (2.11) as

$$d = \left\lceil \frac{K}{K'} \frac{K_1'}{K_1} \right\rceil$$

will result in a stopband gain that is lower than specified, while matching passband gain and transition bandwidth. Often it may be more desirable to instead match passband and stopband gains and obtain a transition narrower than specified. In this case, the degree equation has to be solved for a new κ, using κ_1 and d. One possible method to achieve just that [Ant79] is via the representation of the elliptic sine function in terms of the *theta functions* θ_0 and θ_1,

$$\mathrm{sn}(\nu, \kappa) = \frac{1}{\sqrt{\kappa}} \frac{\theta_1\left(\frac{\nu}{2K}, o\right)}{\theta_0\left(\frac{\nu}{2K}, o\right)},$$

where $o = e^{-\pi \frac{K'}{K}}$ and with $\mathrm{sn}(K, \kappa) = 1$

$$\kappa = \frac{\theta_1^2\left(\frac{1}{2}, o\right)}{\theta_0^2\left(\frac{1}{2}, o\right)}.$$

For the sake of simplicity, we will define the theta functions θ_0 and θ_1 in terms of their series representation,

$$\begin{aligned}
\theta_0\left(\frac{\nu}{2K}, o\right) &= 1 + 2\sum_{j=1}^{\infty}(-1)^j o^{j^2}\cos\left(2j\frac{\pi\nu}{2K}\right),\\
\theta_1\left(\frac{\nu}{2K}, o\right) &= 2o^{\frac{1}{4}}\sum_{j=0}^{\infty}(-1)^j o^{j(j+1)}\sin\left((2j+1)\frac{\pi\nu}{2K}\right).
\end{aligned}$$

With $\cos(j\pi) = \sin\left(j\pi + \frac{\pi}{2}\right) = (-1)^j$,

$$\kappa = 4\sqrt{o}\left(\frac{\sum_{j=0}^{\infty} o^{j(j+1)}}{1 + 2\sum_{j=1}^{\infty} o^{j^2}}\right)^2.$$

Since K and K' are not available, we may replace them via the degree equation as

$$o = e^{-\pi\frac{K'}{K}} = e^{-\frac{\pi K_1'}{dK_1}}.$$

K_1 and K_1' are available as the complete elliptic integrals of the first kind for moduli κ_1 and κ_1'. The series representations of the theta functions converge rapidly [Ant79] and only few steps are required to achieve sufficient accuracy. Alternatively, κ_1 can be computed from κ and d in similar manner.

2.5 Sakurai-Sugiura-type methods

While the methods of the Sakurai-Sugiura type are, strictly speaking, not approximate projections obtained through spectral filtering, they do share algorithmic similarities, in particular with the Cauchy integral based filtering of the contour integration method, and the related subspace basis can be used in Rayleigh-Ritz subspace iteration [ST07] which is why we will outline them here. We will, however, not go into much detail about the differences in behavior compared to the other methods introduced.

Instead of approximating a filtering function on the spectrum of the matrix pencil (A, B), the origin of these methods is a root finding algorithm where a polynomial is constructed that has the same roots as some target function, making the application of a simpler root finding method for polynomials possible. Since the roots of the characteristic polynomial of a matrix pencil are its eigenvalues, the algorithm can be adapted so that it applies to generalized eigenproblems.

2.5.1 A root finding method for analytic functions

The root finding method for analytic functions described in [DL67] constitutes the first step in a series of modifications that ultimately leads to the formulation of the Rayleigh-Ritz type Sakurai-Sugiura Method.

Let $g : \mathbb{C} \longrightarrow \mathbb{C}$ be analytic inside a simple closed curve $\mathcal{C}$ in $\mathbb{C}$ and also analytic on $\mathcal{C}$ itself. Let further $z_1, \ldots, z_k$ be the k roots of g inside $\mathcal{C}$, counted with their respective multiplicity and let no roots of g be located on the contour $\mathcal{C}$. Choose

$$s_\rho := \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} z^\rho \frac{g'(z)}{g(z)}\, dz \tag{2.12}$$

such that the integrand has simple poles in z_i, $i = 1, \ldots, k$. If $\eta \leq k$ roots are distinct, let $\hat{z}_1, \ldots, \hat{z}_\eta$ be these distinct roots and $\mu_1, \ldots, \mu_\eta$ their multiplicity, such that $\sum_{i=1}^{\eta} \mu_i = k$. Then $g(z)$ and $g'(z)$ can be written [Ahl79]

$$\begin{aligned} g(z) &= (z - \hat{z}_i)^{\mu_i} h(z), \quad h(\hat{z}_i) \neq 0 \\ g'(z) &= \mu_i (z - \hat{z}_i)^{\mu_i - 1} h(z) + (z - \hat{z}_i)^{\mu_i} h'(z) \end{aligned}$$

for any distinct pole $\hat{z}_i$ with multiplicity μ_i and therefore the poles of

$$\frac{g'(z)}{g(z)} = \frac{\mu_i}{z - \hat{z}_i} + \frac{h'(z)}{h(z)} = \frac{1}{z - \hat{z}_i} \frac{\mu_i h(z) + (z - \hat{z}_i) h'(z)}{h(z)} \tag{2.13}$$

are simple. A generalization of Corollary 2.1 makes it possible to compute this integral.

Theorem 2.2 (Residue theorem [Ahl79], without winding number) *Let g be a function that is analytic almost everywhere. In particular, let there be η isolated singularities a_i in the interior of a positively oriented simple closed curve $\mathcal{C}$ in $\mathbb{C}$. Then*

$$\mathcal{P} := \oint_{\mathcal{C}} g(z) dz = 2\pi\mathbf{i} \sum_{i=1}^{\eta} \operatorname{Res}(g, a_i).$$

In particular $\mathcal{P} = 0$ if $\eta = 0$. The *residue* of g at a_i, $\operatorname{Res}(g, a_i)$, for simple poles can be computed as [Ahl79]

$$\operatorname{Res}(g, a_i) = (z - a_i) g(z) \Big|_{z\, =\, a_i}.$$

The deliberate choice of the form of the s_p from Equation (2.12) ensures evaluation to the power sums of the roots using the above theorem and applying the rule for simple poles. With Equation (2.13) follows

$$\operatorname{Res}\left(z^\rho \frac{g'(z)}{g(z)}, \hat{z}_i \right) = \hat{z}_i^\rho \mu_i$$

and thus

$$s_\rho = \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} z^\rho \frac{g'(z)}{g(z)}\, dz = \sum_{i=1}^{\eta} \operatorname{Res}\left(z^\rho \frac{g'(z)}{g(z)}, \hat{z}_i \right) = \sum_{i=1}^{\eta} \mu_i \hat{z}_i^\rho = \sum_{i=1}^{k} z_i^\rho.$$

The power sums of the roots z_i appear in Newton's identities to find the coefficients c_j of a polynomial

$$\prod_{i=1}^{k}(z - z_i) = \sum_{j=0}^{k} c_j z^j$$

given only the power sums $s_0, \dots, s_k$. The degree k of the polynomial is known by $s_0 = k$. Letting $c_k = 1$ (the roots of a function are obviously invariant with respect to scaling), the coefficients of the polynomial are given by the solution of the triangular linear system [CS82]

$$\begin{pmatrix} 1 & & & & \\ s_1 & 2 & & & \\ s_2 & s_1 & 3 & & \\ \vdots & \vdots & \ddots & \ddots & \\ s_{k-1} & s_{k-2} & \cdots & s_1 & k \end{pmatrix} \begin{pmatrix} c_{k-1} \\ c_{k-2} \\ c_{k-3} \\ \vdots \\ c_0 \end{pmatrix} = - \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_k \end{pmatrix}. \tag{2.14}$$

This leaves the determination of the roots of the resulting polynomial as remaining (much simpler) task. Due to their characteristic form, the integrals s_ρ are often referred to as *moments*,[8] s_ρ in particular being the ρ-th moment.

2.5.2 An improved root finding method

In [KSB99] the root finding method is refined to circumvent the ill-conditioning in case of clustered roots within the integration contour. To this end, finding the distinct roots $\hat{z}_1, \dots, \hat{z}_\eta$ via a polynomial of degree η with simple roots $\hat{z}_i$ first and deducing their multiplicities $\mu_1, \dots, \mu_\eta$ separately is proposed. The derivation of such a polynomial, however, is not as straight forward as it was for the original method since now the s_ρ do not meet the requirements for computing the polynomial of degree $\eta < k$ using Newton's identities,

$$s_\rho = \sum_{i=1}^{\eta} \mu_i \hat{z}_i^\rho \neq \sum_{i=1}^{\eta} \hat{z}_i^\rho. \tag{2.15}$$

The difficulty lies solely in relating the polynomial coefficients c_i to the s_ρ again. Let $\varphi(z) = \sum_{i=0}^{\eta} c_i z^i$ be a polynomial of degree η. Surely, the condition

$$\sum_{i=1}^{\eta} \mu_i \hat{z}_i^\rho \varphi(\hat{z}_i) = 0$$

is fulfilled if the $\hat{z}_i$ are the η roots of φ. Of course, this is not necessarily the only situation in which this condition can be fulfilled. For finding the coefficients of φ,

[8] Cf. the definition of the term moment in statistics and mechanics.

fix the scaling of φ with $c_\eta = 1$ and require

$$\begin{aligned}
&\sum_{i=1}^{\eta} \mu_i \hat{z}_i^\rho \varphi(\hat{z}_i) = \sum_{i=1}^{\eta} \mu_i \hat{z}_i^\rho \sum_{j=0}^{\eta} c_j \hat{z}_i^j = \sum_{j=0}^{\eta} c_j \sum_{i=1}^{\eta} \mu_i \hat{z}_i^{\rho+j} = \sum_{j=0}^{\eta} c_j s_{\rho+j} = 0 \\
\iff &\sum_{j=0}^{\eta-1} c_j s_{\rho+j} = -s_{\rho+\eta} \qquad (2.16)
\end{aligned}$$

for $\rho = 0, \dots, \eta - 1$, which yields the order-η linear system

$$\underbrace{\begin{pmatrix} s_0 & s_1 & \cdots & s_{\eta-1} \\ s_1 & s_2 & \cdots & s_\eta \\ \vdots & \vdots & \iddots & \vdots \\ s_{\eta-1} & s_\eta & \cdots & s_{2\eta-2} \end{pmatrix}}_{=:\, H_\eta} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{\eta-1} \end{pmatrix} = - \begin{pmatrix} s_\eta \\ s_{\eta+1} \\ \vdots \\ s_{2\eta-1} \end{pmatrix} \qquad (2.17)$$

using the moments $s_0, \dots, s_{2\eta-1}$. The matrix H_η has *Hankel form*. If H_η is regular, a unique solution for φ exists; this solution must be for φ to have roots $\hat{z}_1, \dots, \hat{z}_\eta$ since at least this solution always exists. Indeed, it can be shown [KSB99, Thm. 2.1] that H_t has rank η for any $t \geq \eta$.

The roots of a polynomial φ can be obtained as the eigenvalues of its *companion matrix* [HJ13]

$$C_\varphi = \begin{pmatrix} 0 & \cdots & 0 & -c_0 \\ 1 & \cdots & 0 & -c_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & -c_{\eta-1} \end{pmatrix}.$$

Let (λ, x) be an eigenpair of C_φ, then

$$C_\varphi x = \lambda x \iff H_\eta C_\varphi x = \lambda H_\eta x.$$

The roots of φ can therefore be found by solving the generalized eigenproblem of the matrix pencil $(H_\eta C_\varphi, H_\eta)$ [KSB99, Thm. 4.1]. With Equation (2.16) for $\rho = 0, \dots, \eta - 1$ follows

$$H_\eta C_\varphi = \begin{pmatrix} s_1 & s_2 & \cdots & s_\eta \\ s_2 & s_3 & \cdots & s_{\eta+1} \\ \vdots & \vdots & \iddots & \vdots \\ s_\eta & s_{\eta+1} & \cdots & s_{2\eta-1} \end{pmatrix}.$$

The matrix pencil $(H_\eta C_\varphi, H_\eta)$ can therefore be built from just the 2η moments $s_0, \dots, s_{2\eta-1}$ and the matrix $H_\eta C_\varphi$ again has Hankel form. To acquire the multiplicities $\mu_1, \dots, \mu_\eta$ via the now known roots $\hat{z}_1, \dots, \hat{z}_\eta$, from Equation (2.15) arises the linear system

$$\begin{pmatrix} 1 & \cdots & 1 \\ \hat{z}_1 & \cdots & \hat{z}_\eta \\ \vdots & \ddots & \vdots \\ \hat{z}_1^{\eta-1} & \cdots & \hat{z}_\eta^{\eta-1} \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_\eta \end{pmatrix} = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{\eta-1} \end{pmatrix}$$

to be solved for the μ_i. The matrix is of transposed *Vandermonde* form.

2.5.3 Application to generalized eigenvalue problems

To apply the method introduced in the previous section to generalized eigenproblems [SS03a], it is necessary to construct a scalar function w.r.t. (A, B) with k poles $\lambda_1, \dots, \lambda_k$ in $\mathcal{C}$. To this end, let $B^{-1}A$ have the eigendecomposition[9] $X\Lambda X^{-1}$. Then

$$(zB - A)^{-1}B = X(zI - \Lambda)^{-1}X^{-1}$$

with eigenvalues $1/(z - \lambda)$. In particular, $zB - A$ is singular for $z = \lambda_1, \dots, \lambda_k$. Using two random non-zero vectors v and y to obtain a scalar function

$$g(z) = v^H(zB - A)^{-1}By = v^H X(zI - \Lambda)^{-1}X^{-1}y \tag{2.18}$$

and letting $[v_1', \dots, v_n'] := v^H X$ and $[y_1', \dots, y_n']^T := X^{-1}y$ yields

$$g(z) = \sum_{i=1}^{n} \frac{v_i' \, y_i'}{z - \lambda_i}.$$

Let $\xi_1, \dots, \xi_\eta$ be the distinct eigenvalues of (A, B) in $\mathcal{C}$. The function $g(z)$ has only simple poles since multiplication with any $(z - \xi_i)$ removes all occurrences of ξ_i. For any $\xi_t = \lambda_j = \dots = \lambda_{j+\mu}$ with multiplicity μ, the residue of $z^\rho g(z)$ at ξ_t is therefore

$$\operatorname{Res}(z^\rho g(z), \xi_j) = \xi_j^\rho \underbrace{\sum_{i=1}^{\mu} v_{j+i}' \, y_{j+i}'}_{=: \, \nu_j}$$

and

$$s_\rho := \frac{1}{2\pi \mathbf{i}} \oint_{\mathcal{C}} z^\rho g(z)\, dz = \sum_{i=1}^{\eta} \nu_i \xi_i^\rho.$$

Note that the ν_i do not represent the multiplicity anymore and that we have to require $\nu_i \neq 0$ for $i = 1, \dots, \eta$. With this, the positions of the distinct eigenvalues of (A, B) in $\mathcal{C}$ can be determined.

2.5.3.1 Blocking and Eigenvectors

The reason for computing a certain amount of moments is for the linear system in Equation (2.17) to have a sufficient amount of equations in order to be able to solve for all coefficients of the polynomial φ. In the case of Equation (2.18), it is easy to find a large number of functions, all sharing the same roots, by varying the vectors v and y. The associated polynomial with identical roots is unique. Combining the moments corresponding to a certain power ρ into a matrix yields the block moments

$$S_\rho = \frac{1}{2\pi \mathbf{i}} \oint_{\mathcal{C}} z^\rho V^H (zB - A)^{-1} BY \, dz.$$

[9] A formulation that allows singular B and defective eigenvalues can be found in [SS03a].

Often, the choice $V = Y$ is made. Building a linear system in accordance with Equation (2.16) using moments of different functions, however, only decreases the required number of moments moderately. A blocked version of the Sakurai-Sugiura method that allows for significant reduction of the number of (block) moments to be computed by employing multiple vectors and considering the method from a spectral operator point of view has been introduced in [ISN10]. Since

$$\begin{aligned} M_\rho &= \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} z^\rho (zB - A)^{-1} B \, dz = \frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} z^\rho X (zI - \Lambda)^{-1} X^{-1} \, dz \\ &= X \operatorname{diag}\left(\frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} z^\rho \frac{1}{z - \lambda_i} \, dz, \ i = 1, \ldots, n \right) X^{-1} \end{aligned}$$

and

$$\frac{1}{2\pi\mathbf{i}} \oint_{\mathcal{C}} z^\rho \frac{1}{z - \lambda_i} \, dz = \operatorname{Res}\left(\frac{z^\rho}{z - \lambda_i}, \lambda_i \right) = \begin{cases} \lambda_i^\rho \text{ if } \lambda_i \in \Omega \\ 0 \text{ otherwise,} \end{cases} \tag{2.19}$$

the matrix operator for the moment ρ in the sense of subspace iteration is

$$\begin{aligned} M_\rho &= X \operatorname{diag}(d_1, \ldots, d_n) X^{-1}, \quad d_i = \begin{cases} \lambda_i^\rho \text{ if } \lambda_i \in \Omega \\ 0 \text{ otherwise} \end{cases} \\ &= X_k \operatorname{diag}(\lambda_1^\rho, \ldots, \lambda_k^\rho) X_k^{-1} \end{aligned}$$

and $S_\rho = V^H M_\rho Y$. Remember that $\mathcal{C} = \partial\Omega$. Let (Λ, W) be the eigenpairs of the matrix pencil $(\mathbb{M}_1, \mathbb{M}_0)$ with

$$\mathbb{M}_\rho = Z^H M_\rho U$$

for vector blocks Z and U with at least k columns. Then

$$\begin{aligned} && \mathbb{M}_1 W &= \mathbb{M}_0 W \Lambda \\ \iff && Z^H X_k \operatorname{diag}(\lambda_1, \ldots, \lambda_k) X_k^{-1} U W &= Z^H X_k X_k^{-1} U W \Lambda \\ \iff && \operatorname{diag}(\lambda_1, \ldots, \lambda_k) X_k^{-1} U W &= X_k^{-1} U W \Lambda \\ \iff && \operatorname{diag}(\lambda_1, \ldots, \lambda_k) \widetilde{W} &= \widetilde{W} \Lambda, \end{aligned}$$

which shows that $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_k)$. In particular, $\widetilde{W} = I$ and thus

$$X_k = X_k X_k^{-1} U W = M_0 Y W.$$

Here we assume that Z and U have full rank and k columns and the inverse of $Z^H X_k$ exists. For larger matrices Z and U, $\mathbb{M}_1$ and $\mathbb{M}_0$ become singular.[10] The problem is again reduced to solving the reduced sized eigenproblem for $(\mathbb{M}_1, \mathbb{M}_0)$.

[10] Since the Rayleigh-Ritz theorem does not require the Rayleigh quotients to be smaller than the original matrices, it implies that $\lambda_1, \ldots, \lambda_k$ are eigenvalues of $(\mathbb{M}_1, \mathbb{M}_0)$ also in the singular case. The regular part may be extracted using a singular value decomposition of $\mathbb{M}_0$ [ISN10].

With

$$\begin{aligned} M_i M_j &= X_k \operatorname{diag}\left(\lambda_1^i, \dots, \lambda_k^i\right) X_k^{-1} X_k \operatorname{diag}\left(\lambda_1^j, \dots, \lambda_k^j\right) X_k^{-1} \\ &= X_k \operatorname{diag}\left(\lambda_1^i, \dots, \lambda_k^i\right) \operatorname{diag}\left(\lambda_1^j, \dots, \lambda_k^j\right) X_k^{-1} \\ &= X_k \operatorname{diag}\left(\lambda_1^{i+j}, \dots, \lambda_k^{i+j}\right) X_k^{-1} \\ &= M_{i+j}, \end{aligned}$$

the specific choice

$$Z^H = \begin{bmatrix} V^H M_0 \\ V^H M_1 \\ \vdots \\ V^H M_{\ell-1} \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} M_0 Y & M_1 Y & \cdots & M_{\ell-1} Y \end{bmatrix}$$

yields the matrices

$$\mathbb{M}_\rho = Z^H M_\rho U = \begin{pmatrix} V^H M_\rho Y & \cdots & V^H M_{\rho+\ell-1} Y \\ \vdots & & \vdots \\ V^H M_{\rho+\ell-1} Y & \cdots & V^H M_{\rho+2\ell-2} Y \end{pmatrix} = \begin{pmatrix} S_\rho & \cdots & S_{\rho+\ell-1} \\ \vdots & & \vdots \\ S_{\rho+\ell-1} & \cdots & S_{\rho+2\ell-2} \end{pmatrix},$$

which now have a blocked Hankel form. The eigenvectors can accordingly be reconstructed as

$$X_k = M_0 U W = M_0 [M_0 Y, \dots, M_{\ell-1} Y] W = [M_0 W, \dots, M_{\ell-1} Y] W = U W.$$

If $V = Y$, then $Z = U$. Note that X_k^{-1} is the left inverse of X_k, see also Sections 1.1.2 and 1.3.3. Without loss of generality, let X be partitioned into two arbitrary block columns and let X^{-1} be partitioned accordingly into two block rows. Then

$$I = X^{-1} X = \begin{pmatrix} X_1^{-1} \\ X_2^{-1} \end{pmatrix} \begin{pmatrix} X_1 & X_2 \end{pmatrix} = \begin{pmatrix} I_1 & 0 \\ 0 & I_2 \end{pmatrix}.$$

The argument can be extended recursively or by reordering until the desired partitioning is reached. In the case of single vectors $V = v$ and $Y = y$, the above reduces to the original method with $\mathbb{M}_1 = H_\eta C_\varphi$ and $\mathbb{M}_0 = H_\eta$. It conveniently shows how to acquire the corresponding eigenvectors as well.

As described in [ST07], with a vector written as linear combination of eigenvectors, $y_j = \sum_{i=1}^n \gamma_{ij} x_i$, and $(zB - A)^{-1} B x_i = 1/(z - \lambda_i) x_i$,

$$M_\rho y_j = \frac{1}{2\pi \mathbf{i}} \oint_{\mathcal{C}} z^\rho \sum_{i=1}^n \gamma_{ij} \frac{1}{z - \lambda_i} x_i \, dz = \sum_{i=1}^n \gamma_i \frac{1}{2\pi \mathbf{i}} \oint_{\mathcal{C}} z^\rho \frac{1}{z - \lambda_i} x_i \, dz \overset{(2.19)}{=} \sum_{i=1}^k \gamma_i \lambda_i^\rho x_i$$

and $[M_0 y_j, \dots, M_{\ell-1} y_j]$ can be written as $X_k \mathcal{D}_j \mathcal{V}_\ell$ where $\mathcal{D}_j = \operatorname{diag}(\gamma_{ij},\ i = 1, \dots, k)$ and $\mathcal{V}_\ell$ is the $k \times \ell$ Vandermonde matrix[11]

$$\mathcal{V}_\ell = \begin{pmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{\ell-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_k & \lambda_k^2 & \cdots & \lambda_k^{\ell-1} \end{pmatrix}.$$

Consequently, we may write a reordered U,

$$U' = [M_0 y_1, \dots, M_{\ell-1} y_1, \dots, M_0 y_\zeta, \dots, M_{\ell-1} y_\zeta]$$

as[12]

$$U' = [X_k \mathcal{D}_1 \mathcal{V}_\ell, \dots, X_k \mathcal{D}_\zeta \mathcal{V}_\ell] = X_k \underbrace{\begin{pmatrix} \mathcal{D}_1 & \cdots & \mathcal{D}_\zeta \end{pmatrix}}_{=:\, \mathbb{D}} \underbrace{\begin{pmatrix} \mathcal{V}_\ell & & \\ & \ddots & \\ & & \mathcal{V}_\ell \end{pmatrix}}_{=:\, \mathbb{V}}.$$

Clearly, $\mathcal{V}_\ell$ has at most rank k and looses rank for every occurrence of multiple eigenvalues, dropping the rank down to η. Increasing the number of moments amounts to adding columns to $\mathcal{V}_\ell$, while adding a right-hand side vector $y_{\zeta+1}$ increases the rank of $\mathbb{V} \in \mathbb{C}^{k\zeta \times \ell\zeta}$ by the rank of $\mathcal{V}_\ell$. Similarly, it may increase the rank of the matrix $\mathbb{D} \in \mathbb{C}^{k \times k\zeta}$ if $\gamma_{i,\zeta+1} \neq 0$ where all $\gamma_{ij} = 0$ for $i = 1, \dots, n$ and $j = 1, \dots, \zeta$. In particular, employing more than one right-hand side vector, $\zeta > 1$, makes it possible to compute eigenvalues with multiplicity larger than one.

Note that, formally, if any $\lambda_i = 0$, then M_0 does not exist since 0^0 is not defined. For all M_ρ, $\rho > 0$, the component in direction x_i is lost. If, however, M_0 is specifically defined identically to the projection operator from the Cauchy filter, thereby implicitly allowing $0^0 = 1$, the method can compute zero-valued eigenvalues if the zeroth moment is used. In practice, while computing the moments via numerical quadrature, integration nodes can be chosen to never be zero and the problem doesn't arise. In this case, the above formal requirement is respected implicitly.

2.5.3.2 Rayleigh-Ritz

The method introduced in [ST07] now applies the above U as a subspace basis in a Rayleigh-Ritz context. While this formulation only employs a single vector with a corresponding number of moments $\rho \in \{0, \dots, \ell - 1\}$ to build a subspace basis, with the above it is also possible to find a basis by increasing the number of right-hand sides $Y = [v_1, \dots, v_\zeta]$ or a combination of both [IS10]. For the columns of U to be a spanning set for the space spanned by the eigenvectors associated with eigenvalues inside $\mathcal{C}$, we generally have to require

[11] Incidentally, the inverse of this Vandermonde matrix gives the eigenvectors of the companion matrix C_φ and, as such, eigenvectors of $(H_\eta C_\varphi, H_\eta)$. We may infer that for the single vector case $X_k \mathcal{D}_j = U \mathcal{V}_\ell^{-1}$ which differs only in scaling of the eigenvectors X_k or W.

[12] Similarly, V may be decomposed with respect to the left eigenvectors of the matrix pencil; the same reasoning applies.

1. $\ell\zeta \geq k$, i.e., the number of columns of $\mathbb{V}$ exceeds k,
2. $\zeta \geq k/\eta$, i.e., the row rank of $\mathbb{V}$ exceeds k,
3. $\mathrm{Rank}(\mathbb{D}) = k$, i.e., $\gamma_{ij} \neq 0$ for at least one y_j for all $i \in \{1, \dots, k\}$, and
4. the condition regarding M_0 from above is met in case $B^{-1}A$ is singular.

As both the Rayleigh-Ritz and the Hankel matrix-based variant of the Sakurai-Sugiura method require the subspace basis to be of full rank for the reduced eigenproblem to be solvable and since U can at most have rank k, some scheme of reduction is required, orthogonalization being one option.

2.5.3.3 Filtering functions

While it is not possible to write down the effective filtering function for Sakurai-Sugiura methods using multiple moments, the moment matrix operators M_ρ have, by construction, the same eigenvectors as (A, B) and constitute a filtering function on the spectrum of (A, B), similarly to the power method but with explicit removal of certain directions. The filter of a single moment can be visualized [ISN10] by depicting how much the absolute value of an eigenvector is scaled depending on the location of its associated eigenvalue, see Figure 2.14.

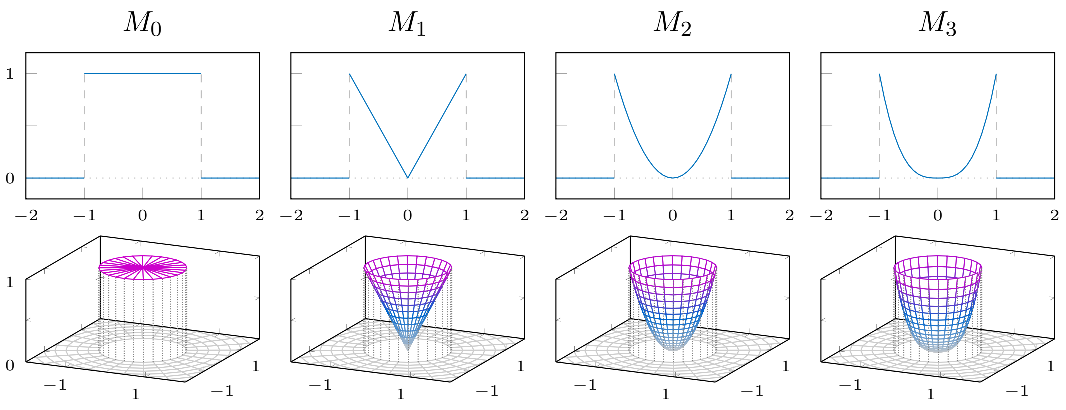


Figure 2.14: *Spectral filtering functions for the moments 0 to 3, absolute value. Top: for the interval* $[-1, 1] \subset \mathbb{R}$*. Bottom: for the unit circle on* $\mathbb{C}$*.*

2.5.3.4 Transformations of the weight function

The weight function z^ρ directly dictates the eigenvalues of the matrix pencil $(\mathbb{M}_1, \mathbb{M}_0)$ since the eigenvalues are the residues of the poles of the first moment, as was seen in Equation (2.19). Therefore, any substitution of z translates directly to the computed eigenvalues of the Hankel matrix based Sakurai-Sugiura methods. In the case of simple shifting and scaling, as it is applied for numerical reasons in, e.g., [ISN10], the computed eigenvalues appear shifted and scaled accordingly. A corresponding back transformation is required to acquire the eigenvalues of (A, B).

On the contrary, applying transformations to the weight function does not require additional measures to be taken in case of Rayleigh-Ritz based variants of the method. Since it is based on the compatibility of eigenvectors alone and the eigenvectors X_k of the non-singular part of M_ρ are, by construction, eigenvectors of $(B, zB - A)$, which again are eigenvectors of (A, B), the computed eigenvalues of the Rayleigh quotients

$$\left(Z^H AU, Z^H BU\right)$$

are not influenced by the weight function, as long as the conditions for providing a suitable subspace basis are not violated. The shape of the filter functions of the ρ moments, however, obviously changes with the contour $\mathcal{C}$. To keep the magnitude of the root powers small and to keep the filtering functions independent of the integration region, shifting and scaling the weight function accordingly is again advisable.

2.5.3.5 Computation

Due to the algorithmic similarity to the Cauchy filtering method, see Section 2.2, all the possible simplifications discussed there also apply to the computation of the subspace basis for the Rayleigh-Ritz type Sakurai-Sugiura methods. Since the solution of the linear systems can be considered to be the most expensive part of the computations and their solution can be reused for the different moments, it is advisable to not parallelize over moments at all as to not solve a particular linear system on more than one process or process group.

For numerical quadrature, the different schemes mentioned earlier in the context of the Cauchy filter, Section 2.2.2, can of course be used again. Similarly, effective filtering functions based on these quadrature schemes can be found (compare Section 4.2.3, Figure 4.9).

2.5.3.6 Relation to spectral projection

Analogously to [IDS16], assuming a perfect filter $\mathcal{X}$, the matrix operator of the moment ρ can be written

$$M_\rho = X\mathcal{X}(\Lambda)\Lambda^\rho X^{-1} = X\mathcal{X}(\Lambda)X^{-1} \underbrace{X\Lambda^\rho X^{-1}}_{= (B^{-1}A)^\rho}$$

and thus

$$\left[M_0 Y \; M_1 Y \; \cdots \; M_{\ell-1} Y\right] = X\mathcal{X}(\Lambda)X^{-1}\left[Y \; B^{-1}AY \; (B^{-1}A)^2 Y \; \cdots \; (B^{-1}A)^{\ell-1} Y\right],$$

which is the block-Krylov space of the matrix $B^{-1}A$ and the vector block Y. It is therefore possible to interpret the Sakurai-Sugiura method as spectral filtering applied to the block-Krylov space above. This understanding translates to an approximate filter as well [IDS16].

2.6 Other filtering methods

As has been stressed above, any approximation of a meaningful filtering function (we have seen from the power iteration that the pass-band not necessarily has to be forced to be close to one for the filter to be effective) of rational form such that it can be expressed in terms of a partial fraction representation may serve to construct matrix operators for computing inner eigenpairs. There are many candidates of approximation methods that meet these criteria; this work by no means intends to mention them all.

2.6.1 Rational filter types

While we did not discuss rational filters

$$f(\lambda) = \frac{p(\lambda)}{q(\lambda)}$$

with $\deg(p) > \deg(q)$, they are equally suitable for use as spectral filters. The remainder of the polynomial long division in this case is a non-trivial polynomial which can be computed efficiently using *Horner's method* (see, e.g., [Knu97]) requiring d multiplications with the matrix pencil, assuming d is the degree of the polynomial. If the eigenproblem is generalized, this includes d solves with B.

Let a matrix polynomial in the matrix argument C of degree d be given as

$$p(C) = \sum_{i=0}^{d} c_i C^i$$

which may be rewritten as

$$p(C) = c_0 + (c_1 + (c_2 + \ldots + (c_{d-1} + c_d C)C)C)C.$$

Letting $\gamma_d = c_d$, $\gamma_{d-1} = c_{d-1} + \gamma_d C$, defines a recursive sequence for the computation of $p(C) = \gamma_0 = c_0 + \gamma_1 C$. Applied to a vector block V, the computation as

$$\begin{aligned}
\Gamma_{d-1} &= c_{d-1}V + c_d C V\\
\Gamma_{d-2} &= c_{d-2}V + C\Gamma_{d-1}\\
&\ldots\\
\Gamma_0 &= c_0 V + C\Gamma_1
\end{aligned}$$

requires the maintenance of three block vectors, one for saving V which is used in every step, and two additional buffers for alternating the Γ_i since matrix-vector multiplication cannot be performed in-place. The coefficients c_i can be computed from the roots of the polynomial using Newton's identities, see Equation (2.14).

If also poles of multiplicity > 1 are to be allowed, the partial fraction form becomes more complex, in particular involving polynomials in the denominators. These cases shall not be discussed here.

2.6.2 Delta filters

The approximation of the window function is not a fundamental necessity of subspace iteration. We may, arguably, not speak of spectral filtering per se in this case, but more unorthodoxly shaped functions may improve convergence or reduce costs in certain cases. From the power iteration method we saw that the most fundamental filter is the matrix spectrum itself. The term *filter* is, of course, used somewhat aberrantly in this case.

Another example is the polynomial approximation of the Dirac δ-function. For Chebyshev polynomials, the analytical computation of the corresponding inner products (see Section 2.1.2) is straight forward,

$$\langle T_i, \delta \rangle = \int_{-1}^{1} \delta(z - c) T_i(z)\, dz = T_i(c)$$

with $w(z) \equiv 1$. The result is a distinct peak, centered at some target c inside $[-1, 1]$, that narrows with increasing degree. Due to a gentler fall-off of filter values with increasing distance to the peak, this filter is only feasible for increasingly small spectral target regions. A summation of identical delta approximations, located at the centers of equal subdivisions of the target interval, again approaches the approximate window function of Chebyshev polynomials with increasing number. The typical Gibbs oscillations can again be smoothed, see Section 2.1.4. Filters of this kind have been used, e.g., in [Li+16].

2.6.3 Least squares

Methods for the approximation of target functions also include strategies for fitting a rational filtering function based on preselected pole locations using a least squares approach [XS16]. The goal is to find coefficients a_j for a rational function

$$L(z) = \sum_{j=1}^{d} \frac{a_j}{z - \times_j}$$

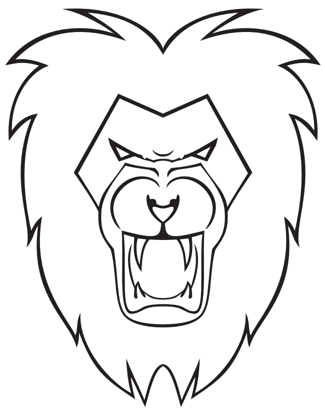
with poles $\times_j$ such that $\|\chi_\omega - L\|_w^2$ is minimized. Here, the target region ω typically is the interval $[-1, 1]$. The norm is given by the function inner product with some weight function w (see Section 2.1.1). For more details, see [XS16]. Filters of this kind have been used alongside Cauchy-filters in [FS12].

2.6.4 Beyond conventional subspace iteration

Due to the compatibility of eigenvectors, other methods for solving eigenvalue problems can be modified to benefit from spectral transformations, such as the filtering function introduced in this chapter, to improve convergence speeds, given that eigenvalues of the original problem can always be obtained from just the eigenvectors via the proper Rayleigh quotients.

The Lanczos algorithm (cf. Section 1.7.2), for example, has been modified to employ a filtering operator in [FS12] and [Li+16].

Chapter 3
BEAST



THIS very short chapter shall, in all brevity, introduce the software that lends its name to the following two chapters as the primary prototype implementation of most of the concepts and algorithms introduced therein, as well as many aspects of the preceding chapters. Originally inspired by the very first FEAST algorithm introduced in 2009 [Pol09], functionalities and features were continuously extended, ultimately giving it its name ***b**eyond F**EAST***, or BEAST. Of course, the original implementation of FEAST [PK15; Pol20] evolved as well and the name is only meant to refer to the basic algorithm. The implications of interval-level parallelism in BEAST will serve as motivation for the following chapters.

3.1 The BEAST framework

BEAST is a feature-rich software library that acts as framework for subspace iteration algorithms, in particular—but in the future not necessarily limited to—spectral projection. Currently, the list of implemented filters contains the polynomial Chebyshev (Section 2.1) and Cauchy (Section 2.2) filters (midpoint and Gauss-Legendre integration flavors), but the other alternatives from Chapter 2 are to follow. Beyond that, potentially arbitrary rational filters can be implemented. Many other features were integrated to improve reliability, robustness, and performance that shall be mentioned here in a short overview. BEAST is written in pure C to allow comparably easy implementation of potential interfaces in C++, Fortran, or Python.

BEAST is intended to solve very large scale Hermitian eigenproblems of standard or generalized form where some or even many inner eigenpairs are to be computed on modern hybrid-parallel high performance supercomputers. Subspace iteration eigensolvers offer great potential for parallelization. In particular interval level parallelism introduces certain difficulties into the concept of spectral projection eigensolvers. First, the choice of filtering function (Chapter 2) affects the interaction between intervals. In which way this interference affects the result of an eigensolver is explored in Chapter 4, in particular with regard to eigenvector orthogonality, which will be the focus of Chapter 5.

BEAST has been developed and used in the BMBF project "Eigenwert-Löser für Petaflop-Anwendungen" (ELPA) [Mar+14] and its follow-up project "ELPA - Algorithmische Erweiterungen und Optimierungen" (ELPA-AEO), as well as the subprojects "Equipping Sparse Solvers for Exascale" (ESSEX) [Alv+14; Wel+20] and ESSEX-II of the DFG priority programme "Software for Exascale Computing" (SPPEXA).

3.1.1 Basics

For basic building blocks, i.e., fundamental operations such as sparse matrix-vector multiplication (also for blocks of vectors), dense matrix-matrix multiplication, as well as column-wise dot products and the associated data structures for sparse and dense matrices, BEAST relies on GHOST, the general, hybrid, and Optimized Sparse Toolkit [KE+; Kre+16; Kre+17], or PHIST, a pipelined hybrid parallel iterative solver toolkit [TR+; Thi+19; Thi+16], which guarantee highest performance on hybrid parallel systems. To not confuse them with the different kernel libraries provided by PHIST (which also includes GHOST), we will refer to them as *back end* libraries.

These operations play a major role in the application of the approximate projector, which typically is the dominant component in terms of performance of the algorithm. As such, the per-node performance is provided by the underlying back end libraries.

3.1.2 Parallelism

An inherent property of the spectral projection algorithm is its potential for multi level parallelism (see also [PK15; Pol20]). In total, we can differentiate four levels. The higher levels encapsulate the lower levels as groups of processes, resulting in a process group hierarchy. These levels of parallelism are generally realized by the back end libraries and BEAST itself via MPI [MPI15].

An additional level of parallelism that is not mentioned in this hierarchy is shared memory multi threading. Both GHOST and PHIST provide this kind of parallelism transparently, but BEAST itself uses explicit thread-level parallelism whenever applicable and sensible, via OpenMP [Ope18].

Level 0: Basic operations In case of BEAST, the lowest possible level of parallelism is provided by the back end libraries (see above) in terms of matrix and block vector distribution in combination with operations to act on the distributed data. The data distribution scheme is dictated by the governing sparse matrix, subdividing matrix and block vectors into block rows of homogeneous or inhomogeneous height as depicted in Figure 3.1. We will see that this distribution scheme is not only beneficial for sparse matrix vector multiplication, but also simplifies other processes highlighted in the following chapters. For further details on how the back end libraries handle data distribution, parallelization and optimization, the reader is referred to the associated literature, e.g., [Kre+16; Kre+17; Thi+19; Thi+16] and the references therein.

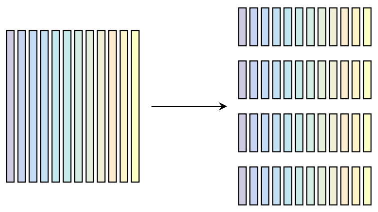


Figure 3.1: *Subdivision into block rows.*

The rather small size of the reduced eigenproblem makes it possible to solve it redundantly on all processes. For larger numbers of eigenpairs, it might make sense to parallelize this step as well. In this case, even if the distribution differs, the parallelism would be classified as level 0, as well. Note that this is a future feature of BEAST.

Level 1: Right-hand sides The approximate projector is applied to every column of the right-hand side vector block (the initial guess or the approximate eigenvectors) in the exact same way. Subsets of vectors can therefore be handled independently, see Figure 3.2. In case the number of processes exceeds the scaling potential of the projector application, more processes can be used efficiently by limiting the number of processes for the projector application this way. We call the process

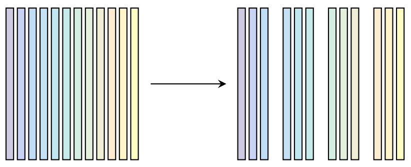


Figure 3.2: *Subdivision into blocks of right-hand sides.*

groups responsible for applying the approximate projector to a vector subblock *prides*. Choosing the data layout identical for each group, meaning that the number of processes per group is identical, the distribution of right-hand sides and the collection of results can be performed as `scatterv` and `gatherv` operations (see

[MPI15]). This communication is performed between related processes of every group, i.e., processes with the same inner-pride rank. We will refer to process groups of this orthogonal group structure as *coalitions*.

Level 2: Rational approximation poles For the evaluation of rational matrix functions such as those from Chapter 2, the solution of a linear system is required for each pole of the function. The system matrices differ depending on pole location while the right-hand sides are identical. By broadcasting the right-hand side vector block to other groups of processes, linear systems are solved independently using a suitable sparse direct or iterative solver. Figure 3.3 shows the half-contour and independently computable poles for an interval.

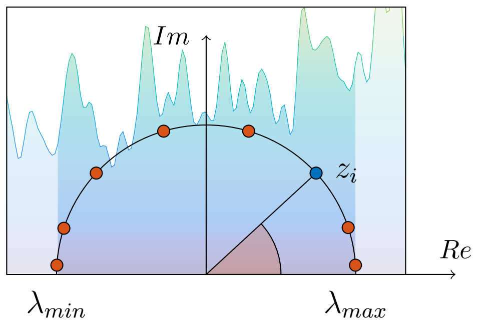


Figure 3.3: *Individually computable poles (Cauchy, degree* 16*); in the background an exemplary spectral density of the interval is shown, see also Figure 3.4.*

The different solutions then have to be accumulated. If the data distribution layout is identical for every group, i.e., the number of processes (or subgroups) is identical, the summation can be performed as reduction operation (see [MPI15]). We will call process groups that handle linear systems for a single pole, or a list of poles, *factions*. In case a single pole is assigned to each faction, preliminary steps such as constructing the system matrix or performing a factorization have to be processed only once. For a list of poles, if all the factorizations cannot be held in memory, refactorization is required in every iteration. For filters that do not involve linear system solves, such as the polynomial Chebyshev filter from Section 2.1, this level of parallelization has to be omitted.

Level 3: Intervals Since we deal exclusively with Hermitian matrices and definite pairs, eigenvalues are real and the spectral region of interest is an interval. Subdividing the complex plane in case of non-Hermitian matrices, however, can be achieved with the help of custom contours for Cauchy filters [PK15; Pol20]. Using other filters that are not based on an integration contour is possible, but involves overlap between the intervals. In Section 5.3.9 we explore a method for handling overlap and duplicates that could be extended to the complex plane, but that will be a task for the future and not be pursued further in this work.

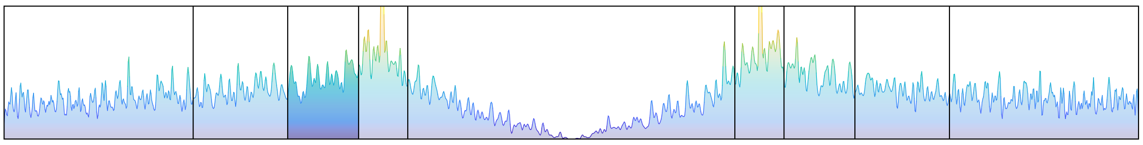


Figure 3.4: *Exemplary interval subdivision; not based on actual density. The interval from Figure 3.3 is highlighted.*

The first step for a multi-interval setup is the subdivision of the target interval. The choice of the intervals themselves is not trivial; all process groups should be assigned approximately the same basis size. Therefore, for the sake of load balancing, any information about the spectrum may be used to decide on the subdivisions. Estimating the density of the spectrum using the kernel polynomial method (KPM) [Wei+06] is a good method for a low resolution spectral overview in case $B = I$ (see also [LSY16] and [XLS18]). Figure 3.4 shows a possible inhomogeneous subdivision based on spectral density. The KPM interprets the spectrum of a matrix as superposition of Dirac delta impulses. The resulting function is approximated by a polynomial, not unlike the filter approximation from Section 2.1. Since, with the spectral function being a sequence of delta impulses on the eigenvalues, the computation of the inner products for the polynomial coefficients is equivalent to computing the trace of the matrix polynomial, those terms of the coefficients can be approximated by stochastic trace estimations of the matrix polynomial. Once the approximate coefficients are obtained, an inverse discrete cosine transform (IDCT) is employed to find a representation of the approximated spectral function. Since the resolution typically is low and the detail is blurred over the real axis, single eigenvalues are not separated, and we instead obtain the estimated spectral density, or density of states (DOS). For generalized eigenproblems, the solution of linear systems is required [FTS10; DPS16] to obtain similar information. Overlap also plays an important role (see Section 5.3.9). A flexible mode for asynchronous scheduling of many intervals can alleviate the impact of uneven workload distribution.

The interval parallelization layer subdivides the processes into groups, which we will call *packs*. There are no constraints for the number of processes per pack, contrary to prides and factions. BEAST implements two strategies for handling multiple intervals. For the static subdivision approach, the number of intervals and the number of process groups is identical. Each group handles exactly one interval. The dynamic approach manages many intervals in a manager-worker style mechanism. One process, the *tamer*, coordinates a number of packs, distributes intervals and the required information, keeps track of the overall progress, and decides how to handle the results, based on the information returned by the packs. This also allows for dynamic reaction to unforeseeable events, such as subspace overfill due to large clusters, empty intervals, or other conditions which require additional measures. The subdivision of an interval, in case the maximum number of vectors was surpassed without reaching the number required for the subspace, for example, can be met with subdivision of the original interval, such that the spectrum is split and the respective searchspaces may now be large enough to compute both intervals. This process can be repeated for the subintervals in case the number of vectors is still too small. In the worst case, this method can at least isolate the problematic cluster and report the respective spectral region as incomputable with the given resources. For this, a certain maximum number of subdivisions serves as terminator. In case the number of vectors can be expanded, because the overall available memory is not yet depleted, it is, of course, more efficient to perform this expansion on-the-fly, see Section 4.2.1.

3.1.2.1 Process group hierarchy

Since it is not required to run the core algorithm on multiple groups redundantly, the different process groups are organized in a manager-worker relationship where the first pride of the first faction runs the core algorithm, provides data and information to the subgroups, and gathers the results. A second level manager-worker scheme is given by the dynamic interval assignment described above. Every group type can be of size one, effectively disabling the respective parallelization layer. Figure 3.5 illustrates the process group hierarchy.

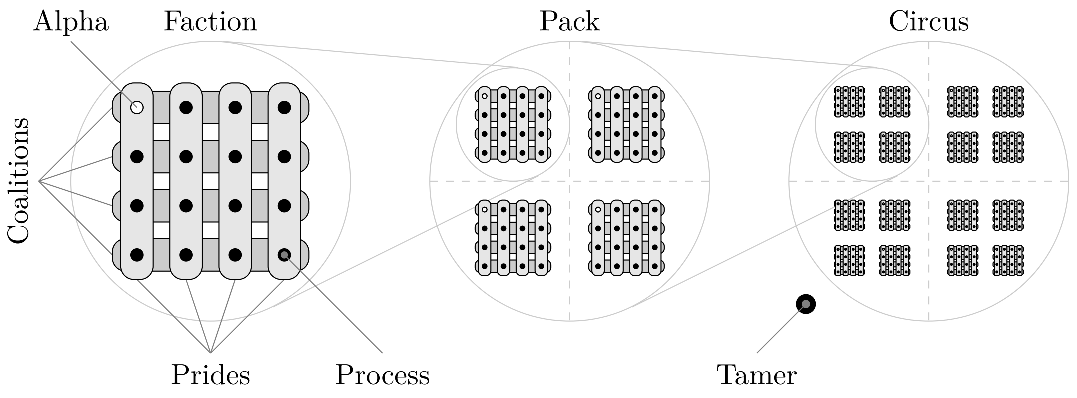


Figure 3.5: *Process group hierarchy. Note that the packs do not necessarily need to be of the same size. If the total number of processes does not meet the constraints for factions and prides, some processes might be left unused. For filters without poles, the faction-layer is omitted; packs and factions are identical in this case. For static interval parallelism, there also is no tamer process.*

Distributing and gathering information to and from the different process groups and subgroups involves fan-out and fan-in style communication patterns, where information is initially available on the first process or process group (we will call them alpha processes or alpha groups), which then relay the information to their group members of alpha processes of the subgroups, possibly subdividing the data in the process. Gathering information reverses this approach. For parallelization level three (intervals), there is no additional function to alpha processes or alpha process groups other than implementing multi-stage broadcasts, scatter and gather operations, or reduction operations. Here, only alpha processes of the packs directly communicate with the tamer. The tamer does not have to take care of providing the information to the whole pack, this is handled by the alpha processes in the background in a separate step. This strict separation of groups and duties eases implementation and should not have detrimental effects on performance. In case of the levels two and one, however, the alpha pride of the alpha faction is the process group running the core algorithm and therefore source for the distribution of right-hand sides and target for gathering the subspace basis vectors. This communication occurs inside the coalitions and between related coalitions of different factions. We can think about prides, coalitions, and factions as three-dimensional process grid, where different stages of the communication are performed along two of the three dimensions (coalition and faction; communication along the pride dimension is restricted to basic operations, i.e., level zero).

The data distribution for factions and prides is performed in unison. Reversing the order of operations, from first broadcasting data from the alpha pride of the alpha faction (which is running the core algorithm) to the alpha prides of all other factions followed by a scatter operation where every alpha pride provides data to other prides of the faction, to first scattering data withing the alpha faction followed by a broadcast from every process of the alpha faction to the associated processes in all other factions. This allows to bypass the storing of intermediate vector block with full number of columns on the alpha prides. Similarly, when accumulating the results, performing the reduction across all factions to the alpha faction first makes an intermediate full-size vector block unnecessary. An additional communication mode using MPI inter-communicators can drive the workers for exclusive projector application while the alpha pride running the core algorithm remains separated and does not participate in projector computations such that no additional memory is consumed. This can be of use if memory is not sufficient otherwise.

Inter-pack communication uses special implementations of send and receive algorithms for communication between (disjoint or overlapping) process groups of different size. This pattern is covered via an MPI all-to-all communication, either using intra- or inter-communicators, but many communication pairs would get assigned a communication volume of zero. In most cases, one process in one side is paired with at most two processes on the other side. The implementations handle subdivision and reassignment of local data blocks and non-blocking communication steps to prevent serialization due to message dependency chains.

3.1.2.2 Interval communication protocol

BEAST uses a fire-and-forget messaging system, i.e., the very small messages are sent in non-blocking fashion and the request handle is stored in a data structure that is periodically checked for completed messages. This holds for all types of flow control messages. Figure 3.6 gives a simplified overview over the basic program flow. The tamer manages a queue of unprocessed intervals and a dynamically linked data structure for the chain of intervals with proper ordering. If unprocessed intervals are available, the interval information is sent (pitched) to the pack alphas, which distribute it to the rest of the pack (the pack following). The tamer can feed the

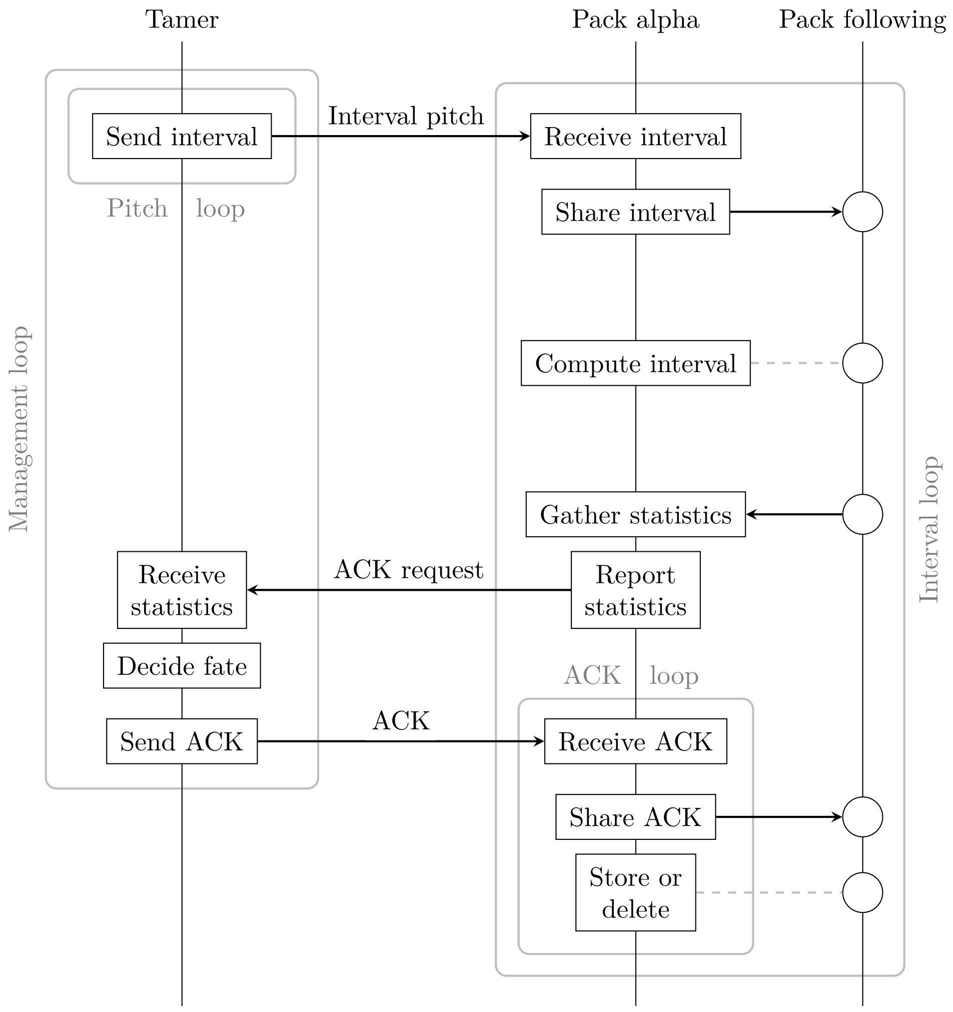


Figure 3.6: *Simplified dynamic interval parallelism protocol. ACK (acknowledge) messages consult the tamer to decide how to proceed with a given interval's result. Not shown is the termination protocol, which also handles potentially remaining ACKs. All messages are sent in non-blocking fashion, while the main loops continue, such that there are no additional waiting times and the relation between result and ACK on tamer and pack has to be encoded in the messages.*

packs in this way to maintain a given interval fill level; typically one interval at a time is enough. Information on the completed intervals are sent back to the tamer since comprehensive knowledge of the specifics of all intervals and their relation is held solely by the tamer. Once the statistics of a completed interval are received by the tamer, it decides how to handle the results. Possible options are store, forget (delete), split, retry, and expand to react to several conditions, some of which have been mentioned above. The verdict is transmitted to the pack which implements the order. The tamer's management loop runs at high frequency.

Since the pack alphas only receive one interval, the MPI message buffer has to hold the (potentially) pending incoming interval messages. Once the interval computation finished and after possible additional information is gathered from the pack following, the results are temporarily stored and the tamer is consulted for a decision on how to proceed. However, only the request is issued in a non-blocking way and the system continues with the next interval. Messages received from the tamer that hold information on what to do with a set of results (ACK) are checked periodically. During such a check, as many ACKs as possible are processed and the required action is transmitted to the rest of the pack.

In all cases, a pointer to the respective element of a data structure is passed along to prevent having to search data structures for every message received. Transmitting pointers in a shared-memory context usually raises eyebrows and several alarms in the programmer's mind. Understandably so, given that pointers only have meaning on the specific process that allocated the memory they are pointing to and are completely meaningless on every other process. That is, until the communication protocol ensures that they are sent unmodified, e.g., as a sequence of bytes, back to the process they originated from to identify an object without having to traverse an additional data structure or use a look-up table and are never dereferenced anywhere else. This, of course, requires that the maximum length of addresses on all participating processes must be known, which can easily be determined in an initial communication step, before the associated datatype is created. Relying on transferred pointers in particular means that a result stored on the pack is temporarily forgotten until the respective ACK message reminds the pack of where to find the result. In the context of fault tolerance, this behavior is only semi-safe, given that a pointer can potentially be lost at any time, even though the error might be recoverable. A local list of intervals acts as safety net, which, under normal operation conditions, never has to be traversed to find an interval, other than at the very end and, of course, in case of an error condition. A result is removed from the list of pending results and added to the list of finished intervals when an ACK is received from the tamer and the order is to store the result. The results are currently permanently stored on the alpha pride of each pack.

3.1.3 Orthogonalization layer

The concurrent computation of intervals can be performed completely asynchronous, but it is a well-known fact for independently computed eigenvectors to suffer from bad orthogonality, depending on the gap between eigenvalues of different intervals and other factors (see [GKL11; Krä+13; Krä14] and Section 4.5).

An orthogonalization phase (currently only for static interval parallelism) can be activated on the faction and pride layer to ensure orthogonality of eigenvectors. The topic is outlined in much more detail in Chapter 5.

3.2 Meet the BEAST – short feature overview

Some features included in BEAST will be mentioned in later sections or at least have some relation to the topic. Others will play no major role for the remainder of this work. We will shortly outline both.

3.2.1 Related features

Control over and optimization of the number of iterated basis vectors is ensured by multiple components. On the one hand, the number of iterated vectors can be reduced by locking single, already converged vectors in later iterations, accompanied by continuous orthogonalization [Krä14], see also Section 4.2.2.2. On the other hand, an SVD-based analysis of the Rayleigh quotients provides an association of the iterated eigenpairs with filter values and, with this, an estimation of the number of eigenpairs actually contained in the target interval. This allows for a careful assessment of the number of vectors required against faster convergence speeds [GKL12; Gal+14; Krä14], see also Section 4.2.2.4 and Section 4.2.2.5. In particular the latter component is indispensable from a numerical point of view to prevent rank deficient vectors sets for use in the Rayleigh-Ritz procedure [GKL11; Krä+13; Krä14], see also Section 4.2.2.1 and Section 4.2.2.3. Increasing the number of vectors does not require tamer interaction (or a restart in single interval contexts). In case the number of iterated vectors is identified as being to small, their number is allowed to grow towards a user-specified maximum by appending additional vectors.

For rational filters on integration basis, an iterative implementation of the Sakurai-Sugiura method adds the possibility of reducing the number of right-hand sides during projection in favor of using additional moments [Hub+; Fut+17; Fut+18; Fut+19; GHL18a]. The number of linear systems remains unchanged. See also Section 2.5 and Section 4.2.3.

Typically, operations performed in single precision are significantly faster than operations performed in double precision; for an algorithm which can be partially executed with lesser accuracy without impact on the result, a corresponding performance increase can be expected. In the case of subspace iteration, as used in BEAST, this applies to the early iterations. Therefore, the option to start the algorithm in single precision and automatically switch to double precision as soon as it is necessary is included in BEAST. Single precision execution may be possible only for few iterations but an improvement of the time-to-solution without impact on the final result is generally possible [GHL18d; Alv+19; Wel+20], see also Section 4.3.1.1 and additional results in [GHL18f; GHL18c; GHL18e; GHL18a].

3.2.2 Linear solvers

The choice of a suitable linear solver has always been a difficult one (see also Section 4.2.7). The linear systems arising in the context of rational filters for subspace iteration have proven to be of utmost difficulty, depending to some degree on the distance to the eigenvalues of the eigenproblem. This difficulty is restricted to iterative solvers, which are sensitive to these conditions [GKL12; Krä14], see also Section 4.2.7. Direct solvers have fewer issues in regard to ill-conditioning, but require more memory and become increasingly infeasible with growing system size.

An iterative algorithm, that is robust also for singular systems, is the CARP-CG algorithm [GG10], the conjugate gradient method [HS52] with blocked Kaczmarz row projections and averaging (CARP) [GG05] as preconditioner. An implementation is provided by PHIST and used as a solver option in BEAST. Results achieved with this solver can be found in [Gal+15].

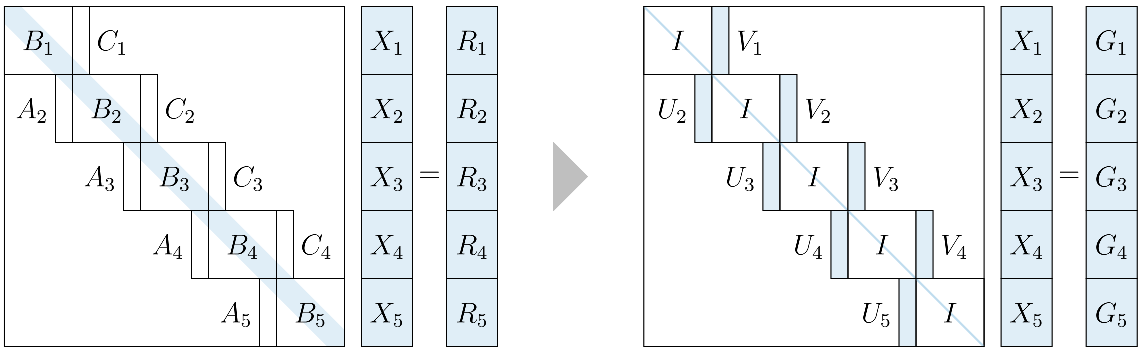
3.2.2.1 Callback interface

To allow simple integration of different linear system solvers, BEAST contains a callback interface consisting of functions that are invoked when entering or leaving key stages of the program flow, e.g., setup and termination phase, when switching to a new shift/pole, when changing floating-point precision, or when the actual linear system solve is required. The interface will be extended further in the future. Solvers for which an implementation of the interface is included currently are the STRUctured Matrices PACKage (STRUMPACK) [Ghy+18] and MUMPS [Agu+20]. STRUMPACK matches the data distribution scheme of BEAST very well, while MUMPS requires gathering the right-hand side vectors on one process.

3.2.2.2 A (hybrid) parallel direct solver for banded linear systems

Assuming that, for achieving reasonable convergence rates, an incomplete LU decomposition preconditioner requires a non-negligible amount of fill-in, for a matrix with narrow banded structure filling its envelope almost completely, a direct banded solver becomes a viable alternative. Such a solver would require not much more memory in the scenario described above and a blocked parallelization for enabling level-3 BLAS operations allows for reasonable node-level performance. Employing a multi-threaded BLAS implementation makes this approach hybrid parallel. A suitable block-parallel algorithm, whose data distribution pattern matches the one used in BEAST well, is described in [PS06] and based on [SK78; LS84]. BEAST includes a rudimentary implementation of the basic algorithm as a fall-back solver, should other solvers fail.

Assuming a set of p processes, decomposing a banded system matrix M into p block rows of not necessarily equal size to solve a linear system $MX = R$ yields the form shown in Figure 3.7, left. Let the bandwidth of the system matrix and the width of the block columns A_i and C_j, $i = 2, \dots, p$, $j = 1, \dots, p-1$, be n_b. Let further the matrix D be the block-diagonal matrix, consisting of the square blocks B_k, $k = 1, \dots, p$. The inverse of D, D^{-1}, is the block diagonal matrix consisting of the square blocks B_k^{-1}, $k = 1, \dots, p$. In the modified system $D^{-1}MX = D^{-1}R$ (Figure 3.7, right), it is $U_i = B_i^{-1}A_i$, $V_j = B_j^{-1}C_j$, and $G_k = B_k^{-1}R_k$. Each block row couples only with variables of itself or the n_b closest rows of adjacent block rows, i.e.,



B_1 C_1 A_2 B_2 C_2 A_3 B_3 C_3 A_4 B_4 C_4 A_5 B_5
X_1 X_2 X_3 X_4 X_5 = R_1 R_2 R_3 R_4 R_5
I V_1 U_2 I V_2 U_3 I V_3 U_4 I V_4 U_5 I
X_1 X_2 X_3 X_4 X_5 = G_1 G_2 G_3 G_4 G_5

Figure 3.7: *Block-wise transformation of banded matrix.*

the last n_b rows of the block row above and the first n_b rows of the block rows below (if they exist). Adopting the notation from [PS06], let us denote the first and last n_b rows of X_k as

$$X_k^{(t)} \quad \text{and} \quad X_k^{(b)},$$

respectively. Analogously, define

$$G_k^{(t)},\ G_k^{(b)},\ V_k^{(t)},\ V_k^{(b)},\ U_k^{(t)},\ \text{and}\ U_k^{(b)}$$

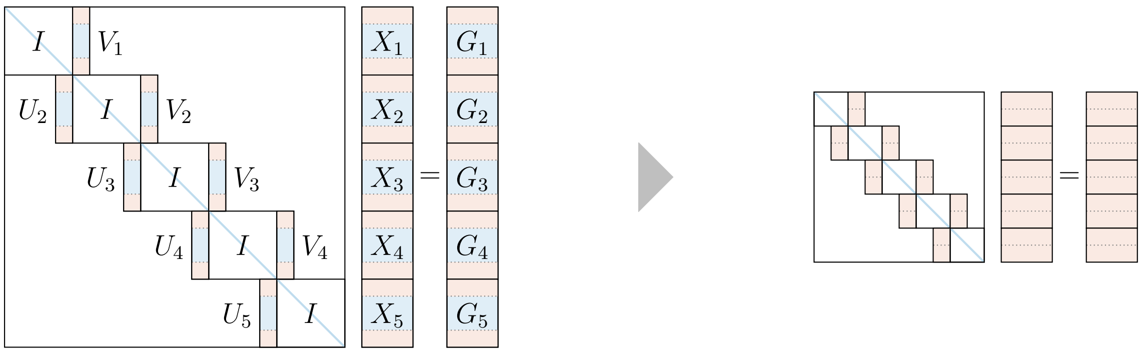
as the top or bottom n_b rows of the respective block. Additionally,

$$U_1^{(t)} = U_1^{(b)} = V_p^{(t)} = V_p^{(b)} = \mathbf{0}.$$

Among the rows of block row k, the first and last n_b rows couple only with

$$X_{k-1}^{(b)},\ X_k^{(t)},\ X_k^{(b)},\ \textit{and}\ X_{k+1}^{(t)}.$$

This allows for computing the respective components of X from just these equations. More precisely, a linear system of reduced size may be formulated (by renumbering the unknowns) as pointed out in Figure 3.8, where the block rows now have the form



I V_1 U_2 I V_2 U_3 I V_3 U_4 I V_4 U_5 I
X_1 X_2 X_3 X_4 X_5 = G_1 G_2 G_3 G_4 G_5
=

Figure 3.8: *Reduction to smaller system.*

$$\begin{bmatrix} U_k^{(t)} & I & & V_k^{(t)} \\ U_k^{(b)} & & I & V_k^{(b)} \end{bmatrix}, \quad \begin{bmatrix} X_k^{(t)} \\ X_k^{(b)} \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} G_k^{(t)} \\ G_k^{(b)} \end{bmatrix}$$

for matrix, result and right-hand side, respectively. The resulting system of linear equations is block tridiagonal and again has p block rows. Contrary to [PS06], we dot not remove the rows corresponding to $X_1^{(t)}$ and $X_p^{(b)}$ from the reduced system as to not having to redistribute data for a subsequent parallel solution of the reduced linear system, although they can be computed from $X_2^{(t)}$ respectively $X_{p-1}^{(b)}$ since

$$X_1 = G_1 - V_1 X_2^{(t)} \quad \text{and} \quad X_p = G_p - U_p X_{p-1}^{(b)}.$$

The reduced system may be solved in any way, direct or iterative (if applicable). If n_b is reasonably small, a blocked version of recursive doubling, see e.g. [HJ88], may be used, which is currently the case for the implementation found in BEAST.

Once the solutions

$$X_k^{(t)} \quad \text{and} \quad X_k^{(b)},$$

are found, the remaining parts of X can be computed via

$$X_k^{(c)} = G_k^{(c)} - U_k^{(c)} X_{k-1}^{(b)} - V_k^{(c)} X_{k-1}^{(t)}$$

since

$$U_k^{(c)} X_{k-1}^{(b)} + X_k^{(c)} + V_k^{(c)} X_{k-1}^{(t)} = G_k^{(c)},$$

where

$$U_k^{(c)},\ V_k^{(c)},\ X_k^{(c)} \text{ and } G_k^{(c)}$$

are the remaining center blocks of U_k, V_k, X_k, and G_k.

The basic algorithm is rather old and many improvements have been made over time, see for example [PS06; SPS18]. Implementing more efficient versions or including existing implementations via the callback interface from Section 3.2.2.1 will be a task for the future. We end the discussion of the band solver with a small scaling test, Figure 3.9. The strong scaling limit is reached at 64 processes already; it is likely that we also see the limits of the recursive doubling solver for the reduced system here. Performance does not change linearly with the number of right-hand sides and parallelization over blocks of vectors is not as efficient as parallelization over rational poles or intervals. The latter two are the proper method to improve scaling when the limit of the solver in use is reached.

3.2.3 Unrelated features

Optimization potential of the filters themselves is exploited by filter dependent heuristics and extensions. For filters based on polynomials or integration, adaptive control of the polynomial degree or integration order enables optimization of the time-to-solution by choosing suitable parameters [GKL18]; *(B. Lang, personal communication)*. Additionally, the dampening properties of polynomial filters can be significantly improved by a retroactive heuristic optimization of the coefficients, such that the same speed of convergence can be achieved with lower polynomial degree [Gal+17]; *(B. Lang, personal communication)*.

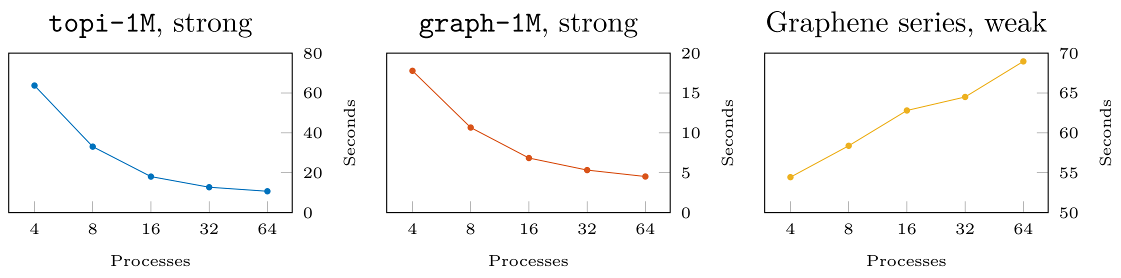


Figure 3.9: *Scaling behavior of the banded solver. Strong scaling for the matrices* `topi-1M` *(bandwidth* 524*,* 300 *right-hand sides, complex values) and* `graph-1M` *(bandwidth* 247*,* 105 *right-hand sides, real values). For the weak scaling, the matrix size was doubled each step, starting at* 1 048 576*, up to* 16 777 216*, such that rows per process remain equal. Number of right-hand sides* (256) *and bandwidth* (512) *were kept constant. The timings are averaged over* 8 *different systems for graphene-type matrices and* 16 *systems for the topological insulator* `topi-1M`*. The tests were performed on the Emmy HPC cluster at Friedrich-Alexander-Universität Erlangen-Nürnberg.*

On machines with many components, an execution halting hardware fault or faulty computations due to hardware defects have to be expected. To counteract the severity of these cases, options for detecting incorrect results and to save the progress of the computations are integrated. An extension of the right-hand sides during application of the approximate projector by linear combinations of regular basis vectors allows to check results after or during computation of the projection *(Suggested, B. Lang, personal communication)*.

The most time-consuming part of the algorithm is the application of an approximate projector P to a set of right-hand sides Y,

$$U = P(A, B)Y.$$

Let $L(Y)$ be any linear combination of the columns of Y and choose $\tilde{Y} = [Y \;\, L(Y)]$. Then

$$\tilde{U} = P(A, B)\tilde{Y} = P(A, B)[Y \;\, L(Y)] = [\underbrace{P(A, B)Y}_{=U} \;\, \underbrace{L(P(A, B)Y)}_{=L(U)}].$$

$L(U)$ can act as checksum for detecting errors in the computation of the projection if $L(Y)$ is appended as additional column to Y. BEAST supports appending and checking checksum columns at various stages throughout the layers of the system.

With every checksum column being a linear combination of columns of Y, in vary rare cases errors may cancel each other out in the representation of the checksum, leading to undetected corruption of the results. To further reduce the probability of such cancellations, more than one column may be appended. If each column is produced by a different linear combination, for errors to cancel out in all representations is far less likely.

The process subdivision scheme from Section 3.1.2 results in a total of four levels of checksum creation and verification; an additional level is given by the non-parallel blocking of right-hand sides [Pie+16]. With level 0 representing no checksum verification at all, the five levels are:

Level 0 No checksums.

Level 1 Checksum columns are appended once per interval, i.e., checksums are generated and verified before and after the complete application of the approximate projector in each iteration of the eigensolver.

Level 2 Checksum columns are appended by every pride to their portion of right-hand sides.

Level 3 Checksum columns are appended by the factions to their copy of the right-hand sides, i.e., for every pole of a rational filter. This is a verify-only step, since the right-hand sides for all groups are identical and now new vectors can be added during accumulation.

Level 4 Applying the approximate projector to the columns of Y can be performed (sequentially) in a series of subblock columns to reduce the memory consumption for certain linear solvers or to optimize data access patterns. Checksum columns can be added for each such block.

Level 5 For polynomial filters, fine-grained checksum verification can be performed every k-th degree of the approximation, i.e., every k-th multiplication with the system matrix, to reduce recomputation overhead in case or errors. Appending additional checksum columns is not possible. High-frequency verification, of course, increases the overhead as well.

The ordering of levels two and three, pole and right-hand side parallelism, is encouraged by the necessity to transform and transfer results of potential factorizations produced by linear system solvers to smaller process subsets, were they reversed. The distribution of these kinds of data may be complicated, solver-specific, or even completely opaque. The layers are implemented in this way, even if the logical hierarchy from Section 3.1.2 reverses these two layers.

While the checksums are generally verified on the same layer they were generated, a given checksum column may be verified at any layer. That in particular means that level five can be combined with any higher level without adding additional columns, while a combination of higher levels, apart from being redundant, produce multiple checksum columns. Also note that even lower levels of checksum monitoring are not possible since all intervals apply different projectors.

This checksum method detects all kinds of computation errors, however, small errors in early iterations do not seem to impact convergence. In general, one-time errors are often handled best by simply proceeding with the computations, ignoring the error.

Additionally, the plausibility of the results can be inexpensively ensured by analyzing the aforementioned SVD-based filter estimation *(B. Lang, personal communication)*, see also Section 4.2.2.4. The appearance of values that are larger than expected based on the known filtering function indicate an error either during projection or computation of the SVD.

With the inclusion of CRAFT, a library for application-level checkpoint/restart and automatic fault tolerance, [Sha+19], a system to save and restore computation progress in and from persistent memory is provided. With this, the program can be resumed after an unforeseeable halt of the execution. This mechanism is currently only implemented on the single interval level, but will be extended to multi-interval scenarios.

3.2.4 Layer structure

With its several features, stages, and hierarchies, BEAST is structured in nested layers, where each layer handles a certain mechanism and hands control down to deeper layers. The outermost layer is the type-aware user interface, where matrices are explicitly passed as single or double precision, real or complex datatypes. The next layer is the type-oblivious pack subdivision and interval parallelization layer, where data is passed along with no need for explicit knowledge about the datatypes, followed by the faction and pride subdivision layer that separates worker process groups. The interval subdivision layer is also responsible for the final orthogonalization phase. A datatype layer reintroduces datatypes and manages changes in floating-point precision, e.g., from single to double precision, mid-iteration. Due to similarities in requirements, loading checkpoints and resuming a previously aborted run of the eigensolver is also part of this layer. The core algorithm is not quite the innermost layer, despite its name. It includes the management of projection parameters and the number of vectors (including communicating this information to other processes), the Rayleigh-Ritz process, and writing of checkpoints, among other elements. After the algorithmic layer follow layers for the most important and most expensive part of the algorithm, the application of the approximate projector. These layers provide orthogonalization over prides, then over factions (see the rationale above), finally over blocks of right-hand sides (non-parallel for reducing memory consumption and improvement of data access patterns, as has been mentioned above). In case certain layers are disabled or not available, they are skipped, passing control to lower layers immediately.

3.3 Feeding the BEAST – Applications

While BEAST is intended to be a general purpose solver, not tailored to a certain application specific problem class, most results so far have been gathered from the eigenproblems produced by graphene modeling and the simulation of topological insulators. Results for problems from modeling of graphene nanoribbons were presented in [Gal+14; Wel+20]. For problems stemming from topological insulators as well as the scaling behavior of the polynomial filter, for which highly specialized kernel routines were implemented in GHOST and PHIST, see [Wel+20] as well as [GHL18e; GHL18b].

Chapter 4

Taming the BEAST – Quality of results

For the computational solution of eigenvalue problems, two important metrics for the success of such an endeavor exist. Determining the error of the computed eigenpairs requires the knowledge of the exact solutions and is, in general, unobtainable. Estimations and upper bounds for the error have been given in Section 1.6.3, based on the residual of the computed eigenpairs, a replacement for the error metric that can be computed from the approximate results found by the eigensolver and constitutes a metric describing how well this approximate solution solves the eigenequation. The second metric may be less important for some applications, but can be crucial for others. As has been established before, Hermitian matrices have orthogonal eigenvectors and eigenproblems of generalized form for definite matrix pairs produce B-orthogonal eigenvectors. Mathematically, two vectors are either orthogonal or not. Computationally, two vectors may be more or less close to being orthogonal, measured by the angle between them. This *orthogonality* of the computed set of eigenvectors can be determined from the computed approximate solution. Applications may rely on eigenvectors being orthogonal or being close enough to orthogonality, which is why finding the orthogonality of a computed set of eigenvectors is important and ultimately ensuring a certain degree of orthogonality is worthwhile.

This chapter will shed light on the convergence behavior of subspace iteration using the introduced approximate spectral projectors, in particular the achievable residual and orthogonality that can be expected from a subspace iteration based iterative eigensolver and explore ways to guarantee a predefined degree of orthogonality. This will be accomplished by several numerical experiments under controlled conditions. To this end, suitable matrix pencils will be generated from a predefined spectrum and with known eigenvectors. The filtering operator can be built explicitly to bypass possible computational errors during conventional application and is then fed to a bare-bones spectral filtering implementation. A collection of real-world matrices from different fields serves as example to confirm results for eigenproblems from actual applications.

Eigenprose We casually talk about eigenpairs, eigenvalues, eigenvectors, and eigendirections as connected entities, often relating properties of one to the others. So far, for the reader to make this connection has been tacitly assumed.

To an eigenvalue is connected, in our case, a unique eigendirection, represented by an infinite number of possible positive-length eigenvectors which differ only in scaling and/or sign in case the eigenvalue is distinct. In case of an eigenvalue with multiplicity larger than one, an appropriate basis for the eigenspace can be found, and we may relate one basis direction to each of those identical eigenvalues. An eigenvalue and one of its associated eigenvectors form an eigenpair. An eigenvalue resides at a spatial location in the complex plane, or along the real axis in the case of Hermitian matrix pencils. By relation, we may speak about the position of eigendirections or eigenvectors, adjacent eigenpairs or eigendirections, and being located inside the target interval or outside of it, implicitly always referring to the property of the associated eigenvalue. Similarly, we may refer to dampening of eigenvalues, eigenpairs, or eigenvectors, while any dampening applies chiefly to the concept of eigendirections and secondarily to associated eigenvectors at most.

Additionally, new terminology shall be introduced here to ease the discussion that follows.

Searchspace The space whose basis is computed during the Rayleigh-Ritz procedure, which may also be chosen smaller or larger than the targeted amount of eigenpairs, will be called the *searchspace*. Synonymously, we might refer to the basis representing the searchspace itself as a searchspace. The set of vectors that represents the searchspace during subspace iteration and is used for computing the Rayleigh quotients will be called *searchspace vectors* or *searchspace set*. The size of the searchspace is the number of vectors in the searchspace set and the maximum dimension of the searchspace.

Slot For the discussion of the convergence behavior in this chapter, we require a more abstract notion of how the searchspace basis relates to the eigenvectors that are to be approximated. In particular, this terminology is intended to separate the eigendirections that the iterative method will approximate from those it will not. The concept thus is somewhat projective and relies on the assumed invariability of the convergence process in terms of which directions will be approximated by the algorithm. A searchspace set (and the corresponding searchspace) of dimension d will be able to approximate at most d eigendirections. Following the rationale from Chapter 1, this approximation of eigenvectors is achieved by rotation of the searchspace to align with the space spanned by some eigenvectors of the matrix pencil. We will refer to this concept of possible alignment with an eigendirection as *slot* of the searchspace, often—but not necessarily—corresponding to a vector of the searchspace set. However, the mere possibility for the searchspace to accommodate d eigendirections in this manner does not yet guarantee that it will align with d eigendirections over the course of the algorithm. Since for each dimension, the searchspace allows for one eigendirection to be (approximately) contained in it or there being no clear relation to an eigendirection at all, we refer to a slot of the searchspace as being occupied or unoccupied.

Occupation The driving concept behind the rotations of the searchspace is the amplification of the components of the searchspace basis in the directions of some eigenvectors or, from a different point of view, a dampening of all other directions. This modifies the searchspace such that it gradually aligns with the space spanned by the least dampened directions. If the searchspace clearly rotates towards a certain eigendirection such that it will be approximately contained in the searchspace after a number of iterations, we say that the eigendirection (or the associated eigenvalue or eigenvectors) occupies a slot of the searchspace. We may thus also think about occupation as how close an eigendirection is to being contained in the searchspace. What it means for a slot to be unoccupied is not inherently clear yet, but we will see that sometimes certain Ritz values will show no sign of convergence. Furthermore, this binary notion of slot occupation is blurred when dealing with multiple eigenvalues, and we will have to refine this definition at a later point.

Decay A filtering function that approaches zero in the stopband towards the ends of the spectral range of the filtered spectrum (and possibly beyond) will be dubbed *decaying*, as opposed to filters that equioscillate over the stopband. Note that the filter's value may again grow outside the spectral range, as is the case with filters based on Chebyshev polynomials. Therefore, Cauchy-, Polynomial-, Butterworth-, and Chebyshev-Type-I filters are to be considered decaying, while Zolotarev-, Elliptic-, and Chebyshev-Type-II filters are not decaying. The dampening properties of decaying filters improve with increased distance to the target interval.

Cluster Groups of eigenvalues are typically considered *clustered* if there is little or no difference in value. This often reflects the multiplicity of an eigenvalue, but numerically the Ritz values may be segregated, either by floating-point inaccuracies for real multiplicities, or actual (albeit very small) distance. As to what distance should be allowed for eigenvalues in order to still be considered clustered is up to the specific situation and information that is to be gained from it. In our case, we do not need a sharp distinction and a cluster is just a group of eigenvalues more densely packed than the remaining eigenvalues inside a spectral interval of interest.

Saturation While this term may be used for a plethora of different conditions, we will use it here to describe a Ritz pair that has reached its minimum possible residual. In this case, the residual stagnates, possibly varying slightly within a certain small range without further improving or deteriorating. The maximum residual in the range of these variations is called *saturation residual*.

Residual drop rate Sometimes also *residual drop* or just *drop*. The change in residual before and after the application of the projection operator and application of the Rayleigh Ritz procedure, i.e., the change in residual from one iteration of a subspace iteration eigensolver to the next, assuming that both Ritz values can be associated with the same eigenvalue.

Dominance This term has been used before and its semantics are intuitively clear. Globally, dominance can be understood simply as the value of the filter function at the position of the associated eigenvalue. Dominance of one direction over other directions describes the ratio of the values of the filter at the respective positions. It therefore describes by how much a direction is dampened during the application of the filter, possibly in relation to other directions. The most dominant directions in a global sense will occupy the searchspace and converge the quickest.

Eigencount The number of eigenvalues contained in a given spectral region, e.g., the target interval of a subspace iteration eigensolver for Hermitian matrix pencils.

4.1 Experiments

The analysis in this chapter is largely experiment-driven. Many factors may play a role in the speed of convergence and the final quality of the results that can be obtained from a subspace iteration algorithm. A non-exhaustive list includes spectral distribution of the matrix pencil, initial guess for the eigenvectors, filtering function, or numerical errors emerging in the different steps of the algorithm and in particular during application of the projection operator, which typically involves either many matrix-vector multiplications or several linear system solves or even both. A framework to explore the properties of subspace iteration would require bypassing the most important aspects on this list to highlight the influence of others. An ideal filtering function, e.g., requires the knowledge of the eigenvalues of the matrix pencil and its application requires the eigenvectors. Similarly, circumventing linear solves or many matrix multiplications in the application of an approximate filter requires the explicit construction of the projection operator matrix. The filtering functions available from Chapter 2 have different properties that we may easily exploit using this method. To enforce certain spectral properties, the spectrum of the matrix pencil would have to be predefined. To this end, many matrix pencils used in the experiments have to be generated in a way that mimics real-world problems.

4.1.1 Synthesis of non-trivial sparse definite matrix pencils with predefined spectrum

For the following experiments, we require matrices with well defined spectral properties to dissect the correlations between them and the results' quality criteria. For this purpose, we aim to generate sparse matrices, real or complex, from a given spectrum Λ where A and B are required to be symmetric or Hermitian and B additionally is required to be positive definite, all while keeping the spectral deviation from the target spectrum as small as possible. For the real case, this is easily accomplished via diagonal matrices, such that

$$A_d = B_d\Lambda$$

holds. Choosing B_d randomly (and real) and shifting its entries appropriately yields a diagonal matrix A_d, such that the matrix pencil (A_d, B_d) has the desired properties. Diagonal matrices are computationally beneficial to a degree that they do not reflect the behavior of real-world examples in all cases. This in particular refers to the numerical errors introduced during computations and their impact on the residual. As we will see in Experiment 4.17, the sparsity of a matrix has an effect on the residual that can be achieved, even if the impact remains rather small. The number of floating-point operations also seems to affect the variations in residual among the converged Ritz values. This can make the evaluation of residual bounds difficult, and we may be tempted to believe that a smaller residual can be achieved. While we aim to bypass certain sources of errors, in particular those introduced by the application of different types of filter, the inherent consequences of numerical computation must not be ignored if we are interested in practical results. Rarely will a real-world matrix be diagonal. Furthermore, experiments considering the sparsity of a matrix are not possible at all, if all matrices are diagonal only. We therefore choose a sparsity of $2 \sim 3\%$, values already quite large for a matrix that is considered sparse. Additionally, generating complex valued problems is inherently impossible using this approach. We therefore also require our matrices to have a non-trivial structure. A similarity transformation (see Section 1.2.2), in particular an orthonormal matrix Q, retains the spectrum of the matrix pencil when applied to both matrices,

$$\underbrace{QA_dQ^H}_{=:\,A} = QB_d\Lambda Q^H = \underbrace{QB_dQ^H}_{=:\,B} Q\Lambda Q^H.$$

Remember that $Q^HQ = I$ and A as well as B are Hermitian since A_d and B_d are diagonal. Subspace plane rotation matrices as used for Jacobi (or Givens) rotations, given by a matrix

$$Q_{ij} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\overline{s} & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}$$

with off-diagonal entries only in rows and columns i and $j \neq i$ are orthogonal since

$$\begin{pmatrix} c \\ -\overline{s} \end{pmatrix}^H \begin{pmatrix} s \\ c \end{pmatrix} = 0$$

if c is real and orthonormal if

$$\left\| \begin{pmatrix} c \\ s \end{pmatrix} \right\|_2 = 1.$$

While they typically are used with carefully chosen values for c and s to eliminate off-diagonal non-zeros, we will use them here to introduce off-diagonal non-zeros, allowing for a mostly random choice of c and s. For Hermitian matrices we choose s non-real. Successive application of these transformations with random values for i and j generates a random symmetric sparse matrix structure. The values of j in relation to i may be chosen such that off-diagonal entries are, on average, kept closer to the main diagonal. Since reordering matrices are orthonormal, a band reduction algorithm may be applied. MATLAB [MLB], e.g., supplies the reverse Cuthill-McKee reordering or variations of the approximate minimum degree reordering. Using just the approach as described here, the structure of A and B remains identical, but this is of no concern for the following experiments. In case a structural difference is desired, additional rotations, deliberately chosen to introduce zeros in either A or B (see Section 5.2.2), will typically not also introduce zeros in the respective other matrix. For example, a few rotations to introduce zeros in B will not add zeros in A but instead introduce additional non-zeros.

The eigenvectors X of the generated matrix pencil are given in sparse form by the product of the sequence of applied similarity transformations and by construction are both I-orthonormal and B-orthogonal,[1] but not B-normal. B-orthonormality can be established by scaling with respect to B. The eigenvectors can be used to explicitly compute the matrix operator of an approximate or ideal filtering function. In particular, the filtering properties can be chosen infinitely sharp with constant dampening of unwanted directions.

4.1.2 Matrix test set

Beyond generated matrices with known spectra, a set of test matrices with different sizes will be used to confirm the validity of certain deductions with problems from real applications. Table 4.1 lists name, size, target interval, and eigencount inside the target interval for the respective matrix. Decimal numbers in tables will be

Name	Size	Interval	Eigenpairs
`laser`	3 002	[−5.033651644876957e−01, −2.688920370800423e−01]	1 000
`SiH4`	5 041	[−1.831832550688610e−01, −3.438302367677931e−02]	731
`linverse`	11 999	[−3.049133393553184e−01, −2.350212902114284e−01]	1 739
`Pres_Poisson`	14 822	[−9.511245554968437e−01, −8.811715627407559e−01]	1 746
`Si5H12`	19 896	[−2.042532397249384e−01, −8.673585236243539e−02]	2 250
`brainpc2`	27 607	[+1.848435756778146e−01, +2.343686520490190e−01]	2 792
`rgg_n_2_15_s0`	32 768	[−7.492505037717052e−01, −6.914155231208449e−01]	3 723
`SiO`	33 401	[−2.725881280626155e−01, −1.516415844705570e−01]	4 196
`Andrews`	60 000	[−4.255272272030095e−01, −3.432940637079993e−01]	7 290
`Si34H36`	97 569	[−1.918520641752265e−01, −8.074090437135376e−02]	10 455
`fe_rotor`	99 617	[−3.999986667124474e−01, −3.461078701061172e−01]	10 521
`GraI-119k`	119 908	[+3.571287929836170e−01, +4.956607232057880e−01]	16 118

Table 4.1: *Standard eigenproblem test set "The Dirty Dozen".*

[1] Since $X^H B X = X^H X B_d X^H X = B_d$.

written in e-notation where de$\pm b$ is used as a shorthand for $d \cdot 10^{\pm b}$, e.g., 5.43e−21. To cover a wider variety of eigenproblems, matrices from different backgrounds were included in the test set. The set consists of eleven test matrices from the GHS_indef (`laser`, `linverse`, `brainpc2`), PARSEC (`SiH4`, `Si5H12`, `SiO`, `Si34H36`), ACUSIM (`Pres_Poisson`), DIMACS10 (`rgg_n_2_15_s0`, `fe_rotor`), and Andrews (`Andrews`) matrix groups of the SuiteSparse Matrix Collection (formerly the University of Florida Sparse Matrix Collection [DH11]) and one test matrix from graphene modeling, `GraI-119k` [Net+09], almost all of which were also used in [GKL18]. All matrices have been scaled and shifted for use with polynomial filters, i.e., the spectrum is mapped into $[-1, 1]$, and spectral positions are given with respect to the transformed matrix. These matrices will predominantly be used in the analysis of orthogonalities, the heuristic estimation of achievable orthogonality and the empirical derivation of a suitable orthogonalization order. The default data sets are obtained by subdividing the target interval into 64 sub-intervals which are then processed independently. Two different strategies for subdivision are used; the *even* distribution of eigenpairs produces intervals of different size with equal amounts of eigenpairs per interval, based on the known spectra of the matrices; the *uneven* distribution of eigenpairs produces intervals of equal sizes with varying number of eigenpairs per interval. Both distribution strategies are performed with and without locking (see Section 4.2.2.2), resulting in a total of four data sets.

Experiment 4.1 — Test set calibration
We perform calibration runs with BEAST from Chapter 3 for all combinations of matrix and subdivision strategy, with and without locking. For comparison with orthogonalized or otherwise modified results under different conditions, the range of residuals and orthogonalities achieved in these runs are the main point of interest. Polynomial filters, for which the degree has been adaptively determined on-the-fly [GKL18] for each case separately, was used, and proper searchspace sizes have been deduced explicitly from the known spectra. In all instances, the subspace iteration process is not continued to small residuals to produce more difficult conditions that may better reflect actual applications. The obtained residual ranges for all matrices and data sets are listed in Table 4.2. Orthogonalities are not listed here, but in Tables 5.4 and 5.5 in Section 5.3.8, where they will be of relevance for the first time.

Additionally, two larger matrices that are not listed in Table 4.2, a real matrix of size 1 199 948 from Graphene modeling [Net+09] (`graph-1M`) and a complex matrix of size 1 048 576 from the simulation of topological insulators [HK10] (`topi-1M`) provide very dense spectral regions, as can be seen in the spectral density plots of Figure 4.1, as ultimate test. However, these sizes are by far not the largest problems that have been tackled using subspace iteration, but typically less dense regions are the spectral target for such sizes and only few eigenpairs are computed. The target intervals are marked in the background, but are too small to be visualized as more than a line. A zoomed representation does not reveal additional information, as the resolution is not high enough. For `graph-1M` the interval is $[0.83075, 0.83275]$ with

Name	locking				no locking			
	even		uneven		even		uneven	
	min res	max res	min res	max res	min res	max res	min res	max res
`laser`	8.999e−14	1.475e−09	1.196e−13	1.418e−09	7.741e−16	1.202e−09	1.097e−15	1.359e−09
`SiH4`	3.727e−12	8.980e−08	2.706e−13	8.876e−10	7.355e−16	6.483e−08	8.863e−16	8.155e−09
`linverse`	1.798e−13	3.499e−07	4.139e−13	3.441e−09	6.461e−17	2.504e−07	1.753e−16	2.932e−09
`Pres_Poisson`	1.150e−12	1.393e−08	2.357e−13	1.391e−08	8.777e−16	1.332e−08	7.373e−16	1.188e−08
`Si5H12`	2.138e−14	4.010e−09	4.573e−14	3.925e−09	1.136e−15	2.714e−09	1.365e−15	2.997e−09
`brainpc2`	7.377e−14	6.391e−09	1.177e−13	6.462e−09	3.003e−15	5.776e−09	3.483e−15	5.444e−09
`rgg_n_2_15_s0`	4.588e−13	2.439e−08	2.336e−12	2.419e−08	1.732e−15	1.898e−08	2.384e−15	2.261e−08
`SiO`	9.972e−14	8.995e−09	1.999e−14	8.948e−09	1.118e−15	5.445e−09	1.215e−15	7.518e−09
`Andrews`	3.750e−13	2.536e−08	2.687e−13	2.523e−08	8.296e−16	2.362e−08	8.035e−16	2.476e−08
`Si34H36`	7.879e−14	1.822e−08	4.115e−14	1.802e−08	1.331e−15	1.292e−08	1.358e−15	1.600e−08
`fe_rotor`	5.259e−14	3.929e−08	1.026e−13	3.948e−08	1.500e−15	3.387e−08	1.492e−15	2.726e−08
`GraI-119k`	9.284e−15	5.549e−08	7.212e−15	5.232e−08	1.708e−15	5.549e−08	1.261e−15	2.183e−08

Table 4.2: *Final residuals for the four data sets of the matrix test set.*

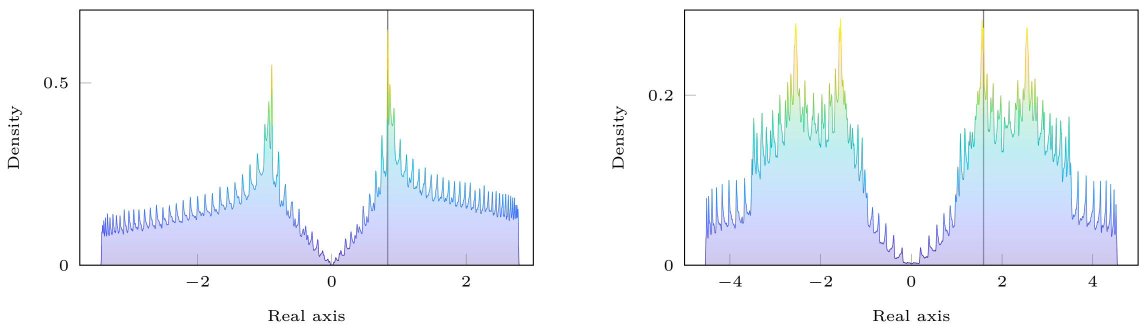


Figure 4.1: *Low resolution density of states for the matrices* **`graph-1M`** *(left) and* **`topi-1M`** *(right), produced using the KPM-DOS method [Wei+06]. The density is normalized such that it integrates to one.*

1 995 eigenpairs and for `topi-1M` the interval is $[1.59, 1.594]$ with 1 780 eigenpairs. Results for these matrices will be presented in Section 5.3.16.

4.1.3 Ritz value pairing

During the analysis of the speed of convergence of distinct directions during subspace iteration, the residual drop rate is the only quantifiable indicator of progress. However, the process of decreasing residuals of Ritz pairs is difficult to track, since there is no direct relation between an eigenpair and a Ritz pair other than their proximity in relation to the residual of the Ritz pair (see Section 1.6.3). While the eigensolver used for the reduced eigenproblem often retains ordering, eigenvalues may switch positions or move along the spectral axis as iterations progress. If the residual is small enough, a Ritz pair may be unmistakably assignable to an eigenpair and, by extension, to a Ritz pair from an earlier iteration whose residual was low enough as well. In early passes of the subspace iteration process, this is often not possible, in particular if the dampening of the filter is weak. If eigenvalues are densely clustered, assignment will only be possible in later iterations, if a residual that would allow separation is reached at all. In experiments dealing with residual drop rates, often the first iterations therefore cannot be considered to serve as valid sources of data. Similarly, as soon as Ritz values reach saturation, the computed drop rates only reflect the small variations of the saturation residual of the respective Ritz value and are therefore omitted.

4.2 Convergence revisited

Let us revisit convergence and dampening of eigendirections by a filter function. Here, the term *convergence* is not used as a binary property, say, for reaching a specified target residual, but a continuous decrease in residual, no matter how small or large. Therefore, convergence can be slow or fast and start or stop at certain levels. As outlined in Section 1.6.1, the speed of convergence during subspace iteration is chiefly related to the separation of eigenvalues,

$$\left|\frac{\lambda_{m+1}}{\lambda_i}\right|$$

for $i = 1, \ldots, m$, if the projection operator is the orthogonal projector onto the invariant subspace associated with the m largest magnitude eigenvalues, when trying to compute exactly these m eigenpairs. If, in order to compute a sequence of m eigenpairs at an arbitrary position inside the spectrum, a suitable filtering method is used, convergence is now related to the separation of eigenvalues of the projecting operator,

$$\left|\frac{f(\lambda_{m+1})}{f(\lambda_i)}\right|$$

for $i = 1, \ldots, m$, where f shall be the filtering function and we assume the eigenvalues of the matrix pencil to be ordered such that the eigenvalues of the projection operator are sorted by absolute value in descending order,

$$|f(\lambda_1)| \geq \ldots \geq |f(\lambda_m)| \geq \ldots,$$

since the projection operator replaces the matrix pencil in the subspace iteration process. We have seen that the corresponding eigenvalues of the original matrix pencil are obtained by the compatibility of their eigenvectors in the Rayleigh-Ritz process. If, furthermore, the filtering function is not ideal and $m_0 > m$ vectors are iterated, we have to relate the convergence speed to

$$\left|\frac{f(\lambda_{m_0+1})}{f(\lambda_i)}\right|$$

for $i = 1, \dots, m_0$, where the filter is chosen such that the convergence of the first m eigenpairs is, ideally, much faster than the convergence of the remaining unwanted eigenpairs. This is nothing more than the simple application of Theorem 1.2 to spectral filtering with some number of vectors m_0, where the subspace iteration actually tries to compute eigenpairs of the projection operator—hence the effect of the filter function—and only obtains eigenpairs of the original matrix pencil by choice of the Rayleigh quotients. A very similar statement has been derived in [TP] more formally.

This view on the progress of convergence has some implications for the understanding of the behavior of subspace iteration. Slowly converging directions will appear if eigenvalues $f(\lambda_i)$ of the projection operator with index $i \leq m_0$ are close in absolute value to the *convergence reference value* $f(\lambda_{m_0+1})$. The separation of eigenvalues of the original matrix pencil around the interval boundaries has no impact on convergence if $m_0 > m$ other than that the values of the filtering function at that position may result in slower convergence of these particular Ritz pairs. Instead, the convergence of every iterated eigenpair depends on the value of the filtering function at the positions of the eigenvalues of the original matrix pencil and, with this, on its spectral distribution.

Experiment 4.2 — Residual drop rates

We generate a matrix pencil of size 1000 *with evenly distributed random spectrum in* $[-1, 1]$. *The central* $m = 20$ *eigenpairs with a searchspace size of* $m_0 = 26$ *are to be computed over* 50 *iterations using a Cauchy filter of degree* 4 *with Gauß-Legendre integration rule. We compute the achieved residual drop rate between iterations if the Ritz values can be associated with each other. From the filtering function and the known eigenvalues of the matrix pencil, we estimate the drop rates for all Ritz pairs of the searchspace. The results are shown in Figure 4.2. Additionally, the ranges in which we expect the drop rates of Ritz pairs considered inside and outside the target interval are explicitly marked.*

It may take a few iterations for a Ritz pair to assume the predicted drop rate. Faster converging Ritz pairs reach this state sooner than slower converging Ritz pairs. Particularly in earlier iterations, other factors play a role, often leading to rates of convergence that are higher than anticipated, while convergence comes to a hold once saturation is reached. For particularly good dampening, saturation might be reached before the drop rate can settle at the estimated value.

In the extreme case, an ideal filter, all drop rates assume the prescribed dampening of the filter; this is not shown here. Repeating the experiment with different filters additionally shows that the predicted drop rate is reached faster with filters that produce better dampening in general. Smaller deviations from the estimated values may also be caused by the quality of the initial guess vectors. In terms of slots and occupation we now have to assume that the m_0 most dominant directions will occupy the m_0 slots of the searchspace and the searchspace will not align with the $n - m_0$ remaining directions. We can also assume that among the m_0 most dominant

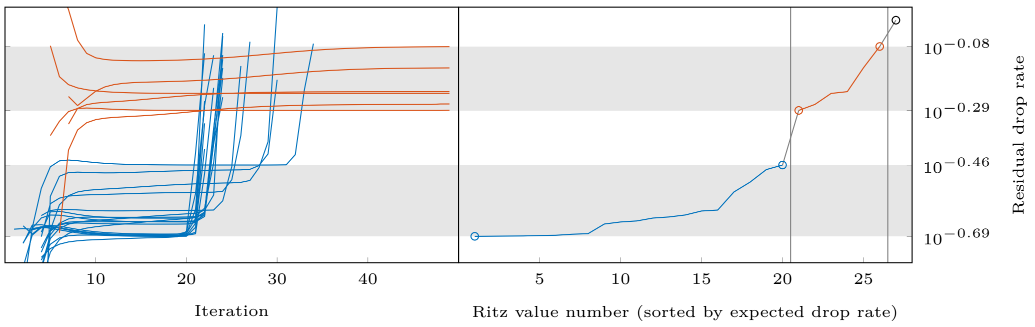


Figure 4.2: *Measured (left) and estimated (right) residual drop rates for every Ritz value inside a searchspace of size* 26*. Left: Ritz values inside (blue) and outside (red) the target interval. Right: The drop rates for Ritz values inside (blue) and outside (red) the target interval are separated by vertical lines. The convergence reference value at index* $m_0 + 1$ *is marked with a black circle.*

directions there may be those whose convergence speed, i.e., the speed at which the searchspace aligns with an eigendirection, is very low, such that virtually no alignment towards these directions will occur. These dimensions of the searchspace remain ambivalent as to what eigendirection they will align to. We will analyze this situation further in Section 4.2.4.

4.2.1 Undersized searchspaces

As has been discussed in [Krä+13] and [Krä14], the searchspace reaches critical mass with its size chosen as the exact number of eigenvalues contained in the target interval. Beyond this size, convergence is generally possible and below this size, convergence has to be considered practically impossible. In [Krä14] it has been shown that choosing too small a searchspace results in the computed subspace being embedded in the desired subspace but rotating with the remaining degrees of freedom. This is purely a side effect of the filtering function. Under ideal filtering conditions, or with a high-order approximation of the window function such that $f(\lambda_i)$ is approximately equal for all eigenvalues λ_i represented in the searchspace, the failure to converge is supported by the above considerations regarding convergence rates, which indicate little to no convergence in this case. No direction will establish dominance over another.

In the case of approximate filters, assuming that the number of directions with equally minimal dampening (or, depending on the type of filter in use, maximum amplification) fit the searchspace, dominance of these few directions may eventually be established, albeit with insufferably slow progress, governed by the differences in magnitude of the filtering function inside the passband, as indicated in Section 4.2. This effect is more pronounced for low degree filters with larger variations inside the passband. From this point of view, too small a searchspace is a problem of little or no dampening among the participating directions. Since convergence depends on the eigenvalues of the filtering operator and not the eigenvalues of the original eigenproblem, eigenvalues that are exposed to the same or similar levels of dampening can be considered clustered in this regard (ignoring the sign of the filter values, which has no effect on dominance). It is then inherently clear that the filter flanks, fitted to the target eigenpairs and being the most rapid change in filter value, seem to define the critical searchspace size.

4.2.1.1 On-the-fly increase of searchspace size

As has been established before, convergence speed in undersized searchspaces can be very slow or even come to a halt, depending on the filter in use. Assuming this situation can be detected and based on the understanding of the convergence behavior so far, it should be possible to extend the searchspace set with additional columns, opening up slots for occupation by additional directions. Conceptually, this is nothing other than starting with an initial guess that just so happens to contain a remarkably good representation of some of the eigendirections, assuming at least some convergence could occur before. We expect the convergence speed of directions iterated so far to increase, while newly added directions would start from scratch. We also expect the well represented directions to start off with a better residual, depending on their quality, i.e., the distances to the associated eigenvectors as apparent from the theorem on convergence Theorem 1.2, such that not all progress from earlier iterations is lost. Let us confirm these assumptions with an experiment.

Experiment 4.3 — On-the-fly increase of searchspace size
We generate a definite pair (A, B) *of size* 1000 *with an evenly spaced spectrum in* $[-1/2, 1/2]$. *A Butterworth filter of degree* 4 *is applied over* 100 *iterations to compute the* $m = 20$ *central eigenvalues. The size of the searchspace is initially chosen as* $m_0 = 10$ *and is increased to* $m_0 = 25$ *at iteration* 30 *by appending random vectors to the computed approximate eigenvectors. The filtering function and spectral distribution are chosen such that slight convergence is possible in the first* 30 *iterations. The results can be found in Figure 4.3.*

It can be determined from the results depicted by Figure 4.3 that not all information from earlier iterations is lost with the change of searchspace size. Previously iterated directions maintain—at least in part—their residual, and the loss in accuracy is nowhere near a complete restart with an initial searchspace size of $m_0 = 25$. For both phases, before and after the extension, residual drop rates assume the respective estimated values after some iterations.

4.2.1.2 Detection of undersized searchspaces

An early on or even a priori detection of undersized searchspaces is typically impossible, unless some information on the spectral distribution, e.g., the density of states derived from the KPM [Wei+06], is available (see also Section 3.1.2).

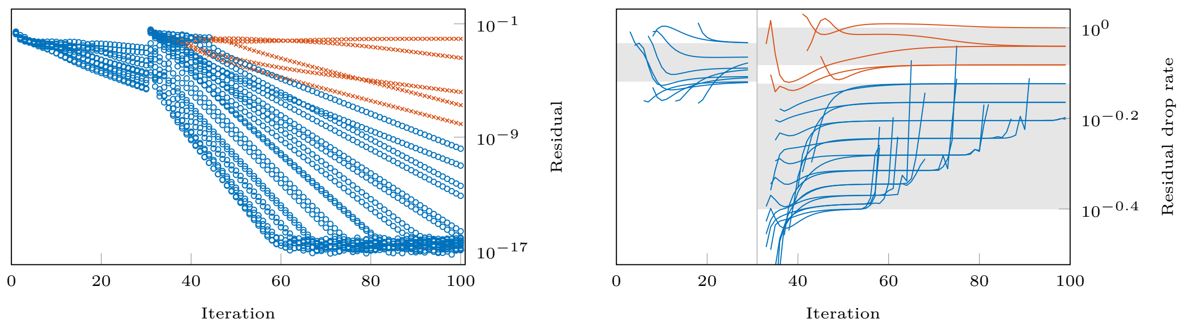


Figure 4.3: *Residual history (left) and drop rates (right) for a one-time on-thy-fly increase of searchspace size. Ritz values inside the target interval are shown in blue, Ritz values outside the target interval in red.*

A readily available indicator is the number of Ritz values inside the target interval. While not a particularly accurate source for the exact number of directions inside the target interval—especially in early iterations, since the most dominant directions are the directions inside the target interval—having no Ritz values outside the target interval occupy a slot is a good indicator for undersized searchspaces. But even if outside Ritz values do occupy slots, the searchspace might still be too small for achieving suitable residual drop rates depending on filter and spectral distribution.

A more accurate estimation can be obtained from the eigenvalues of the second Rayleigh quotient of the reduced-size eigenproblem. This method will be explored in the following section.

4.2.2 Enlarged searchspaces

The effect of oversized searchspaces has been discussed in [Krä+13], [Krä14], [Pie+16], and [TP]. Following Section 4.2, the observed improvement of convergence speed in larger searchspaces is again a consequence of the combination of spectral distribution and filtering function.

Let again the eigenvalues of the matrix pencil be ordered such that the eigenvalues of the projection operator descend in absolute value,

$$|f(\lambda_1)| \geq \ldots \geq |f(\lambda_m)| \geq \cdots \geq |f(\lambda_{m_0})| \geq \ldots,$$

where, as before, m is the number of eigenvalues inside the target interval and $m_0 \geq m$ is the searchspace size. If we assume the largest magnitude eigenvalues of the projection operator to occupy the slots of the searchspace[2] in accordance with Theorem 1.2, convergence of all iterated directions is mandated by the convergence reference value $f(\lambda_{m_0+1})$. If the filtering function of the projection operator is also decaying, $f(\lambda_{m_0+1})$ typically becomes smaller as m_0 grows, improving convergence

[2] For the typical filter shape, eigenpairs inside the target interval are more dominant than eigenpairs outside the target interval, but the considerations here do not necessarily require this.

speed of all directions previously occupying the (smaller) searchspace. For non-decaying filters, this effect does not occur, or only in a small range of searchspace enlargement; improvement can be expected until $f(\lambda_{m_0+1})$ reaches the maximum stopband gain. Oscillations of the filter in pass- and stopband play a role in that they skew the intuition of which directions will occupy the searchspace or which directions will converge the fastest. Not always will the most dominant outside directions also be the closest to the target interval. Extreme cases are equioscillating filters since the oscillation frequency decreases with increased distance to the target interval, possibly injecting far-away directions into the searchspace.

Experiment 4.4 — Residual drop rates and searchspace size

We generate a definite pair (A, B) *of size* 1000 *with an evenly spaced spectrum in* $[-1/2, 1/2]$. *A Chebyshev filter of degree* 200 *is applied over 50 iterations in form of the explicit projection operator matrix. The* $m = 20$ *innermost eigenvalues are computed with searchspace sizes*

$$m_0 \in \left\{m, \frac{5m}{4}, \frac{3m}{2}, 2m\right\}$$

and the drop rates are compared to the estimated drop rates for the best Ritz value inside the interval (blue), the worst Ritz value inside the interval (red), the best Ritz value outside the interval (yellow), and the worst Ritz value outside the interval (purple); the latter two do not exist for $m_0 = m$. *Figure 4.4 summarizes the results. For the second set of plots, a Zolotarev filter of degree* 4 *with* $\kappa = 0.01$ *was used to compare the previous results to a non-decaying filter. We can expect the drop rates to improve with increased searchspace size, as long as the* $(m_0 + 1)$*-th eigenvalue of the projection operator still decreases. Beyond that point, convergence speed does not improve any further.*

Repeating the experiment with evenly distributed random spectra reveals that the estimation is most accurate if no Ritz phantoms are produced by the combination of spectral distribution and filter. Ritz phantoms are more likely to occur if the filtering properties are weak and disturb the convergence of Ritz values close by. A follow-up experiment will detail this effect in a later section, see Experiment 4.9 in Section 4.2.4.1.

While enlarged searchspaces benefit convergence of eigenpairs inside the target interval, computation time and memory consumption, in particular during the application of the projection, increase accordingly. As to what extent this holds true in practice depends on the filter in use and the method of its application, as well as the general implementation. For certain filter types, searchspaces exceeding a certain size have no additional benefit. Additionally, larger searchspaces increase the probability of Ritz phantoms (see Section 4.2.4). If more directions outside the target interval are iterated, the drop rates of these directions are low since the filter is not steep between λ_{m_0+1} and the eigenvalues of the additional directions. Since for decaying filters the steepness of the filtering curve fades with increased distance,

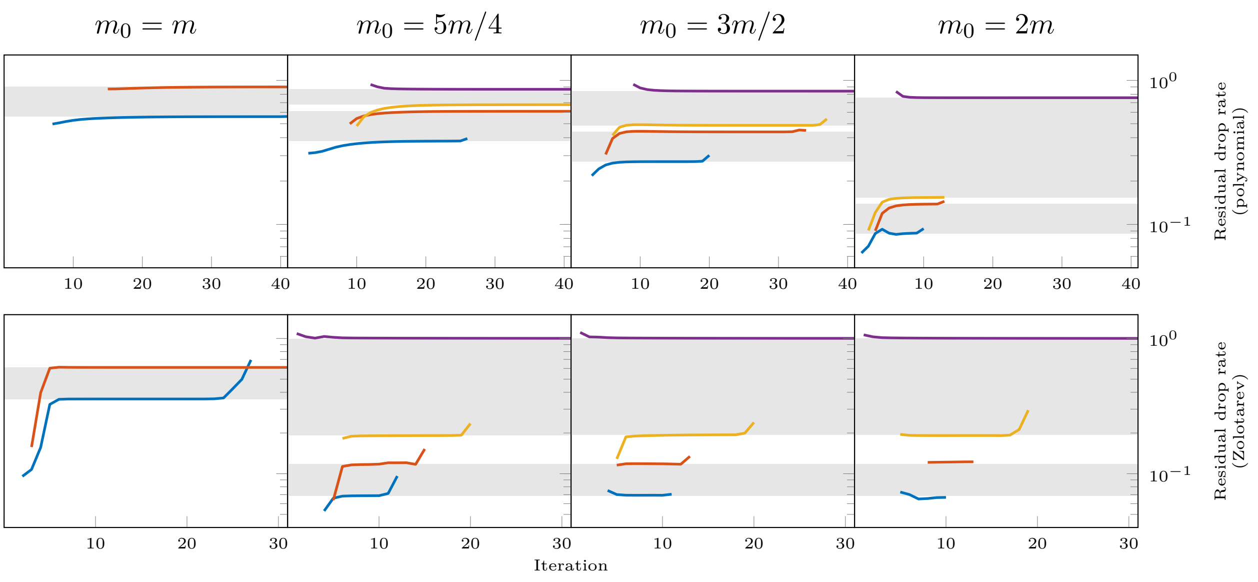


Figure 4.4: *Residual drop rates for different m_0. Top: Chebyshev polynomial filter (decaying). Bottom: Zolotarev filter (not decaying). Shown are the drop rates for the best (blue) and worst (red) Ritz value inside the interval as well as the best (yellow) and worst (purple) Ritz value outside the interval (the latter two of which do not exist for $m_0 = m$); the gray areas indicate the estimated drop rates inside and outside the interval.*

the gain from additional slots in the searchspace decreases and the improvement of drop rates becomes increasingly lower.

4.2.2.1 On-the-fly restriction of searchspace size

The specification of an optimal searchspace size requires the knowledge of the positions of eigenvalues. However, reducing the size of the searchspace during a run of an iterative eigensolver can be reasonable for at least four reasons.

- Retaining full numerical rank of the searchspace set.
- Removing Ritz phantoms and/or slowly converging directions.
- Reducing the number of vectors to be processed by the algorithm.
- Locking directions that have reached the desired residual.

The removal of directions from the set of iterated approximate eigenvectors is trivial: the removal of single columns is approximately equivalent to orthogonalization against the associated directions. This is because the approximate eigenvectors are essentially B-orthogonal after the first iteration. With the primitive Ritz vectors W of the reduced Rayleigh-Ritz eigenvalue problem

$$A_U W = B_U W \Lambda_U$$

for the matrix pencil

$$\left(A_U = U^H AU, B_U = U^H BU\right)$$

being B_U-orthogonal,

$$W^H B_U W = I,$$

since A_U is Hermitian and B_U Hermitian positive definite,[3] with $X = UW$ it is

$$X^H BX = W^H U^H BUW = W^H B_U W = I.$$

In practice, orthogonality may still be lacking after one iteration, but is typically established properly after the second iteration.

4.2.2.2 Locking

One instance of the method outlined above is the locking mechanism [Saa11; Krä14] that aims at improving performance by removing directions with sufficiently small residual from the searchspace to iterate on fewer vectors in following iterations. However, these removed directions will, in all likelihood, not be the least dominant directions; it is quite the opposite—dominant directions will reach the target residual first. Since orthogonality is never perfect and small components of the allegedly removed directions remain, due to the convergence behavior of subspace iteration discussed earlier, previously removed directions are bound to reappear and replace the least dominant directions in the then reduced-size searchspace eventually. To prevent this from happening, frequent orthogonalization of the searchspace set against all locked directions is required to maintain the dampening enforced this way. Indeed, since the removal of vectors from an orthogonal set of vectors, i.e., the orthogonalization against certain directions, essentially hides these directions from the subspace iteration process; it can be seen as equivalent to directions being almost eliminated by the filter itself, which has the same effect. The difference is that locking also reduces the searchspace size, while not interfering with the directions that occupy the searchspace.

4.2.2.3 Rank deficiency

In most cases, the searchspace set is a basis for the searchspace. If this searchspace set—the set of vectors that spans the searchspace—becomes rank deficient, the searchspace looses dimensions and the Rayleigh quotients form an eigenproblem that is singular, which might be unsolvable by available standard algorithms without extracting its regular part. In order to produce a non-singular reduced eigenproblem, rank deficiency has to be detected and the number of searchspace vectors has to be reduced before the small-scale eigenproblem is generated by removing selected vectors accordingly.

The reason for rank deficiency of the searchspace set is the loss of directions among the vectors due to strong dampening of components by the filter function or removal

[3] $y^H By > 0 \implies x^H U^H BUx > 0$ for $y = Ux$ if $x \neq 0$.

of directions by orthogonalization without adjusting the number of searchspace vectors. If a direction is, either over several iterations or due to strong filtering properties, reduced to numerically irrelevant noise, it can be considered vanished. The searchspace then is essentially orthogonal to this direction. If the removal of the direction is not maintained, e.g., if it was caused by a one-time orthogonalization, the slot that now has to be considered unoccupied will eventually be occupied by the removed directions again, since small components of these directions remain. As long as the removed directions are sufficiently dampened, other directions might temporarily occupy the free slots.

Experiment 4.5 — Slot swapping

We generate a definite pair of size 1000 *with evenly spaced spectrum in* $[-1, 1]$. *With a searchspace size of* 30, *we compute the innermost* 20 *eigenpairs using a Butterworth filter of degree* 12. *To simulate a case where directions are evicted from the searchspace for other, far less dominant directions to temporarily occupy the slots, we orthogonalize the searchspace set against the known eigenvectors of the six least dominant directions inside the target interval, leaving* 14 *eigenpairs inside and ten eigenpairs outside the interval to converge (rather) normally. The last point in time to perform the orthogonalization without causing rank deficiency in the searchspace set is the beginning of iteration four. Figure 4.5 shows how the procedure wreaks havoc in the searchspace.*

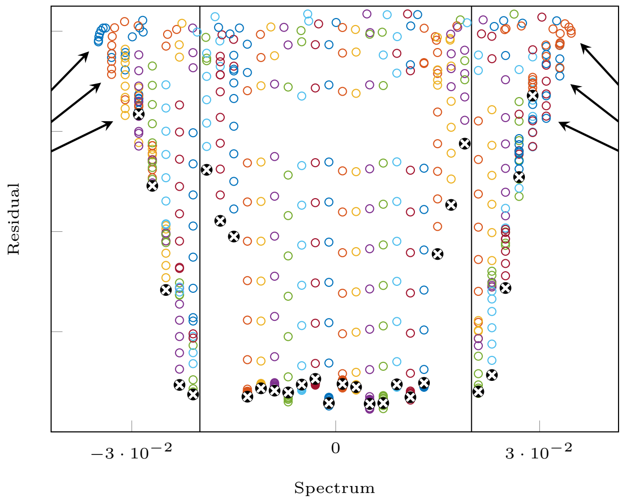


Figure 4.5: *Slot swapping. The final residual of all slots is marked by* ⊗. *The Ritz values temporarily occupying the slots are marked with arrows, and the target interval is indicated by vertical lines. The different Ritz values are identified by color.*

It can clearly be seen that the six Ritz values inside the interval at the boundaries are disturbed severely. All other Ritz values react with a jump in residual after iteration three. Six additional Ritz values emerge and start converging against the next most dominant directions. At about iteration 13, the original eigenvectors have gained enough dominance and the Ritz values snap back to their original positions, where they converge with the respective delay. A very close look reveals that the movement back to their original position takes a few iterations in which the Ritz values seem to hop from slot to slot. This is because the gradual gain of dominance by the original directions evacuates the temporal directions one by one.

If the removal was caused by the filter, it affects the least dominant directions of the basis and is permanent. The slot then cannot be occupied by a different direction; since the most dominant directions occupy the searchspace, all other directions are less dominant and are equally or more heavily dampened than the least dominant direction in the basis. This is not to be confused with the emergence of Ritz phantoms (see Section 4.2.4).

Experiment 4.6 — Filter induced rank deficiency

We generate a definite pair of size 1000 *with evenly spaced spectrum in* $[-1, 1]$. *With a searchspace size of* 80, *we compute the innermost* 20 *eigenpairs using*

- *a Butterworth filter of degree* 4,
- *a Zolotarev filter of degree* 4, *and*
- *an ideal filter with fixed dampening of* 10^{-1}

over 100 *iterations. Since the Rayleigh-Ritz procedure or orthogonalization of the searchspace set interferes with the representation of the directions, we iterate on the searchspace set exclusively, in the vein of Algorithm 1.4, but without orthogonalization. Figure 4.6 shows the development of the searchspace size, which is reduced whenever rank deficiency is detected.*

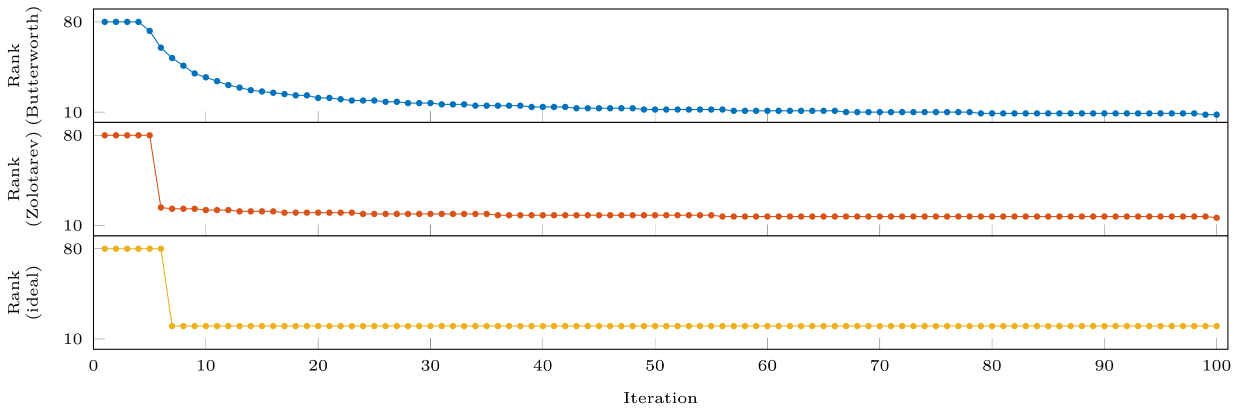


Figure 4.6: *Development of the rank of the searchspace set. Shown are searchspace sizes (here equivalent to numerical rank) over* 100 *iterations. Top: Butterworth, degree* 4. *Middle: Zolotarev, degree* 4. *Bottom: Ideal, dampening* 10^{-1}.

If the filter is decaying, rank deficiency—also depending on the size of searchspace—occurs gradually depending on the differences in filter value. Stronger dampening properties cause earlier rank deficiency. In the case of non-decaying filters, where dominance is almost equal for many directions, rank deficiency causes larger reductions in size in a single iteration due to approximately equal associated filter values. The fewer gradual changes in filter value are reflected by few smaller reductions in size afterwards. Ideal filters, accordingly, experience rank reduction to the exact size once.

We also see that the rank drops below the number of contained eigenpairs. This is due to the bypass of the Rayleigh-Ritz procedure (or basis orthogonalization), which otherwise retains an orthogonal basis and thus rescales the weak directions in the process. In this case, rank deficiency due to strong filtering properties are likely to occur once in the very first iteration because the directions are essentially removed in one fell swoop. We expect rank deficiency of this kind to be more likely for decaying filters since farther directions are often suppressed much more strongly.

If the removal was caused by orthogonalization that is maintained for all following iterations, it is permanent and eventually the next most dominant directions previously not part of the searchspace will occupy the now free slots, affecting the convergence behavior of all iterated directions (we have seen this in Experiment 4.5, although orthogonality was not maintained). If the orthogonalization is performed from the very beginning, directions can be blocked from appearing altogether [GKL12] (see also Section 5.3.1).

No matter the cause, reoccupation may take more than one iteration and may not prevent rank deficiency from occurring in the iteration following the removal (see Experiment 4.5 where later orthogonalization causes rank deficiencies due to the basis being more well-established; the original directions require some iterations to reclaim their slot). To deal with this rank deficiency of the searchspace set, an alternative reduced-size basis may be constructed that includes the remaining directions of the former basis. A commonly used method to achieve just that is, again, orthogonalization, which will reveal the rank deficiency and can reduce the basis. Conventional orthogonalization, however, can be expensive if the problem size is large and the searchspace set is, accordingly, a tall and (often) skinny block of vectors. While specialized algorithms for this case exist [Dem+12], we are able to extract the required information from the Rayleigh quotients [Krä14]. The Rayleigh quotient of the matrix B, $B_U = U^H BU$, can reveal the rank of U. Assuming a reduced (thin) SVD of U (see also Section 5.2.8),

$$U = V\Sigma W^H,$$

where $V \in \mathbb{C}^{n\times m}$ orthonormal, $\Sigma \in \mathbb{R}^{m\times m}$ diagonal, and $W \in \mathbb{C}^{m\times m}$ orthonormal, a reduced matrix may be written

$$U^H U = W\Sigma V^H V\Sigma W^H = W\Sigma^2 W^H$$

that evidently has the same rank as U, but is normally not computed during the Rayleigh-Ritz process unless $B = I$. Certainly, U and B_U have the same rank as well, with B being positive definite. With an SVD of B_U

$$B_U = W\Lambda_W W^H$$

where left and right singular vectors are identical due to B_U being Hermitian, the equivalent eigenproblem

$$B_U W = W \Lambda_W$$

can be used to obtain the rank of B_U and thus the rank of U. Let the rank of U by examination of Λ_W be k. A subspace basis of reduced size k can now be obtained by letting

$$U_k \coloneqq UW(1\colon k)$$

assuming that the eigenvalues and with them their associated eigenvectors are ordered descendingly, such that $W(1\colon k)$ represents the k eigenvectors associated with the k largest eigenvalues of B_U. Then U and U_k span the same space if the rank determination was correct.[4] Indeed, every matrix of size $m \times k$ and full rank may be used instead of $W(1\colon k)$.

This begs the question of what happens if more columns are removed.

4.2.2.4 Further reduction

The question raised above implicitly asks for the connection between the singular values or eigenvalues of B_U and the directions contained in the searchspace. Similarly, further reduction of the searchspace size beyond distinct directions that are saved and available for orthogonalization or maintaining a searchspace set of full rank—be it for speeding up convergence or reducing the number of iterated vectors—necessitates the association of directions and their dominance, i.e., their convergence speed and, equivalently, the associated value of the filter function at their positions. This is a connection that typically cannot be made by looking at just the eigenvectors.

By writing the application of the approximate projection operator P_f to the right-hand sides in terms of its eigendecomposition,

$$U = P_f Y = X f(\Lambda) X^H BY, \tag{4.1}$$

the second Rayleigh quotient may be written

$$\begin{aligned} B_U &= Y^H BX f(\Lambda) X^H BX f(\Lambda) X^H BY \\ &= Y^H BX f^2(\Lambda) X^H BY. \end{aligned} \tag{4.2}$$

If now Y is B-orthonormal and thus $Y^H BY = I$, invoking the Rayleigh-Ritz theorem for the matrix pencil

$$\left(Y^H BP_f^2 Y, Y^H BY\right)$$

and $X_P = YW_P$ being those eigenvectors of (A, B) (and P_f) that the vectors Y are

4) $\exists H$ s.t. $UH = \begin{pmatrix}\widetilde{U} & \mathbf{0}\end{pmatrix}$ with $\widetilde{U} \in \mathbb{C}^{n\times k}$. Then $UHW(1\colon k)$ just recombines the columns of $\widetilde{U}$.

approximations for,

$$\begin{aligned} P_f^2 X_P &= X_P \Lambda_P \\ P_f^2 Y W_P &= Y W_P \Lambda_P \\ Y^H B P_f^2 Y W_P &= Y^H B Y W_P \Lambda_P \\ Y^H B P_f^2 Y W_P &= W_P \Lambda_P \\ Y^H B X f^2(\Lambda) X^H B Y W_P &= W_P \Lambda_P \\ B_U W_P &= W_P \Lambda_P. \end{aligned}$$

Thus, B_U is a Rayleigh quotient for the squared projection operator P_f^2 and there are eigenpairs (W_P, Λ_P) of B_U such that $(Y W_P, \Lambda_P)$ are the eigenpairs of P_f^2 contained in span(Y); refer to Theorem 1.1. Since, from the second iteration onward, the columns of Y are the approximate eigenvectors, we know that span$(X_p) \subseteq$ span(Y)—at least approximately—and we obtain the associated eigenvalues of the projection operator, i.e., the values of the filter function at the positions of the eigenvalues of the original matrix pencil associated with the approximate eigenvectors Y. As we have seen at the beginning of this section, Y is essentially B-orthogonal as of the second iteration. The filter values obtained here are also expected to improve as the approximations Y are improved, with the same convergence rates. While P_f is not Hermitian unless $B = I$, the Rayleigh quotients here are Hermitian by choice of the vectors for left and right multiplication.

We thus have obtained the values of the filter function acting on the iterated eigenpairs and a direct connection between these directions and the searchspace set U. The choice of $W(1\!:\!k)$ from before has been left unexplained. In the light of the relation between U and W we see that

$$U_k^H B U_k = W^H(1\!:\!k) U^H B U W(1\!:\!k) = W^H(1\!:\!k) B_U W(1\!:\!k) = \Lambda_W(1\!:\!k, 1\!:\!k),$$

which also allows to omit the explicit recomputation of B_U using U_k. We can therefore choose U_k to contain selected directions by choosing from the columns of W. Assuming the eigenvalues Λ_k are ordered descendingly, $W(1\!:\!k)$ selects the k most dominant and fastest converging directions.

4.2.2.5 Optimization of searchspace size

The knowledge of the (approximate) filter values of the iterated directions allows for the acquisition of additional information that can be used for better control of the subspace iteration's progression.

First, knowing the filtering function that is applied in each iteration approximately, and in particular its values at the boundaries of the interval of interest, gives a reliable estimate of the number of eigenpairs inside this interval by counting the eigenvalues of B_U that exceed the boundary values [GKL12; TP; Krä14]. The situation becomes problematic if the values on both boundaries differ; this can be the case for the polynomial Chebyshev filter if the target interval is off-center with respect to the interval of approximation, or the filter is intended to capture complex eigenpairs of non-Hermitian problems and the steepness of the filter flanks on the boundary of the target region differs largely (remember Chapter 2). As has been mentioned before, exactly what parts of the filter are to be considered inside or outside the target interval may differ on a case-to-case basis.

Second, while the common factor determining the speed of convergence, $f(\lambda_{m_0+1})$, is still unknown, at least a statement about the minimum possible approximate convergence speed can be made since

$$\left|\frac{f(\lambda_{m_0+1})}{f(\lambda_i)}\right| \leq \left|\frac{f(\lambda_{m_0})}{f(\lambda_i)}\right| \quad \text{for } i = 1, \dots, m_0$$

where $f(\lambda_{m_0})$ is known. It may be possible to start the subspace iteration with an additional vector for the searchspace set, such that after reduction of the searchspace size by this one vector, a more accurate estimation can be given since the later $(m_0 + 1)$-st value will be approximately known from an earlier iteration.

Continuing this train of thought, the choice for a possible size reduction to dimension k as soon as the eigenvalues of B_U are reliable enough can be made by estimating the new slowest convergence speed as $f(\lambda_k)/f(\lambda_{k+1})$ where $k < m_0$ and all occurring quantities are known at least approximately. With this, it is possible to choose a reduced size k that maximizes the speed of the eigenpair that will converge the slowest. Of course, any reduction will reduce the overall speed of convergence, as has been established previously. However, estimated filter values from B_U for directions that are slow to converge will likewise be converging at a slower rate, making the estimation unreliable in early iterations.

A direct connection between vectors in Y and the eigenvalues of B_U cannot be made since it has to be assumed that the eigenvalues of B_U can be reordered during computation.

4.2.2.6 Addendum

Other methods to extract the regular part of a singular eigenproblem exist. In [ISN10] the following transformation of the reduced size eigenproblem based on the SVD is used. Let the SVD of B_U be

$$B_U = V\Sigma W^H.$$

Then

$$\begin{aligned} A_U H &= B_U H\Lambda \\ A_U H &= V\Sigma W^H H\Lambda \\ V^H A_U H &= \Sigma W^H H\Lambda \\ \Sigma^{-\frac{1}{2}} V^H A_U H &= \Sigma^{\frac{1}{2}} W^H H\Lambda \end{aligned}$$

Replacing H with $H = W\Sigma^{-\frac{1}{2}}\Gamma$ gives

$$\Sigma^{-\frac{1}{2}}V^H A_U W\Sigma^{-\frac{1}{2}}\Gamma = \Sigma^{\frac{1}{2}}W^H W\Sigma^{-\frac{1}{2}}\Gamma\Lambda = \Gamma\Lambda.$$

This is essentially the transformation to standard form from Section 1.2.2. Since B_U is Hermitian, $V = W$, and we can again replace the SVD with an eigenproblem. Since this is computed anyway to obtain the filter values for the searchspace, the above is applicable as replacement for the modified U_k and B_U from before. The reduction can be incorporated via the values of Σ, which again are the filter values. Selecting k values from Σ and the respective vectors from V and W reduces the size of the eigenproblem, and if all small values are discarded, the eigenproblem will be regular. The Ritz vectors are then computed as $X = UW\Sigma^{-\frac{1}{2}}\Gamma$. The method introduced before does require fewer computations while this method also works for non-Hermitian problems.

4.2.3 SSM effective filter

In [GKL12], the method of computing the filter values from Section 4.2.2 was used to find a representation of the Cauchy filter with Gauss-Legendre rule and, in particular, its values at the interval boundaries to ultimately use this information to find estimations of the number of eigenvalues inside the target interval. To achieve this, an isolated eigenvalue with known location was used as the target for a progression of intervals, which may be visualized as an interval, gradually shifted along the real axis. At some point, the isolated eigenvalue enters the interval and, similarly, leaves it again at a different point. The associated singular value of B_U is assumed to be the largest of the singular values since, with a decaying filter, the well separated remaining spectral regions are sure to be much more strongly dampened. With the filter value obtained at different spectral positions relative to the target interval, the filter can be reconstructed. Essentially, instead of sampling the spectrum via a filter, the filter is sampled using the spectrum. The results from [GKL12] were formally confirmed in [TP]. For alternatives on estimating eigenvalue counts, the reader may consult [FTS10] and [DPS16].

It is surely a long stretch to assume that the effective filtering properties of the SSM can somehow be written as a single operator, whose eigendirections are, above all, identical to the target eigenproblem. The latter, at least, is the prerequisite for the application of the Rayleigh-Ritz process to extract eigenpairs of the original matrix pencil from the constructed basis and, as such, can be assumed to be the case. The former, however, is the prerequisite for identifying an effective filtering function for the SSM in the same manner as the filter was constructed in [GKL12].

Experiment 4.7 — Effective multi-moment filter

We construct a Hermitian matrix A of size 1000 *with a distinct isolated eigenvalue at $\lambda = 0$ and a generous separation of* 0.5 *to the remainder of the spectrum. The whole spectrum is contained in $[-2, 2]$. The width of the target interval is chosen as $w = 0.1$ and its center shifted over* 1000 *positions in $[-2w, 2w]$. We choose a Cauchy* *filter with midpoint rule of degree* 8 *so that we can write the filter function of the* ρ*-th moment conveniently as*

$$f_\rho(\lambda) = \left| \frac{\omega^{\rho-1}}{1+\omega^{2d}} \right|$$

with a transformation of the argument as

$$\omega = \frac{2\lambda}{w}$$

to set the cutoff frequency to the interval boundaries. The experiment is performed for $\ell \in \{2, 4, 8\}$ *moments; equivalent results for the Gauss-Legendre rule are included, as well.*

To obtain a new set of right-hand sides for following iterations, usually a random linear combination of the approximate Ritz vectors is used [Sak+19] (T. Sakurai, Y. Futamura, personal communication)*. This ensures the presence of all directions in the new set of vectors. Due to the construction of the searchspace set, the first block of* m *vectors from the basis is regularly Cauchy-filtered and also contains the desired directions. Although it bypasses the Rayleigh-Ritz procedure, this approach turns out to deliver an undisturbed result. From Chapter 1 we know that this kind of iteration requires orthogonalization to prevent all iterated vectors to approximate the most dominant eigenvector. We also require the right-hand sides to form an orthogonal basis.*

The experiments were performed with 4 *right-hand sides over* 4 *iterations. Figure 4.7 summarizes the findings for the midpoint rule, Figure 4.8 for the Gauss-Legendre rule. The disturbances caused by the random recombination of the vectors are displayed in the background. The moments of both rules are shown separately in Figure 4.9. Note that the functions were scaled w.r.t. their largest value, such that their range is* $[0, 1]$.

The filter functions we obtain experimentally with the above approach are plausible and the general validity of the method can be confirmed by producing conventional Cauchy filters with just the first moment. The disturbances can then equivalently be produced by a random recombination of the approximate eigenvectors in each iteration, which adds similar amounts of noise to the determined filter. Since the eigenvector to contribute to a new set of right-hand sides can be selected, for example excluding outside directions, the filtering properties of the disturbed filter can be improved. However, since the fluctuations are caused by a modification of the basis, stagnation at a certain residual may be possible (see Section 4.3.1).

The disturbances produced by the recombination into a new set of right-hand sides can, however, be completely eliminated by maintaining the length of each vector during recombination and injecting each direction only once. Since the searchspace size is a multiple of the number of right-hand sides, the recombination matrix has

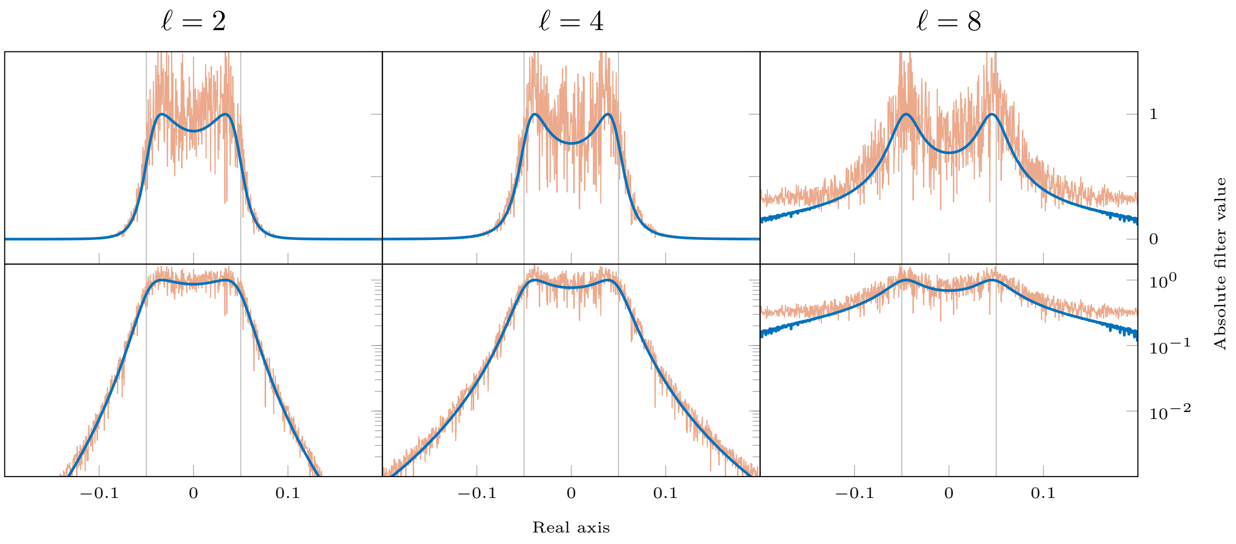


Figure 4.7: *SSM effective filters for the midpoint rule. Blue: undisturbed filter found by prefiltering the right-hand sides with a conventional Cauchy-filter. Red: disturbances, caused by repeated recombination of eigenvectors into right-hand sides.*

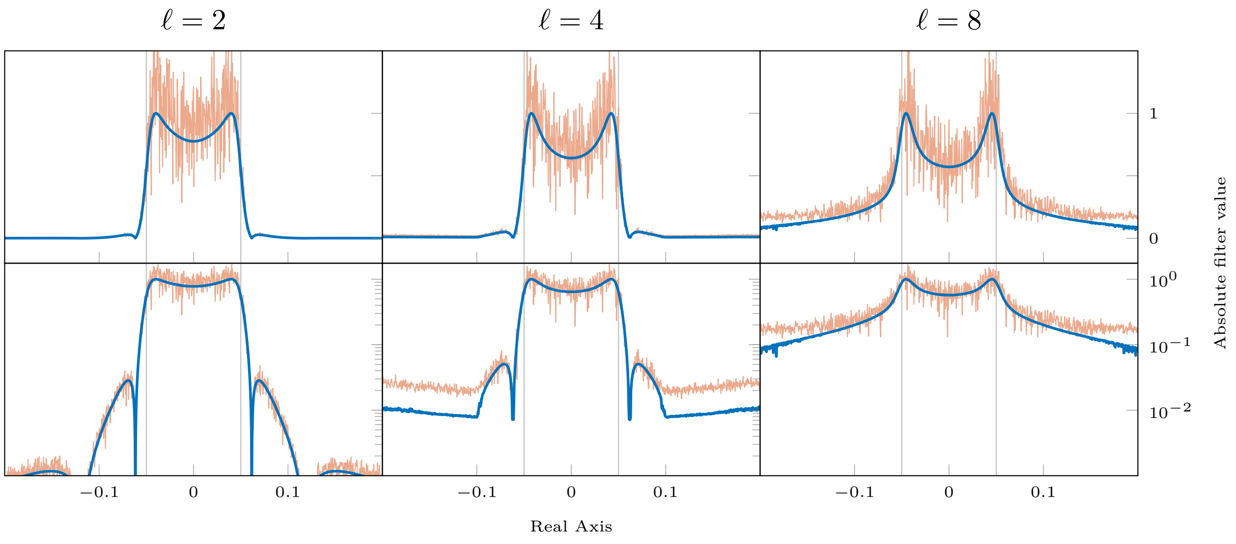


Figure 4.8: *SSM effective filters for the Gauss-Legendre rule. Blue: undisturbed filter found by prefiltering the right-hand sides with a conventional Cauchy-filter. Red: disturbances, caused by repeated recombination of eigenvectors into right-hand sides.*

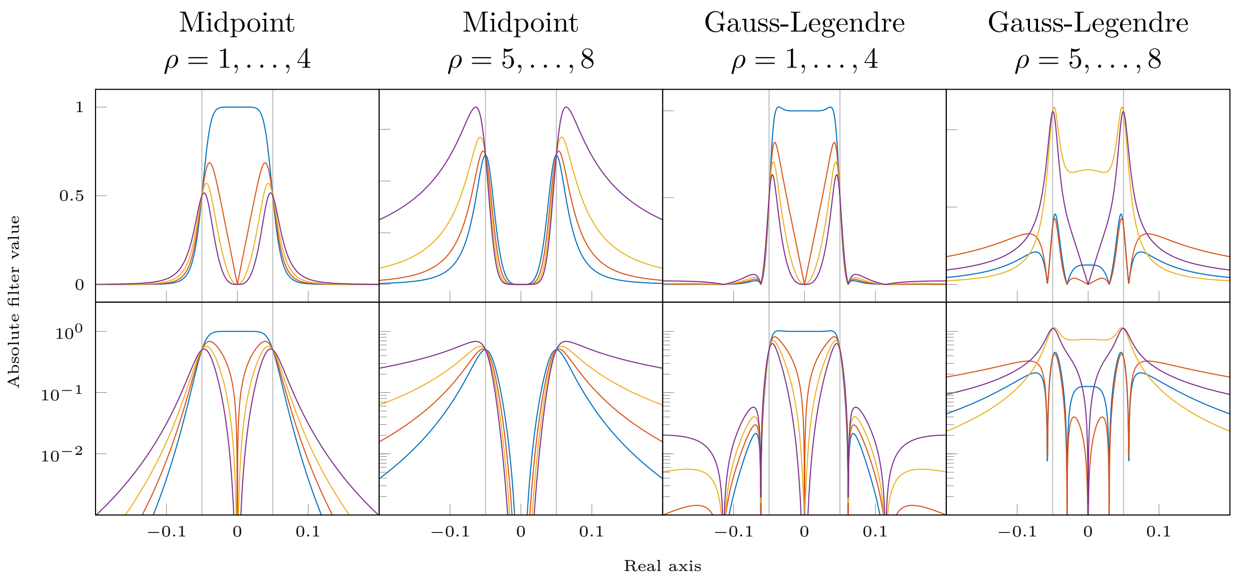


Figure 4.9: *SSM moments (absolute values). Top row: linear filter plots; bottom row: the respective logarithmic plots. Color order is: blue, red, yellow, purple.*

the simple form

$$Y = \boldsymbol{X} \begin{pmatrix} 1 & & & \\ \vdots & & & \\ 1 & & & \\ & 1 & & \\ & \vdots & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & \vdots \\ & & & 1 \end{pmatrix} \qquad \text{or} \qquad Y = \boldsymbol{X} \begin{pmatrix} I \\ \vdots \\ I \end{pmatrix}$$

to combine the approximate eigenvectors $\boldsymbol{X}$ into the set of right-hand sides Y. For ℓ moments and m right-hand sides, in each column $j = 1, \ldots, m$ of the recombination matrix only the rows $(j-1)\ell + 1$ to $(j-1)\ell + \ell$ are unity, all other entries being zero (of course the rows may be permuted). In case directions of $\boldsymbol{X}$ shall be removed, setting the corresponding entries to zero is all that is required. If columns of $\boldsymbol{X}$ were removed previously, the corresponding rows must be removed from the recombination matrix as well. The resulting filter function now matches the one found without recombination exactly (in this particular case, with either none or just a single eigenvalue inside the target interval, rank deficiency issues arise).

The plots also reveal that the filtering properties are weakened by the application of additional moments. Since the filtering function seems to be some combination of the moment filters and these moment filters tend to emphasize the interval boundaries, additional moments flatten the flanks of the filter and reduce the dampening of the outside regions. Moments beyond $\ell = 5$ even emphasize the outside of the target interval and dampen the inside, such that utilization of these moments is not recommended.

Being able to produce a representation of a filter in this way also hints at a relation to spectral projection beyond what was concluded in Section 2.5.3.6. The resulting filter is neither the sum, product, or plain convolution of the moments involved. Its potential analytical form remains unknown.

4.2.4 Ritz phantoms

With searchspace sizes that are larger than the theoretically required minimum, additional directions are included in the iteration. Depending on the filter in use, these typically are directions associated with eigenvalues outside of the target interval. Sometimes these additional Ritz values seem to show no sign of convergence and just shift their position along the real axis. A distinct assignment to an eigendirection is impossible, sometimes even after many iterations.

Following the convergence theorem for subspace iteration Theorem 1.2, stagnation of this kind can occur when some $f(\lambda_i)$, $i \in \{1, \ldots, m_0\}$, are very close to $f(\lambda_{m_0+1})$. The appearance of slowly converging Ritz values is therefore more likely for non-decaying filters since eigenvalues occupying the searchspace then have filter values that are very close to the convergence reference value. This is caused by a large number of filter values having basically identical dominance, which are first to occupy the searchspace beyond the eigenvalues inside the target interval. (see also Section 4.2.6, Figure 4.17). Decaying filters, on the other hand, are more likely to dampen farther away directions strongly, such that after some iterations the searchspace set may become rank deficient.

It is difficult to distinguish actually unoccupied slots with seemingly no relation to the eigenproblem at hand—we have dubbed these unrelated directions Ritz phantoms in the discussion of the Rayleigh-Ritz procedure of the introductory chapter—from slowly converging directions, possibly with larger distance to the target interval, that could be considered to occupy a slot. Indeed, iterating the searchspace set many times, depending on the spectrum and filter at hand, Ritz phantoms eventually may show ever so slight signs of convergence, albeit with very low residual drop rates. It can take many iterations for a direction to clearly represent an eigendirection.

This leads us to believe that Ritz phantoms are just the extreme case of a direction with very low convergence rate whose dominance over other directions is slim to none, resulting in similarly low rates of convergence. There is no circumstantial evidence that there is a more fundamental difference between slowly converging slots and unoccupied phantom slots, other than their theoretical existence as a consequence of the Rayleigh Ritz theorem Theorem 1.1.

De facto, such unrelated directions cannot exist, much less emerge in a searchspace of a subspace iteration algorithm. For eigenproblems with a complete set of eigenvectors, as is the case for Hermitian matrices and definite pairs, the eigenvectors span the n-dimensional vector space such that every direction is a combination of eigendirections. Due to filter functions typically not being optimal in that they never remove directions completely, the searchspace set contains directional components of all eigenvectors of the matrix pencil, many of which, however, may be quickly dampened to indistinguishable noise. A possible exception is the aforementioned gradual dampening of directions such that they are effectively removed, which ultimately may lead to rank deficiency of the searchspace set if these directions are occupying a slot in the searchspace such that its size has to be reduced anyway. This is, however, not the case for Ritz phantoms.

Since any slot in the searchspace is, after all, just a linear combination of eigendirections, comprised of components with length relative to their dominance, instead of considering Ritz phantoms not to represent an eigendirection at all, it may be more reasonable to consider them to represent more than one eigendirection, at least as long as the convergence process does not allow their separation. Since their associated filter value is nearly identical, they may be considered clustered with regard to the projection operator. Effects of clustering will be explored further in Section 4.2.5. Depending on the filter, few directions (decaying) or many directions (non-decaying) may overlap in this manner. Possible remaining directions then may again be separated more clearly by the filter, such that the Ritz phantom is governed by those overlapping directions in unison. During the (slow) progress of convergence, directions may ultimately be separated out, such that the slot now is more clearly associated with the remaining directions. We would expect that, after a sufficient number of iterations, even originally indistinguishable directions will separate.

This may be a possible explanation for the often peculiar movement of these Ritz values that typically follows a not necessarily direct path towards the location of an actual eigenvalue.

Experiment 4.8 — Movement of Ritz phantoms

We generate a definite pair (A, B) *of size* 1000 *with an evenly distributed random spectrum in* $[-5 \cdot 10^{-4}, 5 \cdot 10^{-4}]$. *Using a Zolotarev filter of degree* 12 *with* $\kappa = 0.01$ *on a searchspace of size* 40 *to compute* 20 *central eigenpairs ensures the emergence of Ritz phantoms. The size of the spectral interval plays no further role here. To track the path of the Ritz phantoms,* 1000 *iterations are performed.*

Figure 4.10 shows the paths of the slowly converging directions as their residual plotted against the Ritz value positions. The colors indicate the ordering of the Ritz values. The plot also includes the values of the filter at the spectral positions of the matrix pencil (which is why it appears to be a bit irregular) to emphasize the pursued target positions for the Ritz phantoms. These are the locations of the associated dominant directions of the searchspace, the peaks of the oscillations of the filter.

For all that matters, we will refer to slowly converging directions as *Ritz phantoms* and not distinguish them from absolute stagnation, both in residual and position, which has been observed only for ideal filters on enlarged searchspaces so far.

⊟⊠

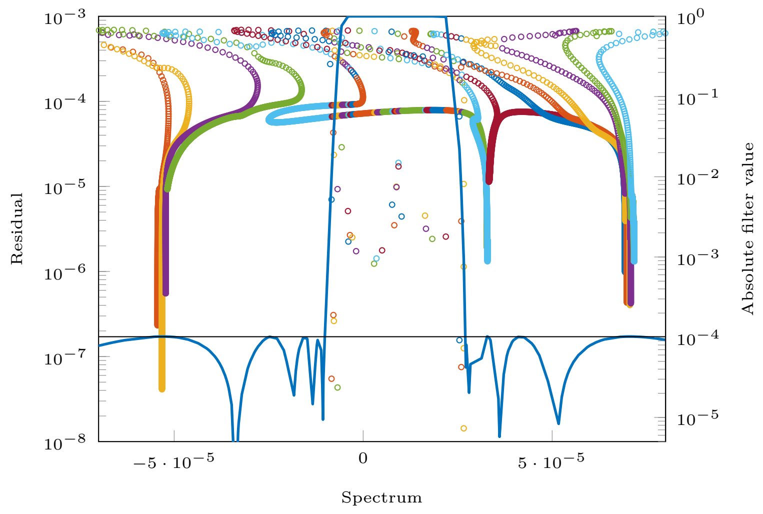


Figure 4.10: *Paths of Ritz phantoms. The logarithmic plot of the filter function at the positions of the eigenvalues is shown as overlay.*

4.2.4.1 Disturbance of convergence

Convergence is obstructed in the presence of Ritz phantoms. This is the conclusion drawn from observations made during experiments involving Ritz phantoms that did deviate from the expected convergence behavior. One of these experiments is presented as an example below.

Experiment 4.9 — Ritz phantom disturbances

We generate a definite pair (A, B) *of size* 1000 *with an evenly distributed random spectrum in* $[-1/2, 1/2]$ *such that two slowly converging directions appear for a searchspace size of* 30 *in combination with a weak filter, i.e., a filter with a gentle slope instead of steep flanks, to compute* 20 *inner eigenpairs. A Butterworth filter of degree* 4 *is applied for slow convergence and increased probability of Ritz phantoms. Figure 4.11 shows the residuals over* 100 *iterations presented in two styles: the left plot shows the residuals of Ritz values inside (blue) and outside (red) the target interval over the iterations; the right plot shows the residuals at the position of the respective Ritz value where the color indicates the ordering of the Ritz values.*

In the left plot, distinct disturbances can be seen at different iterations for different Ritz values. In the right plot, these disturbances occur as clustering of markers, followed by a switch in color. This indicates that the ordering of the Ritz values changed. At the top of the right plot, the movement of the Ritz phantoms can be seen; they ultimately move towards the eigenvalue associated with the most dominant among the overlapping eigendirections. As they pass by the regular Ritz values, they cause a change in the ordering produced by the eigensolver of the projected Rayleigh-Ritz eigenproblem. Thus, both plots together reveal that the observed disturbance of convergence occurs exactly as one of the Ritz phantoms passes the corresponding Ritz

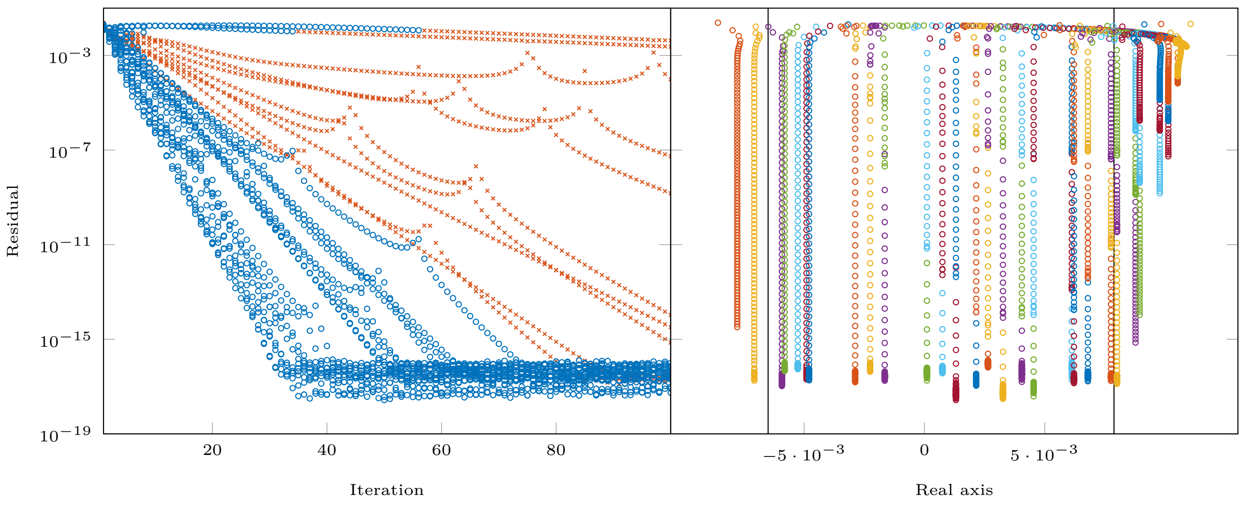


Figure 4.11: *Disturbance of Ritz values by nearby Ritz phantoms. Left: Residuals plotted against iteration number. Right: Residuals plotted against Ritz value position.*

value and that the disturbance is not only related to the presence of Ritz phantoms in the searchspace, but also to its proximity to the remaining Ritz values. The effect observed here is the same one as described in Section 4.3.1, where the proximity of a Ritz phantom raises already stagnating residuals.

Ideal filters The production of Ritz phantoms which are closest to real unoccupied slots, in the sense that no progress can be made, requires an ideal filter where identical filter values for certain directions can be assured. If at least λ_{m_0} and λ_{m_0+1} are chosen to be part of this group, a Ritz phantom occurs due to indistinguishability between the associated directions, caused by identical scaling during the application of the projection operator. This ensures, in theory, absolutely no convergence. And indeed, these Ritz phantoms not only do not improve in terms of residual, their position on the spectral axis is also fixed.

Estimated filter values As before, if Y contains directions with no or weak representation of an eigendirection, Ritz phantoms in the main Rayleigh Ritz process of the original matrix pencil occur, but the related eigenvalues of B_U in the second implicit Rayleigh Ritz process appear to be converging to the correct filter values instead of behaving like a Ritz phantom themselves. This is true, even for artificially created, most stagnant Ritz phantoms (see above). This is related to the difference in eigenvalue distribution of (A, B) and P_f^2. Remember that $X = YW_P$, Section 4.2.2. Generally, we expect the residual

$$\left\| P_f^2 Y W_P - Y W_P \Lambda_P \right\|$$

to behave similarly to $\|AX - BX\Lambda\|$, seeing that the progression and quality of Y is linked to the main Rayleigh-Ritz process. Ritz phantoms represent eigenvalues of the projection operator that are densely clustered or identical which does, of course, not necessarily translate to eigenvalues of (A, B). Eigendirections associated with identical eigenvalues form an eigenspace such that any direction from this eigenspace is a valid eigendirection for any of these eigenvalues (see Section 1.1), which brings us back to the notion of Ritz phantoms representing more than one eigendirection. Assume a distribution of eigenvalues of the projection operator

$$f(\lambda_1) > \ldots > f(\lambda_{m_0}) = f(\lambda_{m_0+1}) = \ldots = f(\lambda_{m_0+k}) > \ldots .$$

In this example, the searchspace of size m_0 is populated by $m_0 - 1$ distinguishable directions with one remaining slot. It is impossible for one of the eigendirections of the $k+1$ identical eigenvalues, that shall here be represented by $f(\lambda_{m_0})$, to clearly occupy this slot. Since all remaining eigenvalues are smaller, the projection process flattens the single remaining vector of the searchspace set into the eigenspace of $f(\lambda_{m_0})$, making it a valid eigenvector for any of the $k+1$ identical eigenvalues of P_f^2. At the same time, this vector is almost certainly not a valid eigenvector of (A, B). As such, convergence is possible with regard to P_f^2 where it is not possible with regard to (A, B). A welcome consequence is the convergence of eigenvalues of B_U to the correct filter values, even in the presence of Ritz phantoms. This makes it possible to accurately determine the number of non-phantom eigenpairs inside the target interval in the first place. Were those filter values to converge at the same speed as the Ritz phantoms, with similarly large deviations from the actual filter values, an accurate determination of filter values and therefore the determination of precise eigencounts would be impossible.

At which speed such an underrepresented eigenspace converges is not immediately clear. A reasonable assumption is a rate of

$$\frac{f(\lambda_{m_0+k+1})}{f(\lambda_{m_0+k})},$$

assuming that no full-rank representation of the eigenspace of $f(\lambda_{m_0})$ in the search-space is necessary.

Experiment 4.10 — Convergence of filter values

We generate a definite pair of size 1000 *with evenly distributed random spectrum in* $[-1, 1]$ *and a degree-*4 *Butterworth filter for slow convergence such that the drop rates can be observed over more iterations. The filter has been modified to enforce* $f(\lambda_{m_0}) = f(\lambda_{m_0+1})$. *We compute the* $m = 20$ *innermost eigenpairs with a searchspace of size* 30 *over the course of* 50 *iterations. The results are depicted in Figure 4.12.*

Figure 4.12 shows that all distinct Ritz values converge at rates within the expected range (gray; only best and worst are shown in blue—indeed, all those Ritz values converge at the predicted rate). While the premise is not exactly the same, the application of the convergence theorem for subspace iteration Theorem 1.2 is possible here as well. Additionally, the convergence rate of the irregular Ritz value $f(\lambda_{m_0})$ of multiplicity 2 (red) approaches the predicted rate from above (black).

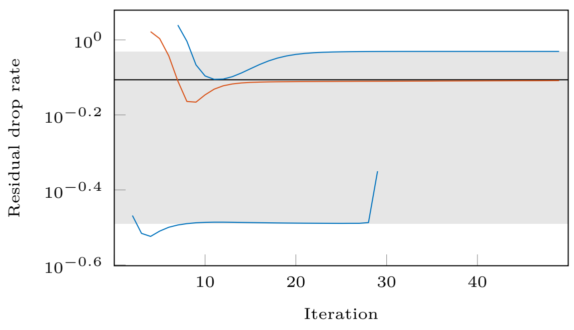


Figure 4.12: *Convergence of approximate filter values of B_U. Shown are the best and worst regularly converging values as well as the special value $f(\lambda_{m_0})$ of multiplicity 2.*

Clustering and multiplicity have similar implications for the main progression of the eigenpairs of (A, B); they are described in the following section.

4.2.5 Clustered eigenvalues

In the latter part of the previous section, we have concluded that eigenpairs that are clustered can be partially represented by a searchspace that does not fit all directions as a consequence of eigenvalue multiplicity. We refer to such a cluster as *underrepresented*. Eigendirections with identical eigenvalue form an eigenspace in which every direction is a valid eigendirection corresponding to the eigenvalue. The subspace iteration process flattens the space spanned by participating directions onto the eigenspace of the eigenvalue since the corresponding filter values are identical as well. We can consider this eigenspace—instead of a single direction—to occupy a slot of the searchspace. When eigenvalues are not identical, but in close proximity to one another, and with Ritz value locations being of finite accuracy, the concepts of multiplicity and eigenspace gain an approximative character.

4.2.5.1 Quasi-multiplicity and Quasi-eigenspaces

By Theorem 1.4 for the distances between Ritz values and their associated eigenvalues, a disk centered on the Ritz value is defined that contains at least one eigenvalue. It is therefore possible for two or more separate Ritz values to be valid representatives of the same eigenvalue or for one Ritz value to represent two or more eigenvalues, depending on distances and residuals. Whether that is truly the case cannot be discerned from the Ritz values alone. Thus, within this region of uncertainty, Ritz values and/or Ritz vectors may be disturbed without invalidating any of the above statements.

Consider two eigenvalues λ_i and λ_j as well as a Ritz value $\boldsymbol{\lambda}$ with residual r such that

$$|\lambda_i - \boldsymbol{\lambda}| < |\lambda_j - \boldsymbol{\lambda}| \leq \left\| B^{-\frac{1}{2}} r \right\|.$$

For both eigenvalues the Ritz value is a reasonable assignee. If the residual becomes smaller, the right inequality breaks no later than

$$\left\|B^{-\frac{1}{2}}r\right\| \le \frac{1}{2}|\lambda_i - \lambda_j|.$$

The same holds for

$$|\lambda_j - \boldsymbol{\lambda}| \le |\lambda_i - \boldsymbol{\lambda}| \le \left\|B^{-\frac{1}{2}}r\right\|.$$

If we assume that $\boldsymbol{\lambda} \to \lambda_i$ with decreasing r, the estimated point of separation will likely be closer to $|\lambda_i - \lambda_j|$.

In other words, reversing the interpretation, a given residual *allows* for some perturbation of the Ritz value's position. Within that leeway granted by the residual, both eigenvalues may be represented by the same Ritz pair. For two potential Ritz vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

$$\|A\boldsymbol{x}_i - B\boldsymbol{x}_i\boldsymbol{\lambda}\| + \|A\boldsymbol{x}_j - B\boldsymbol{x}_j\boldsymbol{\lambda}\| \ge \|A(\boldsymbol{x}_i + \boldsymbol{x}_j) - B(\boldsymbol{x}_i + \boldsymbol{x}_j)\boldsymbol{\lambda}\| =: \left\|B^{-\frac{1}{2}}r\right\|.$$

As such, the residual of a Ritz pair representing two eigendirections can be small if the Ritz vector is a linear combination of those Ritz vectors that, would they both be included in the searchspace set, would produce a reasonably small residual themselves. If the inseparability condition is violated, only one of those residuals would continue to drop while the other is restricted by the distance between the Ritz value and the unpaired eigenvalue. This can be extended to arbitrary numbers of simultaneously represented directions. If the effective Ritz vector could be decomposed into directions of which only one would produce a large residual, the combined Ritz vector would likely produce a similarly large residual.

With this principle of uncertainty, we can think of the approximate Ritz vectors as spanning a *quasi-eigenspace* that represents multiple eigendirections via a Ritz value of *quasi-multiplicity*, all within the range imposed by the current residual. Conjoint convergence as quasi-eigenspace is possible until the residual is small enough to separate the directions. This quasi-eigenspace should therefore be considered to occupy a slot of the searchspace until separation occurs, after which one distinct direction of the (former) quasi-eigenspace takes over the slot.

4.2.5.2 Expulsion of clustered values

The general rule of dominance, the survival of the fittest, applies here as before. Thus, a cluster cannot be decimated through replacement by less dominant directions. It occupies as many slots as possible, up to a maximum number equal to the number of clustered eigenvalues. Three distinct situations are possible.

Case 1 All clustered values fit the searchspace.

Case 2 The cluster is the least dominant element and not all clustered values fit the searchspace.

Case 3 The cluster is the most dominant element, but not all clustered values fit the searchspace.

In the first case, all clustered eigenvalues are represented in the searchspace by a distinct direction in form of a Ritz vector. Peculiarities involving the convergence under these conditions will be discussed at a later point, see Section 4.2.5.3. Cases two and three are fundamentally identical. A fourth case, where the cluster is excluded from the searchspace completely is of no relevance here.

The situation of the cases two and three translates the convergence behavior of the approximate filter values of B_U examined in the previous section to the original eigenvalue problem and permits the following prognosis. While clustered filter values do not imply clustered eigenvalues, the reverse is true for most non-ideal filters. The filter values of clustered eigenvalues are themselves clustered and, albeit differences in density exist, convergence is influenced in similar manner.

Experiment 4.11 — Convergence of clustered values with under-representation

We generate a definite pair of size 1000 *with an evenly spaced spectrum in an interval of size two, centered around one, to make sure that no clustering apart from the intentionally injected cluster is perturbing the results. A Butterworth filter guarantees smooth filter value distribution and direct translation of eigenvalue positions to magnitude of the corresponding filter values. A cluster of size* $k = 10$ *and inner separation* 10^{-10} *is added as the least dominant element in the searchspace. The searchspace and interval are chosen such that only one slot represents the cluster. The overall searchspace size is* 20.

Figure 4.13 shows the residual convergence history (left) and a selection of achieved residual drop rates (right). For Figure 4.14, the experiment was repeated with less densely clustered values at a density of 10^{-5}.

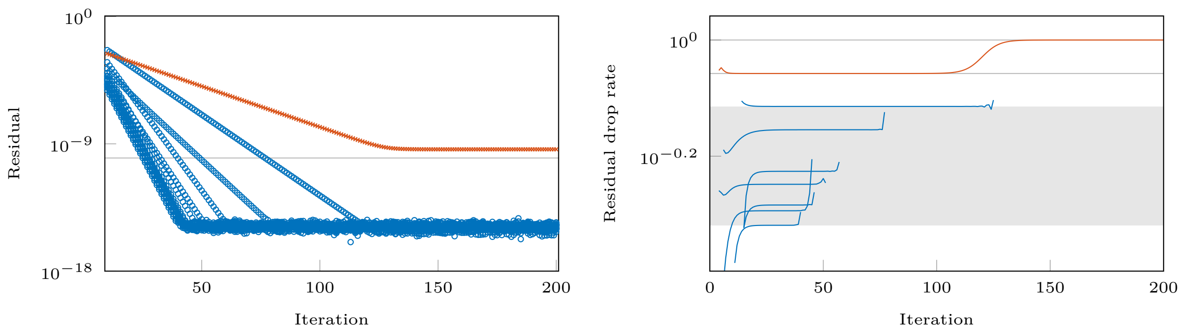


Figure 4.13: *Convergence of clustered values with under-representation. The cluster of size* 10 *and density* 10^{-10} *is represented by only one Ritz pair (red).*

For the experiment depicted in Figure 4.13, convergence proceeds at the expected rates (gray area) for non-clustered slots (blue). The slot representing the cluster (red) converges at what appears to be

$$\frac{f(\lambda_{m_0+k})}{f(\lambda_{m_0})} \approx \cdots \approx \frac{f(\lambda_{m_0+k})}{f(\lambda_{m_0+k-1})},$$

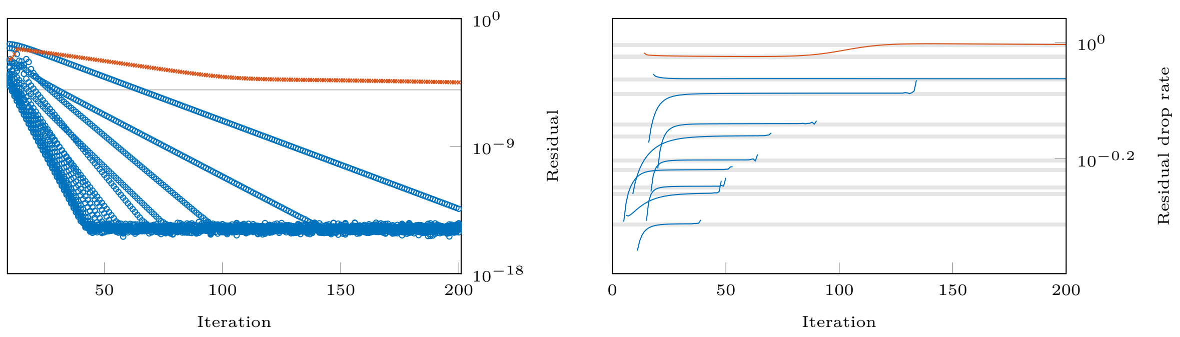


Figure 4.14: *Convergence of clustered values with under-representation. The cluster of size* 10 *and density* 10^{-5} *is represented by only one Ritz pair (red).*

indicated by the lower one of the two gray lines, until the residual approaches the cluster separation after which the convergence rate changes to the expected value (upper gray line). The change in drop rate happens gradually, without any direct jump.

For the experiment depicted in Figure 4.14, convergence of all Ritz pairs appears to be influenced by all clustered values. Due to the clustered values being indistinguishable, this makes sense. The regular Ritz values converge at a rate between

$$\frac{f(\lambda_{m_0+1})}{f(\lambda_i)} \quad \text{and} \quad \frac{f(\lambda_{m_0+k-1})}{f(\lambda_i)},$$

indicated by the gray regions, with a tendency towards worse drop rates, while the Ritz value representing the cluster (red) converges at a rate between

$$\frac{f(\lambda_{m_0+k})}{f(\lambda_{m_0})} \quad \text{and} \quad \frac{f(\lambda_{m_0+k})}{f(\lambda_{m_0+k-1})},$$

again indicated by the corresponding gray regions before and after the split. The arithmetic or geometric mean of the drop rates appears to be a reasonably good overall estimate for the real drop rates, but both are not absolutely accurate.

It is unlikely that the filter values would not be clustered with the eigenvalues being clustered. This situation might occur if the clustered values are located directly on a very sharp flank of a high-degree filter. To answer two questions arising in such scenarios, the modifications of the filter and its application to the spectrum are required.

Will drop rates for regular Ritz values change after the separation of the cluster?
In order to test this, we need the representative Ritz value of the cluster to converge faster, such that the remaining Ritz values have not yet reached saturation when the cluster separates but without having the other directions of the cluster occupying the searchspace. By manually modifying the eigenvalues of the approximate projection operator and, with this, the dominance of the different directions, the Ritz value

representing the cluster (and only this single value) can be allowed to converge much faster than the remaining Ritz values (and the rest of the cluster), at least until the cluster separates. There seems to be no change in the convergence rates of the regular Ritz values at this point or later.

What happens if the filter values are not clustered? Again, this requires the manual modification of the eigenvalues of the approximate projection operator, specifically assigning different convergence speeds to directions of the cluster. As a side effect, the ranges for possible convergence speeds from Experiment 4.11 become larger, given that the convergence speeds of the cluster values are now spread over a wider range. The separation of the cluster can be observed as before. Convergence rates are higher in accordance with the chosen filter values and within the predicted ranges mentioned above. The Ritz value of the cluster converges with a tendency towards the best possible drop rate before separation and with a tendency towards the worst possible drop rate after separation.

Consequences for estimated filter values The observations made here apply to both the primary Rayleigh-Ritz process on (A, B) and the implicit secondary process on P_f^2. With the emergence of a Ritz phantom as described in Section 4.2.4 the associated estimated filter values obtained from B_U would accordingly be assumed to converge slowly. However, if the filter values are identical, faster convergence of this underrepresented cluster has been confirmed possible. We now have to revise this statement in that this accelerated convergence is only possible until a certain residual is reached. This residual limit is related to the density of the cluster, i.e., the separation of the values. If the values are identical, convergence is generally accelerated. With non-zero separation, accelerated convergence can only be maintained up to a certain residual. This is in general a favorable result; the smaller the separation is and the slower convergence should be, the longer these values can converge at accelerated speeds. We therefore can expect improved accuracy for estimated filter values of Ritz phantoms.

4.2.5.3 Clustered super convergence

If another Ritz value $\boldsymbol{\lambda}'$ is introduced that fulfills the same conditions as $\boldsymbol{\lambda}$ from before, exchanging both in the relations from the start of this section does not change the bounds on the residual. For all that matters, $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ may be considered interchangeable. During subspace iteration, the emergence of two Ritz values that are associated with the same eigenvalue is prevented by either orthogonalization or the Rayleigh-Ritz procedure. The largest possible distance between $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ such that both of them still may represent both eigenvalues is

$$|\boldsymbol{\lambda} - \boldsymbol{\lambda}'| \leq \left\|B^{-\frac{1}{2}} r\right\| + \left\|B^{-\frac{1}{2}} r'\right\| - |\lambda_i - \lambda_j|.$$

We may think of both Ritz values simultaneously representing both eigenvalues via linear combination of their Ritz vectors. While both vectors are kept orthogonal among themselves and to all other vectors by the eigensolver for the reduced-size Ritz eigenproblem, they could, potentially, freely rotate within the span of their Ritz vectors.

Thus, Ritz values in close proximity may be considered as being of a quasi-multiplicity larger than one with their Ritz vectors forming a quasi-eigenspace. Depending on the spectral distribution, these quasi-eigenspaces get subdivided as the corresponding residuals are surpassed. Some peculiarities in convergence behavior can be attributed to this behavior.

Experiment 4.12 — Clustered super convergence

We generate a definite pair of size 1000 *with evenly spaced spectrum in an interval of size two, centered around one. A Butterworth filter is used for slower convergence over many iterations. A cluster of size k with density d is added as least dominant element in the searchspace; searchspace size and interval are chosen such that values inside the target interval and the clustered values are occupying all slots. Figure 4.15 shows the convergence history of two scenarios, $k = 2$ with $d = 10^{-10}$ on the left and $k = 10$ with $d = 10^{-15}$ on the right.*

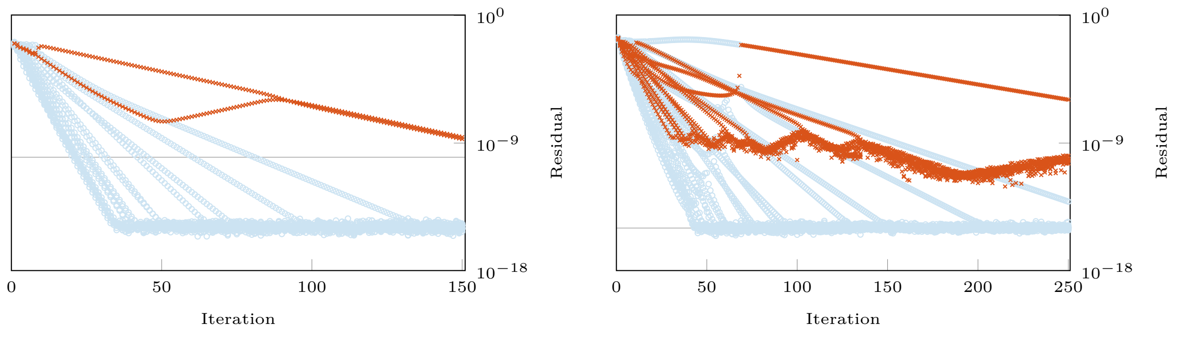


Figure 4.15: *Convergence of clustered values with full representation. The cluster (red) of size $k = 2$ (left) / $k = 10$ (right) and density $d = 10^{-10}$ (left) / $d = 10^{-15}$ (right) shows atypical convergence behavior.*

Only one Ritz pair among the clustered values converges at the estimated rate while the remaining values of the cluster converge at much higher speeds, but only until a certain residual is reached. Beyond this point they seem to reunite with the next slowest value from the cluster, a process during which the residual increases again. The residual at which the turn over occurs is influenced by the cluster separation.

Since the spectrum of any matrix can be considered more or less clustered, the mechanism described here applies to all Ritz pairs of all iterations. Starting from few quasi-eigenspaces comprised by large residuals, continued iteration repeatedly separates them, based on the spectral distribution, into more and more quasi-eigenspaces until they—ideally—represent only a single direction.

If convergence speed is increased, this also applies to the slowest converging clustered values, allowing them to catch up faster. This, in turn, reduces the rebound time of clustered values with accelerated convergence. Since accelerated values rebound around a certain residual, the effect gradually vanishes with increased convergence speeds.

4.2.5.4 Disturbances through Ritz phantoms

The analysis of the behavior of clustered Ritz values may also shed a light on why the convergence of Ritz values is disturbed when a Ritz phantom is in close proximity. In the reduced eigenproblem (A_U, B_U), the Ritz phantom and the nearby regular Ritz value form a cluster with possibly very small distance. The situation differs from before in that one residual is considerably larger than the other. If we assume the two associated primitive Ritz vectors to span a quasi-eigenspace with regard to the (almost equal) Ritz values, both influencing each other does not seem impossible. The primitive Ritz vectors may change direction within the quasi-eigenspace, explaining a temporary increase in residual. This may well be a byproduct of the eigensolver enforcing orthogonality among both vectors.

4.2.6 Comparison of filter functions

The speed of convergence that any filtering function will provide can be estimated from the filtering function in combination with a spectral distribution and, if the filter is decaying, a searchspace size. For non-decaying filters it can be assumed that residual drop is limited by the maximum stopband gain.

A comparison of filtering functions may be conducted by fixing the workload; in this case, the number of linear solves, i.e., the degree of the filter. Filters based on polynomial approximation have to be excluded from this comparison, since they rely on matrix vector multiplication alone. This also neglects the important aspect of the difficulty of the linear systems arising in the application of the filter, which is related to the number of iterations an iterative linear solver requires to reach a suitable residual.[5] As an indicator, we will include the minimum distance of the filter poles to the real axis. To conduct the comparison, we fix the transition bandwidth to obtain the associated stopband gain. The Cauchy filter has to act as reference since there these values are dictated by the filter degree.

4.2.6.1 Definition of transition band and stopband gain

By construction, the transition band for filters from electronic filter design (Section 2.4) is known; for Zolotarev filters, the transition band is explicitly given through the parameter κ. For Cauchy and Chebyshev polynomial filters, the transition band has to be derived from the maximum oscillation amplitude next to the filter's flanks and is defined solely through the filter's degree. For the following comparison, we find these values via a root finding algorithm applied to the derivative of the filtering

5) With iterative linear solvers, where matrix vector multiplications are the major workload, comparison with polynomial based filters is possible again.

function for Cauchy filters on a circular contour over $[-1, 1]$ with Gauss-Legendre rule of degrees 2–32, see Table 4.3. The deviations from zero respectively one are identical. Since the oscillations decay in stop- and passband, we will list first (δ_1) and second order deviations (δ_2), the sum of which gives the total oscillation range. See Figure 4.16 for clarification.

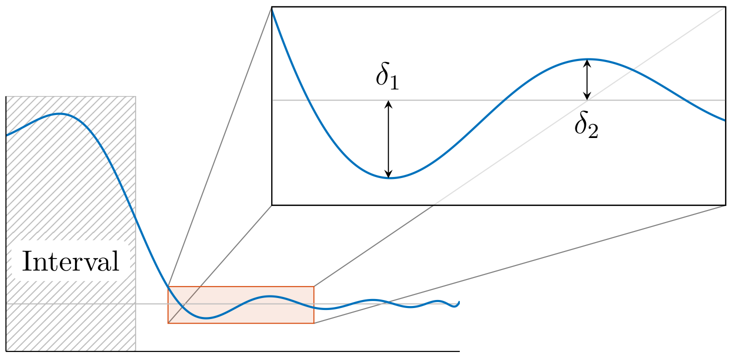


Figure 4.16: *Exaggerated depiction of the filter deviations δ_1 and δ_2 from zero in the stop-band. The deviations from one in the passband are identical in magnitude, δ_1 being the positive deviation, δ_2 the negative deviation.*

Degree	δ_1	δ_2	f_p	f_s
2	0	0	—	—
4	1.513478030173369e−02	0	4.905288111002327e−01	1.857192322077595e+00
6	1.957776350723739e−02	2.453811436999645e−04	7.051153779587295e−01	1.351019874275442e+00
8	2.150652768468497e−02	4.858103437515304e−04	8.119039455581168e−01	1.195912530809244e+00
10	2.252169669840243e−02	6.567219584557721e−04	8.706581789242298e−01	1.126116594149748e+00
12	2.311814598731410e−02	7.719818204051654e−04	9.059541901132968e−01	1.088328611863879e+00
14	2.349707556942965e−02	8.509878644063773e−04	9.286743279905780e−01	1.065444703868107e+00
16	2.375234504673240e−02	9.067621283803950e−04	9.441095469580231e−01	1.050492409508766e+00
18	2.393230151981124e−02	9.473203792700891e−04	9.550535057171183e−01	1.040167375609645e+00
20	2.406383706919518e−02	9.776154002443527e−04	9.630855245051573e−01	1.032730454306030e+00
22	2.416285334169493e−02	1.000783187234927e−03	9.691500447281927e−01	1.027192254396067e+00
24	2.423923276491330e−02	1.018867359921322e−03	9.738388114970027e−01	1.022954959994240e+00
26	2.429937532521365e−02	1.033237762895239e−03	9.775373901448535e−01	1.019639521398144e+00
28	2.434757193162140e−02	1.044836896434254e−03	9.805054996212751e−01	1.016995915102813e+00
30	2.438678575722864e−02	1.054328765719342e−03	9.829231558326313e−01	1.014853687800546e+00
32	2.441911592986706e−02	1.062191242318128e−03	9.849182595835504e−01	1.013093328470374e+00

Table 4.3: *Cauchy filter deviations and resulting transition bands.*

4.2.6.2 Fair conditions

Cauchy and Zolotarev filters oscillate about zero in their stopbands. Scaling and shifting the Cauchy filter to fit the value range between zero and one with the help of Table 4.3 for a given degree,

$$\frac{f(\lambda)+\delta_1}{1+2\delta_1},$$

as is the case with all filters from electronic filter design, eliminates its decaying property. With this, the comparison would be flawed. Note that the scaling itself has no influence on the resulting estimated convergence speed; it cancels out in the division. A proper shift for the electronic filters, however, reduces the maximum stopband gain by one half, since we don't have to prevent negative filter values in the context of subspace iteration. We therefore shift all filters such that they oscillate about zero in the stopband and rescale them to oscillate about one in the passband. Additionally, for the Cauchy and Zolotarev filters to match the electronic filters, the transition band is defined to start at $f_p = 1$. The modified values for f_s of the Cauchy filters are easily computed from Table 4.3,

$$f'(\lambda) = f(f_p\lambda) \quad \longrightarrow \quad f'_p = 1,\ f'_s = \frac{f_s}{f_p}$$

such that $f'(1) = 1-\delta_2$ and $f'(f'_s) = \delta_1$. For the Zolotarev filter, this transformation is equivalent to a scaling with G, if $[1/G, G]$ is the untransformed transition interval. Let f'_s be the transformed end of the desired transition band. Then G is given by solving

$$G^2 - 1 = f'_s - 1 \quad \Rightarrow \quad G = \sqrt{f'_s}.$$

From G, the elliptic modulus κ can be computed and the filter poles and weights have to be transformed per scaling with G. The new transition interval is then $[1, G^2]$. Of course, fixing f_p in this way has no influence on the values of the filter and only plays a role for convergence speed if we assume λ_{m_0+1} to be located within some distance of f'_s.

For type-I Chebyshev and elliptic filters, larger oscillations in the passband can be allowed to improve transition band and maximum stopband gain at the price of reducing the wost-case residual drop rate inside the target interval by a small amount. Controlling the passband ripple is a unique feature to some electronic filters that may benefit them in this comparison.

We therefore fix the allowed oscillation in the passband to 10%. This choice is rather arbitrary. Even large oscillations in the passband only have a comparatively small effect on convergence, as long as the filter values do not drop to values too close to zero. For the sake of defining a compact interval where eigenpairs inside this interval converge faster than all other eigenpairs, the value of the filter function inside the interval should not drop below the function's value at the interval boundaries. Beyond this point, the approximated spectral region is no longer an interval and a collection of chunks from the original interval would be missing. If the eigenvalues in the transition band are more dominant than the eigenvalues dampened by the oscillations, the latter might not even appear in the searchspace, depending on the searchspace size. This would also affect the evaluation of approximate filter values to obtain an eigencount, following Section 4.2.2.4 and Section 4.2.4.1, which relies on comparing estimated filter values with the boundary values of the filter. For Cauchy filters, something similar is possible by changing the integration contour to an ellipse. This case shall be considered separately.

4.2.6.3 Strict searchspace size

In this first comparison, we assume $\lambda_{m_0+1} = f_s$ such that the value of the filter function at that position is the maximum stopband gain of the filter. Whether the transition band does contain eigenpairs is not of importance, the searchspace size is assumed to be chosen accordingly. We summarize the best and worst possible residual drop rates of eigenpairs inside the passband,

$$\min_{|\lambda|<f_p} \left| \frac{f_s}{\lambda} \right| \quad \text{and} \quad \max_{|\lambda|<f_p} \left| \frac{f_s}{\lambda} \right|,$$

in Tables 4.4 and 4.5, respectively. The differences arise solely due to the oscillations in the passband. Table 4.6 lists the minimum distance of the filter poles to the real axis.

This setup does not reflect the decaying property of some of the filters, which is why the improvements with increasing degree are small in some cases. Furthermore, it is clear that oscillations in the passband have only small influence on the difference in convergence, unless the filter gain differs in orders of magnitude inside the passband. It should be noted that, without a shift of the filter function to halve the stopband gain, Chebyshev filters of type I and II produce the same drop rates under these conditions.

Degree	Cauchy	Zolotarev	Butterworth	Chebyshev (I)	Chebyshev (II)	Elliptic
4	1.4909133e−02	8.9573292e−03	4.1961965e−02	1.1619177e−02	5.8435374e−03	1.4682727e−03
6	1.9201834e−02	5.5680001e−03	1.5390488e−01	1.7639255e−02	8.8981059e−03	5.6399569e−04
8	2.1053734e−02	2.9618265e−03	2.8883734e−01	1.9448193e−02	9.8195835e−03	1.5881780e−04
10	2.2025641e−02	1.5270911e−03	4.0719232e−01	1.9799866e−02	9.9989219e−03	4.2102859e−05
12	2.2595773e−02	7.8596795e−04	4.9907712e−01	1.9655660e−02	9.9253752e−03	1.1136796e−05
14	2.2957638e−02	4.0746514e−04	5.6801879e−01	1.9351791e−02	9.7704332e−03	2.9909292e−06
16	2.3201260e−02	2.1336735e−04	6.1984046e−01	1.9011015e−02	9.5967296e−03	8.1981043e−07
18	2.3372933e−02	1.1291903e−04	6.5938901e−01	1.8678834e−02	9.4274641e−03	2.2956452e−07
20	2.3498375e−02	6.0380746e−05	6.9015390e−01	1.8371113e−02	9.2707135e−03	6.5632942e−08
22	2.3592784e−02	3.2604768e−05	7.1455196e−01	1.8091950e−02	9.1285517e−03	1.9136524e−08
24	2.3665596e−02	1.7767883e−05	7.3425243e−01	1.7840773e−02	9.0006763e−03	5.6827601e−09
26	2.3722923e−02	9.7652415e−06	7.5042109e−01	1.7615292e−02	8.8859101e−03	1.7165124e−09
28	2.3768857e−02	5.4095315e−06	7.6388550e−01	1.7412734e−02	8.7828336e−03	5.2674027e−10
30	2.3806228e−02	3.0187261e−06	7.7524374e−01	1.7230361e−02	8.6900471e−03	1.6402972e−10
32	2.3837036e−02	1.6961156e−06	7.8493578e−01	1.7065671e−02	8.6062717e−03	5.1782722e−11

Table 4.4: *Best residual drop rates.*

Degree	Cauchy	Zolotarev	Butterworth	Chebyshev (I)	Chebyshev (II)	Elliptic
4	1.5134780e−02	9.1207239e−03	4.6624406e−02	1.2910197e−02	6.4970378e−03	1.6316803e−03
6	1.9582568e−02	5.6307036e−03	1.7100542e−01	1.9599173e−02	9.8965688e−03	6.2670115e−04
8	2.1516980e−02	2.9794759e−03	3.2093037e−01	2.1609104e−02	1.0922565e−02	1.7646734e−04
10	2.2536496e−02	1.5317694e−03	4.5243591e−01	2.1999851e−02	1.1122269e−02	4.6781173e−05
12	2.3136006e−02	7.8720538e−04	5.5453014e−01	2.1839622e−02	1.1040370e−02	1.2374233e−05
14	2.3517088e−02	4.0779747e−04	6.3113199e−01	2.1501990e−02	1.0867835e−02	3.3232557e−06
16	2.3773902e−02	2.1345844e−04	6.8871162e−01	2.1123350e−02	1.0674415e−02	9.1090056e−07
18	2.3954994e−02	1.1294453e−04	7.3265446e−01	2.0754260e−02	1.0485944e−02	2.5507169e−07
20	2.4087385e−02	6.0388038e−05	7.6683767e−01	2.0412348e−02	1.0311414e−02	7.2925492e−08
22	2.4187059e−02	3.2606894e−05	7.9394662e−01	2.0102166e−02	1.0153133e−02	2.1262804e−08
24	2.4263954e−02	1.7768514e−05	8.1583604e−01	1.9823081e−02	1.0010763e−02	6.3141779e−09
26	2.4324508e−02	9.7654322e−06	8.3380121e−01	1.9572547e−02	9.8829912e−03	1.9072360e−09
28	2.4373037e−02	5.4095900e−06	8.4876167e−01	1.9347482e−02	9.7682365e−03	5.8526696e−10
30	2.4412524e−02	3.0187443e−06	8.6138193e−01	1.9144846e−02	9.6649400e−03	1.8225524e−10
32	2.4445081e−02	1.6961213e−06	8.7215087e−01	1.8961857e−02	9.5716771e−03	5.7536358e−11

Table 4.5: *Worst residual drop rates.*

Degree	Cauchy	Zolotarev	Butterworth	Chebyshev (I)	Chebyshev (II)	Elliptic
4	1.2561759e+00	1.1802759e+00	1.2247448e+00	7.3523425e−01	3.1011790e−01	7.1317626e−01
6	4.9170871e−01	4.1075537e−01	7.2112478e−01	3.2197743e−01	2.8438360e−01	2.7081071e−01
8	2.6653518e−01	1.9377919e−01	5.0363972e−01	1.8002708e−01	2.0045553e−01	1.2808220e−01
10	1.6865341e−01	1.0833140e−01	3.8495203e−01	1.1488016e−01	1.4144062e−01	6.9510912e−02
12	1.1686884e−01	6.7407208e−02	3.1082535e−01	7.9647817e−02	1.0354183e−01	4.1423401e−02
14	8.5989225e−02	4.5130458e−02	2.6033409e−01	5.8458425e−02	7.8610566e−02	2.6420239e−02
16	6.6026348e−02	3.1867683e−02	2.2380814e−01	4.4728021e−02	6.1549697e−02	1.7745991e−02
18	5.2345685e−02	2.3431800e−02	1.9619308e−01	3.5324756e−02	4.9429613e−02	1.2416475e−02
20	4.2546640e−02	1.7786386e−02	1.7460013e−01	2.8603806e−02	4.0535907e−02	8.9798936e−03
22	3.5280066e−02	1.3852990e−02	1.5726237e−01	2.3633841e−02	3.3827584e−02	6.6748708e−03
24	2.9738430e−02	1.1021085e−02	1.4304008e−01	1.9855362e−02	2.8647941e−02	5.0772985e−03
26	2.5413497e−02	8.9259746e−03	1.3116587e−01	1.6915768e−02	2.4567888e−02	3.9389176e−03
28	2.1972009e−02	7.3399604e−03	1.2110451e−01	1.4583884e−02	2.1298155e−02	3.1082279e−03
30	1.9187858e−02	6.1155171e−03	1.1247154e−01	1.2703022e−02	1.8638292e−02	2.4894380e−03
32	1.6903131e−02	5.1540023e−03	1.0498375e−01	1.1163930e−02	1.6445990e−02	2.0200974e−03

Table 4.6: *Minimum pole distances to the real axis.*

4.2.6.4 Relaxed searchspace size

Convergence rates depend heavily on the spectral distribution. Whether one filter will perform better than others hinges on the position of λ_{m_0+1} and, with this, also on the size of the searchspace. To emphasize at what point one filter becomes better than another, we compare filtering functions in a suitable representation. Since the most dominant directions occupy the searchspace, the sorted filtering functions give more of an idea of possible convergence speeds, given that they are directly proportional to $f(\lambda_{m_0+1})$. Figure 4.17 shows the filters at degree 16. The first plot depicts the filters in a conventional logarithmic way. The transition band is indicated as well. It is not easy to infer which directions would populate the searchspace and what drop rates can be expected given a certain searchspace size, even though the dampening allows for some assumptions. The second plot assumes that the target interval covers the central 2% of the spectrum and depicts the first 1 000 values of the filter function, evaluated at 5 000 equidistant points along the spectral line and sorted descendingly. The coverage assumption assures the consideration of far-away filter values in the sorted representation; otherwise, low filter values gather at the right side of the plot, indicating better drop rates, although this is only realistic if the spectral range matches the plot range.[6] The x-axis remains unlabeled as it does not represent any meaningful quantity anymore. The corresponding transition band, however, can still be indicated since values to the left or right are guaranteed to be larger or smaller, respectively. Eigenvalues are considered to be populating the available searchspace starting from the left in this representation. The third plot shows an identical representation of the filter functions, but here each line indicates an eigenvalue of a randomly chosen exemplary spectral distribution.[7] Choosing any such line (e.g., the line highlighted in red) as λ_{m_0+1} implies that all eigenvalues (lines) to the left are contained in a searchspace of suitable size and the residual drop rates to be expected are chiefly influenced by the filter's value at that position, the filter with the lowest value resulting in the best drop rates. It is then obvious that choosing larger searchspace sizes (lines farther to the right) influences which filter has to be considered best.[8] Filters having the benefit of decay eventually outperform other filters if the searchspace is chosen large enough and the spectral distribution allows it. Choosing different parameters mixes up these relations further. This includes elliptic contours for Cauchy filters and passband oscillations for the electronic filters, but in particular the transition band, which we have fixed for all these comparisons. In combination with enlarged searchspaces however, tight transition bands are of lesser importance and relaxing the constraints improves overall dampening. Filters which are influenced by this parameter are the non-decaying filter types (Zolotarev, Chebyshev-II, and elliptic). Other filters are either smooth or there is no direct way to define a transition bandwidth (Cauchy). Figure 4.18 compares these parameter modifications for filters of degree 16. Throughout, increasing passband oscillations,

[6] For equioscillating filters for example, the potential eigendirections associated with the smaller filter values caused by the oscillations in the stopband will never appear in the searchspace since almost all other eigenvalues are more dominant. This can only be the case if the searchspace is large, close to the size of the eigenproblem.

[7] These semantics need further clarification. Since the ordering by filter value is individual for each filter, each line represents a *different* eigenvalue for each filter function, i.e., the original positions of the eigenvalues are different for each filter. This defines the occupancy of the searchspace and reflects the filter properties when the searchspace size is increased in a more general way, removing the dependency on spectral distribution to a degree. A similar comparison for one specific spectral distribution will differ from case to case. Such a comparison would be possible by sorting only the eigenvalues of the projection operator matrix.

[8] The strict comparison of Section 4.2.6.3 corresponds to placing the red line at the position corresponding to f_s, placing a line at position 0 for best drop rates, and placing a line at the position corresponding to $f_p = 1$ for worst drop rates.

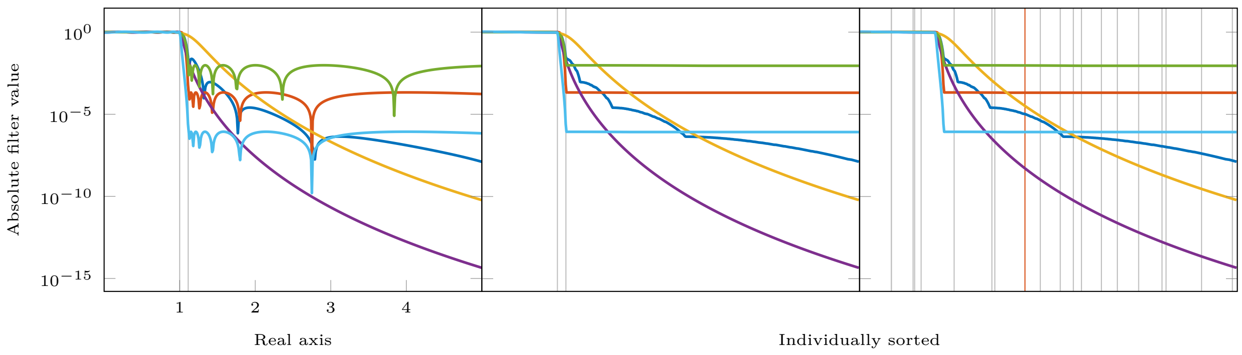


Figure 4.17: *Comparison of filters of degree* 16. *Depicted are Cauchy (blue), Zolotarev (red), Butterworth (yellow), Chebyshev-I (purple), Chebyshev-II (green), and elliptic (light blue) with the setup and parameters chosen as described.*

be it via elliptic contours for Cauchy filters or explicitly for electronic filters, improves the dampening properties of the filter. The same holds for wider transition bands. The Zolotarev filter has to maintain its low passband oscillation strength due to which the dampening properties in the transition band towards the passband become considerably weaker, a constraint that does not apply to filters where passband oscillations can be specified. We may summarize the findings as follows. Allowing

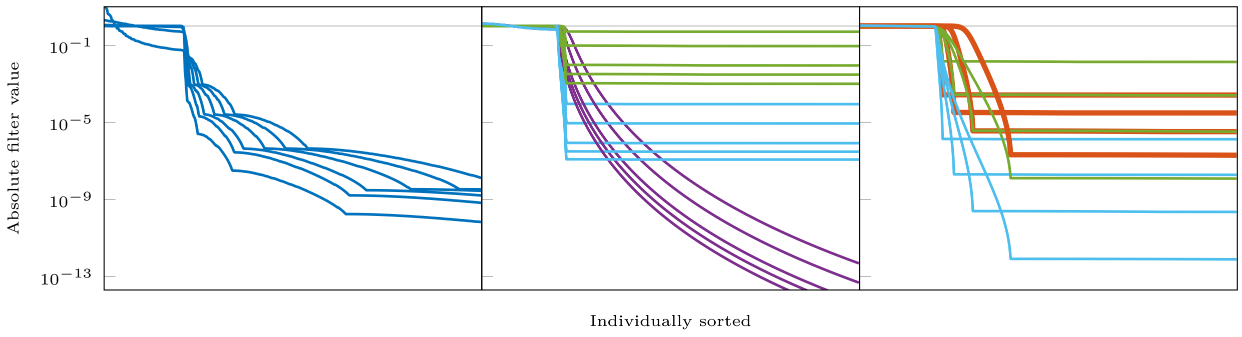


Figure 4.18: *Filter parameter modification; all filters are of degree* 16. *Left: Semi-minor axis length* $\ell = 1.0, 0.75, 0.5, 0.25, 0.1, 0.01$ *for a Cauchy filter. Center: Passband oscillations of* $0.1, 1, 10, 25, 50$ *percent for a Chebyshev-I (purple), Chebyshev-II (green), and elliptic (light blue) filter. Right: Transition bandwidth of* $0.1, 0.25, 0.5, 2.0$ *for a Zolotarev (red), Chebyshev-II (green), and elliptic (light blue) filter.*

passband oscillations is a good way to improve filtering properties. Even passband oscillations at 50% of the total (unscaled) filter gain do not change the filtering magnitude for eigenvalues inside the target interval by much. For Cauchy filters, the strength of oscillation for a certain semi-minor axis length depends on the degree, higher degrees allowing narrower ellipses without impeding the filtering magnitude in the passband. The transition bandwidth can be chosen very lax for compatible filter types, if the searchspace is large enough. It requires some knowledge about the spectral distribution to make sure that λ_{m_0+1} is located beyond f_s in the sorted representation of the filter for optimal performance. Filters that allow the separate modification of these parameters have to be considered superior.

4.2.6.5 Choice of interval

Given a transition band, the actual position of the target interval's boundaries may be chosen at some position inside of it. If all eigenpairs inside the target interval should converge with approximately equal speeds, interval boundaries should be mapped close to f_p. This is because eigenvalues in the transition band converge at slower speeds and choosing the interval to contain (parts of) the transition band will include these eigenvalues in the interval. This choice, however, can diminish overall convergence speed since the filter function is expanded slightly, resulting at larger filter values at λ_{m_0+1}. In the literature, typical choices are the half-points of the filter function, if they can be uniquely determined, see, e.g., [Krä+13; Krä14; Pol09; Güt+15]. This will cause eigenvalues located close to the interval boundaries to converge slower while still being considered inside the interval.

4.2.7 Difficulty of linear systems

A factor that, so far, did not play a role in the analysis conducted and has yet to be explored is the inherent difficulty of linear systems depending on the pole location when employing iterative linear solvers. Linear systems arising in the context of spectral filtering have the form

$$(zB - A)X = BY \iff \left(zI - B^{-1}A\right)X = Y \tag{4.3}$$

where shifts z are the poles of the partial fraction form of the filtering function. The spectrum of the resulting matrix operator for the linear systems then is a mapping of the spectrum of the matrix pencil where z is mapped to zero and, consequentially, the eigenvalue of the matrix pencil closest to z becomes the smallest magnitude eigenvalue of the effective operator, its magnitude being the distance of its preimage to z. The system matrix thus is not Hermitian anymore, unless z is real.

For normal matrices and thus for Hermitian matrices, the condition number is typically defined as [Saa03; GV13]

$$\kappa(A) = \left|\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}\right|$$

and a general indicator of how well an iterative linear solver would perform, i.e., how quickly it would converge. The condition number is a quantification of the condition of a problem that is to be solved by a numerical algorithm. It describes the sensitivity of the solution of the problem with respect to perturbations of the input data. The condition number often appears during the analysis of the stability of numerical algorithms and, for linear systems, relates the relative residual to the relative error. The system matrix from Equation (4.3), however, is not normal and the condition number can be written

$$\kappa(A) = \left\|A\right\|\left\|A^{-1}\right\|$$

or in terms of singular values. It is therefore difficult to relate the possible performance of an iterative solver to the distance of z to an eigenvalue alone.

Figure 4.19 conveys a notion of the difficulty of shifted linear systems in terms of GMRES iterations (without restarts) for an example matrix pencil of size 1000. The

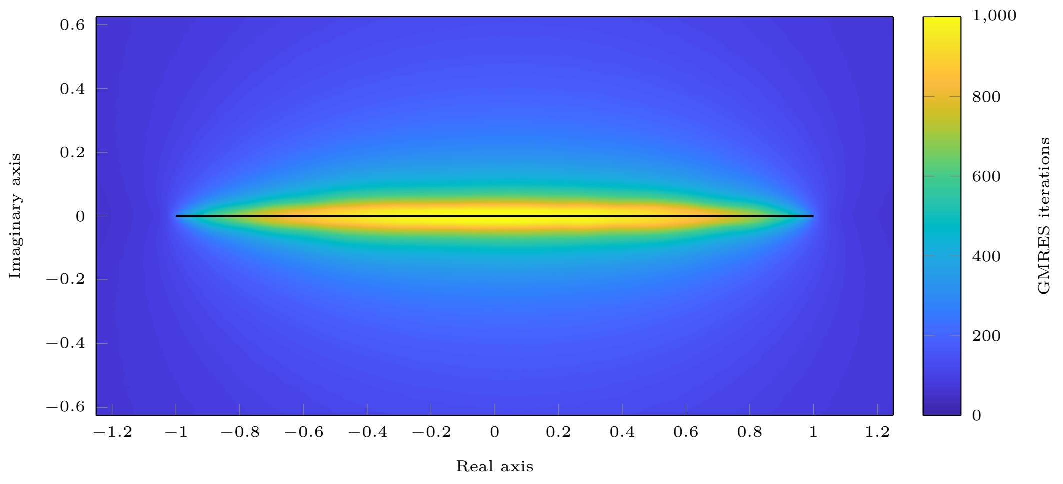


Figure 4.19: *Difficulty of an exemplary linear system in term of pole position z in the complex plane.*

spectrum was chosen evenly distributed in $[-1, 1]$, indicated as a black line. Plotting the number of required iterations in a contour plot reveals that not only the smallest distance to any of the eigenvalues plays a role, but the proximity to the spectrum as a whole, increasing the number of iterations for shifts closer to the center of the spectrum. The absolute position of the spectrum has no influence on this condition.

Figure 4.20 is intended to provide a more detailed analysis of the impact of multiplicity and density on the difficulty of the linear systems. To this end, the spectrum has been modified to free the innermost third of the spectral range and place a cluster of 2, 10, and 100 eigenvalues (plot rows) at the center. Furthermore, the density of the clustered eigenvalues is varied as 0, 10^{-10}, 10^{-5}, and 10^{-3} (plot columns). Since changes in Figure 4.20 are subtle, we have zoomed in on the area around the clustered eigenvalues at the center with increasing resolution towards the center in Figure 4.21. For visibility, it was necessary to apply different color range mappings to each plot. The number of iterations for the shifts appearing within each row of Figure 4.21 is roughly equal, while the range of required iterations between rows, from top to bottom, is mostly disjoint, each row starting around where the previous row stopped. It is obvious that the multiplicity of the cluster clearly has an impact on the required number of iterations, even if clustered values

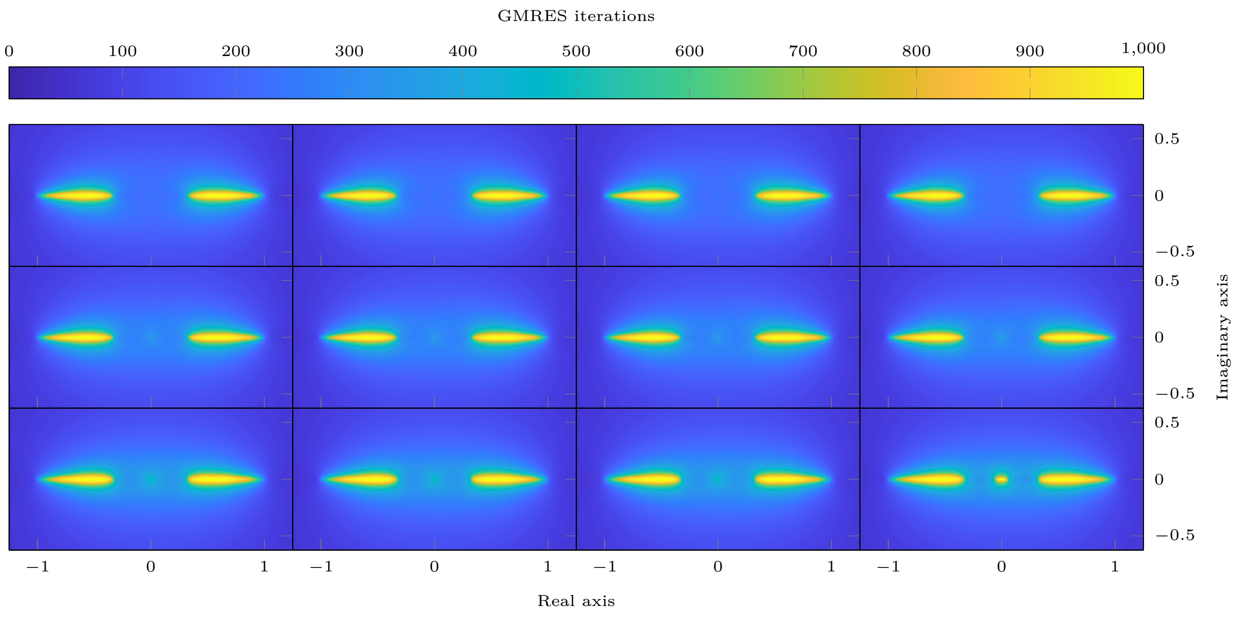


Figure 4.20: *Difficulty of linear systems per pole position z in the complex plane in terms of GMRES iterations required for reaching a relative residual of* 10^{-10}. *Rows: clusters of* 2, 10, *and* 100 *eigenvalues. Columns: inner cluster separation* 0, 10^{-10}, 10^{-5}, *and* 10^{-3}.

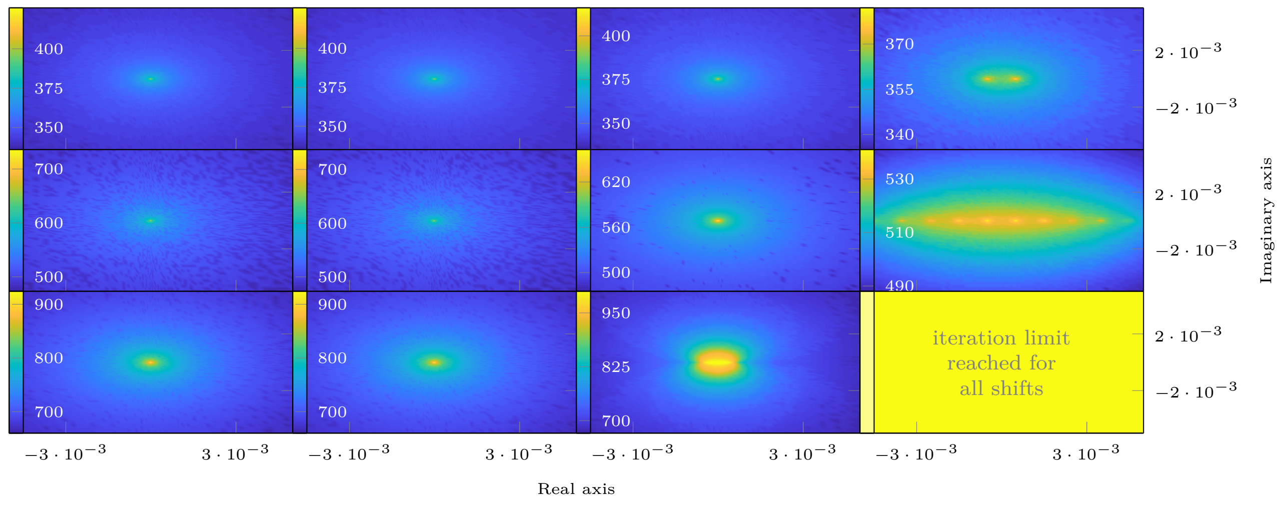


Figure 4.21: *Difficulty of linear systems (zoomed) per pole position z in the complex plane in terms of GMRES iterations (refer to the color bars) required for reaching a relative residual of* 10^{-10}. *Rows: clusters of* 2, 10, *and* 100 *eigenvalues. Columns: inner cluster separation* 0, 10^{-10}, 10^{-5}, *and* 10^{-3}.

are not separated in any way. The inner separation among the clustered eigenvalues influences in what proximity the effect is recognizable, but also the overall difficulty, increasing the number of iterations required in an area around the clustered values. Densely clustered values contract the area of increased difficulty around the eigenvalue cluster. The resolution for the plots is limited, even in the zoomed-in version, such that local differences in iteration number cannot always be resolved.

Approaches for resolving the difficulties have been hinted at in [Pol13], where linear solves with comparably high relative residuals could achieve reasonably low eigenpair residuals, see also Section 4.3.1.2. In [GMP18], a generalization of residual inverse iteration [Neu85] is used as a nonlinear eigenvalue solver based on contour integration. This technique allows low accuracy solves of the emerging linear systems [Pol20].

4.3 Achievable residual

In [Krä+13] and [Krä14], limits for the residual that can be expected from a subspace iteration algorithm under certain conditions have been proposed, which act as somewhat pessimistic upper bound, aiming to specify a residual that can be reached under any circumstances. The proposed limit has the form

$$\boxed{1} \qquad \varepsilon n \left\| B^{-1} A \right\|_2 .$$

A less pessimistic limit in case the interval boundaries are not too small is given as

$$\boxed{2} \qquad \varepsilon n \min\{|\lambda_{\min}|, |\lambda^{\max}|\}.$$

In both cases, ε should not be chosen smaller than machine precision (see below).

4.3.1 Absolute rock bottom

From the general theory for the rate of convergence discussed in Section 1.6.1 and in the first part of this chapter, we can infer that merely disturbing the filtering function that serves as a basis for a projection algorithm is not enough to impose a hard limit on the achievable accuracy of a spectral projection eigensolver. The result is nothing more than a filtering function with diminished dampening properties in regions where the disturbance is large compared to the value of the filtering function. Due to this, convergence is slowed down, but not stopped. We do, however, have to expect to ultimately encounter a limit based on the numerical accuracy the computations are executed with. Due to the design of floating-point representations, relative accuracy is retained if the operands do not differ severely in magnitude, meaning that computations dealing exclusively with small numbers exhibit higher accuracy on an absolute scale. This is related to a term often referred to as *machine precision* $\boldsymbol{\varepsilon}(1)$, the distance between unity and the next representable floating-point number of the same precision, or, more intuitively, the accuracy which can be expected when performing computations involving numbers roughly of unit magnitude.[9] For computations in different magnitudes, $\varepsilon(m)$ is defined as the difference between m and the next representable floating-point number (twice the maximum absolute error). If less information is used for encoding numbers, $\varepsilon(m)$ will accordingly be larger, e.g., considering IEEE 754 double precision vs. single precision. During complex computations, these round-off errors can accumulate and diminish accuracy further. Factors that add to the overall error are thus, among others, matrix sparsity or searchspace size. For the sake of simplicity, we henceforth let $\varepsilon = \varepsilon(1)$. A comprehensive coverage of floating-point arithmetic, rounding errors, and accuracy of numerical computations can be found, e.g., in [Hig02] or [Gol91].

The limited precision of floating-point arithmetic has to be interpreted as systematic error during filter application (among other aspects of the algorithm) and consequently as disturbance of the projected searchspace set. It is the cause for the inevitable stagnation of the iterative process. The available precision, however, is not the only cause for a systematic error that can be introduced into the searchspace set. In [Krä+13] and [Krä14], it has been pointed out that the accuracy of the linear solves involved in the application of rational filters imposes a limit on the achievable residual of spectral projection eigensolvers (see also [TP] where this is derived more formally). This in particular relates to iterative linear solvers with prescribed residual limit. The error introduced during the linear system solves is in type identical to the error introduced by limited precision, but potentially larger in magnitude.

In order to identify the residual that at least is achievable by an iterative eigensolver, we shall assume that the projector can be applied without introducing any unnecessary additional error—beyond the inevitable numerical errors caused by the basic algorithm and the application of the approximate projector as single matrix multiplication—into the searchspace set. In order to achieve this, we reduce the computational effort to a single sparse matrix multiplication by explicitly constructing the filtering operator matrix. This also allows us to use theoretically ideal filtering functions to rule out the appearance of Ritz phantoms and not having to distinguish between eigenpairs inside or outside the target interval. In fact, an ideal filter with dampening > 0 can be understood as disturbed filter itself. As such, the filter has no influence on the achievable residual beyond the emergence of Ritz phantoms.

Experiment 4.13 — Projector disturbances

We generate a definite pair of size 1000 *with evenly distributed random spectrum in* $[-1, 1]$ *and an ideal filter with dampening* 10^{-2} *to a searchspace of size* $m_0 = m$. *We compute the* $m = 20$ *innermost eigenpairs over the course of* 20 *iterations to assure that all pairs have converged to the best possible residual. The constructed projection operator* P *is disturbed as*

$$P + eE$$

[9] Often, the term *machine precision* describes the maximum relative error, or unit roundoff, and thus half this distance.

where E is a matrix with evenly distributed random entries in $[-1,1]$ and $e = 10^{-7}$, 10^{-9}, 10^{-11}, 10^{-13}, and 10^{-15} is a scalar, such that eE is a matrix with random entries of at most magnitude e. For comparison, the experiment is repeated completely undisturbed.

The left plot of Figure 4.22 shows the progression of residuals as an area representing the range between minimum and maximum residual for each of the configurations mentioned above. Each line indicates the norm of the error matrix applied to the respective projection operator; the lowest overall residual range belongs to the undisturbed projector. The right plot compares the residuals of a filter with dampening 10^{-5} where the projector has been randomly disturbed with magnitude 10^{-2} (blue) and an undisturbed projector of power 10^{-2} (red). The results are basically identical; we can expect slightly faster convergence from the disturbed projector since its dampening might be below 10^{-2} for some directions. Repeating the experiment with different filters or even with an ideal filter of infinite dampening gives identical results.

Figure 4.23 gives a visual representation of the assumed filtering effect after one (left) and two (right) applications of the projection operator, where the accumulative filtering effect is interpreted as powers of the filtering function, as would be the case with conventional power iteration using the projection operator. A disturbance of the filtering function itself does not inhibit convergence as its powers also affect the accumulative (disturbed) dampening (blue) in the same manner. Instead, we have to understand the error as being reapplied with full strength in every step (purple).

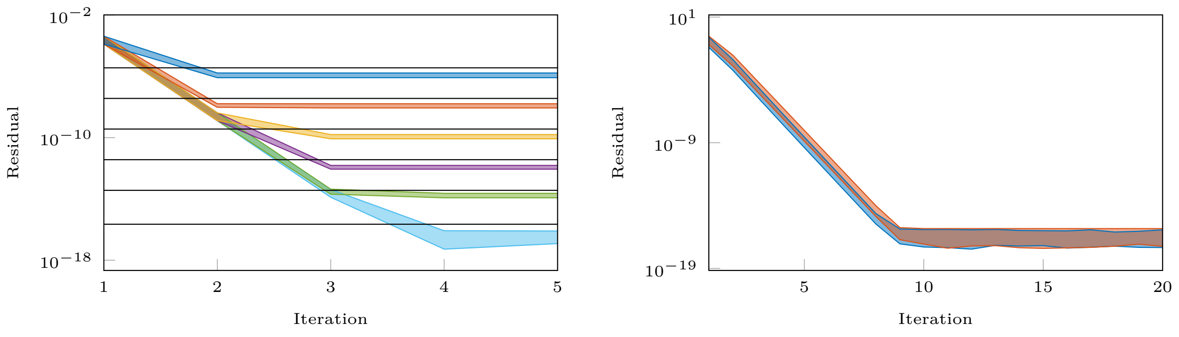


Figure 4.22: *Left: residual barrier with disturbed projection operators with error norms $3.6\cdot10^{-6}$, $3.6\cdot10^{-8}$, $3.6\cdot10^{-10}$, $3.6\cdot10^{-12}$, and $3.6\cdot10^{-14}$ (top-down). Right: residual barrier for an ideal filter with dampening factor 10^{-2} (blue) and the corresponding barrier for an ideal filter with dampening factor 10^{-5} which has been disturbed with magnitude 10^{-2} (red).*

In the absence of Ritz phantoms, the norm of an error matrix E applied to the projection operator P is a reasonable upper bound for the minimum residual that can be achieved under these conditions. This norm is, of course, typically not known. For errors imposed by the finite precision of the computations, we expect a relation to $\varepsilon(\|B^{-1}A\|)$, where the matrix norm, for definite pairs the largest magnitude eigenvalue, describes the general magnitude of the computations. In [Krä+13; Krä14]

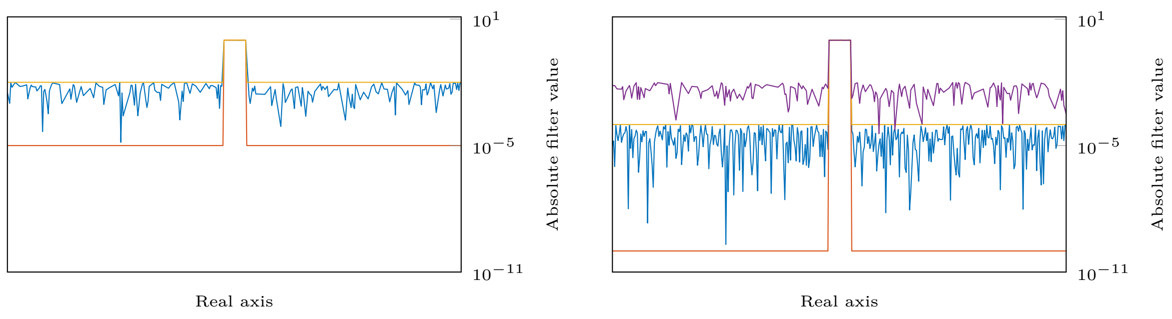


Figure 4.23: *Left: ideal filter with dampening* 10^{-2} *(yellow) and* 10^{-5} *(red), as well as an ideal filter with dampening* 10^{-5} *disturbed with magnitude* 10^{-2}*.*
Right: assumed effective dampening of two consecutive projector applications; in addition to the left plot, the filtering effect of a disturbed projection operator (purple) is shown.

it has been pointed out that the application of an iterative linear solver to the linear systems arising from the Cauchy filter formulation mimics this behavior.

From the discussion of the effect of spectral filtering, it is clear that the best residual that can be achieved does not depend on the choice of filter. There is however one inhibiting factor that can be understood as a consequence of the combination of chosen filter and spectral distribution of the matrix pencil in question: the influence of possible Ritz phantoms on the residual of Ritz pairs, even if those have reached the smallest possible residual already. The effect described here has in principle been described in Experiment 4.9 before, but the following experiment will show that the presence of Ritz phantoms does not only influence convergence, but also the overall achievable residual, even if the smallest possible residual was reached many iterations ago.

Experiment 4.14 — Influence of Ritz phantoms

We generate a definite pair of size 1000 *with evenly distributed random spectrum in* $[-1, 1]$ *and apply a Zolotarev filter of degree* 4 *with* $\kappa = 0.01$ *to compute the innermost* 20 *eigenpairs with a searchspace size of* 26 *over* 500 *iterations. This setup ensures the emergence of Ritz phantoms.*

In a second instance, we place the spectrum in $[-5 \cdot 10^{-4}, 5 \cdot 10^{-4}]$ *and compute the innermost* 20 *eigenpairs with a searchspace size of* 40 *over* 1000 *iterations using a Zolotarev filter of degree* 12*. This particular setup creates a Ritz phantom that crosses the target interval after many iterations. We have seen a very similar setup before in Experiment* 4.8*.*

The left plot of Figure 4.24 shows the first instance, the right plot shows iterations 300–800 *of the second instance.*

The significant aspect of Figure 4.24 is the presence of Ritz phantoms inside the target interval coinciding with significant disturbances in the residual. This even is the case if the smallest attainable residual has been reached by all directions,

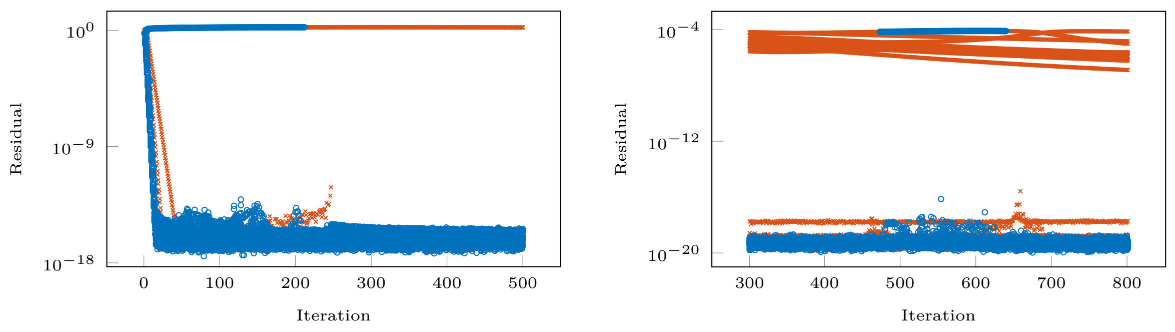


Figure 4.24: *Left: Ritz phantoms disturbing the terminal residuals. Right: Ritz phantoms disturbing the terminal residuals in a very late iteration.*

indicating that a stationary Ritz phantom would impair the best possible residual negatively. With this in mind, we may consider an ideal filter with constant dampening to perform the worst among all the filters introduced here with respect to the smallest achievable residual, since it produces Ritz phantoms independently of the matrix spectrum as soon as the searchspace is oversized. It also further justifies considering the explicit removal of such directions, as has been described before, to improve overall results. In case large numbers of Ritz phantoms populate the searchspace, either provoked by choosing oversized searchspaces or a non-decaying filter or both, the achievable residual can become arbitrarily large. Under practical conditions, this can be prevented by detecting and removing those directions, see Section 4.2.2. For a decaying filter, the appearance of Ritz phantoms is less likely to happen, unless enforced.

We can deduce from the above that a systematic error of the described nature does not inhibit convergence until the respective residual limit is reached. Two important consequences arise that bear potential for computational optimization.

4.3.1.1 Mixed precision

Seeing that nothing can be gained from performing computations in double precision as long as the single precision residual barrier has not yet been broken, early iterations may be carried out in pure single precision without any penalty. For optimal results, the first iteration to be performed in double precision should be the one to surpass the single precision barrier. This may be achieved by estimating the residual drop rates from earlier iterations (remembering that the rates are constant in principle for a given filter), or by inspection of the estimated filter values from eigenvalues of B_U. Nonetheless, a late change of precision does not necessarily impair overall convergence, as the following experiment will demonstrate.

Experiment 4.15 — Mixed precision

With a size 1 048 576 *topological insulator matrix similar to the* `topi-1M` *test matrix from Section 4.1.2, multiple runs are performed with identical setup, differing only in the precision of computations (single or double) or the iteration number where* *a switch from single to double precision occurs. The* 65 *innermost eigenpairs are computed using a polynomial filter of degree* 135 *and a generous searchspace size of* 256*. The larger searchspace size moves the general performance emphasis from the projector application to the overhead produced due to the reduced eigenproblem, inner products, and other factors. To establish baseline results, computations are performed in pure double precision and pure single precision. In additional runs, precision then is switched from single to double in iterations* 7*,* 9*, or* 11*. We monitor the minimum residual of the Ritz vectors as well as the geometric mean residual of all directions inside the target interval. The results are presented in Figure 4.25. The data for this experiment has also been shown in [Wel+20].*

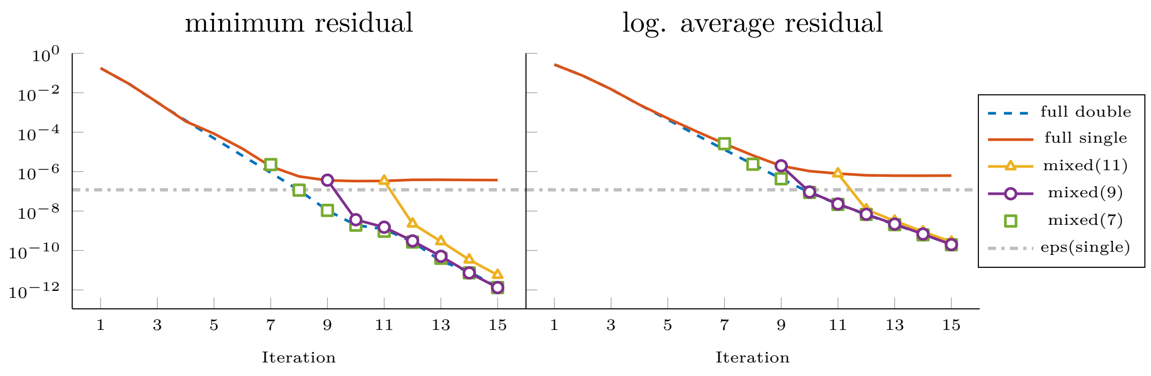


Figure 4.25: *Mixed precision results over 15 iterations. Left: minimum residual; right: geometric mean residual.*

The experiments shows that a slightly late switch has practically no impact on the overall progress since the change in precision is accompanied by a jump in residual, catching up to the full double precision run. For the average residual, even a very late switch is not very detrimental. This allows to detect the deceleration of convergence as the single precision limit is approached in order to trigger the change in precision.

Consequently, nothing can be gained by performing computations in higher precision, e.g., quadruple precision, until we aim to exceed the respective residual limitation imposed by the double precision floating-point system (see also [GHL18d; Alv+19]).

4.3.1.2 Linear solving effort

The linear systems arising in the application of rational filters are among the most difficult problems for iterative linear solvers. The results from above imply that linear systems in early iterations do not have to be solved to high accuracy. The requirements of accuracy posed to the linear solver grow as the eigensolver iterations proceed. As a consequence, the linear systems have to be solved to lower and lower absolute residual in each new eigensolver iteration. Indeed, we can assume that, if the results of the iterative linear solver from an earlier iteration can serve as suitable initial guess for solves in subsequent iterations, an iterative linear solver will only have to reduce the relative residual by a fixed order of magnitude in each iteration for convergence to proceed uninhibited. This, at least, requires the solution vectors to be assignable over iterations, e.g., when the ordering of directions changes due to the proximity of a Ritz phantom or in early iterations, or when the searchspace size is reduced or extended.

Progress in this regard has been hinted on in [Pol13], where results could be achieved with moderate relative residual. Not much information is given, though.

4.3.2 Kick-off residual

An a priori estimation of the residual that can be expected after the first iteration of the subspace iteration, which we will name *kick-off residual*, is not of particular importance, since the residuals are computed at the end of the iteration anyway. Its knowledge does serve a purpose, though, if we are to judge whether a problem is solvable at all under the assumption of a certain bound for the best possible residual.

Judging from Theorem 1.2, we can expect the kick-off residuals to be related to the residual drop rates associated with the respective eigenvalues as dictated by the filter and the representation of the eigendirections in the initial guess vectors. The latter part has been confirmed in Experiment 4.3 and is in particular true as every iteration may be interpreted as an initial iteration with increasingly well represented eigendirections in its initial guess vectors, which are nothing but the approximate eigenvectors themselves. This representation appears as a constant multiplier in Theorem 1.2, describing the difference between an eigenvector x_i and the direction contained in the initial guess vectors that, assuming precise projection, would be projected onto the eigenvector, $Ps_i = x_i$.

With higher eigenvalue densities, Theorem 1.4 allows the residual of a more or less randomly placed Ritz value to be relatively small since an eigenvalue can be found in its vicinity. The following experiment gives an example of this effect.

Experiment 4.16 — Effect of spectral density on kick-off residual

We generate a matrix pencil of size 1000 *with evenly spaced spectrum in an interval of varying size,*

$$100, 10, 1, 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}, \text{ and } 10^{-11},$$

around a central point $c = 200$. *This ensures that the matrix norm and the magnitude of computations have no significant effect on the results. We aim to compute the innermost* 20 *eigenpairs using an ideal filter with a fixed dampening of strength* 10^{-4} *applied to a searchspace of exact size* 20 *to eliminate differences of convergence speed, potentially caused by variations of the filter.*[10] *The eigenvectors used to generate*

[10] This is not necessarily needed for all filter types. Many filters scale with their target interval and will not disturb convergence speeds if the relative distribution of eigenvalues is constant, the only exception being the Chebyshev polynomial filter.

the matrix pencil are generated only once and are therefore identical for all runs of the eigensolver. Vectors for the initial guess are chosen at random, but identical for all runs of the eigensolver to fix the representation of eigendirections in the basis. The best and worst residual inside the target interval are recorded after the first iteration. Figure 4.26 shows the corresponding drop of the kick-off residual for increasing spectral density (and decreasing spectral range).

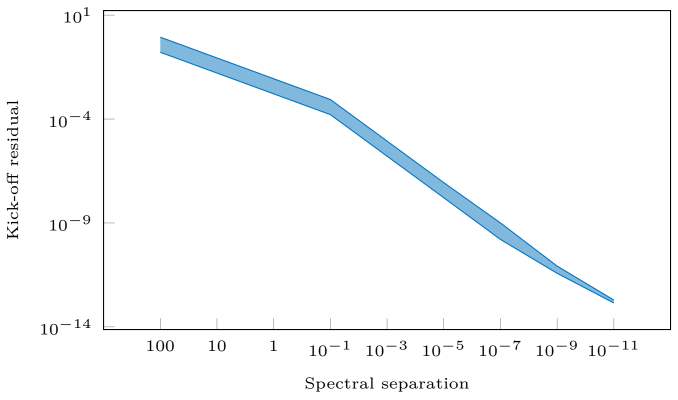


Figure 4.26: *Kick-off residual ranges for different spectral ranges and densities.*

The effect can only be observed if the spectral distribution is dense in general. For small matrices this translates to generally small spectral ranges. Additional experiments show that clustering the eigenvalues inside or around the target interval alone is not sufficient.

4.3.3 Saturation residual

We have seen above that, under optimal conditions, the achievable residual is at least related to $\varepsilon(\|B^{-1}A\|)$ (which will be referred to as $\boxed{3}$). This does not account for the possible influence of the spectral composition, presence of Ritz phantoms, or other inhibiting factors. In order to produce reasonable results for the achievable *worst-case residual*, i.e., the largest residual among Ritz values that have reached saturation, henceforth *saturation residual*, the emergence and movement of Ritz phantoms inside the target interval in later iterations should be prevented, as the amplitude of the disturbances seems unpredictable. In order to achieve undisturbed convergence, we may employ ideal filters with strictly sized searchspaces or decaying filters with more generously sized searchspaces. The occurrence of slowly converging directions is of no concern, if the associated Ritz values appear outside the target interval (with some distance to the interval boundaries) or have reached saturation in the last few iterations. While the first condition is unpredictable, the second one may require iteration ad infinitum. However, the spectrum might be prepared to ensure larger distances to λ_{m_0+1}. In the following, experiments will be conducted that vary parameters related to the quantity that imposes a limit on the saturation residual: the magnitude of floating-point computations in terms of the positions of iterated directions and the norm of the matrix pencil. It is roughly $\boldsymbol{\varepsilon}(m) \approx m\boldsymbol{\varepsilon}(1)$. The limit $\boxed{1}$ mentioned in Section 4.3 therefore reflects machine precision in terms of the largest magnitude eigenvalue of the matrix pencil with an additional factor of n, the size of the matrix pencil, to account for error accumulation. If the position of the spectrum is moved along the real line, the norm of the matrix pencil changes accordingly. The change in the magnitude of computed eigenvalues, however, may differ if the spectral range is large compared to the magnitude of its center.

Experiment 4.17 — Variation in spectral position

We generate a matrix pencil of size 1000 *with evenly distributed random spectrum in an interval of size one around a central point* c*. This central point is chosen from the sequence*

$$c = 10^4, 10^3, 10^2, 10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 0$$

to produce a progression of eigenvalue problems with differing relations between norm and magnitude of the computed eigenvalues. An ideal filter with explicit dampening of 10^{-5} *is applied and the searchspace size is chosen exactly to reliably prevent Ritz phantoms that otherwise may obstruct convergence. Fixing the spectral range, i.e., the extent of the spectrum, the variation of the center of the spectrum influences the norm of the matrix pencil in a different way than it influences the magnitude of the target interval at the center of the spectrum. Shifting the spectrum towards zero, the norm of the matrix pencil becomes dominant as the magnitude of the target interval's boundaries becomes small compared to the magnitude of the total spectral range. The searchspace is iterated* 15 *times to assure that saturation is reached for all Ritz values. The final maximum saturation residual is chosen as the maximum residual among the last* 10 *iterations.*

Figure 4.27 shows four instances of this experiment which differ in matrix sparsity, s*, and the number of eigenpairs computed,* m*.*

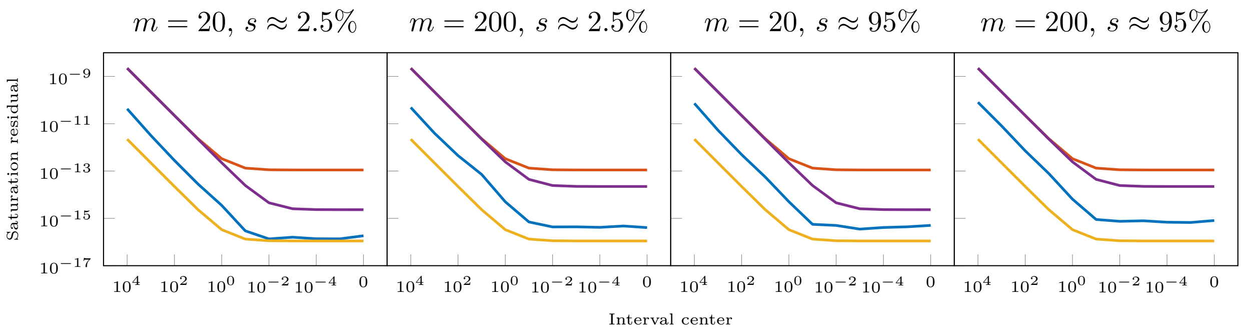


Figure 4.27: *Saturation residual (blue) for variations of the spectral center. Also shown are the limits* $\boxed{1}$ *(red),* $\boxed{2}$ *(purple), and* $\boxed{3}$ *(yellow) with* $\varepsilon = \boldsymbol{\varepsilon}$.

The plots show how both factors influence the saturation residual. The effect is more pronounced if the eigenvalues are small compared to the norm of the matrix pencil. As the magnitude of the target interval continues to decrease, the saturation residuals stagnate as soon as the norm of the matrix pencil ceases to change in magnitude. The saturation residual then approaches the unscaled machine precision (yellow) more closely.

For comparison, the following experiment shows that the variation of the extent of the spectrum changes the norm of the matrix pencil and the computed eigenvalues in relation to the magnitude of the spectral center.

Experiment 4.18 — Variation in spectral range

With a fixed spectral position, varying the spectral range influences the norm and target interval of a matrix pencil in a similar way. We repeat Experiment 4.17 with fixed spectral centers and vary the spectral interval's size,

$$l = 10^2, 10^1, 10^0, 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}.$$

Figure 4.28 shows the results for $c = 0$ and $c = 100$. In the first case, 200 eigenpairs were computed to separate the saturation residual (blue) from the unscaled machine precision (yellow). Since the norm of the matrix pencil is dominant, the saturation residual approaches the unscaled machine precision too closely otherwise. In the second case, the magnitudes of the norm and target interval are identical.

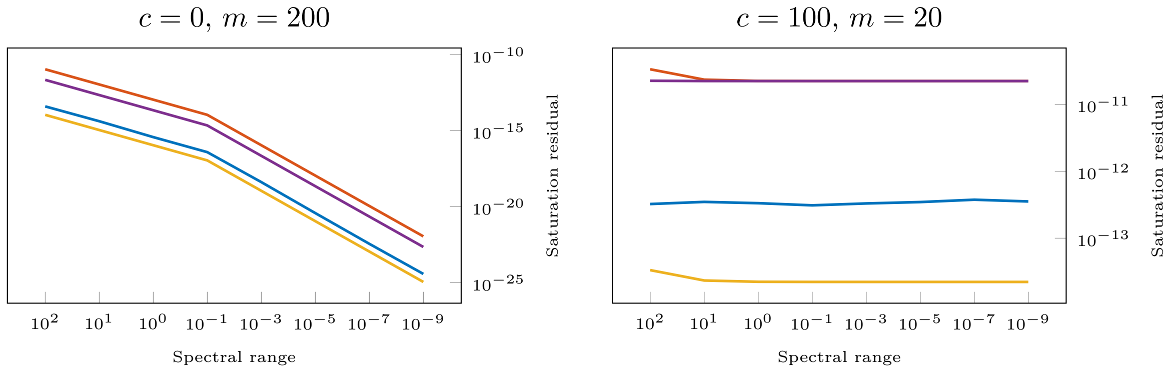


Figure 4.28: *Saturation residual (blue) for variations of the spectral range. Also shown are the limits $\boxed{1}$ (red), $\boxed{2}$ (purple), and $\boxed{3}$ (yellow) with $\varepsilon = \boldsymbol{\varepsilon}$.*

Decreasing the sparsity of the matrix pencil or increasing the number of targeted eigenvalues does indeed raise the saturation residual. However, the factor n is overly pessimistic in most situations, in particular if the iterated eigenpairs are small compared to the norm of the matrix pencil. However, it may allow few Ritz phantoms to be present in the target interval without breaking the predicted bound.

4.4 Termination criteria

Closely linked to the residual that may be achieved by an eigensolver is the criterion for its termination. An algorithm for computing eigenpairs inside a target interval should terminate only if all eigenpairs inside the target interval are found or if the prescribed residual accuracy cannot be reached for all eigenpairs in the target interval and the method stagnates. In the latter case, whether to include the not fully converged eigenpairs in the result has to be decided. Additionally, non-algorithmic criteria may exist, such as a limited number of total iterations or certain error conditions, which shall not be discussed here.

Previous discussions involving suitable stopping criteria can be found in [Krä+13; GKL12; Krä14], which typically are residual-based and combined with a prediction of the eigencount of the target interval. Generally, while a major indicator of a Ritz phantom is a much larger residual, judging where to draw the line that reliably separates Ritz phantoms is difficult if not impossible, in particular under conditions which we have seen by now and where Ritz phantoms may converge rather far due to cluster super-convergence or the achievable residual not being very low to begin with. The usage of an eigencount estimation alleviates this problem in part, as it defines the number of Ritz values that should converge and, with this, whether iteration still has to be continued. While the analysis of Ritz values—be it by counting values inside the target interval or computing filter values based on their position—is unreliable due to possible Ritz phantoms located inside the target interval, the estimated filter values obtained from the second Ralyeigh quotient B_U (see Section 4.2.2) can expose these values which are the least dominant in the searchspace and thus have small associated filter values. The result is a rather reliable count for the number of eigenpairs in the target interval.

Still, a mismatch between the number of converged eigenpairs and the number derived from the filter values by even just one eigenvalue can cause problems. In case of an overestimation, the algorithm expects a Ritz value to converge that may (virtually) never arrive at the expected residual. Something similar happens when the target residual cannot be reached due to the limiting factors outlined before. In this case, the method stagnates, but the estimated count will never be reached. On the other hand, too small an estimation may lead to premature termination.

The quality of the estimated filter values or the estimated count of eigenpairs derived from it can be inaccurate for mainly two reasons, besides the general formal condition of having completed at least one iteration to fulfill the criteria from Section 4.2.2. Either the filter values are not precise enough to properly count eigenpairs very close to the interval boundaries or the filter may be asymmetrical, which may be the case for filters based on polynomial approximation. The latter can easily be corrected by computing a spectral transformation that not only maps the spectrum into the unit interval, but also centers the target interval around zero (see Section 2.1.5).

Let the spectrum be contained in $[\lambda^-, \lambda^+]$ and let $[\lambda_{\min}, \lambda^{\max}]$ denote the target interval. The spectral transformation for symmetrical target intervals in $[-1, 1]$ can then be written as

$$\frac{2\lambda - \lambda^{\max} - \lambda_{\min}}{\lambda^+ - \lambda^- + |\lambda^{\max} + \lambda_{\min} - \lambda^+ - \lambda^-|}$$

for any λ.

The first point made above, however, can only be mitigated by extending the definition of *inside the target interval* to *possibly inside the target interval*, an approach that is based on Theorem 1.4 and has been used, e.g., in [Pie+16]. As such, at least one eigenvalue λ is located inside a disk

$$\left[\boldsymbol{\lambda} - \left\|B^{-\frac{1}{2}}r\right\|, \boldsymbol{\lambda} + \left\|B^{-\frac{1}{2}}r\right\|\right]$$

around a computed Ritz value $\boldsymbol{\lambda}$. If a Ritz value has a distance not farther than the radius of this disk to the interval boundaries, it has to be considered part of the target interval in lieu of a more precise classification. In theory, the same holds for the estimated filter values, but their residual is unknown and not easily computed, as it requires the application of the projector.

While the inclusion of possibly contained Ritz values mitigates the risk of missing eigenpairs by terminating too early, it cannot account for Ritz phantoms, bringing us back to the accuracy of the eigencount. In [GKL18], methods to judge the validity of the predicted eigencount are added to enhance the termination logic. These include, in addition to a requirement for the searchspace and, therefore, for the estimated filter values to be reasonably far converged, comparisons of consecutive eigencount estimations as well as monitoring the searchspace population and their residual. To this end, the *smallest not converged residual* (SNC) inside the target interval is used as an indicator *(B. Lang, personal communication)*. The SNC is used to monitor the speed at which the searchspace is still converging. Of course, the SNC does not always refer to the same Ritz value. If

- a Ritz value inside the target interval converges,
- a Ritz value leaves the target interval,
- a Ritz value enters the target interval, or
- the current SNC reaches saturation before reaching the target residual,

the SNC may switch the Ritz value it refers to. The criterion to abort iteration is then a jump to a considerably larger residual, which would allegedly identify the SNC as a Ritz phantom. Choosing the size of this jump poses the same problem as choosing a hard residual barrier for identifying Ritz phantoms: any fixed choice will likely fail sooner or later. Assuming Ritz phantoms never leave the initially recorded residual range of the searchspace also is not quite true, such that a jump of the SNC into that range cannot be used as an indicator either.

We have, however, described before the fundamental condition that produces a Ritz phantom. It is a Ritz value that converges very slowly. If we define a minimum speed at which we want to see Ritz values in the target interval to converge and the SNC being the fastest among those, any form of stagnation could be detected. This happens under the assumption that none of the Ritz values that have to be computed converges at such a low rate, which is undesirable to begin with. This is very similar to the adaptive approach described in [GKL18], where the residual drop rate computed from the SNC is used to find a close to optimal degree (or number of integration nodes) for the method to deliver the result with the least effort. As such, a stagnation-based termination criterion ties in well with this adaptivity, but has to be disabled as long as the convergence rate is still low.

In order to reliably compute the drop rate from the SNC, we have to exclude cases, where the SNC changes assignment to prevent false positives. The conditions are:

- At least one iteration must have passed to obtain drop rates to begin with.
- There should be converged Ritz values (otherwise the target residual may not be reachable, see below).
- No new values have converged, i.e., the SNC did not change its assignee.
- The number of not converged Ritz values in the interval must not have changed to rule out Ritz values entering or leaving the interval. On the off chance of one Ritz value entering and one Ritz value leaving simultaneously, the criterion may trigger too early.
- The quotient of the old and new SNC residual should be smaller than some ρ close to unity, e.g., $\rho = 1.2$. If ρ is chosen too close to 1, Ritz phantoms may not be detected; if ρ is chosen too far from 1, iteration may be terminated too early.

In almost all instances, a stagnation-based termination criterion is robust in theory, even without the support of an eigencount estimation. There are corner cases, however, such that the eigencount should be used in conjunction.

The SNC might also slip "outside" the target interval (analogous to "inside" from above), which means that all Ritz values inside the target interval have converged, thereby indicating completeness under the assumption of the extended understanding of "inside" the target interval from above.

If the residual reaches saturation before reaching the prescribed limit, the termination criterion will trigger, typically as soon as the first Ritz value stagnates. Deciding which Ritz pairs should be returned as result in this case is difficult as, again, the problem of differentiating between Ritz phantoms and potentially usable values arises. Since the required residual could not be reached, it may be reasonable to declare the run as failed altogether, but since no better residual can be reached, it might make sense to continue iteration until all values have reached saturation.

Two methods to modify the termination conditions come to mind. First, the SNC can be locked on stagnation, causing iteration to continue until all values inside the target interval have stagnated. This method should be used if stagnation sets in without Ritz values having converged before. The estimated eigencount has to be consulted to make sure the interval is not just empty. A second method is the dynamic adjustment of the target residual. Once saturation sets in and the interval is not considered empty, the reachable residual is known, allowing to specify a new limit.

Another case where the residual grows, thereby signaling stagnation, is cluster super-convergence. Since the rebound residual may be larger than the target residual, termination would trigger if the SNC refers to one of the faster converging clustered values. This is, however, unlikely in practice. Eigenvalues inside the interval may converge faster, also accelerating the rebound phases, such that these obstructions of convergence do not cause outside eigenvalues to catch up. It still might be reasonable to not only monitor decrease in residual for stagnation detection, but also to monitor increase in residual, such that fast increasing residuals do not trigger the stagnation condition. We also might require stagnation to occur for at least two consecutive iterations in order to reliably detect real stagnation.

Finally, a situation is imaginable, where an outside eigenvalue, close to the target interval such that it may be interpreted as inside, may converge quicker than eigenvalues actually inside the interval. Normally, the form of the filter does not allow this, but we assume it may happen nonetheless. If this value is mistaken as in-interval target eigenvalue, believing the eigencount estimation, a real in-interval eigenvalue may be missed because of this. In this case, however, the stagnation detector would not trigger, indicating that another Ritz value is still converging.

4.5 Achievable orthogonality

The ability of spectral projection eigensolvers to independently compute portions of the spectrum of a matrix pencil is one of its great strengths. Technically, the targeted spectral regions need not be adjacent or, given a filter function with a more exotic form, contiguous intervals. Here, however, we will assume the target portion of the spectrum to be a contiguous interval, such that it can be subdivided into adjacent subintervals $\mathcal{I}_1, \dots, \mathcal{I}_k$. Each interval contains eigenpairs (X_i, Λ_i), $i = 1, \dots, k$, whose approximations are computed as the Ritz pairs $(\boldsymbol{X}_i, \boldsymbol{\Lambda}_i)$. Independently computed eigenvectors are known to suffer from poor orthogonality, influenced by the distance between the associated eigenvalues, commonly referred to as *gap* (see, e.g., [Par80; Krä14; Krä+13]). This of course is only a problem if the eigenvectors are supposed to be (B-)orthogonal, i.e., for definite pairs.

Cases in which separation into subintervals is impossible are given for multiple eigenvalues; the value of the filter function at that position is equal for all associated directions. A subspace iteration algorithm inherently computes a basis containing all eigendirections with identical eigenvalues and the searchspace size has to be chosen accordingly. This was explored in more detail in the first sections of this chapter.

The term *orthogonality* so far has mostly been used as a binary property, but possible deviations from orthogonality have been mentioned. We now formally define the orthogonality of two vectors z and y as the real number

$$\operatorname{orth}(z, y) = \frac{|\langle z, y \rangle_B|}{\|z\|_B \|y\|_B},$$

fulfilling $\operatorname{orth}(z, y) = 0$ in case z and y are B-orthogonal. We may also explicitly refer to this property as *B-orthogonality*, in case differentiation is necessary. In case $B = I$, the orthogonality of z and y also satisfies

$$\operatorname{orth}(z, y) = \cos(z, y)$$

that is, their *I-orthogonality* is the cosine of the angle between z and y for angles $\leq \pi/2$. In case $B \neq I$, a direct relation to the angle between z and y does not exist.

Analogously, we may define the orthogonality of a block of vectors $Z = [z_1, \dots, z_k]$ as the worst-case orthogonality between any two vectors with different index,

$$\operatorname{orth}(Z) = \max\{\operatorname{orth}(z_i, z_j) \mid i, j = 1, \dots, k \wedge i \neq j\},$$

and the orthogonality between two blocks of vectors Z and $Y = [y_1, \dots, y_\ell]$ as the worst-case orthogonality between any two vectors from different blocks,

$$\operatorname{orth}(Z, Y) = \max\{\operatorname{orth}(z_i, y_j) \mid i = 1, \dots, k\,; j = 1, \dots, \ell\}.$$

To differentiate both, we will refer to the orthogonality of a single vector block as *intra*-orthogonality and to the orthogonality between two vector blocks as *inter*-orthogonality. If the orthogonality of two blocks, e.g., computed eigenvectors from two separate intervals, is analyzed, whether one is orthogonalized against the other or not, we will talk about the results as *interaction* between these two blocks or intervals. If the computed orthogonality, also between single vectors is large, we will say that these blocks or vectors interact strongly; otherwise they interact weakly.

4.5.1 Theoretical limit

To accommodate for orthogonality with respect to B, we may extend the definition of the conventional cosine to

$$\cos_B(Z, Y) \coloneqq \cos \angle_B(Z, Y) \coloneqq \operatorname{orth}(Z, Y)$$

for all B Hermitian and positive definite, and blocks of vectors Z and Y (see, e.g., [Krä14]). Then the B-angle between Z and Y is

$$\angle_B(Z, Y) = \cos^{-1} \operatorname{orth}(Z, Y)$$

and allows for an analogous definition of

$$\sin_B(Z, Y) \coloneqq \sin \angle_B(Z, Y).$$

A theoretical upper bound for the worst-case orthogonality of two blocks of Ritz vectors can be derived from the bound of the angle to their respective eigenvectors Z, the $\sin\theta$ theorem [Krä14; DK70; Nak12]. Formulated for a block of computed Ritz pairs $(\boldsymbol{X}, \boldsymbol{\Lambda})$ with residual $R(\boldsymbol{X}, \boldsymbol{\Lambda})$, it can be written as

$$\sin_B(X, \boldsymbol{X}) \leq \frac{1}{g}\left\|B^{-\frac{1}{2}} R(\boldsymbol{X}, \boldsymbol{\Lambda})\right\| \leq \frac{1}{g}\left\|B^{-\frac{1}{2}}\right\| \|R(\boldsymbol{X}, \boldsymbol{\Lambda})\|, \tag{4.4}$$

where g is the gap between any Ritz value of $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda}_i \mid i \in \Xi)$, $\Xi \subset \{1, \dots, n\}$, and any other eigenvalue λ_j, $j \notin \Xi$, of the matrix pencil that is not associated with a Ritz value of $\boldsymbol{\Lambda}$,

$$g = \min\{|\boldsymbol{\lambda}_i - \lambda_j| : i \in \Xi,\ j \notin \Xi\}.$$

Assuming this worst-case deviation for two separately computed blocks of Ritz pairs $(\boldsymbol{X}_1, \boldsymbol{\Lambda}_1)$ and $(\boldsymbol{X}_2, \boldsymbol{\Lambda}_2)$ and the associated eigenvectors X_1 and X_2 being orthogonal (with respect to B), $\angle_B(X_1, X_2) = \pi/2$, a corresponding worst-case B-angle, or its associated orthogonality, between $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ can be computed as

$$\operatorname{orth}(\boldsymbol{X}_1, \boldsymbol{X}_2) \leq \cos\left(\frac{\pi}{2} - \sin^{-1}\left(\left\|B^{-\frac{1}{2}}\right\|\frac{\|R_1\|}{g_1}\right) - \sin^{-1}\left(\left\|B^{-\frac{1}{2}}\right\|\frac{\|R_2\|}{g_2}\right)\right), \qquad (4.5)$$

where g_1 and g_2 are the respective gaps as defined above, $R_1 = R(\boldsymbol{X}_1, \boldsymbol{\Lambda}_1)$, and $R_2 = R(\boldsymbol{X}_2, \boldsymbol{\Lambda}_2)$. Given the monotonicity of $\sin^{-1}\theta$ inside $[0, 1]$ and the monotonicity of $\cos\theta$ inside $[0, \pi/2]$, the derived bound is sensible.

This interpretation of Equation (4.4) is already somewhat flawed as it is formulated in terms of *canonical angles* between subspaces [Krä14]. Without delving too deep into the definition and theory, the notion conveyed is the maximum angle between any two vectors from the different subspaces, such that this angle becomes zero only if one subspace is completely embedded in the other. Our understanding of the orthogonality between two blocks of vectors, however, is that the angle between these two blocks is the smallest angle between any two vectors from the associated subspaces, such that this angle becomes zero as soon as one direction is shared among the two subspaces. Beyond this, further modifications are necessary in order to replace unknown quantities with known ones.

While the exact values for g_1 and g_2 are unknown, a reasonable estimation is the minimum distance γ of Ritz values from one interval to Ritz values of adjacent intervals, assuming the Ritz values are reasonable approximations of the eigenvalues. If, for an interval, not both neighbors are available, the distance to only one of them will have to suffice. If we intend to estimate the orthogonality between a certain pair of such blocks, taking into account their distance such that we expect better orthogonality between intervals of larger distance, the definition of g from above seems unfit. For an evenly distributed spectrum, e.g., the value of g would be identical for every interval, assuming that all intervals were solved to equal residuals, resulting in equal orthogonality estimations for any pair of intervals.

We therefore replace the numbers g_1 and g_2 by the mutual distance γ of any two intervals whose inter-orthogonality we want to estimate. This is a modification that is not backed up by the $\sin\theta$ theorem, though. First of all, this modification does not take into account the distance to the other neighboring interval or, in case of an interval at the boundary of the covered spectral region, the rest of the spectrum. Assuming that the Ritz values are correct representations of the eigenvalue positions, a potentially smaller gap to the unknown parts of the spectrum may result in an orthogonality that is worse than predicted and thus break the bound. A larger gap, on the other hand, has no effect on the (assumed) validity.

The modification also ignores inaccuracies of the Ritz values. The expected effect is smaller, though; it only plays a role if the distance between the two sets of Ritz values is very small and, in this case, the variations make a proper estimation of the gap futile. However, in all those cases the estimated orthogonality would be large.

The block-residuals may be estimated by the largest per-column residual for the respective interval, in case only those residuals are easily available. Note, however, that the per-column residuals typically are a bit smaller than the block-residual. Whether the bound may be useful in practice can only be confirmed or refuted experimentally.

4.5.2 Estimated upper bound

In addition to the assumed upper bound based on the $\sin\theta$ theorem from above, we will test another empirical bound for the achievable orthogonality. Similar to the saturation residual, we assume the orthogonality to be limited by numerical factors, the magnitude and number of floating-point operations, in addition to the gap between the two sets of Ritz values. This boils down to modifying the limits for the saturation residual from Section 4.3 by the inverse gap γ. Since the less pessimistic of the two presented limits might differ for different intervals $[\lambda_{\min}, \lambda^{\max}]$ and $[\mu_{\min}, \mu^{\max}]$, we assume the worst case. Then, corresponding limits for the orthogonality are

$$\frac{n\varepsilon}{\gamma}\left\|B^{-1}A\right\|_2 \qquad \boxed{1}$$

and

$$\frac{n\varepsilon}{\gamma}\max\{\min\{\lambda_{\min}, \lambda^{\max}\}, \min\{\mu_{\min}, \mu^{\max}\}\} \qquad \boxed{2}$$

with $\varepsilon \geq \boldsymbol{\varepsilon}$. Instead of a pessimistic estimation of the achievable residual, we may base our estimated orthogonality on a readily available and more precise foundation, the residual that actually was achieved by the subspace iteration algorithm. The estimated orthogonality then takes the form

$$\frac{1}{\gamma}\max\{\|R_1\|_2, \|R_2\|_2\}. \qquad \boxed{3}$$

From Equation (4.5), we infer the limit

$$\cos\left(\frac{\pi}{2} - \sin^{-1}\left(\left\|B^{-\frac{1}{2}}\right\|\frac{\|R_1\|}{\gamma}\right) - \sin^{-1}\left(\left\|B^{-\frac{1}{2}}\right\|\frac{\|R_2\|}{\gamma}\right)\right). \qquad \boxed{4}$$

We again assume the worst-case residual of both intervals, and for all following experiments, $\|R\|$ with $R = [r_1, \ldots, r_k]$ is replaced by $\max_{i=1}^{k}\|r_i\|$.

Experiment 4.19 — Interval gap and searchspace occupation (ideal filter)

In order to verify the influence of the gap between intervals and the limits described above, we set up two adjacent intervals with controllable gap on an otherwise evenly distributed spectrum. The gap size (relative to the spectral range) is then reduced stepwise by one order of magnitude,

$$\gamma \in \left\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}\right\}.$$

In all cases, the generated matrix pencil is of size 1000 *and the innermost* 20 *eigenpairs are computed, separated into two sets of* 10 *eigenpairs with a searchspace size per interval of* 10 *using ideal filters with a prescribed dampening of* 10^{-5}. *Figure 4.29 shows results for differently assembled searchspaces.*

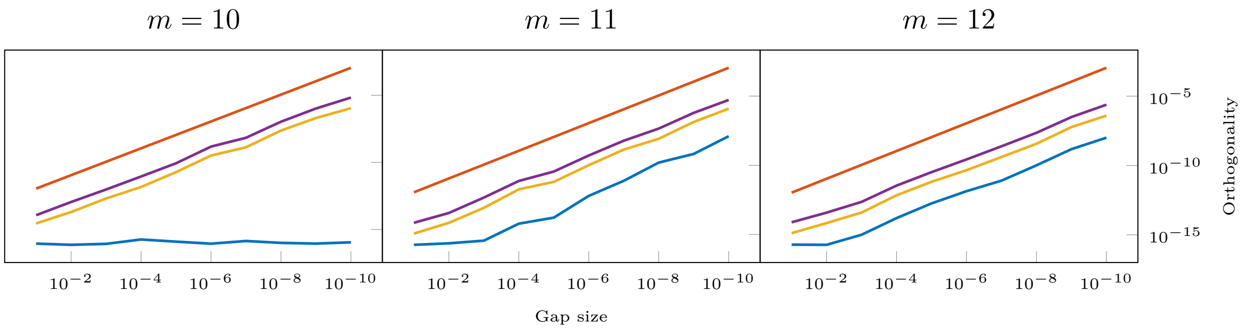


Figure 4.29: *Gap influence for different searchspace combinations.*
Also shown: the orthogonality limits $\boxed{1}$ *(red),* $\boxed{3}$ *(yellow),* $\boxed{4}$ *(purple).*

The orthogonality between the two intervals after 50 iterations does not show any influence of the gap size (Figure 4.29, left), and orthogonality is unimpaired, if the searchspaces are disjoint, i.e., do not contain shared directions. For further verification, the experiment is modified to include one (the nearest) direction of the respective other interval (Figure 4.29, center). The influence of the gap is now clearly visible, but the results are not very consistent. Other factors such as the (randomly chosen) initial guess for the searchspace set play a role. We have chosen a sequence for display here, where the orthogonality clearly shows an influence due to the gap size as a worst-case scenario. Including more directions increases consistency (Figure 4.29, right, for two additional directions).

It is difficult to reenact the setup from Experiment 4.19 with non-ideal filters; the limiting factor λ_{m_0+1} will almost always be the closest eigenvalue of the neighboring interval with distance γ, causing convergence to slow down significantly. We therefore modify the setup of Experiment 4.19 to verify influence of shared directions on inter-orthogonality.

4.5.3 Directional overlap

In the following, we will have to distinguish between the directions of the searchspace that belong to the target interval and the remaining outside directions that are included with larger searchspace sizes. We will employ the term *target searchspace* when referring to the former. The target searchspace is embedded in the searchspace. For the measured orthogonality, only directions of the target searchspace are considered. We will also use the term *overlap* in conjunction with target searchspaces and searchspaces. An overlap refers to shared directions between two spaces. This means that one or more of the iterated directions of two searchspaces or target searchspaces are associated with the same eigendirection and two Ritz pairs will be generated independently as approximations of the associated eigenpair. On the contrary, we speak of *disjoint* spaces if no directions are shared.

Experiment 4.20 — Interval gap and searchspace occupation (non-ideal)
We repeat Experiment 4.19 with a spectrum in the range of 10^{-5} *around one to generate detrimental conditions for orthogonality between the two intervals. The filter in use here is a Cauchy filter with Gauss-Legendre rule of degree* 8*. We then monitor which directions exactly are occupying the searchspaces* ($m = 15$)*. See Figure 4.30 for the results.*

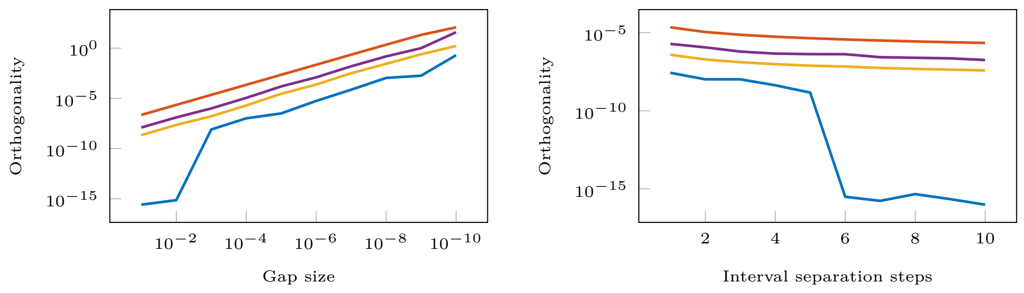


Figure 4.30: *Gap influence for different searchspace combinations.*
Also shown: the orthogonality limits $\boxed{1}$ *(red),* $\boxed{3}$ *(yellow),* $\boxed{4}$ *(purple).*

The first two runs for the gap sizes 10^{-1} and 10^{-2} exhibit a significantly improved orthogonality (Figure 4.30, left). Indeed, the first occurrence of shared directions among the target searchspace of an interval and the respective searchspace of the neighboring interval (and vice versa) is the third run with gap size 10^{-3} (the first non-ideal orthogonality in the plot).

Additionally, for an evenly distributed spectrum without gap, we successively move apart the target intervals, leaving an increasing number of eigenvalues in-between the two intervals while monitoring the directions occupying the searchspace ($m = 20$). The first occurrence of the target searchspaces being disjoint from the respective neighboring searchspace is the sixth interval separation (Figure 4.30, right; the first ideal orthogonality in the plot).

In conclusion, we see that diminishing orthogonality is chiefly caused by overlap in the target directions occupying the searchspaces of independently computed intervals. In practice, this will likely always be the case for adjacent intervals, but not necessarily for intervals that are not directly adjacent. Orthogonality is unimpaired if the overlap involves only the non-target portion of the searchspaces. We will refer to overlap that impairs orthogonality as *critical overlap*. Whether this condition is fulfilled can be determined by inspecting the (non-phantom) Ritz values of both intervals. Only if neighboring Ritz values happen to be located in the target interval (or vice versa)—with an error margin in the order of their residual—has the gap to be considered for further orthogonality estimations. The presence of Ritz phantoms with very slow convergence that are related to some eigendirection may be involved in the critical overlap, even though their Ritz values are not indicative of that. On the other hand, Ritz phantoms may indicate critical overlap, even though they do not belong to a critically overlapping direction. This is yet another reason to remove Ritz phantoms from the searchspace as early as possible during iteration.

The influence of overlap has implications on the choice of the filter function. Some filter functions, when acting on an oversized searchspace, tend to include eigenvalues that are located close to the target interval first. This is the case, first and foremost, for the Butterworth filter, but, to a certain degree, also for the other decaying filters. Non-decaying filters tend to include eigenpairs that are farther away from the target interval. We will call this property of a filter *locality*. This is, again, strongly connected to searchspace size. Larger searchspaces are likely to include directions that are located farther away from the target interval, causing diminished orthogonality over larger distances between intervals.

The concept of subspace overlap may make non-decaying filters seem unfavorable if orthogonality is to be accounted for, but, of course, there is no reason to combine a non-decaying filter with a largely oversized searchspace. In fact, tight searchspace sizes that work well with non-decaying filters are also beneficial for orthogonality since the chance of creating an overlap is reduced.

While Experiment 4.19 and Experiment 4.20 already include a first analysis of the usability of the limits introduced above, we verify the influence of two parameters separately: the dependency on $\|B^{-1}A\|_2$ and possible effects of convergence failure, which here shall describe the failure to reach the saturation residual for at least some of the slots. To explore whether distance and/or number of overlapping directions is a condition for the quality of orthogonality, we employ ideal filters again, explicitly selecting certain directions to be included in the searchspace.

Experiment 4.21 — Overlap and ideal orthogonality

We repeat Experiment 4.17 with modified setup to analyze the effect of overlap on inter-orthogonality in relation to the norm of the matrix pencil. To emphasize the effect on orthogonality, we create conditions under which we have to expect heavily diminished orthogonality by scaling the size of the target interval down to 10^{-5}. *In order to control the directions that occupy a slot in the searchspace directly, we create suitable ideal filters with a dampening of* 10^{-5}. *The distance of both intervals is in the order of* 10^{-8}. *At first, both filters are set up to include only directions inside the target intervals. The searchspace is then extended by explicitly chosen additional directions.*

a) We extend each searchspace by the direction located between the two intervals that is closest to, but not included in, the respective neighboring interval. While the spectral regions of the two searchspaces now overlap, directions are not shared. With our definition from above, the searchspaces do not overlap. The measured orthogonality is ideal.

b) We extend each searchspace by a single shared direction located exactly between *the two intervals. The searchspaces are now considered overlapping. Orthogonality is impaired for larger norms.*

c) We extend each searchspace by all directions located between the two intervals, increasing the number of shared directions. The effect on orthogonality increases.

d) We create critical overlap by having each searchspace include the nearest eigendirection of the respective neighboring interval. The measured orthogonality reaches a worst-case state. Increasing the overlap further, with or without critical directions, has no significant additional influence on orthogonality.

Figure 4.31 outlines the different configurations.

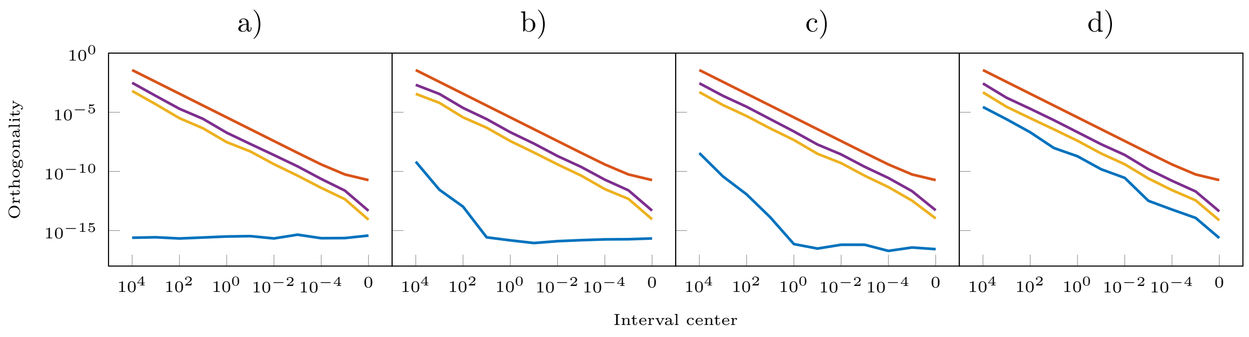


Figure 4.31: *Influence of different types of overlap on inter-orthogonality. Also shown: the orthogonality limits* $\boxed{1}$ *(red),* $\boxed{3}$ *(yellow),* $\boxed{4}$ *(purple). Refer to Experiment* 4.21 *for a description of the scenarios.*

The effect of the dampening strength on overlapping directions can be summarized as follows. If critical overlap occurs in the target searchspaces, orthogonality will be impaired with no beneficial bias due to different strengths of dampening. If the overlap is non-critical, orthogonality is generally better, but is still affected by the norm of the matrix pencil. Dampening of the overlapping direction has a beneficial effect on the overall orthogonality and additional iterations of subspace projection may reduce the orthogonality further. If no overlap occurs, orthogonality can generally be expected in the range of the machine precision $\varepsilon(1)$. Repeating the experiment with different distances does not change the overall outcome, which is why the plots are not included here.

In summary, orthogonality always seems good, if no neighboring directions are occupying a slot in the searchspace. If the searchspace size is not exact, i.e., the searchspace is oversized, orthogonality is still limited by the norm of the matrix pencil, but not by the gap between intervals. If the searchspace size is exact, orthogonality is in the range of machine precision. The orthogonality under non-critical overlap is related to the distance of the overlapping directions to the target directions of the neighboring interval and the norm of the matrix pencil. If the (single) overlapping direction is brought very close to the interval boundaries, orthogonality already suffers. The same is the case for moderate gaps but large norms.

The analysis for subspace overlap, while promising, requires the reliable removal of Ritz phantoms for an effective classification. In general, it is safer to rely on complete disjunction of Ritz value ranges to determine whether orthogonalization has to be conducted or not. It then may be possible to limit the orthogonalization procedure to intervals only a few steps away.

Experiment 4.22 — Overlap-based exclusion

For the four data sets of the matrix test set, we record whether the searchspaces of two intervals overlap by tracking the subspace range of all Ritz values for an interval. For simplicity, we use the strictest possible overlap detection method, only checking the disjunction of a pair of subspace ranges. We then determine the largest worst-case orthogonality among any two intervals that was accepted by this criterion, as well as the smallest worst-case orthogonality that was rejected. Rejected interval pairs would require explicit orthogonalization as per the subspace overlap condition. Table 4.7 summarizes all the values found.

Name	locking				no locking			
	even		uneven		even		uneven	
	min reject	max allow	min reject	max allow	min reject	max allow	min reject	max allow
`laser`	7.198e−08	5.987e−08	8.317e−08	8.744e−08	1.492e−09	4.719e−09	4.096e−09	4.833e−09
`SiH4`	8.572e−08	4.647e−06	1.242e−09	2.148e−07	2.984e−09	2.478e−07	5.588e−10	4.677e−07
`linverse`	4.614e−05	3.398e−05	2.790e−09	2.328e−06	3.298e−13	7.724e−10	6.854e−17	3.893e−08
`Pres_Poisson`	3.658e−06	4.879e−06	3.722e−06	9.415e−09	1.031e−07	1.915e−09	7.755e−08	1.298e−09
`Si5H12`	6.584e−07	9.303e−07	2.494e−07	5.653e−11	1.340e−08	2.771e−10	1.217e−08	5.815e−12
`brainpc2`	2.546e−06	1.473e−05	2.930e−06	1.172e−05	8.854e−08	1.647e−09	9.355e−08	5.468e−09
`rgg_n_2_15_s0`	1.424e−05	1.401e−09	1.368e−05	1.235e−09	4.092e−07	2.453e−11	1.764e−07	5.407e−11
`SiO`	1.647e−06	8.637e−10	1.279e−06	1.992e−06	3.491e−08	1.879e−11	6.507e−09	5.301e−11
`Andrews`	9.021e−06	1.130e−09	8.778e−06	9.001e−10	5.735e−08	6.507e−11	8.372e−08	5.307e−11
`Si34H36`	3.301e−06	2.659e−10	2.926e−06	5.012e−10	6.641e−09	3.845e−12	3.481e−08	1.379e−11
`fe_rotor`	2.160e−05	1.583e−08	1.930e−05	1.035e−08	3.475e−07	1.023e−10	6.719e−08	6.443e−10
`GraI-119k`	2.537e−06	1.060e−08	2.484e−06	1.479e−09	2.929e−09	3.986e−09	1.023e−09	8.245e−14

Table 4.7: *Subspace overlap evaluation. Smallest rejected and largest accepted worst-case orthogonality among all intervals for the four data sets of the matrices from the test set.*

While the smallest rejected orthogonality being small is not a problem per se, large values for the largest accepted orthogonality indicate a failure of the classification. From Table 4.7 it is easy to see that the overlap condition is not feasible in practice. This is due to the subspace iteration process not being allowed to progress to saturation, contrary to the experiments above. In practice, iteration will be stopped before saturation is reached in virtually all cases. We therefore instead have to rely on the estimated bounds formulated earlier.

4.5.3.1 Verification of orthogonality bounds

The purpose of a reasonably tight estimation of the inter-orthogonality of two intervals through cheap computations is the exclusion of certain orthogonalization steps, i.e., interval pairs. To propagate the required information, only the exchange of smaller sets of data such as residuals, eigenspace and interval ranges, and the number of vectors is required for any pair of intervals. While inter-orthogonality can easily be measured for all pairs of intervals, the required effort is larger; vectors have to be communicated and inner products have to be computed.

The proposed a priori orthogonality bounds 1–4 are pessimistic by nature. They are based on roughly the same criteria as the residual bounds and do not contain considerations of factors that may benefit orthogonality, such as disjoint searchspaces. Each bound has its pros and cons.

- The limit 1 does have a benefit compared to the other limits introduced here: it is monotonous with respect to the gap. Relying on this limit then allows to completely skip orthogonalization steps against intervals of larger distance. Being very pessimistic does reduce its usability, though. In addition, it represents an assumed maximum stagnation residual that can be reached under all circumstances. It does not take into account stopping iteration at a possibly much larger residual. As such, it has been broken repeatedly in experiments with the matrix test set.
- Limit 2 is not included in any of the experiments since its missing dependency on the norm of the matrix pencil has proven fatal.
- A strong point of limit 3 is that, even if the iteration does not work out as intended, be it because of Ritz phantoms included in the final result or eigenpairs that did not converge well due to some reason, the limit never seems to break. Since it largely relies on the residual and a large residual is typically accompanied by a comparatively bad orthogonality, it still works in many kinds of failure states.
- The bound obtained from limit 4 is very similar to 3 if the norm of B is ignored; the availability of this norm is a general problem wherever the B-residual is required. The definition of the *gap* becomes somewhat difficult in case an oversized searchspace is iterated, assuming that directions may not have converged very far or Ritz phantoms are present. In any case, the smallest distance between any two non-phantom Ritz values may be used. This, however, can only ever result in smaller gaps, making the estimation even more pessimistic. From this point of view, basing the computation of the gap on Ritz values inside the target interval is not unreasonable.

All upper bounds examined so far have in common that they can be very pessimistic. This is particularly true if there is no overlap, or at least no critical overlap, of the searchspaces.

One of the most interesting matrices from the test set (Section 4.1.2) is `linverse`. The orthogonalities from the four data sets are shown in Figure 4.32. In particular

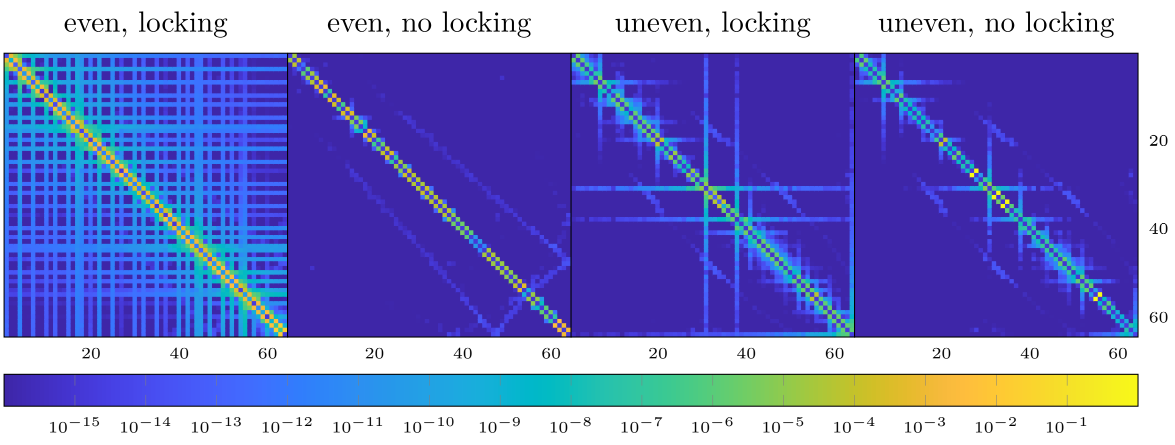


Figure 4.32: *Orthogonalities produced by the* `linverse` *matrix. Shown are the worst orthogonalities of all possible interval combinations.*

with locking, many intervals have a far reach in regard to influences on orthogonality. This is particularly noticeable if the distribution of eigenpairs is even, possibly cutting into dense clusters while specifying intervals. Subdividing the interval into equal-length sections is less likely to hit a cluster. The spectrum of `linverse` is composed of a sequence of small clusters and while some intervals are clearly conspicuous, especially with uneven distribution, the spectral distribution is not a clear indicator. The intervals 31 and 38, as well as 64 interact with almost all other intervals. While the boundaries of intervals 31 and 64 cut clusters, the boundaries of interval 38 do not. Further, there are additional intervals whose boundaries cut clusters, but who do not interact over large distances. Figure 4.33 gives an overview over all interval distances for all data sets from the test set. There is no indication the distance correlates with the patterns observed in Figure 4.32, other than for direct neighbors and duplicate

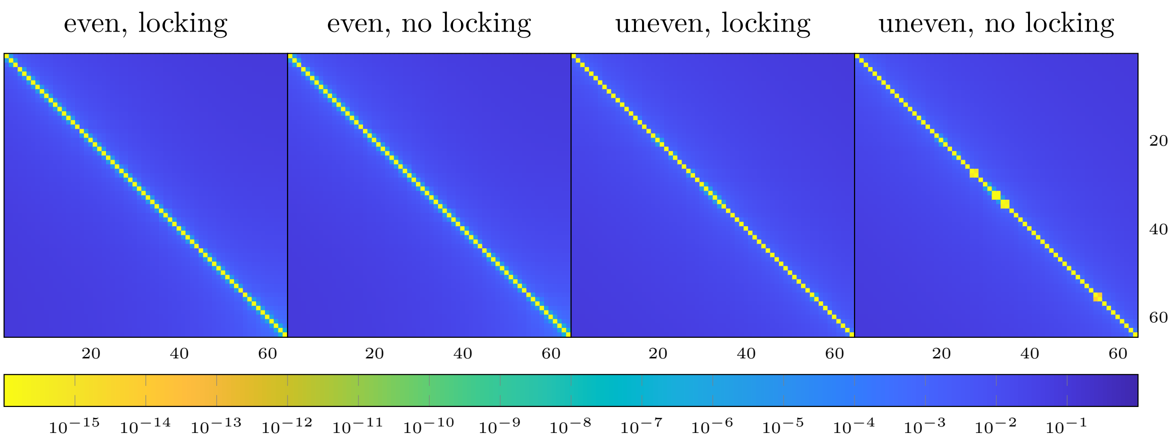


Figure 4.33: *Interval distances of the* `linverse` *matrix. Shown are the minimum distances between Ritz values of all possible interval combinations. Note that the color map has been reversed for improved visibility.*

eigenpairs (fourth picture). It is more likely that orthogonality of far away intervals is related to the number of vectors and general density in the respective intervals, but an obvious direct correlation could not yet be found. Nonetheless, the rather large orthogonalities may serve as a good test for the orthogonality bounds.

Experiment 4.23 — Quality of orthogonality bounds

With the results for the four data sets obtained from the matrix test set (see Section 4.1.2), we can apply the estimated orthogonality bounds $\boxed{3}$ *and* $\boxed{4}$ *and compare them to the computed real inter-orthogonality between all interval pairs. The bounds* $\boxed{1}$ *and* $\boxed{2}$ *are excluded for the reasons mentioned earlier. The visual representation easily skews the intuition of whether a bound is tight or pessimistic, and under which conditions; the ordering of the single interval interactions is crucial. Figure 4.34 shows orthogonalities and bounds for all* 2016 *interval pairs of the* `linverse` *data set with even eigenpair distribution and locking. The plots are sorted by distance of computed eigenpairs, inter-orthogonality, maximum residual, and bound* $\boxed{3}$.

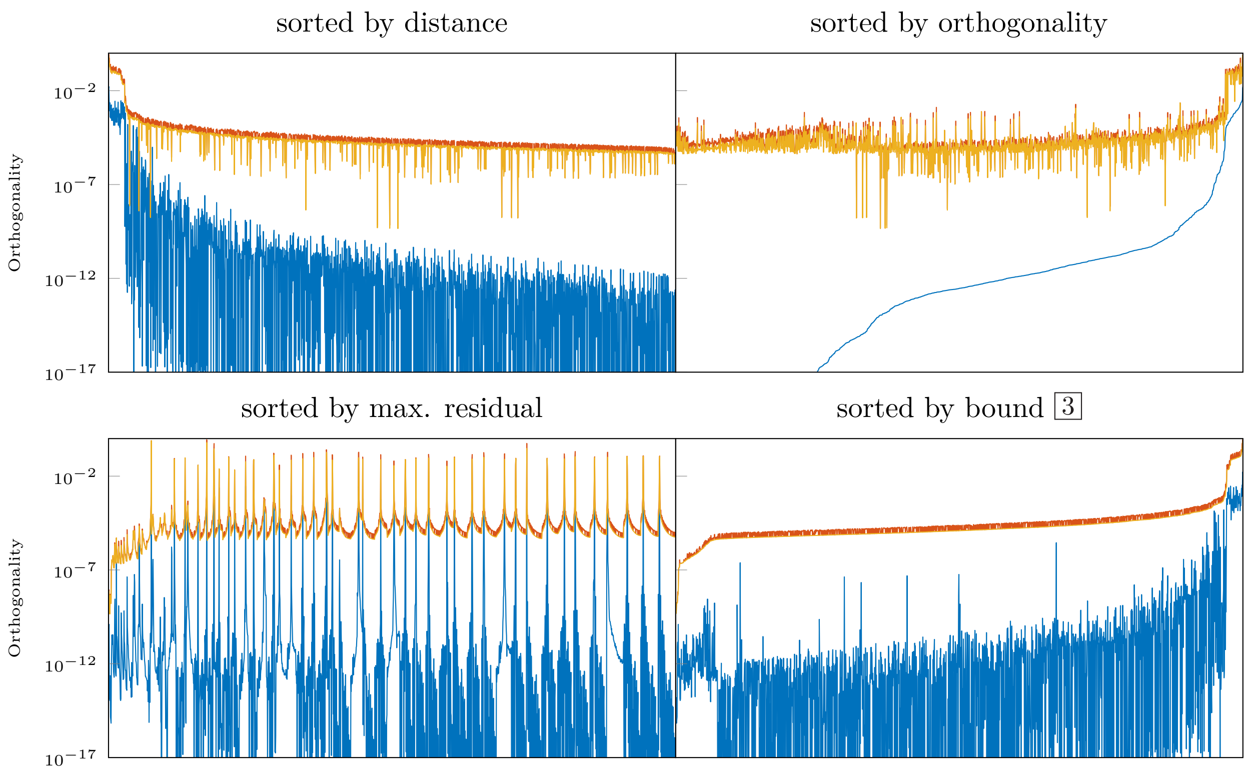


Figure 4.34: *Orthogonality bounds for* `linverse`*(even).*

While the fluctuations in the data appearing in the plots of Figure 4.34 tend to hide how pessimistic the bounds are, the second plot, sorted by orthogonality, reveals how large the discrepancy between estimated and achieved orthogonality can become. The third plot shows that at least interval pairs with low orthogonality are caught well by both bounds. The fourth plot indicates how similar bounds $\boxed{3}$ and $\boxed{4}$ are. If $B \neq I$ for $\boxed{4}$, $\boxed{3}$ would have to use $\left\|B^{-\frac{1}{2}}\right\| \|R\|$ for the residual. In both cases, the result is a shift of the bounds.

4.6 Evolution of orthogonality

So far we only have considered cases where the iteration has been allowed to progress far beyond residual saturation. However, orthogonality may improve further even after the residual has reached saturation if the iteration is continued. This is the orthogonality that has been analyzed here. In practice, the iteration will be stopped if some target residual has been reached.

Many other factors may influence the orthogonality that can be expected after a certain number of iterations or at certain levels of residuals. Before delving into orthogonalization schemes and their consequences, we shed light on some remaining questions regarding the development of orthogonality over the course of several iterations of the eigensolver.

Experiment 4.24 — Evolution of orthogonality under different conditions

We start this experiment by generating a matrix pencil of size 1000 *with evenly spaced spectrum in* $[-1, 1]$. *The central* $m = 20$ *eigenpairs are computed in two separate intervals, separated by a gap of* $g = 10^{-9}$. *An ideal filter with dampening* 10^{-3} *is generated to cover these exact eigenpairs for both intervals and the searchspace is chosen exactly sized to not generate Ritz phantoms yet. However, we choose a set of initial guess vectors that, in a setup with slightly oversized searchspace, will only emit Ritz phantoms that are not located inside the target intervals.*[11] *Since there is no critical overlap and the norm of the matrix pencil is in the order of unity, orthogonality can ultimately reach machine precision. This constitutes the calibration run for the behavior of the inter-orthogonality between both intervals, which converge simultaneously and under the same conditions.*

The conditions are now modified to highlight the influence of certain factors. First, the searchspace is inflated to 1.5 *times its initial size, producing five Ritz phantoms per interval. As mentioned before, those Ritz phantoms are located outside the respective target interval. Then, the convergence speed of inner eigenpairs is modified by changing their filter values. We linearly interpolate filter values between one and* 10^{-2}. *With this, values close to the interval boundaries will converge slower than those located towards the center of the interval. Finally, a disturbance in the order of* 10^{-13} *is applied to the projector, resulting in a disturbance norm of* $3.633 \cdot 10^{-12}$. *Figure 4.35 summarizes the results.*

The first run depicted in Figure 4.35 produces very steady convergence of all eigenpairs. The orthogonality improves at comparable speeds, reaching saturation

[11] The location of Ritz phantoms is not easily predicted; the vectors were found by repeated testing of random vector sets.

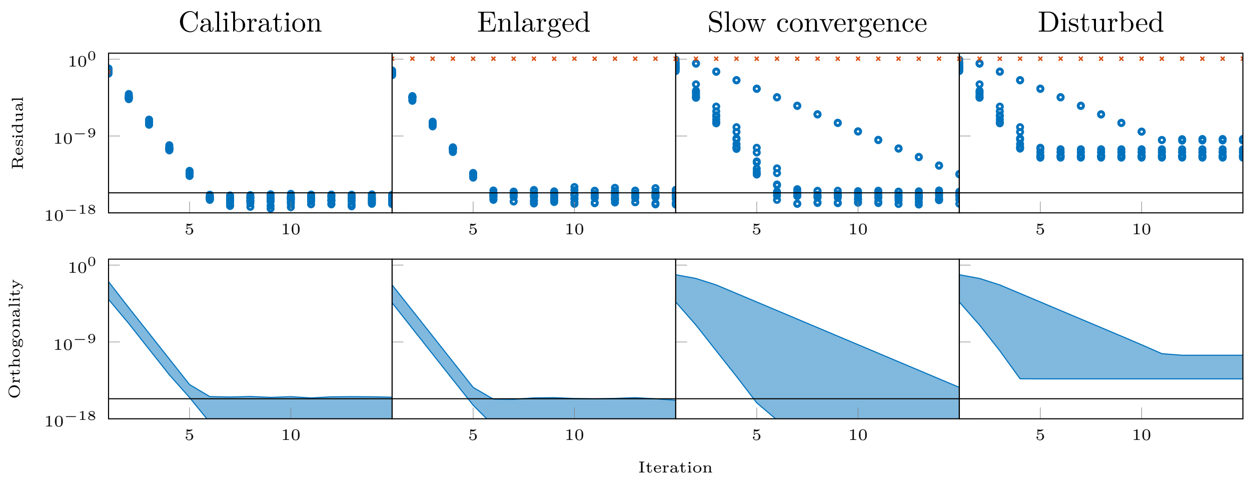


Figure 4.35: *Evolution of the inter-orthogonality of two independently iterated intervals in comparison to the evolution of the residual. The horizontal black line indicates machine precision (double).*

together with the residual. The introduction of Ritz phantoms, as shown in the second plot of Figure 4.35, does not influence the overall result, despite composing an orthogonal set of iterated vectors with the target eigenpairs and showing no signs of convergence themselves. However, the modification of the convergence speed of directions that participate in the computation of orthogonality has an effect on the result. As can be seen from the third plot of Figure 4.35, the worst-case orthogonality is, similarly to the residual, dictated by those directions of the target eigenpairs that converge the slowest. The fourth and final plot of Figure 4.35 shows that the impact of a systematic error, the disturbance of the projector, translates to the orthogonality as well. If the searchspace size is larger than the number of eigenpairs inside the target interval, directions will populate the searchspace that will not take part in the computation of orthogonality, which is reasonable since we are only interested in the orthogonality between vectors from the target intervals. The presence of outside directions and the fact that they typically will be less far converged, i.e., have a larger residual, raises the expectation for them to have an influence on the orthogonality of the whole system.

Once the analysis is extended to actually occupied additional directions, we enter the realm of gap influence. The inhibiting effects of subspace overlap and critical overlap have been discussed before. The above experiment can be extended to analyze the progression of orthogonality under these conditions in order to compare the behavior of the orthogonality.

Experiment 4.25 — Evolution of orthogonality with increasing distance

We generate a matrix pencil of size 1000 *with evenly spaced spectrum in* $[-1, 1]$. *The central* $m = 20$ *eigenpairs are computed in two separate intervals, separated by a gap of* $g = 10^{-3}$, *which is just the spectral distance of all adjacent eigenpairs of the spectrum. Since ideal filters seem to skew the results, we instead employ a*

Butterworth filter of degree 12. We perform four separate runs with distances between the interval boundaries of the two intervals of 0, $6 \cdot 10^{-3}$, $1.8 \cdot 10^{-2}$, *and* $3 \cdot 10^{-2}$. *The first distance, of course, produces critical overlap, while the second distance is exactly large enough for the two intervals to have disjoint searchspaces. The results are presented in Figure 4.36. For comparison, we also decrease the gap from* 10^{-3} *to* 10^{-5}, 10^{-7}, 10^{-9}, *and* 10^{-11}*; the results are shown in Figure 4.37.*

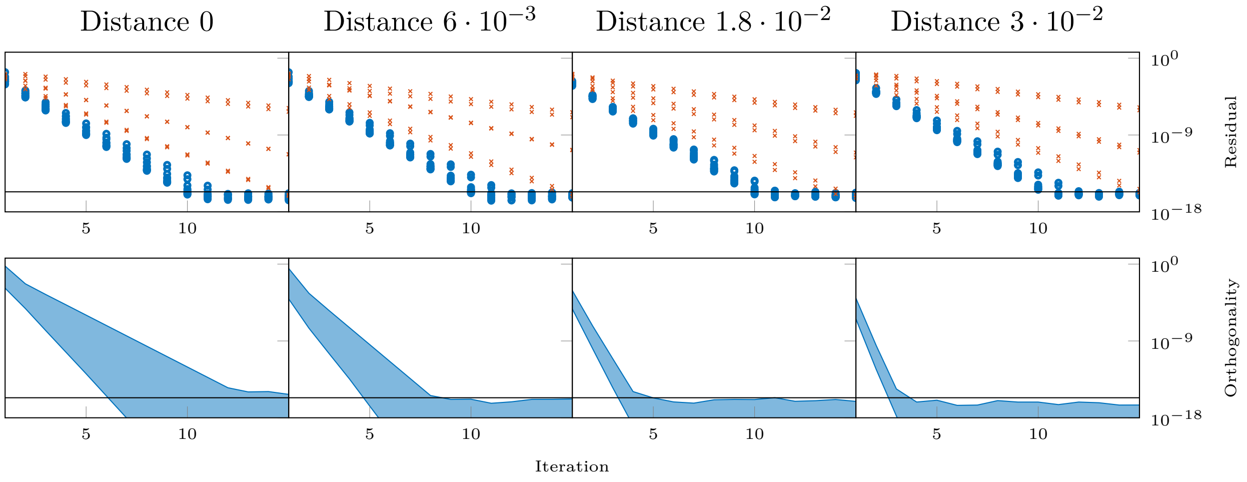


Figure 4.36: *Evolution of the inter-orthogonality of two independently iterated intervals with increasing distance (between interval boundaries) in comparison to the evolution of the residual. The horizontal black line indicates machine precision (double).*

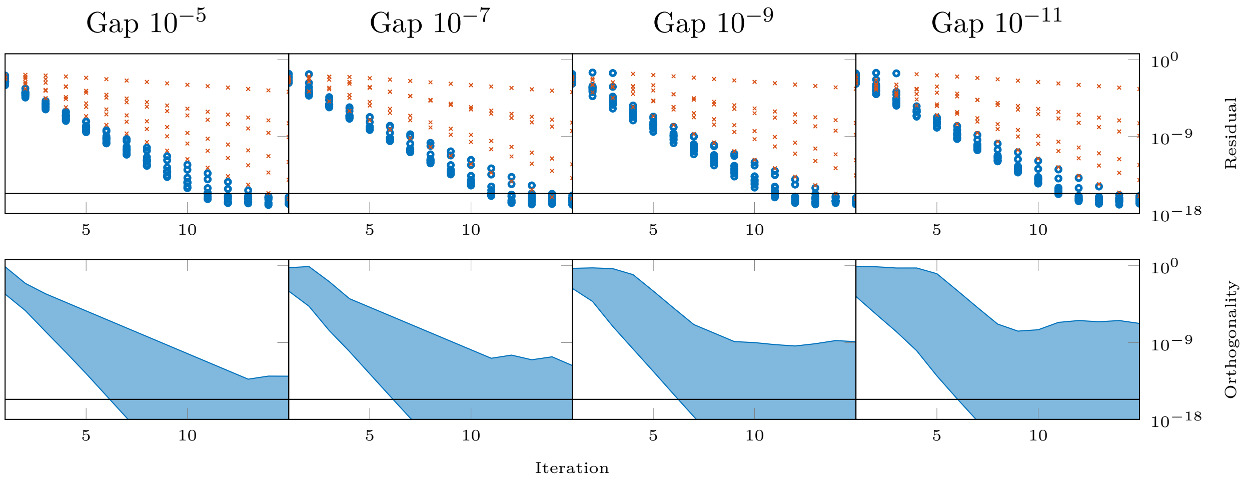


Figure 4.37: *Evolution of the inter-orthogonality of two independently iterated intervals with decreasing gap in comparison to the evolution of the residual. The horizontal black line indicates machine precision (double).*

For the first plot of Figure 4.36, distance zero, orthogonality improves at a similar rate as the residual. Without overlap, increasing the distance, as shown in the remaining plots of Figure 4.36, improves the initial orthogonality and increases convergence speed of the orthogonality while there is essentially no difference for the development of the residual among the four cases shown. This is likely the reason why estimations based on residual are that pessimistic. The orthogonality convergence speed being roughly comparable to residual convergence speed in case of subspace overlap can be seen in Figure 4.37. The convergence speed does not become much worse beyond a gap of 10^{-5}, the curve does not flatten out. Instead, the gap between intervals only defines a hard limit and quickly becomes dominant. The convergence of orthogonality is delayed due to Ritz phantoms in the interval.

4.7 Remarks

We have seen in Experiment 4.24 and Experiment 4.25 that a lower residual drop rate is accompanied by a lower orthogonality drop rate, but without overlapping searchspaces, the convergence speed of inter-orthogonality increases with distance. The results only apply to intervals that converge under identical conditions; intervals at different stages may have other implications on the development of orthogonality. It seems that worst-case orthogonality is dictated by the slowest converging direction that participates in the orthogonalization, as is evident from Experiment 4.24. Other peculiarities of residual convergence, such as cluster super-convergence, carry over to orthogonalities, but typically are much less pronounced compared to the experiments in this chapter, since only fast converging inner directions are considered for orthogonalization. It further is unclear, if the relation between distance and convergence speed is identical for all possible problems, or if other factors play a role. These factors make it difficult to find a more precise estimation of the orthogonality.

A possible estimation would have to be split into two parts, one for subspace overlap, where the gap defines a limit and improvement of orthogonality is tied to the residual, but initial orthogonalities are larger with smaller gaps, and one for disjoint searchspaces. Another factor is the consideration of gap sizes that are of the same order of magnitude as the achieved residual or smaller. A gap measured only from the Ritz values is likely to be no reasonable representation of the real gap in this case. Attempting to infer a usable estimation in this situation is a moot effort.

As has been outlined in larger detail in the previous sections, several cases emerge during subspace iteration that require the explicit orthogonalization of independently computed eigenpairs. Since the orthogonality that can be expected between any two intervals is difficult to estimate, it has become clear that orthogonalization of independently computed approximate eigenvectors is a necessity in general, if orthogonality is of concern for the respective application. In case it can be determined that single orthogonalization steps can be skipped or a certain order of orthogonalization is beneficial, a flexible approach to orthogonalizing a large number of intervals is required.

CHAPTER 5

Taming the BEAST – Orthogonalization

THE need for orthogonal vectors arises on multiple occasions throughout the solution of an eigenproblem using subspace iteration. The specific requirements differ from case to case. This chapter will outline a selection of orthogonalization algorithms with the ultimate goal to orthogonalize a sequence of approximate eigenvector blocks after subspace iteration has been stopped (at any residual) without imminent need for reiteration. Orthogonalization of one block against another will necessarily diminish the intra-orthogonality of the modified block if the inter-orthogonality of the involved remote blocks was poor. Similar effects on the residuals have to be expected. The fact that, after subspace iteration, vector blocks are intra-orthogonal and that orthogonality may be retained to some extent during inter-orthogonalization, however, opens up possibilities of more flexible orthogonalization schemes.

5.1 The many orthogonalities of BEAST

So far, the terms orthogonality and orthogonalization have been used in many places and different contexts. The searchspace set used in the Rayleigh-Ritz procedure may be orthogonalized in general or must be orthogonalized (Figure 5.1, a) to produce a reduced eigenproblem of standard form (while other methods to achieve the same exist); the computed approximate eigenvectors are inherently orthogonal; locking converged eigenpairs necessitates the maintenance of orthogonality of iterated vectors and locked vectors (Figure 5.1, b). Finally, orthogonality of independently computed approximate eigenvectors is influenced by several factors and may require orthogonalization (Figure 5.1, c), which again is expected to have implications on the orthogonality of the eigenvector blocks themselves.

In the case of the generalized Rayleigh-Ritz procedure, orthogonalization of the searchspace set is typically not necessary. The orthogonalization of an iterated approximately orthogonal block of vectors against a locked orthogonal block of vectors is generally unproblematic and trivial. This chapter will therefore be devoted to shouldering the task of cross-interval inter-orthogonalization.

This poses several requirements for the orthogonalization algorithms that may be used, apart from the overall minimum deviation from orthogonality of the result they can provide. Dealing with generalized eigenproblems, the inability to orthogonalize with respect to a Hermitian positive definite matrix B is an exclusion criterion for many algorithms. Additionally, different algorithms may have differing effects on the previously established residual, and we desire to identify a method to reduce potential loss of residual accuracy. Finally, in the context of large problem sizes and high-performance hybrid-parallel computing on many nodes, parallelization potential and communication overhead play an important role towards an efficient parallel eigensolver.

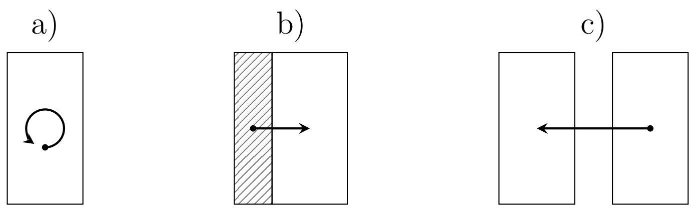


Figure 5.1: *Orthogonalization schemes.*
a) Intra-orthogonalization of a block of vectors.
b) Inter-orthogonalization between different vectors of a single block.
c) Inter-orthogonalization between different blocks of vectors.
Of course, b) and c) are conceptually equivalent.

5.2 Establishing intra-orthogonality

Many methods for orthogonalizing a set of vectors are known, one of which, Gram-Schmidt orthogonalization, has been introduced in Section 1.3.2. While reestablishing intra-orthogonality does not affect previously established inter-orthogonality (in exact arithmetic; with numerical influences, orthogonality might be affected depending on the condition of the matrix [SW06]), there is an additional factor that has to be accounted for in our case. Modification of the approximate eigenvectors is likely to introduce additional errors, reflected by increased residuals. In order to not destroy the accuracy achieved during subspace iteration, we aim at identifying orthogonalization methods that influence residuals as little as possible.

Additionally, if B is not the identity, we require orthogonalization methods that can be modified to produce B-orthogonal bases. A modification of this kind is not always a trivial task. The Gram-Schmidt method of creating orthogonal vector blocks as portrayed in Section 1.3.2 has the intrinsic ability to produce B-orthonormal vectors. For QR decompositions of Givens and Householder type, approaches in these directions have been made, but focus on a different understanding of orthogonality [SS03b; Sin06; SS08].

5.2.1 QR decomposition

As crudely outlined in Section 1.7.1, the QR decomposition of an $n \times m$ matrix Z, $Z = QR$, yields an orthogonal basis Q for the space spanned by Z and an upper triangular matrix R. If Q is square and R has size $n \times m$, we refer to the decomposition as *full* QR decomposition; such a decomposition exists for any matrix [GV13]. A QR decomposition may be constructed by subsequent applications of transformations T_i to the vector block Z in order to bring it to upper triangular form,

$$R = \begin{bmatrix} \widetilde{R} \\ \mathbf{0} \end{bmatrix} = T_k \dots T_1 Z \implies Q = T_1^{-1} \dots T_k^{-1}.$$

The matrix Q is I-orthonormal if all T_i are I-orthonormal and thus $T_i^{-1} = T_i^H$. The structure of R, where only the first m rows are non-zero, shows that only the first m columns of Q are required to reconstruct Z,

$$Z = T_1^{-1} \dots T_k^{-1} \begin{bmatrix} \widetilde{R} \\ \mathbf{0} \end{bmatrix} = \underbrace{T_1^{-1} \dots T_k^{-1} \begin{bmatrix} I \\ \mathbf{0} \end{bmatrix}}_{=: \widetilde{Q}} \widetilde{R}.$$

This is the aforementioned *thin* QR decomposition $Z = \widetilde{Q}\widetilde{R}$. Multiple types of transformations to bring Z to upper triangular form exist. If a QR decomposition is to be used to B-orthonormalize vectors Z, e.g., eigenvectors of a generalized eigenproblem, the transformations must fulfill

$$\widetilde{Q}^H B \widetilde{Q} = I,$$

which is generally the case if

$$T_1^{-H} B T_1^{-1} = I$$

or, equivalently, T_1 is B-orthonormal and all remaining T_i are I-orthonormal.[1] This is often not easily done unless $B^{-\frac{1}{2}}$ is known such that

$$T_1^{-H} B^{-\frac{1}{2}} B B^{-\frac{1}{2}} T_1^{-1} = I$$

and all T_i are I-orthonormal. Nevertheless, we shall introduce the most commonly used algorithms to compute (I-orthogonal) QR factorizations in addition to algorithms that naturally produce B-orthonormality.

5.2.2 QR via Givens rotations

There is no difference between Givens rotations and Jacobi rotations as introduced in Section 4.1.1 apart from the area of their respective application. We follow (roughly)

1) $\begin{bmatrix} I & \mathbf{0} \end{bmatrix} Q^H B Q \begin{bmatrix} I \\ \mathbf{0} \end{bmatrix}$ is obviously B-orthonormal if Q is B-orthonormal.

[Bin+02] and [GV13] to introduce the complex version of Givens QR in pragmatic manner. Let a rotation matrix Q_{ij} be applied to a vector z,

$$Q_{ij} z = y.$$

The factors c and s can be chosen to eliminate the j-th entry of the resulting vector y, i.e., c and s are chosen such that

$$\begin{pmatrix} c & s \\ -\overline{s} & c \end{pmatrix} \begin{pmatrix} z_i \\ z_j \end{pmatrix} = \begin{pmatrix} y_i \\ 0 \end{pmatrix}.$$

By requiring $y_j = -\overline{s} z_i + c z_j = 0$ and $|c|^2 + |s|^2 = 1$ for normalization, we first may choose $c = z_i$ and $s = \overline{z_j}$ and, via normalization, obtain

$$c = \frac{z_i}{\sqrt{|z_i|^2 + |z_j|^2}} \quad \text{and} \quad s = \frac{\overline{z_j}}{\sqrt{|z_i|^2 + |z_j|^2}}$$

since $|a|^2 = |\overline{a}|^2$ for complex numbers a. We also have to require c to be real such that Q_{ij} can be orthogonal. This can be enforced by choosing [Bin+02]

$$c = \frac{|z_i|}{\sqrt{|z_i|^2 + |z_j|^2}} \quad \text{and} \quad s = \frac{z_i}{|z_i|} \frac{\overline{z_j}}{\sqrt{|z_i|^2 + |z_j|^2}}$$

since $a\overline{a} = |a|^2$ for complex numbers a and assuming $z_i \neq 0$. Then,

$$y_i = c z_i + s z_j = \frac{z_i}{|z_i|} \frac{|z_i|^2 + \overline{z_j} z_j}{\sqrt{|z_i|^2 + |z_j|^2}} = \frac{z_i}{|z_i|} \sqrt{|z_i|^2 + |z_j|^2}.$$

If $z_i = 0$ then $c = 0$ and obviously s may be anything of unit absolute value. If $z_j = 0$, nothing has to be done and the rotation can be the identity. For numerical stability (the square root might overflow), different methods of computation are advisable. For details and possible implementations, see [Bin+02; GV13]. Many of these transformations are applied to Z for every column, bottom-up, to finally reach upper triangular form.

5.2.3 QR via Householder reflections

Instead of applying many elementary rotations, transformations can be found that can bring one column of Z to the desired form in just one step. We again follow [GV13]. A *Householder reflection* has the form

$$H = I - \beta v v^H \quad \text{with} \quad \beta = \frac{2}{v^H v},$$

where β is just a normalization factor, chosen such that H is orthonormal, as we will see in the following. The matrix vv^H is obviously Hermitian by inspection of its entries and $\beta v v^H$ is Hermitian since β is real. Thus,

$$\left(I - \beta v v^H\right)^H = I - \left(\beta v v^H\right)^H = I - \beta v v^H$$

and

$$\begin{aligned}\left(I-\beta vv^H\right)^2 &= I-2\beta vv^H+\beta^2 vv^H vv^H = I-\frac{4}{v^Hv}vv^H+v^Hv\frac{4}{\left(v^Hv\right)^2}vv^H\\ &= I-\frac{4}{v^Hv}vv^H+\frac{4}{v^Hv}vv^H = I.\end{aligned}$$

We see that H is orthonormal. The transformation represents a reflection about the hyperplane trough **0** that is orthogonal to v. By some elementary transformations we may write the application of H to a vector z as

$$Hz = z-\frac{2}{\|v\|^2}\left(v^Hz\right)v = z-2\left\langle z,\frac{v}{\|v\|}\right\rangle\frac{v}{\|v\|}.$$

The scalar product signifies the (possibly negative) length of the component of z in the direction v, measured in units of size $\|v\|$. Since v is the normal vector of the hyperplane, this is exactly the distance of the orthogonal projection of z onto the plane. Accordingly, the reflection of z then is

$$z+2\left\langle z,\frac{v}{\|v\|}\right\rangle\frac{-v}{\|v\|}.$$

We now desire

$$Hx = \begin{bmatrix}y_1\\ \mathbf{0}\end{bmatrix}.$$

From the above we have seen that v as well as z and its reflection lie in the same hyperplane, such that the elimination of all entries but the very first one may be enforced by letting

$$\begin{aligned}v = z+\begin{bmatrix}c\\ \mathbf{0}\end{bmatrix} \quad\Longrightarrow\quad Hz &= z-2\frac{v^Hz}{v^Hv}\left(z+\begin{bmatrix}c\\ \mathbf{0}\end{bmatrix}\right) = \left(1-2\frac{v^Hz}{v^Hv}\right)z-2\frac{v^Hz}{v^Hv}\begin{bmatrix}c\\ \mathbf{0}\end{bmatrix}\\ &= \left(1-2\frac{z^Hz+\overline{c}z_1}{z^Hz+\overline{z_1}c+\overline{c}z_1+\overline{c}c}\right)z-2\frac{v^Hz}{v^Hv}\begin{bmatrix}c\\ \mathbf{0}\end{bmatrix}\\ &= \left(\frac{-z^Hz+\overline{z_1}c-\overline{c}z_1+\overline{c}c}{z^Hz+\overline{z_1}c+\overline{c}z_1+\overline{c}c}\right)z-2\frac{v^Hz}{v^Hv}\begin{bmatrix}c\\ \mathbf{0}\end{bmatrix}\end{aligned}$$

and with

$$c = \frac{z_1}{|z_1|}\sqrt{z^Hz} \quad\Longrightarrow\quad -z^Hz+\frac{\overline{z_1}z_1}{|z_1|}\sqrt{z^Hz}-\frac{\overline{z_1}z_1}{|z_1|}\sqrt{z^Hz}+\frac{\overline{z_1}z_1}{|z_1|^2}\sqrt{z^Hz}\sqrt{z^Hz} = 0$$

since z^Hz is real. Then,

$$y_1 = -2c\frac{v^Hz}{v^Hv} = -2c\frac{z^Hz+\overline{c}z_1}{z^Hz+\overline{z_1}c+\overline{c}z_1+\overline{c}c} = -2c\frac{\sqrt{z^Hz}+|z_1|}{\sqrt{z^Hz}+|z_1|+|z_1|+\sqrt{z^Hz}} = -c.$$

There are many pitfalls regarding numerical stability when computing Householder transformations with finite accuracy. For details, see [GV13].

A sequence of Householder transformations can eliminate the entries below the main diagonal for each column of Z from left to right by operating on the trailing sub matrix. In step k, the full-size transformation matrix has the form

$$\begin{bmatrix} I_k & \\ & H_k \end{bmatrix} \quad \text{for} \quad k = 0, 1, \ldots, m-2,$$

where $H_k \in \mathbb{C}^{n-k \times n-k}$ is the appropriate Householder transformation as derived above for the vector $[z_{k+1,k+1}, \ldots, z_{n,k+1}]^T$ and I_0 is to be considered empty.

5.2.4 Gram-Schmidt QR

By writing down the vector modifications of the Gram-Schmidt or modified Gram-Schmidt algorithms from Section 1.3.2,

$$q_i = \frac{1}{\|\hat{v}_i\|}\left(v_i - \sum_{j=1}^{i-1}\langle v_i, q_j\rangle q_j\right) \qquad \Longrightarrow \qquad v_i = q_i\left(\underbrace{\|\hat{v}_i\|}_{r_{ii}} + \sum_{j=1}^{i-1}\underbrace{\langle v_i, q_j\rangle}_{r_{ji}}\right)$$

(where v_i in the scalar product may be replaced with $\hat{v}_i$ to match modified Gram-Schmidt) we automatically obtain the entries r_{ij} of R such that $Z = QR$.

5.2.5 B-orthogonality

Any method that computes the orthogonalization of Z as an explicit right-multiplication with some matrix W^{-1} is theoretically easily modified for B-orthogonalization by the substitution $Z = B^{\frac{1}{2}}Y$. If

$$V = ZW^{-1} \quad \text{and} \quad V^H V = W^{-H}Z^H Z W^{-1} = I$$

then

$$W^{-H}Y^H BYW^{-1} = I,$$

which means that YW^{-1} is B-orthogonal. Similarly, if Y is B-normalized, Z is normalized and vice versa. This would be true also for the above methods, if R^{-1} would ever be computed explicitly. Of course, the computation of $B^{\frac{1}{2}}$ is out of the question as well. Gram-Schmidt QR includes the substitution implicitly via the scalar product. Another (but very similar) way to implicitly include this substitution, is to base the orthogonalization on the inner product $Y^H BY$.

5.2.6 Cholesky QR

The *Cholesky* decomposition (see, e.g., [GV13]) of a Hermitian matrix M has the form

$$M = LL^H,$$

where L is a lower triangular matrix. The diagonal elements of L are real and positive. For a vector block Z as before, $Z^H BZ$ is Hermitian and

$$Z^H BZ = LL^H$$

with $Z = QR$ implies

$$R^H Q^H BQR = R^H R,$$

and thus $R = L^H$ if Q shall be B-orthogonal. With given R we set $Q = ZR^{-1}$. The computation does not involve sequences of transformations other than implicitly via the computation of L. While further details on the computation of the Cholesky decomposition shall not be given here, once L is obtained, the implementation is straightforward and easily parallelizable.[2] Assuming a block row distribution in a parallel context, the only operation to provoke inter-process cooperation is the inner product, if the result is made available on all processes. The computation of the Cholesky factor and the inversion that follows are purely local operations. The Cholesky decomposition can fail in case of very ill-conditioned input matrices, see Section 5.2.13.

5.2.7 Rank revelation

The QR decomposition $Z = QR$ does not generally reveal the rank of Z as the rank of R [GV13] (while it may in many cases). Given that R is triangular, the eigenvalues are found on the diagonal. To aid numerical stability and to move the small eigenvalues of R to the lower right corner, pivoting is employed to effectively compute decompositions [Cha87]

$$Z\Pi = QR,$$

where Π is a permutation matrix reordering the columns of Z. In the case of Householder QR, the division by $\|v\|$ may jeopardize accuracy, since divisions by small numbers constitute an amplification of the numerical error in the numerator [GV13]. This is also the reason why the Cholesky decomposition may become unstable without pivoting. Choosing the column with the largest norm for the next elimination minimizes the problem. For the orthogonalization of eigenvectors, where, under normal circumstances, we can expect full rank and ideal conditioning, stability is of minor concern. A mostly fail-proof method for the determination of the rank of a matrix is the singular value decomposition [GV13], which can also serve as orthogonalization method.

[2] This is highly compatible with the data distribution scheme used in GHOST, PHIST, and BEAST, where block vectors are distributed block row wise and the results of inner products are replicated over all participating processes; see Chapter 3.

5.2.8 Methods based on singular value decomposition

The *singular value decomposition* (SVD, see, e.g., [GV13]) of a matrix $Z \in \mathbb{C}^{n\times m}$ has the form

$$Z = V\Sigma W^H,$$

where $V \in \mathbb{C}^{n\times n}$ is orthonormal, $W \in \mathbb{C}^{m\times m}$ is orthonormal as well, and $\Sigma \in \mathbb{R}^{n\times m}$ is a matrix where only the diagonal of the upper $m \times m$ submatrix is occupied by non-negative entries, the so called *singular values* of Z. Similarly to the QR decomposition, a "thin" variant is implied by the structure of Σ,

$$Z = \widetilde{V}\widetilde{\Sigma}\widetilde{W}^H,$$

where $\widetilde{V} \in \mathbb{C}^{n\times m}$, $\widetilde{W} \in \mathbb{C}^{m\times m}$, and $\widetilde{\Sigma} \in \mathbb{R}^{m\times m}$ is a diagonal matrix. As before, $\widetilde{V}$ and $\widetilde{W}$ are orthonormal and $\widetilde{V}$ spans the same space as Z. Algorithms for the computation of singular value decompositions can be found, e.g., in [GV13].

While direct B-orthogonality is not easily obtained, the *singular values* of a matrix $Z \in \mathbb{C}^{n\times m}$ are defined as the square roots of the eigenvalues of Z^HZ (see, e.g., [Saa11]) and we may analogously define *B-singular values* as the square roots of the eigenvalues of Z^HBZ. For $S = Z^HZ$, if λ_I is an eigenvalue of S, then $\sigma_I = \sqrt{\lambda_I}$ is a singular value of Z, thus, if $S_B = Z^HBZ$, $\sigma_B = \sqrt{\lambda_B}$ is indisputably a singular value of $B^{\frac{1}{2}}Z$. Further, since

$$\left\|B^{\frac{1}{2}}Z\right\| = \sigma_B^{\max} \quad \text{and} \quad \left\|B^{\frac{1}{2}}Z\right\| \leq \left\|B^{\frac{1}{2}}\right\|\|Z\| \qquad \Longrightarrow \qquad \sigma_B^{\max} \leq \sqrt{\lambda^{\max}(B)}\,\sigma_I^{\max}.$$

We may add that the eigenvalues of Z^HBZ are larger or equal to zero and Z^HBZ is thus positive semi-definite since $y^HZ^HBZy \geq 0$ by the definition of the B-scalar product. An orthogonalization method based on the computation of singular values in this manner is described in the following.

5.2.9 SVQB

As with Cholesky QR, the decomposition $Z = V\Sigma W^H$ gives

$$Z_B = Z^HBZ = W\Sigma V^HBV\Sigma W^H = W\Sigma^2W^H$$

if V is supposed to be B-orthogonal, similarly to Section 4.2.2. The Hermitian eigenproblem $Z_BW = W\Sigma^2$ can be solved for (W, Σ^2) and $V = ZW\Sigma^{-1}$; the inversion of Σ is trivial. In [SW06], a scaled version is proposed, modifying the method to

$$\operatorname{diag}(Z_B)^{-\frac{1}{2}}Z_B\operatorname{diag}(Z_B)^{-\frac{1}{2}}W = W\Sigma^2 \quad \text{or} \quad Z_BW' = \operatorname{diag}(Z_B)W'\Sigma^2$$

such that $V = Z\operatorname{diag}(Z_B)^{-\frac{1}{2}}W\Sigma^{-1}$ or $V = ZW'\Sigma^{-1}$. Note that $\operatorname{diag}(Z_B)$ is, of course, real and constitutes a normalization of Z. It also is the standardization method from Section 1.2.2 again with $W' = \operatorname{diag}(Z_B)^{-\frac{1}{2}}W$. The difference is that W' is $\operatorname{diag}(Z_B)$-orthonormal while W is I-orthonormal. Instead of an SVD, only singular values are computed here, as mentioned in the introduction. In case $B \neq I$, an SVD does not even exist in the required form.

5.2.10 B-orthogonal QR via SVQB

The combination of different methods allows to obtain a QR decomposition from the SVQB by combining it with a smaller QR decomposition. For the SVQB method, the factorization $Z = VH$ implies $H = \Sigma W^H$ respectively $H = \Sigma W^H \operatorname{diag}(Z_B)^{\frac{1}{2}}$ (or $H = \Sigma W'^H \operatorname{diag}(Z_B)$). Now any QR decomposition of $H = QR$ may be used to produce a full size B-orthonormal[3] QR decomposition of $Z = VQR$.

While it is possible to chain basically arbitrary orthogonalization algorithms this way, overall orthogonality is unlikely to be improved. Consider $Z = VH$ and a chained decomposition $H = V_1 H_1$. Then, for $V_1^H V^H B V V_1$ to be closer to the identity than $V^H BV$, V_1 would have to be laid out to be approximately $V^H BV$-orthogonal, which is generally not the case.

5.2.11 Parallel block-orthogonalization

A reasonably simple method to orthogonalize a tall and skinny block of vectors that is distributed block row wise over p processes such that the local number of rows n_l is larger than or equal to the number of columns m, is the embarrassingly parallel computation of local orthogonalizations, but only if the resulting block vector Q is *not* required to span the same space as Z. Certainly, the (properly scaled) concatenation of orthonormal subblocks Q_i is itself orthonormal,

$$\frac{1}{p}\begin{bmatrix} Q_1^H & \dots & Q_p^H \end{bmatrix}\begin{bmatrix} Q_1 \\ \vdots \\ Q_p \end{bmatrix} = \frac{1}{p}\sum_{k=1}^{p} I = I,$$

but the inclusion of a matrix $B \neq I$ is not easily possible. If, however, $B = I$ and the only goal is to create an orthogonal set of vectors in parallel, for example to obtain an orthogonal set of vectors as initial searchspace set, then this approach is feasible.

5.2.12 TSQR

Not quite a novel QR algorithm of its own, the *tall skinny QR* algorithm [Dem+12; Hoe11] has been developed with the goal to optimize the computation of a QR factorization for execution in massively parallel environments in case Z has significantly more rows than columns. It is a blocked method that employs existing sequential and parallel QR algorithms to construct the QR decomposition of a larger matrix. In this regard, the algorithm is a combination of the chaining of algorithms from Section 5.2.10 and the subdivision parallelization method from Section 5.2.11. It cannot easily be adapted for B-orthogonality for the same reason.

The TSQR algorithm assumes a one-dimensional block row distribution,[4] but, in its most basic form, requires the number of block rows (typically equal to the number

[3] Since $Q^H V^H BVQ = Q^H Q = I$.

[4] This is compatible with GHOST, PHIST, and BEAST; see Chapter 3.

of participating processes), p, to be a power of two. The mode of operation can be understood as working level-wise bottom-up on a binary tree of height $h = \log_2(p)$ for p block rows. The first step (on the leaves) computes the local QR decomposition (assuming the number of local rows to be larger than the columns)

$$Z_i^{(0)} = Q_i^{(0)} R_i^{(0)} \quad \text{for} \quad i = 0, \dots, p-1.$$

The following steps successively combine neighboring factors

$$Z_{2^\ell k}^{(\ell)} = \begin{bmatrix} R_{2^\ell k}^{(\ell-1)} \\ R_{2^\ell \left(k+\frac{1}{2}\right)}^{(\ell-1)} \end{bmatrix} \quad \text{for} \quad k = 0, \dots, 2^{h-\ell} - 1 \quad \text{and} \quad \ell = 1, \dots, h$$

and compute their QR decompositions in parallel. During this process, every new R factor has the same constant size m, such that the number of rows in the parallel QR factorizations is $2m$ and constant as well. The procedure stops at the root of the tree with a single QR decomposition that yields the final R factor. The orthogonal vectors Q are multiplications of the intermediate orthogonal vectors, arranged the right way, as is outlined for $p = 4$ in Figure 5.2 (similarly to the picture found in [Dem+12]). The recomposition of Q is then intuitively clear. The completed i-th

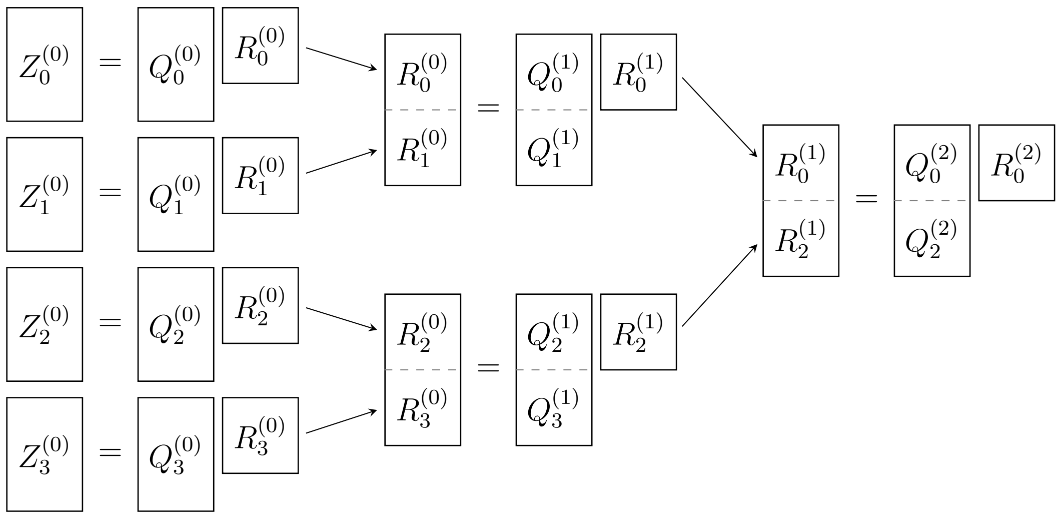

$Z_0^{(0)}$ = $Q_0^{(0)}$ $R_0^{(0)}$
$Z_1^{(0)}$ = $Q_1^{(0)}$ $R_1^{(0)}$
$Z_2^{(0)}$ = $Q_2^{(0)}$ $R_2^{(0)}$
$Z_3^{(0)}$ = $Q_3^{(0)}$ $R_3^{(0)}$
$R_0^{(0)}$ $R_1^{(0)}$ = $Q_0^{(1)}$ $Q_1^{(1)}$ $R_0^{(1)}$
$R_2^{(0)}$ $R_3^{(0)}$ = $Q_2^{(1)}$ $Q_3^{(1)}$ $R_2^{(1)}$
$R_0^{(1)}$ $R_2^{(1)}$ = $Q_0^{(2)}$ $Q_2^{(2)}$ $R_0^{(2)}$

Figure 5.2: *TSQR binary tree scheme with four processes.*

block Q_i of Q is built as

$$Q_i = Q_i^{(0)} \prod_{\ell=0}^{h-1} Q_k^{\ell+1} \quad \text{with} \quad k = \left\lfloor \frac{i}{2^\ell} \right\rfloor.$$

A sequential version works similarly by combining factors of different stages, as indicated in Figure 5.3. Note that not all intermediate blocks are stacked R-type factors in this case. Modifications of this kind may be introduced into the decomposition

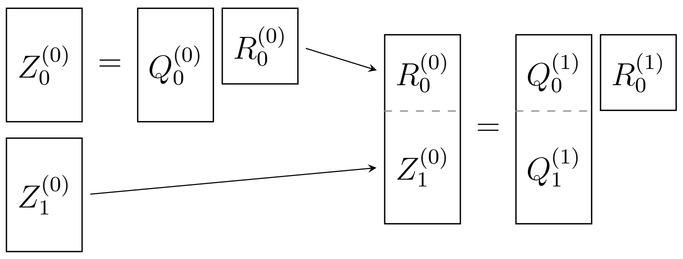


Figure 5.3: *Single step of the TSQR sequential scheme.*

scheme at any stage (not necessarily stage zero as in the example above) and for any block. In general, this is only necessary if the initial number of vector blocks is not a power of two.

For the involved parallel QR decompositions, any algorithm may be used, (including TSQR itself, if at least one block is not yet triangular). In the original proposal, a version of Householder QR is used that can exploit the structure of the intermediate vector blocks, two triangular matrices stacked on top each other [Dem+12]. Technically, the decompositions computed in the several stages of the algorithm need not be QR decompositions at all. Indeed, only the last stage has to yield a QR decomposition in order to produce a valid QR decomposition in total. Different stages may employ any orthogonal decomposition, like an SVQB variant, SVD, or similar. Whether or not this is a good idea is debatable.

5.2.13 Condition and multi-orthogonalization

An analysis of the stability of an orthogonalization algorithm typically amounts to bounding the deviation of the inner product of the resulting vectors with themselves from identity in the presence of inexact floating point arithmetic with machine precision ε. Not uncommonly, the condition number of the input matrix Z, which we define here as the relation between its smallest and largest singular value (see also Section 1.3.2),

$$\kappa(Z) = \frac{\sigma_{\max}(Z)}{\sigma_{\min}(Z)},$$

becomes a dominant factor in the resulting expression, essentially defining the magnitude of the worst possible error that could potentially be made. This may be intuitively understood, seeing that the eigenvalues of $Z^H Z$ are the squared singular values of Z and approach unity as Z approaches ideal orthogonality. As such, the condition number of a matrix is related to its intra-orthogonality and an orthogonal matrix has ideal unit condition. The condition number serves as an indicator of the accuracy of a numeric algorithm that is applied to some input. Its more general definition describes the fluctuations in the result in relation to the fluctuations introduced in the input (see, e.g., [Hig02; Saa03; GV13]). A summary of the stability bounds of the methods discussed here will follow in a later section.

The above reveals the general drawback of inner product based orthogonalization methods such as Cholesky QR or SVQB, the squared condition of the matrix $Z^H Z$ that then often is to be decomposed by algorithms that themselves can react sensitively to ill conditioned matrices.[5] This is the case for clearly inner product based methods, but also for methods where this relation is less obvious, such as the Gram-Schmidt family [Hof89]. However, the numerical stability, i.e., the deviation from orthogonality, of an orthogonalization routine can be improved simply by performing the orthogonalization (or single orthogonalizations that appear in the algorithm itself) twice (or more). Since the condition of the resulting vectors can be assumed to be better than the condition of the initial vectors, an additional pass of the same algorithm will now produce errors that, in magnitude, are bound by an expression related to the improved condition of the new input vectors. In the case of Cholesky QR, for example, the stability of the algorithm hinges on the stability of the underlying Cholesky factorization, a process that may break if the matrix is close to being singular, i.e., it has an eigenvalue near zero. Since for Cholesky QR the eigenvalues of the matrix in question are the squared singular values of the input vector block Z, the problem even intensifies, as has been mentioned above. This can be mitigated by performing the algorithms twice [Fuk+14], but the success of the factorization can still not be guaranteed and consequentially, an algorithm including an additional pass with shift modification has been proposed [Fuk+20]. For Gram-Schmidt, a modification that includes two or more orthogonalization passes for each column [Dan+76; Hof89; Dem+12] can increase stability to machine precision. These methods are typically referred to as *iterative* Gram-Schmidt.

Any of the introduced algorithms may simply be repeated to improve orthogonality. Of course, the computational cost of more complex orthogonalization methods increases rapidly. If many orthogonalizations have to be performed, a cost-efficient method has to be preferred. In the case of possible other iterative orthogonalization methods that inherently are not designed to reach orthogonality in one step but to improve orthogonality by some margin, multiple applications are inherently required.

5.2.14 Weak Gram-Schmidt

Since the Gram-Schmidt orthogonalization procedure is based on the computation of the inner products of the involved vectors, it is not possible to use a potentially optimized block inner product similarly to Cholesky QR[6] since modifications of later vectors are always based on results from earlier vectors. If we were to compute the Gram-matrix $Z_B = Z^H B Z$ en bloc, orthogonality would not be preserved by the algorithm. Consider the Algorithm 1.2 operating on a priori scalar products $\langle v_i, v_j \rangle_B$ resulting in corrections $v_i \leftarrow v_i - \langle v_i, v_j \rangle_B v_j$ that are not based on the intermediate

[5] If such an algorithm is used to orthogonalize eigenvectors, where we can assume that the vector block has full rank and B is positive definite, stability is not in immediate jeopardy.

[6] In the sense of computing the complete inner product of all vectors, as is the case with Cholesky QR. Of course internally blocked Gram-Schmidt implementations exist and make use of block inner products, see, e.g., [Oli+00; Ste08]. Also compare Section 5.3.5.

orthogonal subset of vectors. This can be understood as step-wise orthogonalization such that orthogonality of the first pair of vectors is established and all subsequent modifications have (presumably) no beneficial effect towards orthogonality. Successive application of the algorithm produces one additional orthogonal vector in each pass, until, after m steps, all vectors are orthogonal.

If there is, however, already some level of orthogonality, this method has the property to improve orthogonality proportionally to the initial orthogonality. We will call this method *weak Gram-Schmidt*. The improvement of orthogonality is quickly explained by manually following the first few steps. Assume the vectors z_i to be normalized and let the v_i be the processed vectors. Further, denote the *Gram factors* as $a_{i,j} = \langle z_j, z_i \rangle_B$. We obtain the vectors

$$\begin{aligned}
v_1 &= z_1 \\
v_2 &= z_2 - a_{1,2}\, z_1 \\
v_3 &= z_3 - a_{1,3}\, z_1 - a_{2,3}\, z_2 \\
&\;\vdots \\
v_m &= z_m - \sum_{k=1}^{m-1} a_{k,m}\, z_k
\end{aligned}$$

and, assuming $j < i$, compute their Gram factors[7]

$$\begin{aligned}
v_j^H B v_i &= \left(z_j^H - \sum_{k=1}^{j-1} \overline{a_{k,j}}\, z_k^H \right) B \left(z_i - \sum_{k=1}^{i-1} a_{k,i}\, z_k \right) \\
&= z_j^H B z_i - \sum_{k=1}^{i-1} a_{k,i}\, z_j^H B z_k - \sum_{k=1}^{j-1} \overline{a_{k,j}}\, z_k^H B z_i + \sum_{k=1}^{j-1} \sum_{\ell=1}^{i-1} \overline{a_{k,j}}\, a_{\ell,i}\, z_k^H B z_\ell \\
&= a_{j,i} - a_{j,i}\, a_{j,j} - \sum_{\substack{k=1\\k\neq j}}^{i-1} a_{k,i}\, a_{j,k} - \sum_{k=1}^{j-1} \overline{a_{k,j}}\, a_{k,i} + \sum_{k=1}^{j-1} \overline{a_{k,j}}\, a_{k,i}\, a_{k,k} \\
&\qquad + \sum_{k=1}^{j-1} \sum_{\substack{\ell=1\\ \ell\neq k}}^{i-1} \overline{a_{k,j}}\, a_{\ell,i}\, a_{k,\ell} \\
&= - \sum_{\substack{k=1\\k\neq j}}^{i-1} a_{k,i}\, a_{j,k} + \sum_{k=1}^{j-1} \sum_{\substack{\ell=1\\ \ell\neq k}}^{i-1} a_{j,k}\, a_{\ell,i}\, a_{k,\ell}
\end{aligned}$$

from which we see that the Gram factors improve if all $a_{i,j}$ are small enough. This means that the sum of the absolute values of the occurring Gram factors must be smaller than the original orthogonality to guarantee progression; otherwise, improvement of the Gram factors is pure luck. Expressions for the Gram factors between vectors of higher indices are more complex, but only consist of products of orthogonalities, which we consider orders of magnitude smaller, with the original orthogonality

[7] Remember $a_{i,i} = 1$ and $a_{i,j} = \overline{a_{j,i}}$

canceling out. We may also deduce—in the worst case—that there occur at most $(i-2)$ terms with a product of two Gram factors and at most $(i-2)(j-1)$ terms with a product of three Gram factors.

Let us have a look at the Gram factor for $i = m$ and $j = m-1$, which contains the most terms overall. If we assume $0 < |a_{i,j}| \leq a$ for all vector pairs, m vectors in total, and, at the same time that no further terms cancel out, we might require

$$a > (m-2)^2 a^3 + (m-2)a^2 \quad \Longrightarrow \quad a < \frac{-1+\sqrt{5}}{2(m-2)} \tag{5.1}$$

for guaranteed improvement of the absolute values of the Gram factors (not orthogonalities yet) towards machine precision. These numbers become smaller as the number of vectors increases, but Figure 5.4 shows that even with 1000 or 10 000 vectors, the requirement is bearable, at least in not too extreme cases. The rate of

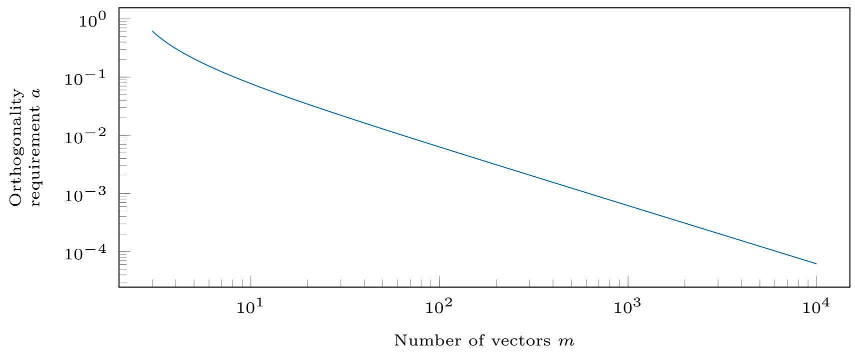


Figure 5.4: *Minimum required initial orthogonality a for weak Gram-Schmidt as a function of the number of vectors $m \geq 3$.*

improvement for one step of weak Gram-Schmidt with initial worst-case orthogonality among all vectors a is then at least

$$(m-2)a + (m-2)^2 a^2$$

with new worst-case Gram factor $(m-2)a^2 + (m-2)^2 a^3$; in practice, Gram factors typically will be better than this bound. Figure 5.5 displays the estimated and achieved Gram factors for one step of weak Gram-Schmidt for different initial orthogonalities a. Results are shown for $m = 10$, 100, and 1000 for vectors of size 10 000. With increasing numbers of vectors the deviation of the estimation from the actually computed value grows and the achieved Gram factors are better than predicted. This is caused by cancellation, in particular if more terms are involved, and due to not all vector pairs having the exact same worst-case orthogonality. For the same reasons, the initial orthogonality requirement is less strict and improvement is possible, even if the postulated requirement is not met. In particular, vectors with lower indices are subject to less strict constraints and improvement is therefore larger.

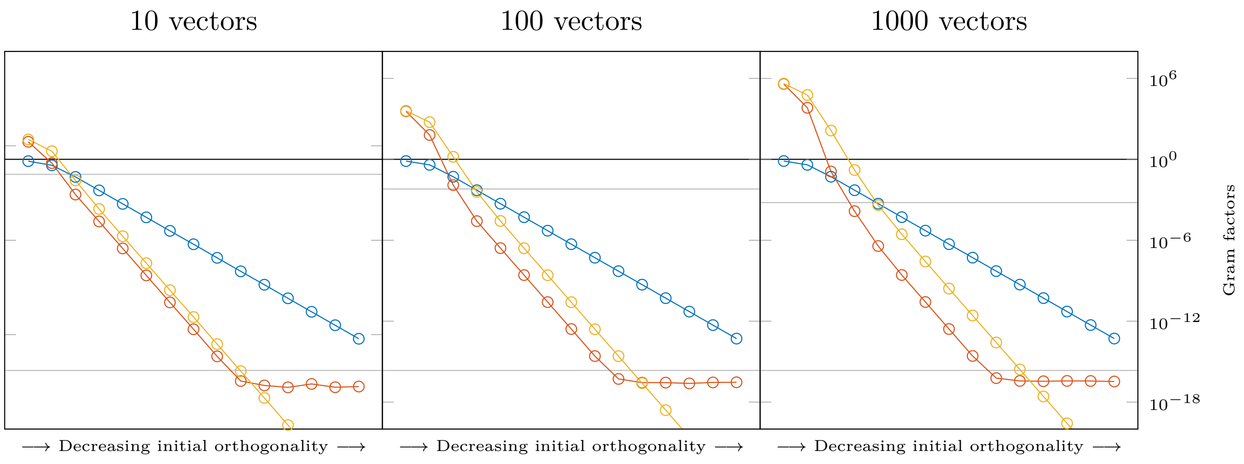


Figure 5.5: *Gram factor improvement of weak Gram-Schmidt. Shown are the initial orthogonality (blue), the predicted result (yellow), and the achieved result (red). Also shown are the orthogonality requirement a (upper gray line), machine epsilon (lower gray line), and one for reference (horizontal black line).*

To find the largest Gram factor that guarantees reaching machine precision (for the worst-case Gram factor) in one step, we can require

$$(m-2)a^2 + (m-2)^2 a^3 = \varepsilon$$

and solve for a, for example with an iterative root finding method. As before, this limit ensures reaching ε, but larger initial Gram factors may work as well. Figure 5.6 shows the values for up to 10 000 vectors. The general shape of the curve is identical to Figure 5.4, but the scaling on the y-axis is different.

Although improvement of Gram factors does not necessarily imply improvement of orthogonality, the above tacitly ignores the normalization that is required to obtain usable values for the orthogonality from the absolute values of the Gram factors. The squared norm of vector v_i, similar to before, is given by

$$\begin{aligned} v_i^H B v_i &= \left(z_i^H - \sum_{k=1}^{i-1} \overline{a_{k,i}}\, z_k^H\right) B \left(z_i - \sum_{k=1}^{i-1} a_{k,i}\, z_k\right) \\ &= 1 - \sum_{k=1}^{i-1} a_{k,i}\, \overline{a_{k,i}} - \sum_{k=1}^{i-1} \overline{a_{k,i}}\, a_{k,i} + \sum_{k=1}^{i-1} \overline{a_{k,i}}\, a_{k,i} + \sum_{k=1}^{i-1} \sum_{\substack{j=1 \\ j\neq k}}^{i-1} \overline{a_{k,i}}\, a_{j,i}\, a_{k,j} \\ &= 1 - \sum_{k=1}^{i-1} |a_{k,i}|^2 + \sum_{k=1}^{i-1} \sum_{\substack{j=1 \\ j\neq k}}^{i-1} a_{i,k}\, a_{j,i}\, a_{k,j}. \end{aligned} \tag{5.2}$$

It is clear that the norm approaches unity as the $a_{i,j}$ become smaller. By the definition of the scalar product from Section 1.1.1 and with B positive definite, Equation (5.2) must be larger than zero (assuming $v_i \neq \mathbf{0}$). The maximum deviation from unity

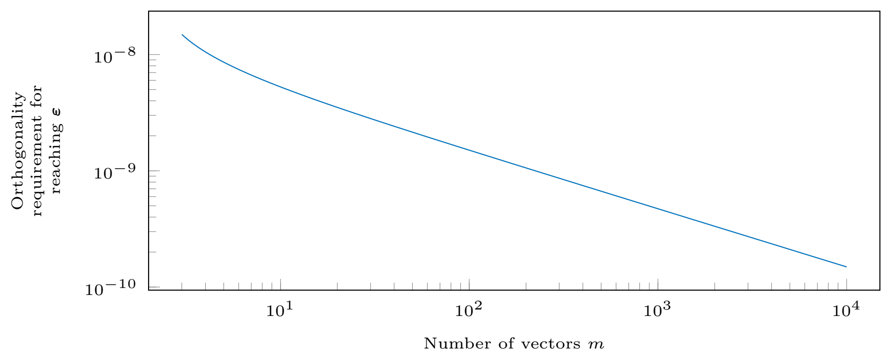


Figure 5.6: *Minimum required initial orthogonality a for weak Gram-Schmidt to reach machine precision as a function of the number of vectors $m \geq 3$.*

can be estimated by assuming no cancellation, again with $a \geq |a_{i,j}| > 0$, which leads to

$$\left|1 - \|v_i\|^2\right| < (i-1)a^2 + \left(i^2 - 3i + 2\right)a^3. \tag{5.3}$$

The expression on the right is unbounded. In lieu of a more sophisticated bound, we employ Equation (5.1) for i vectors as a hard limit for a. Then, the maximum deviation under these restrictions for i vectors is

$$\Upsilon(i) := (i-1)\frac{\left(-1+\sqrt{5}\right)^2}{4(i-2)^2} + \left(i^2 - 3i + 2\right)\frac{\left(-1+\sqrt{5}\right)^3}{8(i-2)^3}. \tag{5.4}$$

and $1 - \Upsilon(i)$ quickly approaches unity for an increasing number of vectors.

Figure 5.7 shows that accounting for the norm as in Equation (5.3) to obtain actual orthogonalities only influences the cases where the new maximum Gram factor would have been larger than one, besides making the estimation slightly more pessimistic. Accounting for the norm in terms of Equation (5.4) for the respective number of vectors (not shown) makes the estimation slightly more pessimistic in a similar way, but otherwise has no additional influence. Due to the norm approaching unity, if orthogonality is established via weak Gram-Schmidt and the normality of the input vectors is guaranteed, the method also produces normalized vectors through subsequent applications.

Convergence is faster the smaller the $a_{i,j}$ are. In particular, if the $a_{i,j}$ are reasonable small already, very few steps of weak Gram-Schmidt suffice to reach machine precision. Considering that methods like Cholesky QR often require multiple applications for stability reasons, weak Gram-Schmidt can be less expensive. The lion's share is again the initial inner product. Let

$$g_{i,j} = \frac{z_i^H B z_j}{\|z_i\|^2 \|z_j\|}.$$

Normalization of the input vectors Z and application of the Gram factors a_{ij} may

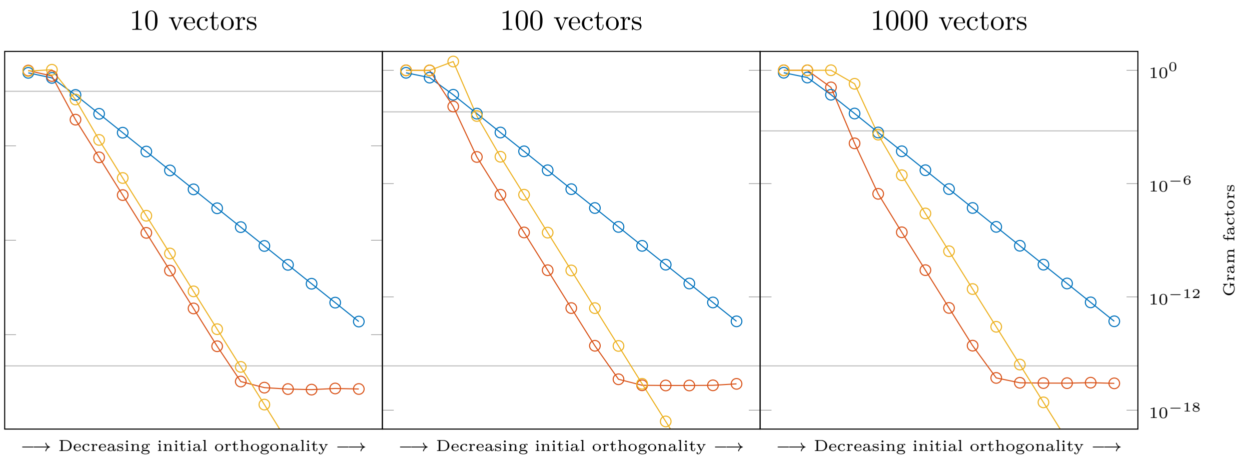


Figure 5.7: *Orthogonality improvement of weak Gram-Schmidt. Shown are the initial orthogonality (blue), the predicted result (yellow), and the achieved result (red). Also shown are the orthogonality requirement a (upper gray line) and machine epsilon (lower gray line).*

be combined in a single $m \times m$ matrix

$$G = \begin{pmatrix} \frac{1}{\|z_1\|} & -g_{1,2} & -g_{1,3} & \cdots & -g_{1,m} \\ & \frac{1}{\|z_2\|} & -g_{2,3} & \cdots & -g_{2,m} \\ & & \ddots & \ddots & \vdots \\ & & & \frac{1}{\|z_{m-1}\|} & -g_{m-1,m} \\ & & & & \frac{1}{\|z_m\|} \end{pmatrix},$$

whose right-multiplication onto Z does not require any additional communication if Z is distributed block-row-wise and Z_B is replicated on every process. Note that the norms $\|z_i\|$ are found on the diagonal of Z_B. Its application ZG may also be written as

$$S = (s_{i,j}) := D^{-\frac{1}{2}} Z_B D^{-\frac{1}{2}} = D^{-\frac{1}{2}} Z^H B Z D^{-\frac{1}{2}}$$

$$ZG = ZD^{-\frac{1}{2}} \begin{pmatrix} 1 & -s_{1,2} & -s_{1,3} & \cdots & -s_{1,m} \\ & 1 & -s_{2,3} & \cdots & -s_{2,m} \\ & & \ddots & \ddots & \vdots \\ & & & 1 & -s_{m-1,m} \\ & & & & 1 \end{pmatrix},$$

with $D = \operatorname{diag}(Z_B)$. The occurrences of $D^{-\frac{1}{2}}$ are the normalizations of all occurrences of the original vectors Z.

With the same approach as outlined in [SW06], we may formulate an upper bound on the orthogonality achieved by weak Gram-Schmidt when computations are performed with finite accuracy. Assume Z to be normalized and thus let $S = Z^H B Z$.

The floating point representation of S with error $\mathrm{e}(S)$ is written as

$$\mathtt{S} = S + \mathrm{e}(S),$$

with

$$\|\mathrm{e}(S)\| \le c_1\varepsilon\|S\| = c_1\varepsilon\lambda^{\max}(S) \le c_1\varepsilon\|B\|\|Z\|^2.$$

The ultimate goal is to formulate a bound for the orthogonality of the floating point representation $\mathtt{V}$ of the resulting vectors. We write $\mathtt{V} = V + \mathrm{e}(V)$ analogous to $\mathtt{S}$ and thus

$$\left\|I - \mathtt{V}^H B\mathtt{V}\right\| = \left\|I - V^H BV + \mathrm{e}(V)^H BV + V^H B\,\mathrm{e}(V) + \mathrm{e}(V)^H B\,\mathrm{e}(V)\right\|.$$

This is, so far, mostly identical to [SW06]. With $V = Z\mathtt{G}$, we end up in a situation similar to before—estimations of the norm of $\mathtt{G}$ involve orthogonalities and require us to again assume that all vectors exhibit some maximum orthogonality $\mathtt{a} = a + \mathrm{e}(a) \ge 0$ to produce a worst-case scenario. We can then find a rough upper bound for the largest eigenvalue of $\mathtt{G}$ by invoking Gershgorin's circle theorem on $\mathtt{G}^H\mathtt{G}$,

$$\|\mathtt{G}\| \le \sqrt{1 + (m-1)\mathtt{a} + \frac{1}{2}m(m-1)\mathtt{a}^2}$$

since $\|\mathtt{G}\| = \sqrt{\lambda^{\max}\left(\mathtt{G}^H\mathtt{G}\right)}$. In this case, it is also

$$\begin{aligned} \|S\| &\le 1 + (m-1)a \quad \text{and} \\ \left\|B^{\frac{1}{2}}Z\right\| &\le \sqrt{1 + (m-1)a}. \end{aligned}$$

We may also write

$$\begin{aligned} \left\|B^{\frac{1}{2}}V\right\| &\le \left\|B^{\frac{1}{2}}Z\right\|\|\mathtt{G}\| \\ \left\|B^{\frac{1}{2}}\,\mathrm{e}(V)\right\| &\le c_2\varepsilon\left\|B^{\frac{1}{2}}Z\right\|\|\mathtt{G}\|. \end{aligned}$$

We thus obtain

$$\begin{aligned} \left\|I - \mathtt{V}^H B\mathtt{V}\right\| &\le \left\|I - \mathtt{G}^H Z^H BZ\mathtt{G}\right\| + 2c_2\varepsilon\left\|B^{\frac{1}{2}}Z\right\|^2\|\mathtt{G}\|^2 + c_2^2\varepsilon^2\left\|B^{\frac{1}{2}}Z\right\|^2\|\mathtt{G}\|^2 \\ &= \left\|I - \mathtt{G}^H\mathtt{S}\mathtt{G} + \mathtt{G}\,\mathrm{e}(S)\mathtt{G}\right\| + 2c_2\varepsilon\left\|B^{\frac{1}{2}}Z\right\|^2\|\mathtt{G}\|^2 + c_2^2\varepsilon^2\left\|B^{\frac{1}{2}}Z\right\|^2\|\mathtt{G}\|^2 \\ &\le \left\|I - \mathtt{G}^H\mathtt{S}\mathtt{G}\right\| + c_1\varepsilon\|S\|\|\mathtt{G}\|^2 + \left(2c_2\varepsilon + c_2^2\varepsilon^2(1)\right)\left\|B^{\frac{1}{2}}Z\right\|^2\|\mathtt{G}\|^2. \end{aligned}$$

The norm of $I - \mathtt{G}^H\mathtt{S}\mathtt{G}$ can be obtained similarly to the norm of $\mathtt{G}$ (somewhat simplified due to $\mathtt{G}^H\mathtt{S}\mathtt{G}$ being Hermitian),

$$\left\|I - \mathtt{G}^H\mathtt{S}\mathtt{G}\right\| \le (m-1)^2\mathtt{a}^2 + \frac{1}{2}m(m-2)(m-1)\mathtt{a}^3.$$

In contrast to the analysis of a single worst-case Gram factor of $V^H BV$ before, it is not surprising that, when looking at the full norm, we catch an additional factor m in the bound, making this estimation more pessimistic. The worst off-diagonal entry of $\mathtt{G}^H\mathtt{S}\mathtt{G}$, however, can indeed again be bounded by $(m-2)\mathtt{a}^2 + (m-2)^2\mathtt{a}^3$.

Without writing down the somewhat unsightly fully resolved bound, it can already be seen that, assuming $a \to 0$ and $\mathtt{a} \to c_3\varepsilon$ (by repeated iterations of weak Gram-Schmidt; we have seen that there are conditions under which this cannot be achieved), the floating-point induced deviation from orthogonality approaches the order of machine precision (up to a constant, a dependency on m, and higher order terms). Dropping all terms with higher powers of ε, we may summarize the bound as

$$\left\| I - \mathtt{V}^H B \mathtt{V} \right\| \lesssim \mathcal{O}\left(m^2 a^2\right) + \mathcal{O}\left(m^3 a^3\right) + \varepsilon\left(\mathcal{O}(ma) + \mathcal{O}\left(m^2 a^2\right) + \mathcal{O}\left(m^3 a^3\right)\right).$$

Here, we assume that $\mathrm{e}(a) \le c_3\varepsilon a$ with $a > 0$. Powers of a are always paired with the respective powers of m; a linear term only appears factored with machine precision.

5.2.14.1 Computational effort

Quite a few of the methods introduced above require the computation of an inner product $Z^H BZ$ which requires $2m \cdot \mathrm{nnz}(B)$ operations (additions and multiplications) for the sparse matrix-vector multiplication and $2nm^2$ operations (additions and multiplications)[8] for the multiplication of Z^H and BZ. Since the result is Hermitian, it suffices to only compute the upper triangular part, reducing the cost for the dense multiplication to roughly nm^2 operations (nm operations more if normalization cannot be assumed, nm operations less otherwise).

For the construction of the weak Gram matrix G, at least $(m^2+m)/2$ divisions, m square roots, and $(m^2-m)/2$ multiplications[9] are required if the vectors of Z are not normalized. If the normality of Z can be assumed, these operations can be omitted and the matrix is built from the Gram factors alone with the diagonal entries being unity. In this case, no arithmetic operations are required to construct G, but its $(m^2+m)/2$ entries still have to be written to memory. The process of building G is trivially parallelizable and may be carried out by multiple threads.

What is left is the multiplication of Z with an upper triangular matrix, at a cost of $nm^2 + nm$ multiplications and additions ($nm^2 - nm$ if normalization can be assumed). Assuming $\mathrm{nnz}(B) \in \mathcal{O}(n)$, the overall asymptotic complexity is $\mathcal{O}(nm^2)$.

Weak Gram-Schmidt may require more than one iteration to reach the desired orthogonality. While other methods may require multiple iterations due to stability concerns, the cheap computation of weak Gram-Schmidt is eventually overshadowed by the number of iterations it may require. In this case it may still serve to improve a result obtained from, say, Cholesky QR in a cheap manner instead of repetitive execution of Cholesky QR itself.

[8] We ignore the different counts of multiplications and additions in the scalar products and assume their number to be equal for the sake of simplicity.

[9] If the square roots are stored separately. Otherwise the number of multiplications duplicates.

5.2.15 Residual

Intra-orthogonalization can affect the residual of the modified vectors. For weak Gram-Schmidt, for example, the residual for an orthogonalized vector q_i becomes

$$\|Aq_i - \lambda_i Bq_i\| \leq \|Ax_i - \lambda_i Bx_i\| + \sum_{k=1}^{i-1} a_{k,i} \, \|Ax_k - \lambda_i Bx_k\|.$$

The effect of normalization on the residual has been omitted, following the previous section. For other QR-type algorithms, the residual can be formulated in similar manner, given that R is a triangular matrix and can be applied via its inverse, which is again triangular. The results may differ for other algorithms. In order to determine which algorithms produces the least disturbance to the residual, we compare a selection of algorithms.

Experiment 5.1 — Influence of local reorthogonalization on the residual

We generate a matrix of size 5000 *with a distinct cluster around unity. In order to include methods that do not support B-orthogonalization, a standard eigenproblem is used. The internal cluster density is set to* 10^{-10}*. The complete spectrum is enclosed in* $[0, 2]$ *and the non-clustered parts are evenly distributed in the two outer regions. The* 160 *clustered eigenpairs are computed independently over* 8 *intervals,* $\mathcal{I}_1, \dots, \mathcal{I}_8$*. To produce a more difficult situation, the intervals were not allowed to converge to machine precision, but stopped after few iterations. Overall progress is slowed down by using a Cauchy filter of the Gauss-Legendre type with only four poles. The already tight clustering of the eigenpairs lowers the kick-off residual considerably and increasing the density further while targeting a comparatively large residual is neither possible nor necessary. The residual produced is of the order of* $5 \cdot 10^{-12}$ *for the boundary intervals and around* $5 \cdot 10^{-10}$ *to* $5 \cdot 10^{-9}$ *for inner intervals. While this seems to indicate that the computed eigenvalues cannot be properly separated, the values are well separated in practice.*

We choose two out of the eight intervals such that a single inter-orthogonalization operation evokes the maximum impact on intra-orthogonality and residual of the modified vectors in order to maximize the deviation applied by reestablishing intra-orthogonality. Here, this is the orthogonalization of $\mathcal{I}_1$ *against* $\mathcal{I}_2$*; consecutive orthogonalization against additional intervals adds little to the situation. Due to high spectral density and comparatively moderate target residual the effect is severe. The residual of* $\mathcal{I}_1$ *is raised to almost* $2 \cdot 10^{-9}$ *and intra-orthogonality is raised to about* 10^{-2}*, see Figure 5.8. Also shown in this figure are the results of reorthogonalization using several of the previously introduced methods (see Table 5.1 for an overview). Monitored are the worst-case, i.e., the maximum per-eigenpair residual and the worst-case, i.e., maximum orthogonality among any two eigenvectors of the modified interval* $\mathcal{I}_1$*.*

It is striking that none of the tested methods (with the exception of all methods based on the SVD) seem to impair the residual beyond what was caused by inter-orthogonalization. Repeating the experiment under different conditions substantiates

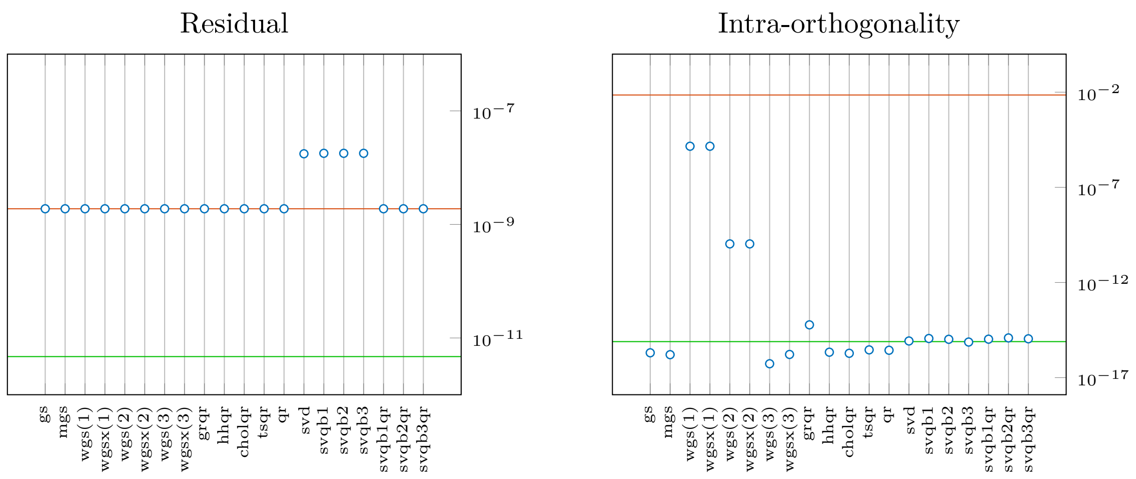


Figure 5.8: *The effect of local (intra-)reorthogonalization. Left: residual of the modified interval. Shown are the initial worst-case residual (green), the worst-case residual after inter-orthogonalization (red), and the worst-case residual after intra-orthogonalization (blue circles). Right: orthogonality of the modified interval. Shown are the initial orthogonality (green), the orthogonality after inter-orthogonalization (red), and the orthogonality after intra-orthogonalization (blue circles).*

Shortcut	Method
gs	Classical Gram-Schmidt
mgs	Modified Gram-Schmidt
wgs(1)	Weak Gram-Schmidt
wgsx(1)	Explicit implementation of weak Gram-Schmidt
wgs(2)	Weak Gram-Schmidt (two times)
wgsx(2)	Explicit implementation of weak Gram-Schmidt (two times)
wgs(3)	Weak Gram-Schmidt (three times)
wgsx(3)	Explicit implementation of weak Gram-Schmidt (three times)
grqr	Givens QR (unoptimized)
hhqr	Householder QR
cholqr	Cholesky QR
tsqr	Basic TSQR implementation without specialized triangular QR
qr	MATLAB QR [MLB]
svd	MATLAB SVD [MLB]
svqb1	First version of SVQB
svqb2	Second version of SVQB
svqb3	Third version of SVQB
svqb1qr	First version of SVQB with chained QR
svqb2qr	Second version of SVQB with chained QR
svqb3qr	Third version of SVQB with chained QR

Table 5.1: *Method name abbreviations. The* wgsx *versions are implementations of weak Gram-Schmidt analogous to classical Gram-Schmidt as opposed to the direct matrix construction method from Section 5.2.14. The* svqb* *versions are the three versions introduced in Section 5.2.9 in order of appearance.*

this result. In terms of orthogonality, the results thoroughly are what was to be expected. The somewhat weaker orthogonality achieved by Givens QR may have to be attributed to a slightly suboptimal testing implementation. Weak Gram-Schmidt delivers results within the predicted per-step improvement, resulting in a total of three steps being necessary in order to achieve sufficient orthogonality in this extreme situation.

The residual being seemingly invariant under local reorthogonalization is not quite what would be expected. Due to the high spectral density throughout the interval, we expect the residual of eigenvalues in combination with unassociated vectors to be comparably small and with the orthogonalities still well below unity, the product of both appears to be insignificant compared to the already diminished eigenpair residuals. Enforcing larger changes in residual involves less densely clustered intervals while narrowing the distance at the boundary. However, for a large change to occur during reorthogonalization, a large change during inter-orthogonalization is required and in all cases it seems that this initial disturbance is dominant over any additional local disturbance caused by reorthogonalization.

A reasonable explanation for the somewhat faster convergence of the boundary intervals in this specific situation can be given by analyzing the filter type and spectrum in and around the intervals. The spectral distribution clusters exactly the eigenpairs computed by the eight intervals. For a decaying filter, the additional values occupying the searchspace are likely to be closer to the interval in question if clustered values are located on both sides or the respective target sub interval. If clustered values are located only on one side, as is the case with the two boundary intervals, only values from the cluster side will occupy the searchspace, allowing more values from this side to be included compared to inner intervals, ultimately leading to larger convergence speeds since occupying values are farther away from the interval where the filter has decayed further. Non-clustered values are unlikely to be included in the searchspaces because they are located farther away and thus are, in general, subject to stronger dampening than clustered values.

An explanation for the unsuitability of SVD-based methods can be found in the structure of the matrix (implicitly) applied to Z in order to obtain Q. For QR-type methods, this is the inverse of R that, itself being a triangular matrix, merely scales the first vector of Z, leaving its direction untouched. Subsequent vectors are built as linear combinations of the current vector and previous vectors. In particular, all vectors are orthogonal to z_1. If the transformation is not triangular, this relation to the eigenvectors is lost.

5.2.16 Fitness for purpose

After examining the several methods for local reorthogonalization and their properties, not all methods are suited for this purpose and among those that are, we may choose the ones that are most efficiently computed. Criteria will be the ability to establish B-orthogonality, disturbance of the residual, potential for parallelization, stability (in regard to the expected deviation from orthogonality of the result) and computational cost. Table 5.2 summarizes the first four of these criteria for the orthogonalization methods introduced so far. Where possible, the bounds were

Method	B-orthogonality	Residual invariant	Comm. avoiding	stability
Gram-Schmidt	✓	✓	✗	$\frac{\xi}{1-\xi}$ with $\xi = \mathcal{O}\left(\varepsilon\kappa\left(B^{\frac{1}{2}}Z\right)^2\right)$ [Gir+05]
Modified Gram-Schmidt	✓	✓	✗	$\mathcal{O}\left(\varepsilon\kappa\left(B^{\frac{1}{2}}Z\right)\right)$ [Hig02]
Weak Gram-Schmidt	✓	✓	✓	See Section 5.2.14
Givens QR	✗	✓	✗	$\mathcal{O}(\varepsilon)$ [GV13]
Householder QR	✗	✓	✗	$\mathcal{O}(\varepsilon)$ [GV13]
Cholesky QR	✓	✓	✓	$\mathcal{O}\left(\varepsilon\kappa\left(B^{\frac{1}{2}}Z\right)^2\right)$ [Yam+15]
TSQR	✗	✓	✓	$\mathcal{O}(\varepsilon)$ [Dem+12]
SVD	✗	✗	✗	$\mathcal{O}(\varepsilon)^{\dagger}$
SVQB	✓	✗	✓	$\mathcal{O}\left(\varepsilon\kappa\left(B^{\frac{1}{2}}Z\right)^2\right)$ [SW06]
SVQB-QR	✓	✓	✓	$\mathcal{O}\left(\varepsilon\kappa\left(B^{\frac{1}{2}}Z\right)^2\right)^{\ddagger}$

Table 5.2: *Exclusion criteria and stability bounds for the orthogonalization methods introduced before.*

modified for B-orthogonality by substitution of Z. For Givens and Householder QR, the bound follows from all transformation matrices being orthogonal to machine precision [GV13] and the product of such matrices being precise to machine precision as well.

The overall cost of an algorithm is a combination of the cost of the pure arithmetical operations on one process and the communication-induced overhead. For the communication cost, two factors play a role. The first is surely the amount of data that has to be transferred. The second factor is the number of communication steps, i.e., the number of times the latency for establishing communication has to

$^{\dagger)}$ The computation of an SVD is inherently iterative. We can therefore assume that precision in the order of ε can be reached after a sufficient number of iterations.

$^{\ddagger)}$ Right-multiplication of (even a perfectly) orthogonal matrix will not improve the overall deviation from orthogonality. Trying to incorporate a QR-decomposition of H in the analysis of the SVQB [SW06] confirms this early on.

be included in the overall cost. As communication avoiding we here understand algorithms that, assuming a block-row-wise distribution, require few (or even a minimal number of) communications stages (which is often equivalent to synchronization points). Due to their $\log(p)$-level fan-in-style communication patterns, TSQR and all inner-product based methods require the same amount of messages to be sent ($p\log(p)$ in total over $\log(p)$ stages for an all-reduction with butterfly communication pattern to replicate the result on each process). A fan-out is not necessary in this case. In contrast, Gram-Schmidt, modified Gram-Schmidt, and Householder QR require $2n\log(p)$ messages [Dem+12].

Before further narrowing down the list of methods based on these criteria, we exclude some of the methods from the get-go. The SVD is computationally intensive and at the same time is the only method that does not satisfy any of the additional criteria from Table 5.2. Givens QR, while easier to parallelize, is computationally more expensive due to the larger number of transformations that have to be applied while having otherwise similar properties to Householder QR. Givens QR may be preferred if only very few elements need to be eliminated, but otherwise is generally inferior to Householder QR. Table 5.3 summarizes the sequential and parallel cost of the remaining methods. For the computational cost (in terms of the number of

Method	Sequential cost	Parallel cost (p processes)
Classical GS	$2nm^2$ [Hig02]	$\frac{2nm^2}{p}$ [Dem+12]
Modified GS	$2nm^2$ [Hig02]	$\frac{2nm^2}{p}$ [Dem+12]
Weak GS	$2nm^2 + m^2$	$\frac{2nm^2}{p} + m^2$
Householder QR	$4nm^2 - \frac{4}{3}m^3$ [Hig02]	$\frac{4nm^2}{p} - \frac{4m^3}{3p}$ [Dem+12]
Cholesky QR	$2nm^2 + \frac{1}{3}m^3$ [Dem+12]	$\frac{2nm^2}{p} + \frac{1}{3}m^3$ [Dem+12]
TSQR	—	$\frac{6nm^2}{p} + \frac{8}{3}m^3\log(p) - \frac{7}{3}m^3$ [Dem+12]
SVQB	$3nm^2 + \mathcal{O}\left(\frac{4}{3}m^3\right)$	$\frac{3nm^2}{p} + \mathcal{O}\left(\frac{4}{3}m^3\right)$
SVQB-QR	$3nm^2 + \mathcal{O}(4m^3)$	$\frac{3nm^2}{p} + \mathcal{O}(4m^3)$

Table 5.3: *Serial and parallel cost of orthogonalization algorithms (GS = Gram-Schmidt).*

floating point operations), $B = I$ was assumed to keep the comparison fair. The cost for the combined inner and outer product appearing in weak Gram-Schmidt, Cholesky QR, and SVQB is listed as $2nm^2$ to ensure compatibility with the numbers from the listed sources; the $\mathcal{O}(nm/p)$ and other lower order terms are omitted. The explicit construction of the Q factor, which is required for our use-case, is included where the algorithm does not inherently include it (TSQR, Householder QR).

The direct matrix formulation of weak Gram-Schmidt is slightly more expensive than an explicit Gram-Schmidt-style formulation, which of course requires the same amount of operations. For TSQR, a non-parallel application is simply a conventional QR decomposition; listing its sequential cost therefore makes no sense. The parallel Householder QR is Scalapack's PDGEQRF; for the application of the Q factor, we assume a similar application of the transformations which would require about the same amount of operations as the construction of the R factor. For SVQB, the eigendecomposition is accounted for with $\mathcal{O}(4m^3/3)$ operations, indicating a QR-type algorithm with few iterations. For SVQB-QR, additionally the QR decomposition and the application of the corresponding Q factor have to be accounted for. All numbers assume block-row-wise distribution.

For the most general problems, all the above reduces the list of suitable orthogonalization methods to weak Gram-Schmidt, Cholesky QR, and SVQB-QR. The pure SVQB has to be excluded since it disturbes the residual further. Of course, if B-orthogonality is not required, the list has to be extended to include TSQR. SVQB-QR is rather expensive and not more stable than Cholesky QR and is therefore not a favorable method.

Over the disadvantage of squared condition, inner-product-based methods have an additional benefit. The computation of the inner product directly reveals all intra-orthogonalities of the vectors in the block. This information can be used to bypass unnecessary orthogonalizations and is otherwise not directly available. It also allows to easily decide between, e.g., weak Gram-Schmidt and Cholesky QR since the inner product has to be performed for both methods.

5.3 Establishing inter-orthogonality

Given two sets of vectors Z and Y, establishing inter-orthogonality by removing directions of Y from all vectors in Z such that $Y^H V = \mathbf{0}$ can be achieved by

$$V = Z - Y\Big(Y^H BZ\Big),$$

under the condition that Y is orthogonal, $Y^H BY = D$, where D is the diagonal matrix of the norms of the vectors in Y. To see why Y is required to be orthogonal, we may think of the procedure as orthogonalizing every vector of Z separately against all vectors of Y. Directions removed by one step may be reintroduced by a following step if the vectors of Y are not orthogonal. We will use the term *orthogonality provider* (or just *provider* for brevity) for the vectors Y which remain unchanged during the process. Analogously, we will use the term *orthogonality recipient* (or just *recipient*) for the vectors Z which are modified.

An orthogonalization operation also has implications for the intra-orthogonality of V. If Z was orthogonal before, V will, in general, not be orthogonal anymore after applying the above. More precisely,

$$\begin{aligned} V^H BV &= \Big(Z - Y\Big(Y^H BZ\Big)\Big)^H B\Big(Z - Y\Big(Y^H BZ\Big)\Big) \\ &= Z^H BZ - 2\Big(Z^H BY\Big)\Big(Y^H BZ\Big) + \Big(Z^H BY\Big)Y^H BY\Big(Y^H BZ\Big), \end{aligned} \tag{5.5}$$

where we assume Z and Y to be normalized and $Z^H BY$ and $Y^H BZ$ describe the same orthogonality. Missing above is again the normalization of V, which may be written as VN_V, where N_V is the diagonal matrix holding the inverse norms of the columns of V. This amounts to left and right multiplication of the terms in Equation (5.5) with N_V. Since normalization is required for the computation of orthogonalities, we have to account for it when estimating the final orthogonality of V.

Let Z and Y have an equal number of columns m. In the worst case, i.e., the largest possible entry of the matrix in Equation (5.5), scaled by the square of the smallest possible norm for columns v_i of V, the orthogonality of the updated V is

$$\operatorname{orth}(V) \leq \frac{\operatorname{orth}(Z) + 3m \operatorname{orth}(Z,Y)^2 + (m-1)m \operatorname{orth}(Z,Y)^2 \operatorname{orth}(Y)}{\min_i \|v_i\|^2}$$

under the assumption that the largest absolute values in the Gram matrices align and do not cancel. The value in the denominator is the smallest possible entry on the diagonal of the matrix in Equation (5.5), which is not easily estimated. We also know that the diagonal entries of the involved matrices are real and positive.

If we further assume that Z and Y are orthonormal, the above simplifies to

$$\operatorname{orth}(V) \leq \frac{m \operatorname{orth}(Z,Y)^2}{1 - m \operatorname{orth}(Z,Y)^2},$$

but we have to assume that $m \operatorname{orth}(Z,Y)^2 < 1$.

Reestablishing intra-orthogonality in exact arithmetic does not influence the inter-orthogonality between V and Y as their respective subspaces are orthogonal and reorthogonalization of V happens only inside the subspace of V. For our purpose, two subspaces are orthogonal if any direction from one subspace is orthogonal to every direction from the other and vice versa. Inter-orthogonality may be affected in a numerical context if the condition of the orthogonality recipient is large [SW06].

Possible effects on the residuals of the modified vectors are akin to Section 5.2.15; here, every vector of Z is orthogonalized against every vector of Y with m_Y vectors in total. The formulation for a single vector v_i becomes

$$\|Av_i - \lambda_i Bv_i\| \leq \|Az_i - \lambda_i Bz_i\| + \sum_{k=1}^{m_Y} \langle y_k, z_i \rangle_B \|Ay_k - \lambda_i By_k\|.$$

5.3.1 Intermediate orthogonalization

Subspace iteration lends itself to intermediate orthogonalization phases, e.g., after each iteration, in order to maintain orthogonality among several intervals. The question arises, whether this approach is worthwhile, whether convergence is influenced, and whether orthogonality is maintained if neither of the participating vector blocks is well converged yet.

Experiment 5.2 — Intermediate orthogonalization

We generate a matrix pair of size 1000 *with evenly distributed spectrum apart from the innermost two eigenvalues that are placed at distances*

$$g \in \left\{10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}\right\}.$$

Two intervals spanning the left resp. right half of the spectral center of 20 *eigenpairs, i.e., bisecting the gap of size g, are co-iterated over* 40 *iterations with a searchspace size of* 15 *each. These runs of the algorithm are then repeated to include an orthogonalization of the right interval against the current state of the left interval at the start of every iteration. The orthogonalization process orthogonalizes the complete set of approximate vectors of the right interval, including directions that are not part of the target interval, against those vectors of the left interval that are (presumably) part of the target interval. This is the only sensible choice for the participating vectors since*

- *the left interval inevitably contains directions of the right interval that shall not be removed,*
- *only the orthogonalities among vectors of the target intervals are of interest, and*
- *directions have to be removed from the entire searchspace set of the right interval in order to be reliably eliminated.*

Whether or not directions from the left outside part of the left interval are removed from the right interval's searchspace does not play any role. Consequently, only the orthogonality among vectors that are part of the target intervals is measured for evaluation. Figure 5.9 compares the orthogonality development with and without intermediate orthogonalization.

It should be noted that in this experiment, both intervals progress at the approximately same speed, meaning that vector sets that are orthogonalized against each other have comparable residual ranges. In practice, progress between intervals may differ largely, even if the iterations are synchronized.

The plots of Figure 5.9 show that orthogonality, with or without intermediate orthogonalization, evolves gradually alongside the convergence progress and cannot be established and/or preserved immediately. Nonetheless, intermediate orthogonalization manages to improve the ultimately achieved worst-case orthogonality to comparable levels for all gap sizes. Improvement, however, stops as soon as the saturation residual is reached by all directions inside the target interval. The plots therefore reveal that orthogonalization using intermediate vectors can only ever improve orthogonality in relation to the currently available residual, i.e., convergence progress. Additionally, the final orthogonality is not as good as expected; it stagnates around values of order 10^{-13}. This is due to the order of operations since orthogonalization occurs at the start of each iteration, but not at the end. We therefore see the influence of the very last iteration on the overall orthogonality as a discrepancy to the expected orthogonality which should be closer to machine precision. It should be

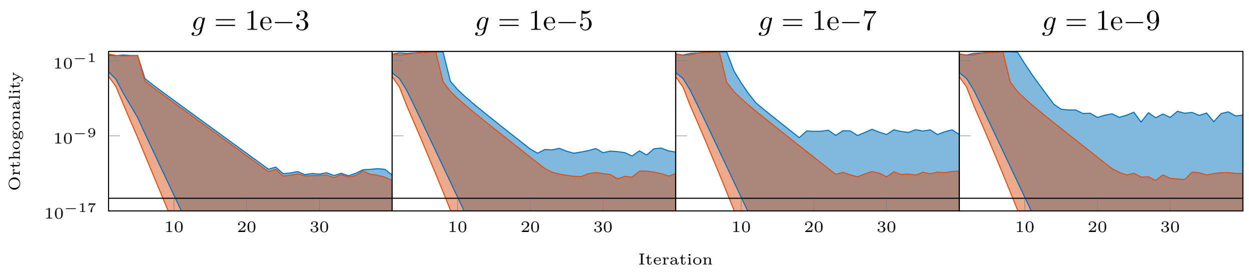


Figure 5.9: *Evolution of orthogonality without (blue) and with (red) intermediate orthogonalization for several gap sizes g over* 40 *iterations. The colored areas enclose the range of orthogonalities between vectors from the two intervals, indicating best and, more importantly, worst orthogonality between the two sets of ten eigenpairs. The black line indicates machine precision.*

emphasized that all eigenpairs have reached saturation at this point; further improvement by additional iterations is not possible. In order to lower the final orthogonality to the expected level, another orthogonalization step is necessary which must not be followed by an additional iteration. This begs the question of whether intermediate orthogonalization is necessary at all, if only the very last orthogonalization processes before stagnation is reached will establish reasonable levels of orthogonality and an additional orthogonalization step is required after termination of the iteration in all instances to reach machine precision.

If Experiment 5.2 is modified such that the left interval is completed first and its converged approximate eigenvectors can be used for orthogonalization, orthogonality can be established and preserved from the beginning (under continuous orthogonalization). Unsurprisingly, even then the accuracy of the final orthogonality is not better than the one achieved via intermediate vectors, given that each additional iteration introduces this particular divergence from orthogonality. A final orthogonalization step without additional iterations of the algorithm would again be required to reach higher levels of orthogonality.

It should be noted that orthogonalization does improve convergence speed (not shown here), in particular if the iterated eigenvectors are orthogonalized against the fully converged vectors of a neighboring interval. The reason for this behavior is directions being "banned" from occupying a slot in the searchspace in favor of different and in general farther away directions with smaller associated filter values. This is akin to reducing the whole eigenproblem by removing certain eigenpairs; they essentially appear not to exist for all that matters. An orthogonalization, i.e., the (approximate) removal of directions can be understood as additional dampening of these exact directions, as if the associated filter value would be very small. Both for ideal filtering and ideal orthogonalization, the affected directions would vanish from the searchspace. The convergence reference value $f(\lambda_{m_0+1})$ changes accordingly. This, of course, can only work well if the directions are removed properly from the orthogonality recipient. The benefit therefore diminishes for less well converged orthogonality providers. This means that with co-iterated intervals, accelerations will be negligible in early iterations and increase in later iterations.

The overall influence on convergence speed gives rise to the idea of artificially accelerating convergence by expunging directions close to the target interval by means of orthogonalizing against the available approximations of these directions in the current searchspace. To not immediately reduce the searchspace size, new random vectors would be appended to directions belonging to the inside of the target interval and the resulting new block would be orthogonalized against the approximate outside vectors. In practice, this approach does not work well for two reasons. First, the slots occupied by outside directions are not well converged for most iterations and are typically less advanced than slots occupied by inside directions. As a consequence, these directions are not removed to sufficient accuracy during orthogonalization. Waiting for these directions to converge further is pointless, if we have to expect the target directions to be (almost) converged at that point. Second, if the slots that serve as orthogonality provider are not saved, even assuming that they are accurate enough to remove the represented directions sufficiently, they will reappear since in the following iteration the less dominant directions now occupying the searchspace would serve as new orthogonality provider, allowing the previously expunged directions to reappear, which they will, considering that they are the more dominant directions. A strategy of this kind can only succeed if the directions that serve as orthogonality provider are known with utmost precision to remove directions permanently and even then is is not guaranteed for them to not reappear due to numerical fluctuations. All in all, while this approach works in principle, it is not practicable beyond the continuous orthogonalization against an already converged set of approximate eigenvectors.

If the two intervals converge at different speeds, such that orthogonalizations against less advanced vector occur, the acceleration effect may be lost.

Experiment 5.3 — Intermediate orthogonalization with uneven convergence

We generate a matrix pair of size 1000 *with evenly distributed random spectrum, but fix the separation of eigenpairs in the target intervals to* $g = 10^{-7}$. *We chose a problem where the left interval contains eigenpairs that converge slower than all eigenpairs in the right interval. Other parameters are adopted from Experiment* 5.2*. To exclude Ritz phantoms in initial iterations from affecting the orthogonalization, we begin continuous orthogonalization at the start of iteration* 10*. We now compare residuals of eigenpairs inside the right interval (the orthogonality recipient) with and without orthogonalization. To analyze the opposite conditions, we switch the role of the intervals by flipping the spectrum. The resulting plots of the residual ranges can be seen in Figure 5.10.*

The apparent stagnation in the first iterations of the right plot is due to a Ritz phantom that moves outside the interval after iteration seven. It indeed appears that skipping the orthogonalizations in the first few iterations has no impact on the result in later iterations; the iteration after the first orthogonalization produces a significant drop in the residuals, beyond what the searchspace modification would

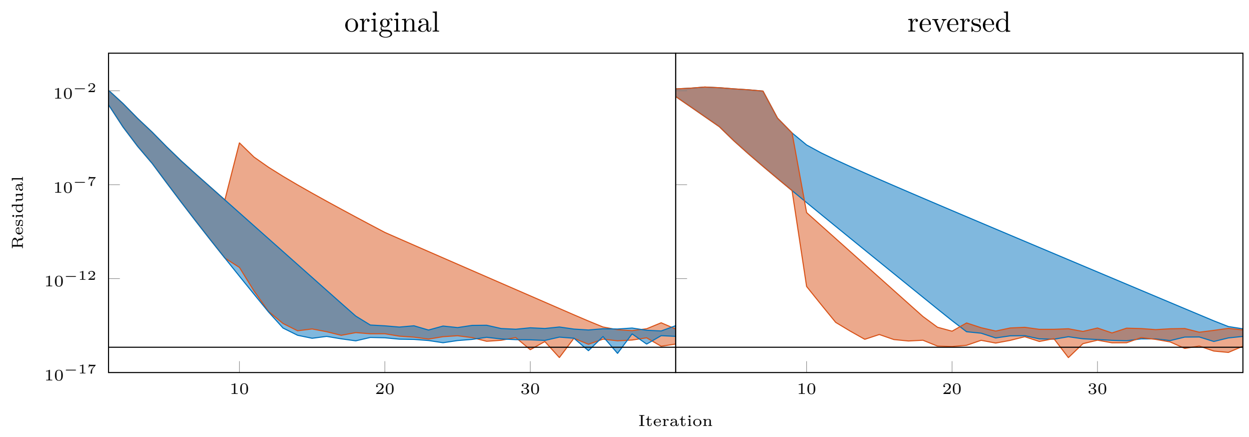


Figure 5.10: *Effects of orthogonalization between intervals with different progress. Shown is the evolution of the residual without (blue) and with (red) orthogonalization, beginning with the tenth iteration. Left: larger residual of orthogonality provider. Right: smaller residual of orthogonality provider.*

allow. We also see that reversing the direction of orthogonalization, i.e., reversing the roles of orthogonality provider and orthogonality recipient, has detrimental effects on the convergence of the modified searchspace.

A final question regarding continuous intermediate orthogonalization that shall be answered by an experiment is whether or not a single orthogonalization at the end of the iterative process will induce a more severe disturbance in the residual than continuous orthogonalizations.

Experiment 5.4 — Residual disturbances of continuous orthogonalizations

Under the same conditions as in Experiment 5.3, the algorithm is stopped after 13 *iterations and an additional orthogonalization of the final approximate eigenpairs is performed. When orthogonalizing the more advanced interval against the less advanced interval, in the case of continuous orthogonalization the worst-case residual grew from* $2.783 \cdot 10^{-7}$ *to* $1.055526655 \cdot 10^{-6}$*. With only a single orthogonalization at the very end, the worst-case residual grew from* $2.743 \cdot 10^{-11}$ *to* $1.055526659 \cdot 10^{-6}$*. The similarity of the final residual in both cases is remarkable. Interestingly enough, repeating the experiment with other spectra and gap sizes confirms this result. Reversing the roles of the interval, however, does not replicate this result if convergence is improved. In the example from above, in the case of continuous orthogonalization the worst-case residual remained at* $7.905 \cdot 10^{-9}$ *with only very little difference. With only a single orthogonalization at the very end, the worst-case residual improved slightly from* $9.266 \cdot 10^{-7}$ *to* $9.182 \cdot 10^{-7}$*, missing the residual achieved under continuous orthogonalization by a non negligible margin.*

The difference in behavior under orthogonalization between inhibitory and beneficially influence may be explained by the general nature of the process. If vectors serve as orthogonality provider that are more advanced than those of the orthogonality recipient, orthogonalization has the effect of dampening directions in favor of new directions with lower dominance. If the orthogonality providers are less advanced, an orthogonalization has merely a disturbing effect without the potential of dampening directions much. Whether this disturbance is added gradually or en bloc does not matter. On the other hand, it is not possible to catch up to the continuous accelerated convergence in the opposite case.

In summary, should the orthogonalization process increase the residual, whether or not orthogonalization was applied continuously does not change the final residual obtained after an a posteriori orthogonalization. However, an acceleration of convergence is only possible with continuous intermediate orthogonalization, performed in the right order.

5.3.2 Post-iteration and retention of orthogonality

In the previous section we have seen that subspace iteration can only retain orthogonality of its searchspace vectors relative to how well neighboring vectors could be removed from it in the first place, with the residual being the indicator of this accuracy. Considering the cost of interrupting the iteration to orthogonalize and the orthogonalization process of many intervals in general, it quickly becomes clear that orthogonalization before every iteration would be too expensive. In particular, if we can foresee that orthogonalization in early iterations only establishes subpar orthogonalities. We therefore consider orthogonalization only after selected iterations, in particular—since rank reduction and the inevitable reduction of the searchspace set will have an impact on convergence due to the many directions shared among the intervals—the last two iterations, in the presence of some convergence criterion.

When orthogonalizing two intervals in between two iterations, an important question is which vectors to orthogonalize. It may be the full basis or only the vectors inside the target interval. We have to remember that both intervals contain components of the neighboring intervals due to the nature of the filter, in particular in the case of decaying filters. Thus, an orthogonalization of only some vectors is ineffective (see also Section 5.3.3) while orthogonalization of all vectors will immediately lead to rank deficiency in the next iteration. From Section 4.2.2 we also know that this kind of elimination of directions will require consecutive orthogonalization for all following iterations, if the orthogonality is to be retained. In more extreme cases, a complex orthogonalization process may significantly diminish the previously established residual, which may require additional iterations of the eigensolver to reestablish the residual. Two questions arise.

- Will the orthogonalized vectors still be good approximations of the eigenvectors, such that convergence is accelerated and very few iterations are needed?
- Will the established orthogonality be retained during post-iteration?

Experiment 5.5 — Single orthogonalization

We repeat Experiment 5.2, but only perform one single orthogonalization at the beginning of iteration 24, which is the first iteration where all target eigenpairs have reached saturation. Figure 5.11 outlines the results.

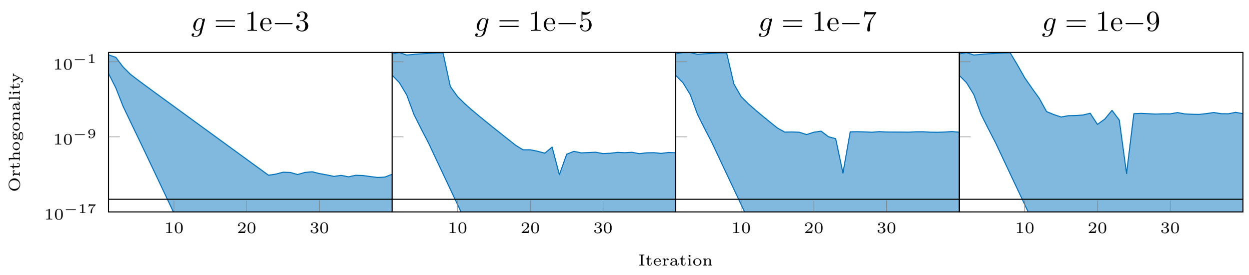


Figure 5.11: *Evolution of orthogonality with orthogonalization before iteration* 24 *for several gap sizes* g *over* 40 *iterations. The black line indicates machine precision.*

The orthogonality after iteration 24 is again of the same order as was the case with continuous orthogonalization in that iteration, which was to be expected. Most notably, the orthogonality cannot be preserved for the following iterations without continuous orthogonalization efforts. Due to the choice of iteration number, the inevitable rank reduction of the searchspace set due to removed directions does not cause much disturbance. If the orthogonalization is performed earlier with less advanced orthogonality providers, the effects on convergence can be chaotic since the directions have to reorganize. Some more dominant directions may have been reduced by the orthogonalization process, such that the associated columns in the basis realign to other directions. Due to deviations in direction, rank reduction may not trigger. While the acceleration of unaffected directions can be observed, new Ritz phantoms may be introduced and convergence of affected directions may be severely hindered. This is not the case or significantly alleviated if orthogonalization is continued in the following iterations or well converged orthogonality providers were used in the first place. If rank reduction is triggered, the number of iterated directions is reduced, affecting the convergence speed of all remaining directions negatively due to the change of the convergence reference value $f(\lambda_{m_0+1})$. In summary, we can confirm the following.

- Identical orders of orthogonality can be reached by a single orthogonalization in absence of previous intermediate orthogonalizations.
- Orthogonality can, in general, not be retained over multiple iterations (there are exceptions if the orthogonality providers and their surrounding searchspace are very well converged).
- In order to reach levels of orthogonality in the order of machine precision, a final orthogonalization step without any additional iteration of the algorithm thereafter is inevitable.

In many practical cases, the need for post-iteration is essentially eliminated. This makes the exploration of a possible purely a posteriori orthogonalization method worthwhile.

5.3.3 Overlapping intervals

Given that the spectral filtering methods described here allow for the computation of eigenpairs over multiple different intervals, the idea of improving inter-orthogonality without explicit communication among the intervals by letting adjacent intervals overlap is not too far fetched at first glance. This, however, is not different in principle from computing two non-overlapping intervals whose searchspaces include directions from the respective neighboring interval. Of course, convergence speed of these directions differ. In the previous chapter, we have seen that under these conditions orthogonality suffers.

From a different standpoint, seeing that the resulting eigenvectors after subspace iteration are themselves intra-orthogonal and inter-orthogonality is still based on the smallest gap between any two computed eigenpairs from different instances of the algorithm (and directional overlap of the searchspaces), we instead have to expect inter-orthogonality to deteriorate in this case.

Overlapping intervals, however, play an important role during orthogonalization after all. Seeing that the uncertainty of the position of an eigenvalue in relation to its associated Ritz value, which is bound by the residual, allows a Ritz value to be located outside the target interval, while its associated eigenvalue is located inside the target interval if the eigenvalue is very close to the interval boundaries, interval overlap becomes an adequate countermeasure, see Section 4.4. This overlap may be chosen individually for each Ritz value, based on the respective residual, essentially requiring the Ritz-disk around the Ritz value to intersect the target interval in any way to include the Ritz value in the accepted set. Interval overlap also invites Ritz pairs of the same eigenpair being computed redundantly in two neighboring intervals. These duplicates may be discerned from unique Ritz pairs by the angle between their eigenvectors, which can be assumed to be comparatively close to zero. On the other hand, assuming that the orthogonality of independently computed intervals can become arbitrarily bad, a clear distinction between duplicates and Ritz values with very little separation may be difficult. The secondary job of the cross-interval orthogonalization thus will be the identification and removal of such duplicates. In Section 5.3.9 we will deal with this particular problem.

5.3.4 Orthogonalization sequences

Performing multiple orthogonalization steps in sequence following Section 1.3.1 has several implications on overall orthogonality. First of all, if a vector z is to be orthogonalized against a vector y in this manner, y must be of unit length. If z is to be orthogonalized against two vectors y_1 and y_2, such that z will be orthogonal to both vectors afterward, y_1 and y_2 have to be orthogonal to begin with. Otherwise, the orthogonalization of z against y_2 will reintroduce directions it shares with y_1 in their common vector space. This leads to the Gram-Schmidt methods of orthogonalization which have been introduced in Section 1.3.2.

In case of Gram-Schmidt style propagation of orthogonality, sequences of blocks $Z_1, \ldots, Z_p$ may be orthogonalized, if the blocks themselves are intra-orthogonalized before some other block is inter-orthogonalized against them. This is required for the same reason as Y needed to be orthogonal in the introductory section. Successive inter-orthogonalizations against blocks which are themselves not inter-orthogonal will (in general) not establish overall orthogonality.

5.3.5 Block Gram-Schmidt

The idea behind Gram-Schmidt or modified Gram-Schmidt is easily applied to blocks of vectors. Additionally, reorthogonalization of a block is required whenever it is to be used for orthogonalization against another block. Algorithm 5.1 outlines the algorithm based on the reordered version of single vector modified Gram-Schmidt, Algorithm 1.3, *without* assuming the vector blocks to be intra-orthogonal and normalized on entry.

```
Input: Vector blocks $Z_1, \ldots, Z_p$
Output: B-orthonormal vector blocks $Z_1, \ldots, Z_p$
1: for $i = 1, \ldots, p$ do
2:     $Z_i \leftarrow$ B-orthonormalize $Z_i$
3:     for $j = i + 1, \ldots, p$ do
4:         $Z_j \leftarrow Z_j - Z_i\left(Z_i^H B Z_j\right)$
```

Algorithm 5.1: *Block modified Gram-Schmidt orthonormalization, reordered.*

5.3.5.1 Parallel application

Conventional application of the modified Gram-Schmidt scheme in a parallel computation context involves broadcasting of vector blocks, similar to the implementation of the single-vector version; the freshly finalized source vector block is broadcast to the processes holding the remaining vector blocks, whereafter the orthogonalization of these blocks against the source block can be performed in parallel. This finalizes the next vector block, which has to be reorthogonalized before it can be broadcast, repeating the cycle. Assuming the vector blocks are held in disjoint groups of processes, groups that hold finalized vectors do not participate in the remainder of the algorithm. This typical starvation of lower-index process groups cannot be avoided.

An alternative implementation makes use of a shifting strategy. The first (intra-orthogonal) vector block is transferred (shifted) to the neighboring process group holding the second vector block. This block can now be orthogonalized while the previously received first vector block is passed to the next process group, which holds the third vector block. If the hardware allows it, orthogonalization of the second block can be performed concurrently with the transfer of the first block. After receiving the first block on the third group, orthogonalization of the third block against the first, sending the first block, and receiving the second block can be performed concurrently. The next step for group three then involves orthogonalization of the third block against the second block and sending the second block before, in a last step, the now finalized third block is passed to the neighboring process group. Proceeding in this manner completes the overall orthogonalization process in $2p - 2$ steps for p vector blocks. For the first vector block to reach the last group, $p - 1$ shifts are required; then $p-2$ additional blocks arrive one after the other. The final step on the last group then includes the orthogonalization of the last block against the previous block and the final reorthogonalization of the last block. While this strategy avoids broadcast operations, significant idle times are caused on the process groups. The first group in particular only needs to send its block once. Similarly the last group has to wait $p - 1$ steps for the first block to arrive.

Figure 5.12 gives an overview over the several steps involved for four concurrent processes, each holding a vector block. The reorthogonalization of the locally main-

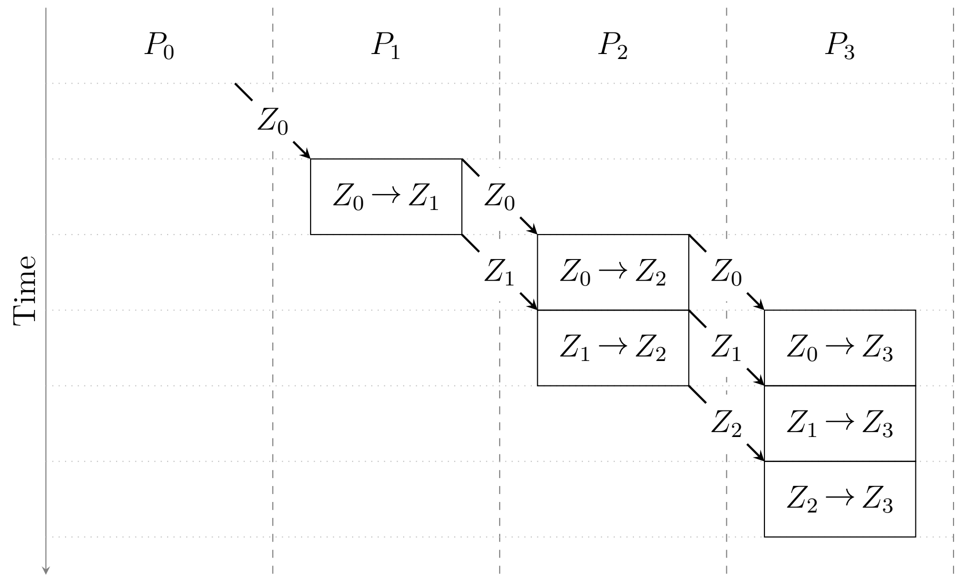


Figure 5.12: *Block Gram-Schmidt shift scheme. Although not explicitly indicated, sending of a vector block implies that it has been reorthogonalized before. If we assume that the vector blocks are orthogonal at the beginning, Z_0 needs no orthogonalization. Every additional process adds two phases to the scheme.*

tained vectors has to occur before the local block is sent as orthogonality provider. It is implicitly included in the last orthogonalization phase on each process to keep the diagram less convoluted. This has no effect on the possible overlap of communication and computation phases as shown in the picture, other than the length of the affected phases, which cannot be expected to be as homogeneous as depicted anyway.

Compared to broadcast-based communication (see Figure 5.13), the same number of phases is required in the shift scheme—under the slightly optimistic assumption that the broadcast can be accomplished as a single phase—since the broadcast has to finish before any computation can take place. Of course, any benefit vanishes if communication and computation cannot take place at the same time.

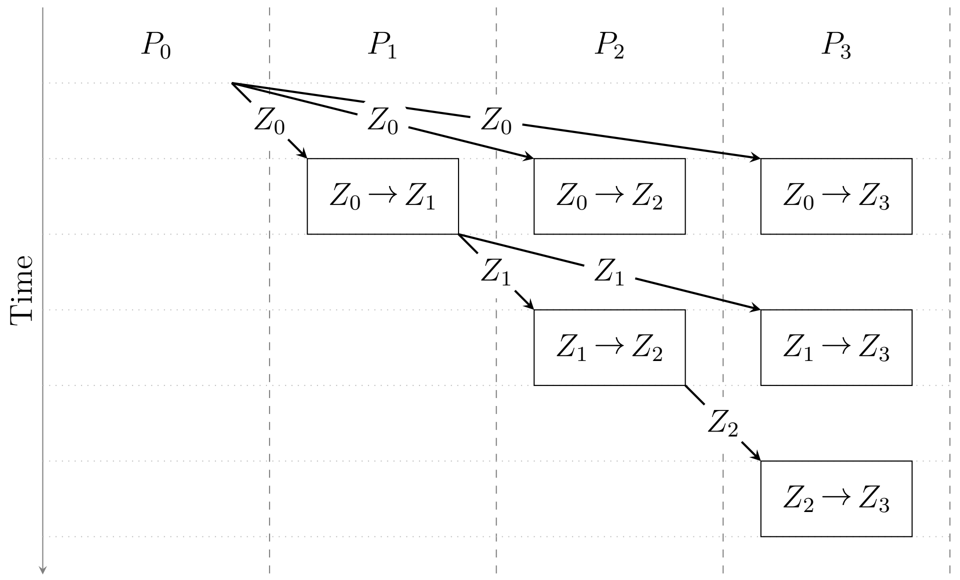


Figure 5.13: *Block Gram-Schmidt broadcast scheme. Although not explicitly indicated, broadcasting a vector block implies that it has been reorthogonalized before. If we assume that the vector blocks are orthogonal at the beginning, Z_0 needs no orthogonalization. Every additional process adds two phases to the scheme.*

5.3.6 Residual

If an orthogonalization operation is applied to a block of approximate eigenvectors $\boldsymbol{X}$ such that $V = \boldsymbol{X} - Y\left(Y^H B\boldsymbol{X}\right)$, the residual of V can be written

$$\begin{aligned}\|AV - BV\boldsymbol{\Lambda}\| &= \left\|(A\boldsymbol{X} - B\boldsymbol{X}\boldsymbol{\Lambda}) - \left(AY\left(Y^H B\boldsymbol{X}\right) - BY\left(Y^H B\boldsymbol{X}\right)\boldsymbol{\Lambda}\right)\right\| \\ &\leq \|(A\boldsymbol{X} - B\boldsymbol{X}\boldsymbol{\Lambda})\| + \left\|\left(AY\left(Y^H B\boldsymbol{X}\right) - BY\left(Y^H B\boldsymbol{X}\right)\boldsymbol{\Lambda}\right)\right\|,\end{aligned}$$

such that the disturbances in residual of the orthogonalized vectors are related to the residuals of the vectors they have been orthogonalized against and the original orthogonality between $\boldsymbol{X}$ and Y. Certainly, the disturbance becomes smaller the more orthogonal $\boldsymbol{X}$ and Y are. For separate vectors $\boldsymbol{x}_i$ and y_j, assuming vector-wise orthogonalization, we may write

$$\|Av_i - \boldsymbol{\lambda}_i Bv_i\| \leq \|A\boldsymbol{x}_i - \boldsymbol{\lambda}_i B\boldsymbol{x}_i\| + \operatorname{orth}(\boldsymbol{x}_i, y_j)\|Ay_j - \boldsymbol{\lambda}_i By_j\|$$

where $\|Ay_j - \boldsymbol{\lambda}_i By_j\|$ has to be considered large since $\boldsymbol{\lambda}_i$ is in general not near the eigenvalue associated with y_j. If it is, however, this term may become smaller, but orthogonality is likely to deteriorate. By successively applying the above for all vectors y_i, we arrive, as before, at

$$\|Av_i - \boldsymbol{\lambda}_i B v_i\| \leq \|A\boldsymbol{x}_i - \boldsymbol{\lambda}_i B\boldsymbol{x}_i\| + \sum_{j=1}^{m} \operatorname{orth}(\boldsymbol{x}_i, y_j)\|Ay_j - \boldsymbol{\lambda}_i B y_j\|$$

if Y has m columns. We got a first taste of the possible disturbances of the residual caused by inter-orthogonalization in Experiment 5.1.

If now more than two blocks of vectors are involved, say $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_p$, each consisting of vectors $\boldsymbol{x}_{i,\ell}$ for $\ell = 1, \ldots, p$, the deviations accumulate accordingly. Let the processed blocks be $V_1, \ldots, V_p$ with vectors $v_{i,\ell}$ and $\boldsymbol{\lambda}_{i,k}$ the respective computed approximate eigenvalue associated with block k, vector i. Assuming a number of m_k vectors for block k,

$$\begin{aligned}\|Av_{i,k} - \boldsymbol{\lambda}_{i,k} B v_{i,k}\| \leq{}& \|A\boldsymbol{x}_{i,k} - \boldsymbol{\lambda}_{i,k} B\boldsymbol{x}_{i,k}\| \\ &+ \sum_{\ell=1}^{k-1}\sum_{j=1}^{m_k} \operatorname{orth}(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,\ell})\|A\boldsymbol{x}_{j,\ell} - \boldsymbol{\lambda}_{i,k} B\boldsymbol{x}_{j,\ell}\|.\end{aligned}$$

The effect of the orthogonalization operations is distributed unevenly over the p blocks. The last block in the sequence will be exposed to more orthogonalizations which disturb its intra-orthogonality and residual.

5.3.7 Orthogonalization order

Oftentimes the order in which vectors or vector blocks are processed—and accordingly the role of orthogonality provider and orthogonality recipient of single orthogonalization operations—is not specifically fixed and may be chosen more or less freely. Additionally, the question of which vector or block should be orthogonalized against which other vectors or blocks may influence the results.

Consider two computed approximations $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to eigenvectors x_1 and x_2. From Section 4.5.1 we know that smaller residuals ensure smaller angles between eigenvectors and their computed approximations. We now assume that, in general, the reverse is true and larger residuals indicate larger angles. In this case, the decision of which vector to orthogonalize against the other becomes important. Figure 5.14 visualizes the two possible outcomes. In this example the angle between $\boldsymbol{x}_1$ and x_1 is smaller than the angle between $\boldsymbol{x}_2$ and x_2. We assume that the respective residuals are indicative of this situation. When orthogonalizing $\boldsymbol{x}_1$ against $\boldsymbol{x}_2$, yielding $\boldsymbol{x}_1'$, the angle between $\boldsymbol{x}_1'$ and $\boldsymbol{x}_1$ is now larger, potentially allowing the new residual to be larger than before. Orthogonalizing $\boldsymbol{x}_2$ against $\boldsymbol{x}_1$ on the other hand, reduces the angle between $\boldsymbol{x}_2'$ and $\boldsymbol{x}_2$, potentially allowing the new residual to be smaller than before. We have to conclude—although the unidirectional character of the relation between residual and angle does not allow for a more substantial statement—that the most reasonable course of action is to prefer orthogonalization of vectors with larger residual against vectors with smaller residual in order to minimize the chance of increasing the residuals involved.

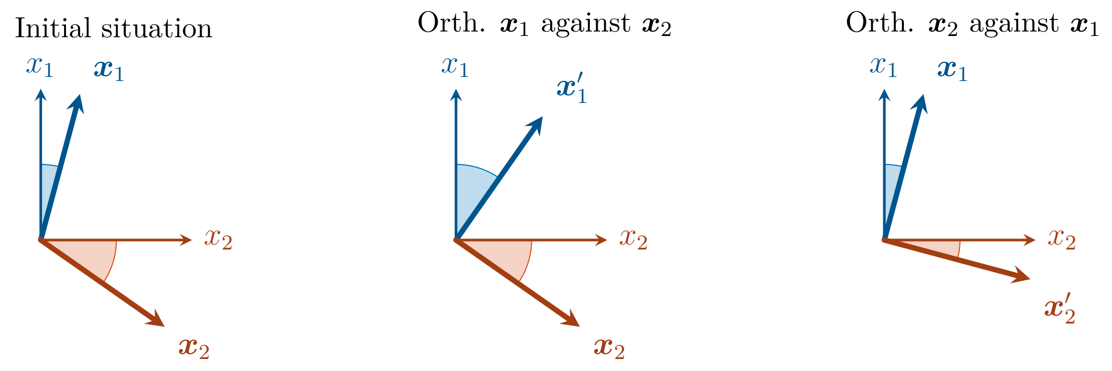

Initial situation
Orth. $\boldsymbol{x}_1$ against $\boldsymbol{x}_2$
Orth. $\boldsymbol{x}_2$ against $\boldsymbol{x}_1$
x_1
$\boldsymbol{x}_1$
x_2
$\boldsymbol{x}_2$
$\boldsymbol{x}_1'$
$\boldsymbol{x}_2'$

Figure 5.14: *The effect of orthogonalization on eigenvector angles.*

A more formal reasoning can be applied *(Suggested, B. Lang, personal communication, 20.4.2020)* by considering the decompositions of two approximate eigenvectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ into a linear combination of exact eigenvectors $x_1, \ldots, x_n$,

$$\boldsymbol{x}_i = x_i + \sum_{k=1}^{n} \alpha_k x_k \quad \text{and} \quad \boldsymbol{x}_j = x_j + \sum_{k=1}^{n} \beta_k x_k,$$

where $|\alpha_k| \ll 1$ and $|\beta_k| \ll 1$. For ease of notation, let $\boldsymbol{r}_i(j) = Ax_j - \boldsymbol{\lambda}_i Bx_j$ be the residual vector of a Ritz value $\boldsymbol{\lambda}_i$ and an exact eigenvector x_j. Let further $(\boldsymbol{\lambda}_i, \boldsymbol{x}_i)$ be the Ritz pair associated with x_i. Considering the eigenvectors normalized, the scalar product of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is

$$\begin{aligned}
\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle_B &= \left(x_j^H + \sum_{k=1}^{n} \overline{\beta_k} x_k^H \right) B \left(x_i + \sum_{k=1}^{n} \alpha_k x_k \right) \\
&= x_j^H B x_i + \sum_{k=1}^{n} \alpha_k x_j^H B x_k + \sum_{k=1}^{n} \overline{\beta_k} x_k^H B x_i + \sum_{k=1}^{n} \sum_{\ell=1}^{n} \alpha_\ell \overline{\beta_k} x_k^H B x_\ell \\
&= \alpha_j + \overline{\beta_i} + \underbrace{\sum_{k=1}^{n} \alpha_k \overline{\beta_k}}_{=:c} .
\end{aligned}$$

The residual of $\boldsymbol{x}_i$ w.r.t. $\boldsymbol{\lambda}_i$ is

$$\|A\boldsymbol{x}_i - \boldsymbol{\lambda}_i B \boldsymbol{x}_i\| = \left\| \boldsymbol{r}_i(i) + \sum_{k=1}^{n} \alpha_k \boldsymbol{r}_i(k) \right\| \leq \left\| \boldsymbol{r}_i(i) + \sum_{\substack{k=1 \\ k \neq j}}^{n} \alpha_k \boldsymbol{r}_i(k) \right\| + \|\alpha_j \boldsymbol{r}_i(j)\|.$$

The residual of

$$\begin{aligned}
\boldsymbol{x}_i' &= \boldsymbol{x}_i - \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle_B \boldsymbol{x}_j \\
&= x_i + \sum_{k=1}^{n} \alpha_k x_k - \alpha_j x_j - \sum_{k=1}^{n} \alpha_j \beta_k x_k - \overline{\beta_i} x_j - \sum_{k=1}^{n} \overline{\beta_i} \beta_k x_k - c x_j - \sum_{k=1}^{n} c \beta_k x_k
\end{aligned}$$

w.r.t. $\boldsymbol{\lambda}_i$ is

$$\begin{aligned}\|A\boldsymbol{x}_i' - \boldsymbol{\lambda}_i B\boldsymbol{x}_i'\| &= \left\|A\boldsymbol{x}_i - \boldsymbol{\lambda}_i B\boldsymbol{x}_i - \left(\alpha_j + \overline{\beta_i} + c\right)(A\boldsymbol{x}_k - \boldsymbol{\lambda}_i B\boldsymbol{x}_k)\right\| \\ &= \left\|\boldsymbol{r}_i(i) + \sum_{k=1}^{n} \alpha_k \boldsymbol{r}_i(k) - \alpha_j \boldsymbol{r}_i(j) - \overline{\beta_i}\boldsymbol{r}_i(j) - C\right\| \\ &= \left\|\boldsymbol{r}_i(i) + \sum_{\substack{k=1\\k\neq j}}^{n} \alpha_k \boldsymbol{r}_i(k) - \overline{\beta_i}\boldsymbol{r}_i(j) - C\right\| \\ &\leq \left\|\boldsymbol{r}_i(i) + \sum_{\substack{k=1\\k\neq j}}^{n} \alpha_k \boldsymbol{r}_i(k)\right\| + \left\|\overline{\beta_i}\boldsymbol{r}_i(j)\right\| + \|C\|,\end{aligned}$$

where

$$C = \sum_{k=1}^{n} \alpha_j \beta_k \boldsymbol{r}_i(k) + \sum_{k=1}^{n} \overline{\beta_i}\beta_k \boldsymbol{r}_i(k) + c\boldsymbol{r}_i(j) + \sum_{k=1}^{n} c\beta_k \boldsymbol{r}_i(k).$$

If we now assume that C is negligible and generally $|\beta_k| < |\alpha_k|$ if the residual of $\boldsymbol{x_j}$ is better than the residual of $\boldsymbol{x_i}$ and $|\beta_k| > |\alpha_k|$ if the residual of $\boldsymbol{x_j}$ is worse than the residual of $\boldsymbol{x_i}$, the bound for the residual of $\boldsymbol{x}_i'$ becomes smaller or larger than the bound on the residual of $\boldsymbol{x_i}$, respectively. The assumption for the coefficients α_k and β_k is not unreasonable since, e.g.,

$$\|A\boldsymbol{x}_i - \boldsymbol{\lambda}_i B\boldsymbol{x}_i\| \leq \|\boldsymbol{r}_i(i)\| + \sum_{k=1}^{n} \|\alpha_k \boldsymbol{r}_i(k)\| \leq \|\boldsymbol{r}_i(i)\| + \sum_{k=1}^{n} |\alpha_k| \|\boldsymbol{r}_i(k)\|.$$

If we further assume that $\boldsymbol{\lambda}_i \approx \lambda_i$, $\alpha_k \approx \alpha\,\forall k$, and $\beta_k \approx \beta\,\forall k$, the dependency becomes clearer. Let $r_i(j) = Ax_j - \lambda_i Bx_j$. Then

$$\|A\boldsymbol{x}_i - \lambda_i B\boldsymbol{x}_i\| = \left\|r_i(i) + \sum_{k=1}^{n} \alpha r_i(k)\right\| = \left\|\alpha \sum_{\substack{k=1\\k\neq i}}^{n} r_i(k)\right\| \leq |\alpha| \sum_{\substack{k=1\\k\neq i}}^{n} \|r_i(k)\|$$

and

$$\begin{aligned}\|A\boldsymbol{x}_i' - \lambda_i B\boldsymbol{x}_i'\| &= \left\|r_i(i) + \sum_{\substack{k=1\\k\neq j}}^{n} \alpha r_i(k) - \overline{\beta} r_i(j) - C\right\| = \left\|\alpha \sum_{\substack{k=1\\k\neq i,j}}^{n} r_i(k) - \overline{\beta} r_i(j) - C\right\| \\ &\leq |\alpha| \sum_{\substack{k=1\\k\neq i,j}}^{n} \|r_i(k)\| + |\beta| r_i(j) + \|C\|.\end{aligned}$$

In practice, eigenpairs might not have converged far enough for these assumptions to hold when orthogonalization is applied, though.

In Section 5.2.15, the form of the R factor matrix or, more precisely, its inverse has proven to be crucial for residual invariance. It leaves the direction of the first vector untouched while all other vectors are constructed being orthogonal to this first vector. With the above, it would seem beneficial to modify the order of orthogonalization, i.e., sort the vectors by residual, best residual first. Since the residual already appears to be invariant, this supposed optimization is of no further importance, though.

When orthogonalizing full blocks of vectors, every vector of the modified block is orthogonalized against every vector from the second block. Within these blocks, vectors have different residuals and for every block there is one vector that shows the worst residual for its block. Following the above, if two blocks Z and Y are to be orthogonalized, there now is a readily available indication of which block should be orthogonalized against which. That is, the block with the larger worst-case residual should be orthogonalized against the block with the lower worst-case residual.

Experiment 5.6 — Influence of inter-orthogonalization on the residual
In Experiment 5.1, the orthogonalization of a vector block with low worst-case residual against a block with considerably larger worst-case residual caused, not least due to close eigenvalue proximity, a significant increase in the residuals of the modified vectors. The original residual of $4.696 \cdot 10^{-12}$ *was diminished to* $1.887 \cdot 10^{-9}$*. While the previous experiment did focus on the effects of intra-orthogonality and these numbers were of secondary interest, we revisit the experiment to confirm the importance of the order in which orthogonalization takes place. Repeating the experiment under the exact same conditions, only reversing the order of orthogonalization, reveals even a slight improvement of the worst-case residual of the modified vectors. With the vector block showing the larger maximum residual being the one modified, the residual improves from initially* $2.412 \cdot 10^{-9}$ *to* $1.474 \cdot 10^{-9}$ *purely due to inter-orthogonalization. We may retrospectively note that a following local reestablishment of intra-orthogonality using non SVD based methods has no effect on the residual in this case as well. Figure 5.15 details all residuals of the modified interval and their orthogonalization-induced changes. The left plot compares residual of the left interval before and after being orthogonalized against the right interval, the right plot accordingly compares residuals of the right interval.*

Deciding on the orthogonalization order solely on basis of worst-case residuals does not take into account the other residuals of the two blocks. Indeed, not always does the above strategy reveal the correct order. Worse still, there are cases where neither order is correct and both possibilities impact the residual negatively. In these situations, the best course of action is to minimize the degradation in residual as much as possible. Since the comparison of the worst-case residual alone does not always lead to the right choice of orthogonality provider and orthogonality recipient, the following heuristic has been found to improve the results slightly in many cases. If the difference in worst-case residual is smaller than one tenth an order of magnitude,

$$\left| \log_{10} \frac{\max(\mathcal{R}_i)}{\max(\mathcal{R}_j)} \right| < \frac{1}{10}, \tag{5.6}$$

where $\mathcal{R}_i$ and $\mathcal{R}_j$ are the sets containing all residuals of the intervals i and j, then

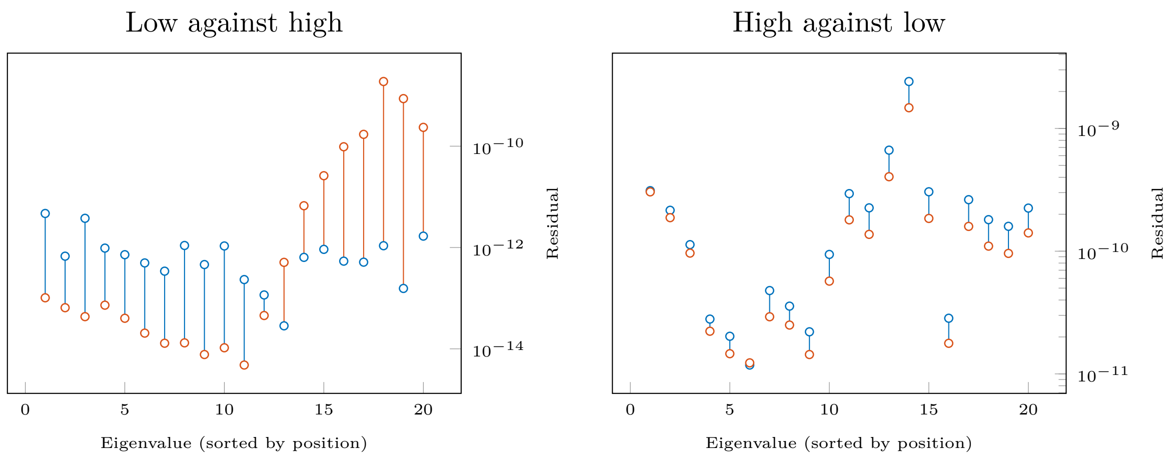


Figure 5.15: *Inter-orthogonalization. Blue: original residual. Red: residual after orthogonalization.*

the smallest algebraic (not geometric) mean residual

$$\frac{1}{m_i}\sum_{k=1}^{m_i} r_{ik}$$

defines the ordering. Here, r_{ik} is the k-th residual of the i-th interval and m_i is the number of eigenpairs computed in interval i. Otherwise, the largest worst-case residual is used to define the ordering, as was the case before. Experiments to identify an optimal value for the switch between worst-case and average residual were performed but their results were inconclusive and a distinct optimal value for all cases could not be found.

5.3.8 A posteriori orthogonalization

While intermediate orthogonalizations, performed with the right ordering, do accelerate convergence, there are reasons for pursuing an approach that does not rely on many continuous orthogonalizations.

- A full orthogonalization sweep for many intervals is expensive, both in computational effort, mostly due to the inner products, and communication overhead.
- Intermediate orthogonalizations require some pattern of synchronization between intervals. In case several intervals are to be processed by the same process group in sequence, this is not possible at all (but would allow orthogonalization against finished sets, which is a possible strategy if the order of sets was chosen carefully).
- To reach machine precision orthogonality, an a posteriori orthogonalization is required anyway.
- Just to reach orthogonality, intermediate orthogonalizations are not necessary.

- An a posteriori approach could be used in other contexts where an orthogonalization is required and which is not necessarily iterative.

To analyze the success of different orderings, we employ the matrix test set from Table 4.1. The general results of computing the specified intervals in 64 subsets is listed in Table 5.4 for an even distribution of eigenpairs among intervals and in Table 5.5 for an uneven distribution, produced by naive equally spaced subdivision of the target interval. Listed are results with and without locking as we expect this parameter to affect residual quality. Locked vectors are not iterated further while they otherwise would improve in residual well beyond the prescribed tolerance. Therefore, with locking, most eigenpairs end up at residuals close to the prescribed tolerance limit where, without locking, at least some vectors will reach much lower residuals.

Name	locking				no locking			
	min res	max res	intra orth	inter orth	min res	max res	intra orth	inter orth
`laser`	8.999e−14	1.475e−09	1.416e−15	2.741e−06	7.741e−16	1.202e−09	1.665e−15	2.303e−06
`SiH4`	3.727e−12	7.992e−08	1.051e−10	2.029e−04	7.355e−16	6.483e−08	1.550e−15	3.454e−04
`linverse`	1.798e−13	3.499e−07	1.424e−07	1.655e−02	6.461e−17	2.504e−07	1.873e−15	3.398e−03
`Pres_Poisson`	1.150e−12	1.393e−08	2.411e−12	7.231e−05	8.777e−16	1.332e−08	2.545e−15	3.709e−05
`Si5H12`	2.138e−14	4.010e−09	2.385e−15	3.250e−06	1.136e−15	2.714e−09	2.815e−15	1.408e−06
`brainpc2`	7.377e−14	6.391e−09	3.506e−11	7.819e−05	3.003e−15	5.776e−09	2.510e−15	3.599e−05
`rgg_n_2_15_s0`	4.588e−13	2.439e−08	2.942e−13	5.532e−05	1.732e−15	1.898e−08	3.301e−15	2.379e−05
`SiO`	9.972e−14	8.995e−09	4.006e−15	9.705e−06	1.118e−15	5.445e−09	4.255e−15	4.022e−06
`Andrews`	3.750e−13	2.536e−08	2.401e−14	2.293e−05	8.296e−16	2.362e−08	3.640e−15	2.054e−05
`Si34H36`	7.879e−14	1.822e−08	5.291e−15	1.144e−05	1.331e−15	1.292e−08	5.046e−15	6.671e−06
`fe_rotor`	5.259e−14	3.929e−08	5.771e−11	6.671e−05	1.500e−15	3.387e−08	3.948e−15	3.864e−05
`GraI-119k`	9.284e−15	5.549e−08	6.438e−15	2.890e−05	1.708e−15	5.549e−08	5.780e−15	2.888e−05

Table 5.4: *Maximum residual and orthogonalities for the matrix test set with even distribution of eigenpairs among* 64 *intervals of different size.*

For both interval setups it can be seen that the locking of eigenvectors, albeit intermediate orthogonalization (against the locked vectors), causes diminished intra-orthogonality in cases where the spectrum is dense. As seen before, any additional iteration is enough to disturb this orthogonality significantly. The locking mechanism orthogonalizes the iterated vectors against the locked ones. As such, the last orthogonalization of a vector occurred in the iteration before it converged and thus any disturbance was produced by the last active iteration of the respective vector. Locking of iterated vectors can be understood as independent iteration of subsets of vectors, even if the initial searchspace set was iterated in unison and only one interval was involved. Local post-orthogonalization is required, also in this case. An a posteriori orthogonalization of the computed eigenvectors did not change the residual, which confirms the previous experiment, where local reorthogonalization did not impact the residual in any significant way (Experiment 5.1). If locking is disabled, all computed sets are reasonably orthogonal.

Name	locking				no locking			
	min res	max res	intra orth	inter orth	min res	max res	intra orth	inter orth
`laser`	1.196e−13	1.418e−09	2.026e−15	6.831e−07	1.097e−15	1.359e−09	2.165e−15	7.285e−07
`SiH4`	2.706e−13	8.876e−10	1.613e−15	1.474e−06	8.863e−16	8.155e−09	1.782e−15	1.903e−05
`linverse`	4.139e−13	3.441e−09	4.213e−10	6.914e−05	1.753e−16	2.932e−09	3.838e−15	6.595e−05
`Pres_Poisson`	2.357e−13	1.391e−08	3.719e−13	3.612e−05	7.373e−16	1.188e−08	2.222e−15	3.328e−05
`Si5H12`	4.573e−14	3.925e−09	2.042e−15	2.961e−06	1.365e−15	2.997e−09	2.756e−15	2.048e−06
`brainpc2`	1.177e−13	6.462e−09	1.766e−13	2.885e−05	3.483e−15	5.444e−09	7.355e−15	1.349e−05
`rgg_n_2_15_s0`	2.336e−12	2.419e−08	1.287e−12	5.182e−05	2.384e−15	2.261e−08	3.303e−15	4.695e−05
`SiO`	1.999e−14	8.948e−09	2.816e−14	9.381e−06	1.215e−15	7.518e−09	3.774e−15	3.185e−06
`Andrews`	2.687e−13	2.523e−08	4.808e−14	2.407e−05	8.035e−16	2.476e−08	3.807e−15	2.161e−05
`Si34H36`	4.115e−14	1.802e−08	4.661e−15	1.349e−05	1.358e−15	1.600e−08	5.256e−15	8.814e−06
`fe_rotor`	1.026e−13	3.948e−08	1.481e−11	4.835e−05	1.492e−15	2.726e−08	3.730e−15	2.477e−05
`GraI-119k`	7.212e−15	5.232e−08	5.516e−15	2.361e−05	1.261e−15	2.183e−08	5.849e−15	5.460e−06

Table 5.5: *Maximum residual and orthogonalities for the matrix test set with uneven distribution of eigenpairs among* 64 *intervals of equal size.*

Block Gram-Schmidt orthogonalizes vector blocks in a strict order. Blocks with higher index are always orthogonalized against blocks with lower index. If a vector block is used as orthogonality provider, it has already been brought to an orthogonal state w.r.t. all vector blocks of lower index before. It is, however, possible to reorder the blocks. The following results compare the orthogonalization strategies listed blow, carried out via block Gram-Schmidt.

Native The ordering as given by the spectrum.

Worst-case residual The ordering defined by the worst residual among all residuals of a vector block.

Worst average residual The ordering defined by the worst arithmetic mean residual of a vector block with fall-back to the worst-case residual, following the heuristic from Equation (5.6).

The feasibility of those purely a posteriori orthogonalization schemes hinges on the possible overall increase of the worst-case residual. This *residual loss* is given as the number

$$\log_{10}\left(\frac{\max(r_{\text{orthogonalized}})}{\max(r_{\text{original}})}\right),$$

that is, the increase of the worst-case residual after orthogonalization in orders of magnitude. A positive number indicates a deterioration in residual, a negative number an improvement. The larger the absolute value of this number is, the larger is the deterioration or improvement.

The following set of tables contains the most important metrics for the success of the orthogonalization after it has been applied: maximum residual, maximum inter-orthogonality, maximum intra-orthogonality, and residual loss. Tables 5.6 and 5.7 contain these numbers for the native ordering. Tables 5.8 and 5.9 cover the worst-case residual ordering. Tables 5.10 and 5.11 show results for the average residual ordering. The residual loss of all tables is again compared directly in Tables 5.12 and 5.13.

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.539e−16	1.992e−16	2.756e−09	+0.271	2.220e−16	2.223e−16	1.481e−09	+0.091
`SiH4`	8.847e−17	9.021e−17	1.249e−07	+0.194	6.765e−17	8.327e−17	4.151e−08	−0.194
`linverse`	7.991e−17	5.489e−17	5.811e−07	+0.220	2.779e−19	5.809e−17	2.495e−07	−0.002
`Pres_Poisson`	3.014e−16	5.940e−16	2.149e−08	+0.188	3.331e−16	4.996e−16	1.219e−08	−0.039
`Si5H12`	5.638e−17	3.556e−17	6.463e−09	+0.207	4.976e−17	3.361e−17	2.264e−09	−0.079
`brainpc2`	5.461e−16	4.823e−16	1.125e−08	+0.246	3.608e−16	5.551e−16	7.749e−09	+0.128
`rgg_n_2_15_s0`	3.192e−16	1.689e−16	4.267e−08	+0.243	3.140e−16	1.596e−16	1.541e−08	−0.090
`SiO`	5.226e−17	4.163e−17	2.174e−08	+0.383	5.378e−17	4.337e−17	4.089e−09	−0.124
`Andrews`	2.201e−17	2.338e−17	3.766e−08	+0.172	2.716e−17	2.016e−17	2.054e−08	−0.061
`Si34H36`	3.816e−17	2.060e−17	1.901e−08	+0.018	3.643e−17	2.082e−17	1.166e−08	−0.045
`fe_rotor`	6.418e−17	3.426e−17	5.286e−08	+0.129	7.286e−17	2.776e−17	3.383e−08	−0.000
`GraI-119k`	1.457e−16	5.378e−17	4.619e−08	−0.080	1.398e−16	5.763e−17	3.915e−08	−0.151

Table 5.6: *Block Gram-Schmidt orthogonalization. Sort method:* ***native****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***even*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.261e−16	2.014e−16	2.096e−09	+0.170	2.776e−16	2.741e−16	1.585e−09	+0.067
`SiH4`	5.985e−17	1.041e−16	1.300e−09	+0.166	8.327e−17	1.249e−16	6.670e−09	−0.087
`linverse`	7.619e−17	2.526e−17	5.037e−09	+0.166	2.096e−17	2.776e−17	3.305e−09	+0.052
`Pres_Poisson`	3.464e−16	5.741e−16	2.669e−08	+0.283	3.266e−16	4.441e−16	1.376e−08	+0.064
`Si5H12`	4.532e−17	3.426e−17	5.392e−09	+0.138	4.814e−17	3.383e−17	2.958e−09	−0.006
`brainpc2`	5.162e−16	4.163e−16	1.014e−08	+0.196	4.214e−16	4.718e−16	5.049e−09	−0.033
`rgg_n_2_15_s0`	3.192e−16	2.407e−16	5.676e−08	+0.370	3.205e−16	1.479e−16	2.111e−08	−0.030
`SiO`	8.500e−17	4.169e−17	1.611e−08	+0.255	5.638e−17	4.857e−17	7.426e−09	−0.005
`Andrews`	1.995e−17	2.114e−17	4.851e−08	+0.284	2.559e−17	2.017e−17	2.476e−08	+0.000
`Si34H36`	4.337e−17	2.255e−17	2.295e−08	+0.105	3.144e−17	2.255e−17	1.156e−08	−0.141
`fe_rotor`	6.418e−17	2.559e−17	5.188e−08	+0.119	6.939e−17	2.689e−17	2.235e−08	−0.086
`GraI-119k`	6.416e−16	2.832e−16	4.353e−08	−0.080	4.372e−16	4.096e−16	1.696e−08	−0.110

Table 5.7: *Block Gram-Schmidt orthogonalization. Sort method:* ***native****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***uneven*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Both inter-orthogonality and intra-orthogonality are in the order of machine precision. There should be no significant differences between the different sorting methods. This makes the residual loss the deciding factor for the quality of the ordering methods. Most obvious, however, is the difference between data sets with locking enabled compared to data sets without locking. The improvement in residual for quickly converging vectors, that otherwise would be prevented by excluding these vectors from

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
laser	3.261e−16	2.519e−16	1.853e−09	+0.099	2.359e−16	2.349e−16	9.852e−10	−0.086
SiH4	8.445e−17	6.939e−17	7.528e−08	−0.026	6.332e−17	8.327e−17	4.151e−08	−0.194
linverse	1.009e−16	1.136e−16	4.725e−07	+0.130	1.554e−19	6.592e−17	1.487e−08	−1.226
Pres_Poisson	3.212e−16	4.996e−16	2.149e−08	+0.188	2.334e−16	6.419e−16	1.034e−08	−0.110
Si5H12	4.337e−17	4.077e−17	5.572e−09	+0.143	3.990e−17	3.513e−17	1.418e−09	−0.282
brainpc2	3.914e−16	3.794e−16	1.440e−08	+0.353	4.855e−16	4.718e−16	5.058e−09	−0.058
rgg_n_2_15_s0	3.400e−16	2.088e−16	4.252e−08	+0.241	2.923e−16	1.470e−16	1.284e−08	−0.170
SiO	5.117e−17	4.337e−17	2.054e−08	+0.359	6.473e−17	3.816e−17	2.775e−09	−0.293
Andrews	2.429e−17	1.999e−17	3.686e−08	+0.162	2.586e−17	2.244e−17	9.060e−09	−0.416
Si34H36	3.361e−17	2.263e−17	1.901e−08	+0.018	3.602e−17	2.082e−17	6.377e−09	−0.307
fe_rotor	7.633e−17	2.602e−17	5.138e−08	+0.117	7.026e−17	2.602e−17	1.595e−08	−0.327
GraI-119k	1.379e−16	5.725e−17	3.808e−08	−0.164	1.674e−16	6.592e−17	2.480e−08	−0.350

Table 5.8: *Block Gram-Schmidt orthogonalization. Sort method:* ***worst-case residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***even*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
laser	3.469e−16	2.392e−16	2.138e−09	+0.178	2.914e−16	2.776e−16	4.723e−10	−0.459
SiH4	6.690e−17	1.041e−16	1.300e−09	+0.166	6.191e−17	1.249e−16	3.819e−09	−0.329
linverse	7.488e−17	4.163e−17	3.829e−09	+0.046	2.115e−17	2.776e−17	1.585e−09	−0.267
Pres_Poisson	3.349e−16	4.679e−16	2.508e−08	+0.256	2.082e−16	4.040e−16	7.556e−09	−0.197
Si5H12	5.334e−17	3.518e−17	5.714e−09	+0.163	4.163e−17	3.383e−17	1.150e−09	−0.416
brainpc2	3.951e−16	4.441e−16	1.014e−08	+0.196	2.908e−16	4.996e−16	3.965e−09	−0.138
rgg_n_2_15_s0	3.227e−16	1.713e−16	4.481e−08	+0.268	3.227e−16	2.030e−16	1.748e−08	−0.112
SiO	5.117e−17	4.510e−17	1.957e−08	+0.340	6.418e−17	4.163e−17	3.041e−09	−0.393
Andrews	2.437e−17	2.179e−17	4.851e−08	+0.284	2.456e−17	2.190e−17	1.438e−08	−0.236
Si34H36	4.012e−17	2.201e−17	2.302e−08	+0.106	3.030e−17	2.288e−17	7.333e−09	−0.339
fe_rotor	5.811e−17	2.667e−17	5.185e−08	+0.118	6.505e−17	2.385e−17	1.814e−08	−0.177
GraI-119k	4.817e−16	3.004e−16	2.969e−08	−0.246	4.111e−16	3.926e−16	1.300e−10	−2.225

Table 5.9: *Block Gram-Schmidt orthogonalization. Sort method:* ***worst-case residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***uneven*** *distribution of eigenpairs among* 64 *intervals of equal size.*

further iterations, also improves the effect of orthogonalization on the residual and is much more noticeable than the nuanced differences between orderings. The step from no ordering to worst-case residual based ordering shows improvements in some cases, but even deterioration in others. The best results are achieved in combination with disabling the locking mechanism. The change to the average residual ordering can again slightly improve results, but not in all cases.

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.192e−16	2.500e−16	1.681e−09	+0.057	2.359e−16	1.891e−16	9.852e−10	−0.086
`SiH4`	8.847e−17	8.327e−17	7.528e−08	−0.026	7.373e−17	9.021e−17	4.151e−08	−0.194
`linverse`	2.964e−16	1.138e−16	4.659e−07	+0.124	1.554e−19	5.910e−17	1.839e−08	−1.134
`Pres_Poisson`	2.769e−16	5.034e−16	1.834e−08	+0.120	3.427e−16	5.863e−16	1.034e−08	−0.110
`Si5H12`	3.946e−17	3.643e−17	4.070e−09	+0.007	5.052e−17	3.144e−17	1.683e−09	−0.208
`brainpc2`	3.878e−16	3.121e−16	9.518e−09	+0.173	4.056e−16	5.829e−16	5.058e−09	−0.058
`rgg_n_2_15_s0`	3.821e−16	1.869e−16	4.267e−08	+0.243	3.469e−16	1.804e−16	1.284e−08	−0.170
`SiO`	5.811e−17	4.163e−17	1.336e−08	+0.172	5.204e−17	4.640e−17	2.775e−09	−0.293
`Andrews`	2.374e−17	2.082e−17	3.766e−08	+0.172	2.602e−17	2.065e−17	9.060e−09	−0.416
`Si34H36`	3.990e−17	2.255e−17	1.888e−08	+0.015	3.990e−17	2.168e−17	6.377e−09	−0.307
`fe_rotor`	6.722e−17	4.380e−17	5.072e−08	+0.111	7.242e−17	3.123e−17	1.595e−08	−0.327
`GraI-119k`	1.457e−16	5.204e−17	4.180e−08	−0.123	1.492e−16	5.551e−17	2.480e−08	−0.350

Table 5.10: *Block Gram-Schmidt orthogonalization. Sort method:* ***worst avg. residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***even*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.192e−16	4.059e−16	1.665e−09	+0.070	2.359e−16	2.498e−16	4.723e−10	−0.459
`SiH4`	6.896e−17	1.110e−16	8.819e−10	−0.003	6.418e−17	9.714e−17	3.819e−09	−0.329
`linverse`	6.290e−17	2.653e−17	3.648e−09	+0.025	2.115e−17	2.776e−17	1.585e−09	−0.267
`Pres_Poisson`	3.412e−16	6.514e−16	2.049e−08	+0.168	2.394e−16	4.002e−16	8.680e−09	−0.136
`Si5H12`	4.554e−17	3.339e−17	4.463e−09	+0.056	4.337e−17	3.469e−17	1.150e−09	−0.416
`brainpc2`	4.287e−16	9.992e−16	8.983e−09	+0.143	3.070e−16	4.718e−16	2.271e−09	−0.380
`rgg_n_2_15_s0`	3.018e−16	1.527e−16	4.066e−08	+0.226	3.478e−16	1.527e−16	1.748e−08	−0.112
`SiO`	6.072e−17	4.857e−17	1.604e−08	+0.253	5.464e−17	3.990e−17	3.041e−09	−0.393
`Andrews`	2.429e−17	2.196e−17	3.629e−08	+0.158	2.526e−17	2.006e−17	1.438e−08	−0.236
`Si34H36`	4.315e−17	2.120e−17	2.671e−08	+0.171	3.144e−17	2.602e−17	7.333e−09	−0.339
`fe_rotor`	7.850e−17	2.776e−17	4.596e−08	+0.066	7.579e−17	2.602e−17	1.814e−08	−0.177
`GraI-119k`	4.244e−16	3.893e−16	3.252e−08	−0.206	4.111e−16	3.402e−16	1.300e−10	−2.225

Table 5.11: *Block Gram-Schmidt orthogonalization. Sort method:* ***worst avg. residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***uneven*** *distribution of eigenpairs among* 64 *intervals of equal size.*

5.3.9 Detection and removal of duplicates

Special cases among the several combinations from the test set are the `SiH4` matrix with the even eigenpair distribution, the `linverse` matrix with the uneven eigenpair distribution, and the `SiO` matrix with the even eigenpair distribution. In all these cases identical eigenpairs have been computed in neighboring intervals due to the spectral density in these regions. We will call these spuriously computed eigenpairs *duplicates*.

Name	locking			no locking		
	native	min.res	avg.res	native	min.res	avg.res
`laser`	+0.271	+0.099	+0.057	+0.091	−0.086	−0.086
`SiH4`	+0.194	−0.026	−0.026	−0.194	−0.194	−0.194
`linverse`	+0.220	+0.130	+0.124	−0.002	−1.226	−1.134
`Pres_Poisson`	+0.188	+0.188	+0.120	−0.039	−0.110	−0.110
`Si5H12`	+0.207	+0.143	+0.007	−0.079	−0.282	−0.208
`brainpc2`	+0.246	+0.353	+0.173	+0.128	−0.058	−0.058
`rgg_n_2_15_s0`	+0.243	+0.241	+0.243	−0.090	−0.170	−0.170
`SiO`	+0.383	+0.359	+0.172	−0.124	−0.293	−0.293
`Andrews`	+0.172	+0.162	+0.172	−0.061	−0.416	−0.416
`Si34H36`	+0.018	+0.018	+0.015	−0.045	−0.307	−0.307
`fe_rotor`	+0.129	+0.117	+0.111	−0.000	−0.327	−0.327
`GraI-119k`	−0.080	−0.164	−0.123	−0.151	−0.350	−0.350

Table 5.12: *Direct comparison of sorting methods for* ***even*** *distribution of eigenpairs.*

Name	locking			no locking		
	native	min.res	avg.res	native	min.res	avg.res
`laser`	+0.170	+0.178	+0.070	+0.067	−0.459	−0.459
`SiH4`	+0.166	+0.166	−0.003	−0.087	−0.329	−0.329
`linverse`	+0.166	+0.046	+0.025	+0.052	−0.267	−0.267
`Pres_Poisson`	+0.283	+0.256	+0.168	+0.064	−0.197	−0.136
`Si5H12`	+0.138	+0.163	+0.056	−0.006	−0.416	−0.416
`brainpc2`	+0.196	+0.196	+0.143	−0.033	−0.138	−0.380
`rgg_n_2_15_s0`	+0.370	+0.268	+0.226	−0.030	−0.112	−0.112
`SiO`	+0.255	+0.340	+0.253	−0.005	−0.393	−0.393
`Andrews`	+0.284	+0.284	+0.158	+0.000	−0.236	−0.236
`Si34H36`	+0.105	+0.106	+0.171	−0.141	−0.339	−0.339
`fe_rotor`	+0.119	+0.118	+0.066	−0.086	−0.177	−0.177
`GraI-119k`	−0.080	−0.246	−0.206	−0.110	−2.225	−2.225

Table 5.13: *Direct comparison of sorting methods for* ***uneven*** *distribution of eigenpairs.*

We will examine the case of `SiH4` as an example. The spectrum is comprised of clusters with zero separation, i.e., real multiplicity. The attempt to separate a target interval into sub intervals which all contain the same amount of eigenpairs based on the known spectrum inevitably will position interval boundaries at the exact locations of eigenpairs. Without overlapping intervals, it is as easy to miss eigenpairs as it is to compute duplicated eigenpairs. Indeed, in the experiment a total of 750 eigenpairs were computed, while only 731 eigenpairs are located inside the target interval. The orthogonalization of all intervals indicated that only 720 unique eigenpairs were computed. The primary observed effect that allows this deduction is a significant jump in post-orthogonalization worst-case residual. While the removal of more vectors did not diminish the (otherwise optimal) final residual—the overall residual could be improved even—removing fewer vectors caused an increase in residual by several orders of magnitude. A close examination of the results and comparison with the known spectrum showed that among the 750 eigenpairs, 27 could be confirmed duplicate and 8 could be confirmed missing, giving a total of $750 - 27 + 8 = 731$. This difference of $27 - 8 = 19$ to the 720 unique eigenpairs identified by the orthogonalization step confirms the overall correctness. The interval of orthogonality that allowed the separation of duplicate eigenpairs was approximately $[5.8 \cdot 10^{-1}, 6.5 \cdot 10^{-1}]$.

When multiple intervals are computed independently, we have to expect duplicate eigenpairs in neighboring intervals not only if the intervals overlap, but also due to dense spectra at the interval boundaries or the overlapping region, such that Ritz values for eigenvalues that are located barely outside the interval appear inside the interval or Ritz values that appear barely outside the interval have to be assumed to belong inside the interval if the residual does not allow a more accurate localization (see also Section 4.4). For the same reasons, without overlap, we have to expect missing eigenpairs in the above cases. Consider two intervals $\mathcal{I}_1$ and $\mathcal{I}_2$ that are directly adjacent, $\mathcal{I}_1$ left of $\mathcal{I}_2$. If the interval boundaries cut densely clustered eigenpairs, deciding which pairs from different intervals should be labeled duplicates can become difficult, since the orthogonality among most or all clustered pairs from different intervals is likely to be large. Surely, a criterion for the classification should be based on the respective residual, instead of being a hard limit on orthogonality. Based on Theorem 1.4, similar to Section 4.2.5.3 and bound $\boxed{3}$ from Section 4.5.2, we can only be sure that two Ritz values $\boldsymbol{\lambda}_j \in \mathcal{I}_1$ and $\boldsymbol{\lambda}_k \in \mathcal{I}_2$ with distance g and residuals r_j and r_k, respectively, are not duplicates if their Ritz disk is disjoint,

$$g > \left\| B^{-\frac{1}{2}} r_j \right\| + \left\| B^{-\frac{1}{2}} r_k \right\|; \tag{5.7}$$

otherwise, we have no way to separate both eigenpairs. Therefore, this will also be the definition of a cluster for the remainder of this chapter. As a consequence, even if both eigenpairs are not duplicates but in very close proximity, there is no way to differentiate them from actual duplicates and their orthogonality can be arbitrarily large. It is then generally unclear, whether clustered values are duplicates and should be removed or not. The limit for the applicability of this method of classification is therefore given by the density of the spectrum in relation to the achievable residual and any attempt at classification beyond this limit is bound to fail with eigenpairs missing due to erroneous removal. Let $O_{i,j} := \boldsymbol{X}_i^H B \boldsymbol{X}_j$ with $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ being the computed eigenvectors from interval $\mathcal{I}_i$ and $\mathcal{I}_j$, respectively, be the *orthogonality matrix* of the intervals $\mathcal{I}_i$ and $\mathcal{I}_j$. We will refer to entries of $O_{i,j}$ for which the associated Ritz pairs fail the criterion from Equation (5.7) as *problematic*, *offending*, or *critical*.

With overlap, if the spectral distribution allows the clustered eigenpairs to be completely included in either of the intervals, such that its separation to the remaining eigenpairs is more clear, even if the other interval only holds part of the cluster, the situation can be resolved. It is then possible to detect which interval holds more of the problematic values and we can assume that this cluster is complete. This is, of course, not known beforehand. We assume the possible overlap of intervals limited such that overlap only occurs between nearest-neighbor intervals. Any significant overlap is likely to include otherwise cut clusters.

The question remains which intervals should be reduced or even if some eigenpairs should be removed from one interval and some from the other. The underlying question here is whether we should put more weight on a good residual or on a good orthogonality. We can assume that the intra-orthogonality of both intervals is sufficiently close to machine precision and all residuals are smaller than a specified limit. Any removal of vectors from the clustered eigenpairs in both intervals will leave vectors of the cluster in both intervals. The inter-orthogonality among those vectors has a chance to be as large as the inter-orthogonality between two duplicate vectors if the spectrum is dense. Consider the examples shown in Figure 5.16. The

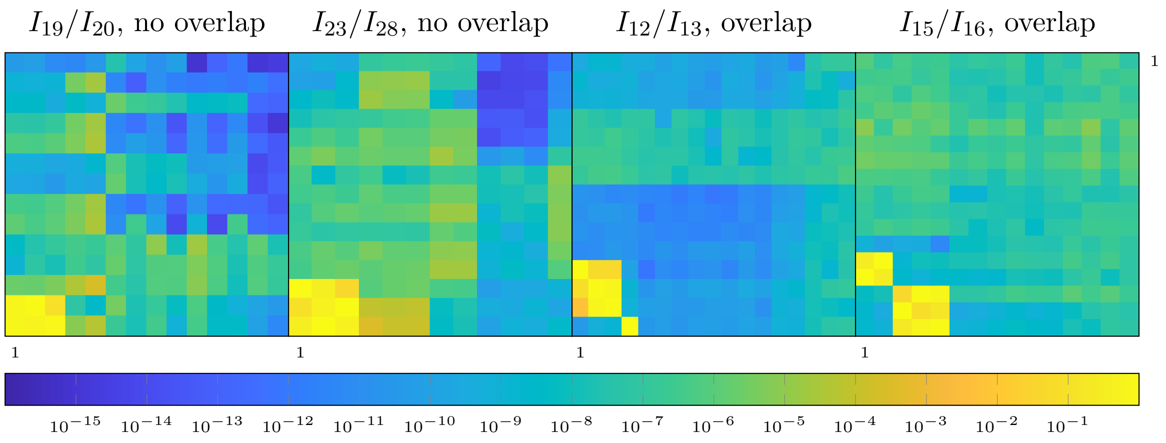


Figure 5.16: *Inter-orthogonalities between different neighboring intervals of the `SiH4` matrix with and without overlap. Note that every block is scaled individually to be quadratic.*

two images on the left show cases from the `SiH4` matrix without overlap. In the case of the first image, the orthogonalities between the last two vectors from the left interval and the first three vectors from the right interval are noticeably larger. We assume the vectors to be sorted by eigenvalue position. Therefore we will find blocks of large orthogonality as block off-diagonal in the lower left corner of the images. It is clear that removing three vectors from the right interval would be wrong since the resulting eigenpair count would be lower compared to removing two vectors from the left interval. In both cases all offending orthogonalities would be eliminated. In the case of the second image, three vectors can be removed from either the left or the right interval. Removing one vector from the right interval and two from the left leaves the possibility of remaining offending orthogonalities. The two remaining images show cases from the `SiH4` matrix with overlap. Here, any cluster is contained either in the left or the right interval. Whenever a cluster is cut by one interval, it has to be contained in the other for the method to work.

Removing rows from the orthogonality matrix $O_{1,2}$ is equivalent to removing vectors from $\boldsymbol{X}_1$ and removing columns from $O_{1,2}$ is equivalent to removing vectors from $\boldsymbol{X}_2$. Without overlap, in order to remove all large orthogonalities, all offending vectors have to be removed either from $\mathcal{I}_1$ *or* $\mathcal{I}_2$, never mixed. The figure shows that some of the larger orthogonalities would remain in $O_{1,2}$ (compare the right two images of Figure 5.16). In case the classification shows one interval to contain less problematic vectors (indicated by either fewer problematic rows or columns in $O_{1,2}$), this should be the interval to be decimated; otherwise we are sure to miss eigenpairs.[10]

If the boundary eigenvalues of both intervals are far enough apart such that they can be separated by the residual following Equation (5.7), this is indicated by the separation into multiple blocks of large orthogonality in $O_{1,2}$ (compare the left two images of Figure 5.16) when interval overlap is applied. In this case the lower left entry (in the representation used here) of $O_{1,2}$ becomes uncritial. We therefore will refer to this entry as *indicator entry*.

Figure 5.17 shows several possible situations. All pictures symbolize orthogonality matrices where the rows relate to the vectors of the left interval and the columns to the vectors of the right interval. The indicator entry is highlighted.

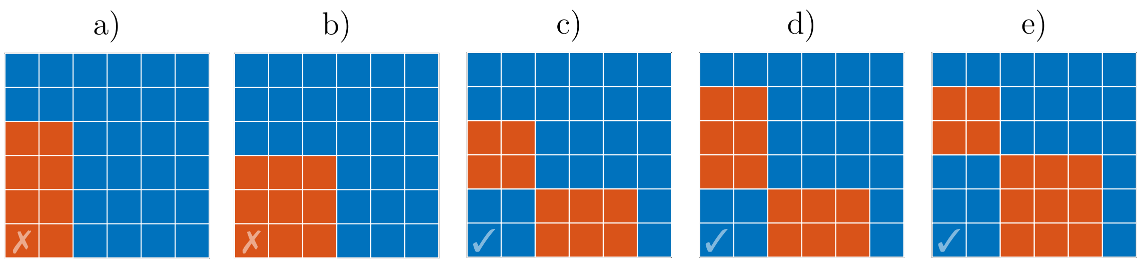


Figure 5.17: *Duplicate detection: orthogonality patterns. Problematic entries are colored in red, unproblematic ones in blue. The indicator entry is marked as problematic (✗) or unproblematic (✓).*

a) The problematic region is rectangular. The left interval contains four vectors that interact strongly with two vectors from the right interval. The cluster is at least of size four, but eigenpairs could be missing. The best course of action is to remove the offending vectors from the right interval and flag possibly missing eigenpairs.
b) The problematic region is a square, meaning that three vectors from both intervals belong to a cluster. It is unclear if the cluster is complete and eigenpairs could be missing. The best course of action is to remove the offending vectors from either interval, maybe decide based on residual or remaining orthogonalities in the rows/columns.
c) The indicator entry is uncritical, indicating that the outermost vectors of both intervals are well separated and no eigenpair was lost. The rectangular shape of the lower block shows that the associated cluster is incomplete in the left

[10] This does, of course, not ensure that no eigenpairs are missing.

interval. The separate blocks can be removed independently from each other but for the rectangular block the two rows must be removed to not loose eigenpairs; the rules and suggestions from a) and b) apply block-wise.

d) No eigenpair was lost since the indicator entry is unproblematic. While the total number of offending rows and columns is equal, the blocks must be removed separately, columns for the left block and rows for the lower block, following the same rules as before. The number of removed vectors then is four (as opposed to five if only rows or columns are removed).

e) No eigenpair was lost; both blocks are square and do not extend to the ends of the matrix. In this case we can remove both blocks in any way, based on residual or remaining orthogonalities in the respective rows or columns.

More difficult cases are possible. If the internal cluster separation is less tight, blocks can be missing the outermost off-diagonal entries. In this case any block detection algorithm should interpret a connected construct as cluster and find the bounding rectangle. This is easily done in practice by employing a flood-fill algorithm that starts with any offending entry and extends a bounding rectangle to all adjacent offenders. It marks all visited entries and terminates once no more offenders are connected. The process is repeated for any not yet visited critical entry until all have been visited. It is likely that the missing entries of a block may not fall under the criterion for problematic values, but still have rather large orthogonalities, such that the above interpretation is the best option.

In the extreme case where a critical block extends to the ends of the orthogonality matrix $O_{1,2}$, the cluster might be cut on the opposite side of the interval, i.e., the boundary opposite of the overlap region between the two intervals under consideration. Even in these cases, the above strategy is correct, as it removes the partial cluster from the interval that holds the least values from the cluster and leaves the resolution of the situation to the interval pair that cut the cluster. This, of course, only holds if the indicator entry of $O_{1,2}$ is unproblematic. If it is not, the cluster fills one of the intervals completely and the associated interval will be wiped completely.

The process of duplicate removal can be integrated naturally in the orthogonalization scheme. Here we generally assume that overlap and duplicates only affect direct neighbors. Otherwise the interval as a whole is degenerate as the residual is not low enough to separate the eigenpairs. Since the order of orthogonalizations—be it native, worst-case residual, or others—is unrelated to the neighborhood relations of intervals and since nearest neighbor interactions may occur late in the scheme, intervals still holding vectors that would be removed as duplicates serve as orthogonality providers for intervals that are not nearest neighbors prior to removal. This is not a problem; whether these directions take part in an orthogonalization against other distinct directions, one or multiple times, does not affect the outcome. Merely the residual of these directions may play a role in the overall results. Tests have indicated that the late removal of these directions may even have a beneficial effect. It therefore is enough to augment the orthogonalization scheme, such that for every nearest neighbor interaction, duplicates are eliminated before the orthogonalization is applied. Since duplicates are detected on a local process or process group, the vectors that have to be removed from the remote partner interval have to be communicated in an additional step.

The alternative approach would be to inspect the residual and decide which interval the corresponding vector is removed from, based on which eigenpair has the larger residual. The large orthogonalities that remain using this approach, however, will have a detrimental effect on the residual during orthogonalization. In order to avoid this source of residual increase, we choose to prefer orthogonality in these considerations. We are still able to use the residual for some of the decisions, whenever the number of offending rows and columns in $O_{1,2}$ is equal and only one block of large orthogonalities is found.

For the three matrices mentioned at the beginning of this section, duplicates were removed prior to orthogonalization since the unprotected orthogonalization of duplicates severely disturbs the results or causes methods for intra-orthogonalization to fail. Repeating these difficult cases with a simple 10% overlap and applying the above method for removing duplicates did yield all eigenpairs, none were missing and none duplicate.

Without overlap, the residual disk criterion, however, is often too pessimistic and the removal of allegedly duplicate eigenpairs also removes actually distinct eigenpairs that just could not be separated in this way. A method for manually finding the correct set of eigenpairs to remove involves searching for a cut-off orthogonality that does not disturb the residual by many orders of magnitude. The jump in residual loss is quite noticeable and the critical orthogonality range for the matrix `SiH4` from the beginning of this section was identified this way.

5.3.10 Butterfly orthogonalization

Similar to the approach of weak Gram-Schmidt from before, less strict orthogonalization schemes have an iterative character since they include orthogonalizations with not yet finalized vector blocks, but allow for more flexibility with regard to orthogonalization patterns and order.

The pattern we will use here works stage-wise from nearest intervals to farthest intervals. For ease of discussion, let a sequence of p intervals $\mathcal{I}_1, \ldots, \mathcal{I}_p$ be given such that

$$[\lambda_{\min}, \lambda^{\max}] = \bigcup_{j=1}^{p} \mathcal{I}_j,$$

and, with $\mathcal{I}_j = [a_j, b_j]$, $b_{j-1} \geq a_j$ for $j = 2, \ldots, p$ such that $\mathcal{I}_{j-1} \cap \mathcal{I}_j = [a_j, b_{j-1}]$.

The k-th neighbors of interval $\mathcal{I}_j$ are $\mathcal{I}_{j-k}$ and $\mathcal{I}_{j+k}$, if they exist. The stages $k = 1, \ldots, p-1$ involve group communication with the k-th neighbors, such that all possible interval pairs have been considered after $p-1$ stages. In every stage, the decision which interval serves as orthogonality provider and which interval serves as orthogonality recipient is made by evaluating certain conditions, such as the worst-case residual of the intervals. The decision is made separately for both neighbors.

The result is a butterfly pattern, as illustrated in Figure 5.18 (left). Since for each

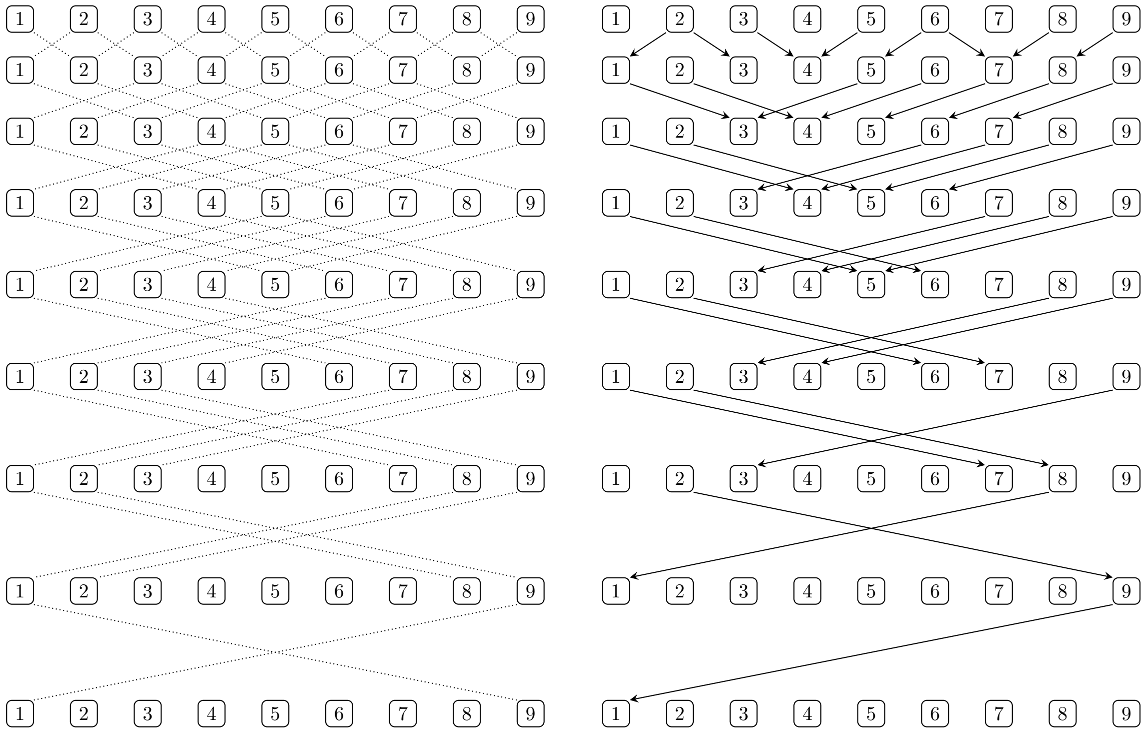


Figure 5.18: *Butterfly orthogonalization pattern. Left: general communication channels for all stages. Right: example realization based on maximum residual ordering.*

stage up to two neighbors have to be contacted, communication must be performed in two phases. Compute the color c of each interval $k = 1, \dots, p$ (or process group) and distance $d = 1, \dots, p-1$ as

$$c(d,k) = \left\lfloor \frac{k-1}{d} \right\rfloor \,\%\, 2$$

where $\%$ denotes the modulo operator for the rest of a an integer division. In the d-th step, if an interval is assigned the color zero, it considers the right d-th neighbor first and the left d-th neighbor second. If it is assigned the color one, the order is reversed.

It is worth noting that for stages $k > (p-1)/2$ communication is collision-free as no interval has two neighbors of this distance. The actual direction of orthogonalization is determined for each interval pair by exchanging the relevant information. This can happen once at the very beginning (updating the information later is not possible) or before each orthogonalization (with up-to-date information). Conditions that do not require additional information to be exchanged, such as “always left” or “always right”, can skip this step. Figure 5.18 (right) shows an example using the maximum residual ordering.

5.3.11 Block weak Gram-Schmidt

Analogous to single vectors, the approach of weak Gram-Schmidt is easily transferred to blocks of vectors. Let

$$V_i = Z_i - \sum_{k=1}^{i-1} Z_k Z_k^H B Z_i.$$

The implications on orthogonality are (almost) exactly the same. Again assume all Z_i to be orthonormal and let $j < i$. It is

$$\begin{aligned} V_j^H B V_i &= Z_j^H B Z_i - Z_j^H B Z_j Z_j^H B Z_i - \sum_{\substack{k=1\\k\neq j}}^{i-1} Z_j^H B Z_k Z_k^H B Z_i - \sum_{k=1}^{j-1} Z_j^H B Z_k Z_k^H B Z_i \\ &\quad + \sum_{k=1}^{j-1} \sum_{\substack{\ell=1\\ \ell\neq k}}^{i-1} Z_j^H B Z_k Z_k^H B Z_\ell Z_\ell^H B Z_i + \sum_{k=1}^{j-1} Z_j^H B Z_k Z_k^H B Z_k Z_k^H B Z_i \\ &= -\sum_{\substack{k=1\\k\neq j}}^{i-1} Z_j^H B Z_k Z_k^H B Z_i + \sum_{k=1}^{j-1} \sum_{\substack{\ell=1\\ \ell\neq k}}^{i-1} Z_j^H B Z_k Z_k^H B Z_\ell Z_\ell^H B Z_i \end{aligned} \tag{5.8}$$

with $(i-2)$ terms containing two Gram matrices and $(j-1)(i-2)$ terms containing three Gram matrices, as before. The matrix multiplications, however, introduce constant factors m, which is not surprising, thinking about block orthogonalizations vector-wise. To reformulate the requirement from Equation (5.1) for p vector blocks yields

$$a > (p-2)^2 m^2 a^3 + (p-2) m a^2 \quad \Longrightarrow \quad a < \frac{-1+\sqrt{5}}{2m(p-2)} \tag{5.9}$$

where a is now the maximum orthogonality between any two vectors of different blocks Z_i and all blocks are assumed to have a constant number of columns m. The requirement on the initial orthogonality increases as the achievable orthogonality decreases with more blocks and columns. This means that if at some point more iterations of weak Gram-Schmidt are required, other methods can be more feasible. Similarly,

$$\begin{aligned} V_i^H B V_i &= Z_i^H B Z_i - \sum_{k=1}^{i-1} Z_i^H B Z_k Z_k^H B Z_i - \sum_{k=1}^{i-1} Z_i^H B Z_k Z_k^H B Z_i \\ &\quad + \sum_{k=1}^{i-1} \sum_{\substack{\ell=1\\ \ell\neq k}}^{i-1} Z_i^H B Z_k Z_k^H B Z_\ell Z_\ell^H B Z_i + \sum_{k=1}^{i-1} Z_i^H B Z_k Z_k^H B Z_k Z_k^H B Z_i \\ &= I - \sum_{k=1}^{i-1} Z_i^H B Z_k Z_k^H B Z_i + \sum_{k=1}^{i-1} \sum_{\substack{\ell=1\\ \ell\neq k}}^{i-1} Z_i^H B Z_k Z_k^H B Z_\ell Z_\ell^H B Z_i, \end{aligned}$$

which reflects the behavior of single-vector weak Gram-Schmidt with $(i-1)$ terms with two Gram matrices and $(i-1)(i-2)$ terms with three Gram matrices. Adjusting the expression for the estimated norm with respect to m for each column v_i of

V_i yields

$$\left|1 - \|v_i\|^2\right| < (i-1)ma^2 + \left(i^2 - 3i + 2\right)m^2a^3$$

where plugging in the Gram factor limit from Equation (5.9) for i vector blocks leads to a quick approach of unity when increasing i or m. The vectors V_i are again not normalized, but similar to the behavior of single-vector weak Gram-Schmidt, it can be seen from the above that normalization plays a minor role in virtually all cases if the input vectors Z_i can be assumed to be normalized.

The above matches the expression from the introductory part of this section for $i = 2$, but, of course, assumes constant orthogonality a of all involved blocks. We will estimate the intra-orthogonality of V_i in a similar manner, looking at the off-diagonal entries. Without any additional cancellation we also obtain

$$(i-1)ma^2 + \left(i^2 - 3i + 2\right)m^2a^3,$$

which, for $i = 2$, again matches the statement from before and otherwise is identical to the norm deviation.

In orthogonalization schemes where unfinished blocks of vectors are used, such as butterfly schemes, the above considerations apply as a worst-case scenario for $i = m$ and $j = m - 1$; due to partially orthogonalized blocks, i.e., blocks that have been orthogonalized against some but not all other blocks, we may expect slightly improved results.

5.3.12 Butterfly results

We repeat the orthogonalization of the data sets from the test set using the butterfly approach with worst-case residual ordering. The results are shown in Tables 5.14 and 5.15. From Table 5.14, it is immediately apparent that a single sweep cannot reduce the orthogonality to machine precision. Particularly difficult are the matrices `SiH4` and `linverse`. Without locking, the result is better, but for the latter matrix still far from optimal. This difficult scenario is created by the choice of interval boundaries by the number of eigenpairs based on the precomputed spectrum that can easily cut dense clusters. Basing the choice of intervals simply on equal ranges on the spectrum disregarding the number of eigenvalues, on the other hand, does not produce this problem here, as apparent from Table 5.15. Of course, cutting no clusters cannot be guaranteed in this case either, but it is less probable. In terms of residual loss, the results are on par with the results from block Gram-Schmidt, see Tables 5.8 and 5.9. The residual loss is even identical in both cases, which comes as no surprise. While the order of the different orthogonalizations is not the same, the direction, i.e., which interval is orthogonalized against which, is identical. We further can assume that the nearest-neighbor interaction plays the dominant role for residual disturbances as inter-orthogonalities are the highest. The suboptimal inter-orthogonality makes an additional iteration necessary; on the other hand, in many cases a single iteration did suffice.

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.886e−16	7.241e−15	1.853e−09	+0.099	3.123e−16	1.874e−16	9.852e−10	−0.086
`SiH4`	7.787e−17	1.106e−11	7.528e−08	−0.026	6.592e−17	4.682e−14	4.151e−08	−0.194
`linverse`	1.264e−16	6.841e−07	4.725e−07	+0.130	1.718e−20	1.620e−10	1.487e−08	−1.226
`Pres_Poisson`	2.348e−16	2.762e−13	2.149e−08	+0.188	2.235e−16	2.016e−14	1.034e−08	−0.110
`Si5H12`	4.077e−17	1.159e−15	5.572e−09	+0.143	5.117e−17	3.383e−17	1.418e−09	−0.282
`brainpc2`	4.760e−16	5.897e−14	1.440e−08	+0.353	3.570e−16	8.612e−16	5.058e−09	−0.058
`rgg_n_2_15_s0`	3.980e−16	8.519e−14	4.252e−08	+0.241	3.027e−16	3.873e−16	1.284e−08	−0.170
`SiO`	5.486e−17	1.839e−15	2.054e−08	+0.359	5.562e−17	4.337e−17	2.775e−09	−0.293
`Andrews`	2.342e−17	4.955e−14	3.686e−08	+0.162	2.564e−17	4.744e−16	9.060e−09	−0.416
`Si34H36`	3.036e−17	5.891e−15	1.901e−08	+0.018	3.296e−17	1.603e−16	6.377e−09	−0.307
`fe_rotor`	7.633e−17	5.798e−13	5.138e−08	+0.117	7.720e−17	2.981e−15	1.595e−08	−0.327
`GraI-119k`	1.336e−16	2.326e−14	3.808e−08	−0.164	1.622e−16	1.649e−15	2.480e−08	−0.350

Table 5.14: *Butterfly orthogonalization, single sweep. Sort method:* ***worst-case residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***even*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.816e−16	8.705e−15	2.138e−09	+0.178	2.290e−16	2.359e−16	4.723e−10	−0.459
`SiH4`	5.985e−17	3.934e−14	1.300e−09	+0.166	6.939e−17	4.929e−14	3.819e−09	−0.329
`linverse`	5.772e−17	7.070e−12	3.829e−09	+0.046	4.103e−17	1.732e−14	1.585e−09	−0.267
`Pres_Poisson`	3.193e−16	6.238e−14	2.508e−08	+0.256	2.845e−16	6.979e−15	7.556e−09	−0.197
`Si5H12`	5.551e−17	5.443e−17	5.714e−09	+0.163	7.947e−17	3.231e−17	1.150e−09	−0.416
`brainpc2`	4.074e−16	2.909e−13	1.014e−08	+0.196	5.282e−16	2.589e−14	3.965e−09	−0.138
`rgg_n_2_15_s0`	2.776e−16	1.382e−13	4.481e−08	+0.268	3.868e−16	9.245e−15	1.748e−08	−0.112
`SiO`	5.833e−17	1.836e−15	1.957e−08	+0.340	5.117e−17	4.163e−17	3.041e−09	−0.393
`Andrews`	2.429e−17	4.184e−14	4.851e−08	+0.284	2.613e−17	2.570e−15	1.438e−08	−0.236
`Si34H36`	3.860e−17	5.207e−15	2.302e−08	+0.106	3.209e−17	2.044e−16	7.333e−09	−0.339
`fe_rotor`	6.592e−17	3.891e−13	5.185e−08	+0.118	7.893e−17	1.679e−14	1.814e−08	−0.177
`GraI-119k`	4.559e−16	4.701e−15	2.969e−08	−0.246	4.396e−16	3.806e−16	1.300e−10	−2.225

Table 5.15: *Butterfly orthogonalization, single sweep. Sort method:* ***worst-case residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***uneven*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Judging from Figure 5.15, it might be reasonable to just consider some of the outermost vectors (boundary vectors) when deciding the direction of orthogonalization. We will test this strategy with just the outermost boundary vector and with five of the boundary vectors. This is a case where a simple ordering of intervals is not possible anymore. Already with only three intervals, there may be no ordering ensuring that the interval with large lower boundary residual is orthogonalized against the interval with small upper boundary residual for every interaction, i.e., whenever the graph of interactions contains a directed cycle. This makes it impossible to employ the block Gram-Schmidt algorithm, at least if the ordering should be respected for all interactions. If we only take into account interaction between direct neighbors, however, ordering is possible again and block Gram-Schmidt can be used. Since we expect the largest impact on the residual from nearest neighbor interactions, the results should be comparable.

The boundary residual approach requires the eigenpairs of all intervals to be ordered by eigenvalue position, which is the case if the eigenpairs of an interval are computed en bloc due to the reduced eigensolver typically ordering its result in this manner. Locking, however, may disturb the ordering if locked vectors are moved to the beginning or end of the vector block (there is, of course, the possibility not to move locked vectors and track them through different means, but this is typically not done due to performance considerations). In this case, eigenpairs have to be sorted again before starting the orthogonalization phase.

Tables 5.16 and 5.17 list the results for a single boundary vector and one sweep of the butterfly scheme, Tables 5.18 and 5.19 show the respective results for five boundary vectors. Of course, the results for inter-orthogonality cannot deviate

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	2.914e−16	7.230e−15	1.364e−09	−0.034	2.845e−16	2.235e−16	9.852e−10	−0.086
`SiH4`	5.670e−17	1.096e−11	8.409e−08	+0.022	6.505e−17	4.680e−14	6.464e−08	−0.001
`linverse`	1.357e−16	1.074e−07	4.671e−07	+0.125	3.861e−19	5.365e−09	1.266e−07	−0.296
`Pres_Poisson`	3.754e−16	2.216e−13	2.159e−08	+0.190	2.194e−16	1.116e−14	1.219e−08	−0.039
`Si5H12`	4.640e−17	1.158e−15	4.070e−09	+0.007	4.163e−17	3.556e−17	2.106e−09	−0.110
`brainpc2`	4.519e−16	2.672e−13	1.088e−08	+0.231	4.591e−16	1.622e−14	5.057e−09	−0.058
`rgg_n_2_15_s0`	3.565e−16	1.262e−13	3.888e−08	+0.203	3.131e−16	3.888e−15	1.867e−08	−0.007
`SiO`	5.898e−17	2.023e−15	2.151e−08	+0.379	5.768e−17	3.816e−17	2.775e−09	−0.293
`Andrews`	2.982e−17	5.094e−14	4.136e−08	+0.212	2.602e−17	6.055e−15	2.054e−08	−0.061
`Si34H36`	3.166e−17	4.171e−15	1.900e−08	+0.018	2.906e−17	3.195e−16	6.768e−09	−0.281
`fe_rotor`	7.286e−17	6.180e−13	5.101e−08	+0.113	6.679e−17	6.186e−14	2.945e−08	−0.061
`GraI-119k`	1.804e−16	3.344e−14	4.180e−08	−0.123	1.579e−16	1.661e−15	3.000e−08	−0.267

Table 5.16: *Butterfly orthogonalization, single sweep. Sort method:* ***boundary residuals, single vector****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***even*** *distribution of eigenpairs among* 64 *intervals of equal size.*

significantly from other orderings using a butterfly scheme. The value of the boundary residual ordering therefore hinges on the residual loss. However, some cases show a detrimental effect of these strategies and only few are significantly better. With these mixed results we cannot claim that this method of sorting is generally better than the worst-case residual or worst average residual methods.

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.331e−16	8.374e−15	2.096e−09	+0.170	2.220e−16	7.615e−15	1.585e−09	+0.067
`SiH4`	8.327e−17	3.934e−14	1.300e−09	+0.166	7.980e−17	4.927e−14	8.121e−09	−0.002
`linverse`	5.152e−17	7.070e−12	4.392e−09	+0.106	4.103e−17	1.732e−14	2.866e−09	−0.010
`Pres_Poisson`	3.140e−16	1.054e−13	2.243e−08	+0.207	2.914e−16	6.979e−15	9.649e−09	−0.090
`Si5H12`	4.380e−17	3.567e−17	4.266e−09	+0.036	5.725e−17	3.556e−17	2.958e−09	−0.006
`brainpc2`	4.324e−16	2.909e−13	9.792e−09	+0.180	4.818e−16	2.588e−14	4.514e−09	−0.081
`rgg_n_2_15_s0`	3.517e−16	3.415e−13	5.676e−08	+0.370	3.105e−16	5.146e−14	2.649e−08	+0.069
`SiO`	5.725e−17	4.626e−14	1.569e−08	+0.244	7.459e−17	4.467e−17	4.572e−09	−0.216
`Andrews`	2.412e−17	4.856e−14	4.851e−08	+0.284	2.580e−17	9.639e−15	2.021e−08	−0.088
`Si34H36`	4.077e−17	5.459e−15	2.671e−08	+0.171	3.123e−17	5.606e−16	1.156e−08	−0.141
`fe_rotor`	6.939e−17	4.918e−13	5.188e−08	+0.119	6.765e−17	3.450e−14	2.264e−08	−0.081
`GraI-119k`	4.080e−16	8.729e−15	5.232e−08	+0.000	4.097e−16	4.389e−16	1.374e−08	−0.201

Table 5.17: *Butterfly orthogonalization, single sweep. Sort method:* ***boundary residuals, single vector****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***uneven*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	2.984e−16	9.298e−14	1.853e−09	+0.099	2.845e−16	1.886e−16	9.852e−10	−0.086
`SiH4`	9.486e−17	1.071e−11	9.359e−08	+0.069	8.327e−17	4.680e−14	6.464e−08	−0.001
`linverse`	1.560e−16	2.303e−07	4.670e−07	+0.125	1.718e−20	1.620e−10	1.487e−08	−1.226
`Pres_Poisson`	3.059e−16	2.762e−13	2.159e−08	+0.190	2.437e−16	1.116e−14	1.219e−08	−0.039
`Si5H12`	4.705e−17	2.860e−16	4.181e−09	+0.018	3.881e−17	3.296e−17	2.106e−09	−0.110
`brainpc2`	4.562e−16	1.178e−13	9.045e−09	+0.151	4.398e−16	1.620e−14	5.057e−09	−0.058
`rgg_n_2_15_s0`	4.241e−16	8.385e−14	3.888e−08	+0.203	3.400e−16	4.770e−15	1.867e−08	−0.007
`SiO`	5.117e−17	1.708e−15	2.054e−08	+0.359	6.072e−17	3.816e−17	2.775e−09	−0.293
`Andrews`	2.580e−17	5.094e−14	4.136e−08	+0.212	2.841e−17	6.058e−15	2.054e−08	−0.061
`Si34H36`	3.188e−17	2.709e−15	1.900e−08	+0.018	3.730e−17	1.622e−16	6.377e−09	−0.307
`fe_rotor`	6.375e−17	6.788e−13	5.138e−08	+0.117	7.112e−17	6.186e−14	2.945e−08	−0.061
`GraI-119k`	1.613e−16	3.344e−14	4.180e−08	−0.123	1.492e−16	1.463e−14	3.915e−08	−0.151

Table 5.18: *Butterfly orthogonalization, single sweep. Sort method:* ***boundary residuals, five vectors****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***even*** *distribution of eigenpairs among* 64 *intervals of equal size.*

5.3.13 Regular residual updates

The orthogonalization ordering schemes introduced above all rely on the residual (typically the worst-case residual) of the involved intervals. We have seen before that the residual is affected by orthogonalization interactions. If the residual is to be the deciding factor, it seems natural that the residual has to be updated whenever the respective interval is to be used as orthogonality provider, but only if the interval

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.747e−16	8.230e−15	1.765e−09	+0.095	3.123e−16	2.533e−16	7.635e−10	−0.250
`SiH4`	6.288e−17	3.933e−14	1.300e−09	+0.166	7.980e−17	3.292e−14	8.121e−09	−0.002
`linverse`	5.152e−17	7.070e−12	4.821e−09	+0.147	4.103e−17	1.732e−14	2.866e−09	−0.010
`Pres_Poisson`	3.959e−16	6.238e−14	2.508e−08	+0.256	2.914e−16	3.682e−15	9.649e−09	−0.090
`Si5H12`	4.337e−17	9.498e−17	5.274e−09	+0.128	4.554e−17	3.383e−17	2.958e−09	−0.006
`brainpc2`	5.044e−16	2.908e−13	9.794e−09	+0.181	6.574e−16	2.588e−14	4.514e−09	−0.081
`rgg_n_2_15_s0`	3.350e−16	3.415e−13	5.213e−08	+0.333	4.293e−16	5.146e−14	2.649e−08	+0.069
`SiO`	4.857e−17	4.627e−14	1.407e−08	+0.197	7.286e−17	4.163e−17	4.572e−09	−0.216
`Andrews`	2.277e−17	4.585e−14	4.851e−08	+0.284	2.602e−17	9.639e−15	2.021e−08	−0.088
`Si34H36`	3.079e−17	5.456e−15	2.671e−08	+0.171	3.469e−17	5.612e−16	1.296e−08	−0.092
`fe_rotor`	6.852e−17	5.564e−13	5.188e−08	+0.119	6.679e−17	3.450e−14	2.615e−08	−0.018
`GraI-119k`	4.900e−16	9.436e−15	5.232e−08	+0.000	4.727e−16	4.958e−16	1.374e−08	−0.201

Table 5.19: *Butterfly orthogonalization, single sweep. Sort method:* ***boundary residuals, five vectors****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***uneven*** *distribution of eigenpairs among* 64 *intervals of equal size.*

order is not decided a priori. Therefore, block Gram-Schmidt will, of course, not yield different results when including residual updates. For an unordered butterfly scheme, however, additional recomputations of the residual may be necessary as each interval may serve as orthogonality provider at any stage.

The final residual achieved after orthogonalization is a crucial output of the overall eigensolver and thus has to be recomputed for any orthogonalization scheme under all circumstances when the orthogonalization phase has finished. For ordered schemes using the block Gram-Schmidt algorithm, this would result in a total of $p-1$ recomputations. The very first interval (in the respective ordering) is not modified and the residual does not change.

We repeat the butterfly scheme with worst-case residual as orthogonalization condition for all data sets, including residual recomputation after every modification of a vector block. The results are listed in Tables 5.20 and 5.21.

Comparing Table 5.20 and Table 5.21 with the original results from Tables 5.14 and 5.15, the changed ordering of intervals in the stages of the butterfly orthogonalization procedure have only little effect on residual loss. In cases where the residual is not improved, the loss of residual is slightly lower in some of the cases. There is no case where an originally positive residual loss becomes a negative loss, though. Considering the cost of many recomputations of the residual, omitting these intermediate steps appears reasonable.

5.3.14 Skipping intra-orthogonalization

In order to serve as orthogonality provider in an inter-orthogonalization interaction, a block of vectors needs to be approximately intra-orthogonal. The effect of orthog-

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.400e−16	7.279e−15	1.681e−09	+0.057	2.776e−16	1.978e−16	9.852e−10	−0.086
`SiH4`	6.787e−17	1.106e−11	7.528e−08	−0.026	7.893e−17	4.682e−14	4.151e−08	−0.194
`linverse`	1.560e−16	2.303e−07	4.725e−07	+0.130	1.718e−20	1.620e−10	1.839e−08	−1.134
`Pres_Poisson`	2.922e−16	2.761e−13	2.149e−08	+0.188	2.514e−16	2.016e−14	1.034e−08	−0.110
`Si5H12`	5.421e−17	1.158e−15	4.070e−09	+0.007	4.391e−17	3.263e−17	1.683e−09	−0.208
`brainpc2`	5.369e−16	5.898e−14	1.061e−08	+0.220	5.128e−16	8.175e−16	5.058e−09	−0.058
`rgg_n_2_15_s0`	3.344e−16	1.200e−13	4.252e−08	+0.241	3.747e−16	7.637e−15	1.346e−08	−0.149
`SiO`	6.852e−17	2.020e−15	2.054e−08	+0.359	5.551e−17	4.163e−17	2.775e−09	−0.293
`Andrews`	2.364e−17	5.720e−14	3.722e−08	+0.167	2.580e−17	4.773e−16	9.060e−09	−0.416
`Si34H36`	3.816e−17	4.280e−15	2.123e−08	+0.066	3.361e−17	1.640e−16	6.377e−09	−0.307
`fe_rotor`	6.505e−17	6.789e−13	5.138e−08	+0.117	7.459e−17	2.976e−15	1.595e−08	−0.327
`GraI-119k`	1.756e−16	3.344e−14	4.180e−08	−0.123	1.716e−16	1.657e−15	2.480e−08	−0.350

Table 5.20: *Butterfly orthogonalization, single sweep, residual update after each modification. Sort method:* ***worst-case residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***even*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	3.400e−16	8.404e−15	2.138e−09	+0.178	2.741e−16	2.177e−16	4.723e−10	−0.459
`SiH4`	5.768e−17	3.934e−14	1.300e−09	+0.166	6.852e−17	4.926e−14	3.819e−09	−0.329
`linverse`	5.772e−17	7.070e−12	3.829e−09	+0.046	4.103e−17	1.732e−14	1.585e−09	−0.267
`Pres_Poisson`	3.509e−16	1.054e−13	2.386e−08	+0.234	2.684e−16	6.979e−15	7.556e−09	−0.197
`Si5H12`	4.770e−17	5.031e−17	4.463e−09	+0.056	4.380e−17	3.426e−17	1.150e−09	−0.416
`brainpc2`	6.068e−16	2.908e−13	1.014e−08	+0.196	5.148e−16	1.127e−14	2.271e−09	−0.380
`rgg_n_2_15_s0`	3.331e−16	1.082e−13	4.239e−08	+0.244	3.469e−16	1.048e−14	1.748e−08	−0.112
`SiO`	7.980e−17	1.836e−15	1.957e−08	+0.340	5.117e−17	3.990e−17	3.041e−09	−0.393
`Andrews`	2.537e−17	4.183e−14	4.851e−08	+0.284	2.640e−17	5.743e−15	1.843e−08	−0.128
`Si34H36`	5.291e−17	5.211e−15	1.910e−08	+0.025	4.250e−17	8.327e−17	4.972e−09	−0.507
`fe_rotor`	6.592e−17	5.564e−13	5.185e−08	+0.118	8.587e−17	1.679e−14	1.814e−08	−0.177
`GraI-119k`	5.821e−16	1.853e−14	2.602e−08	−0.303	4.396e−16	3.224e−16	1.300e−10	−2.225

Table 5.21: *Butterfly orthogonalization, single sweep, residual update after each modification. Sort method:* ***worst-case residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***uneven*** *distribution of eigenpairs among* 64 *intervals of equal size.*

onalizing a vector block Z against a vector block Y where Y is not orthogonal in terms of orthogonality between $V = Z - Y\left(Y^H BZ\right)$ and Y may be described as

$$\begin{aligned} V^H BY &= \left(Z - Y\left(Y^H BZ\right)\right)^H BY \\ &= Z^H BY - \left(Z^H BY\right)\left(Y^H BY\right) \\ &= Z^H BY\left(I - Y^H BY\right). \end{aligned}$$

If we assume Y is normalized,

$$\operatorname{orth}(V, Y) \leq \frac{\operatorname{orth}(Z, Y)\operatorname{orth}(Y)}{\min_i \|z_i\| \|y_i\|}.$$

where z_i are the columns of Z and y_i the columns of Y.

Nonetheless, in all preceding experiments using a butterfly scheme, intra-orthogonalization of modified vector blocks is only performed at the end of the sweep. We therefore compare the preceding results with results that include an intra-orthogonalization after every modification of a vector block. Only a single sweep is applied and the worst-case residual ordering serves as orthogonalization condition. Tables 5.22 and 5.23 summarize the results for all data sets.

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	1.943e−16	7.307e−15	1.853e−09	+0.099	2.359e−16	2.637e−16	9.852e−10	−0.086
`SiH4`	4.163e−17	1.106e−11	7.528e−08	−0.026	4.727e−17	4.681e−14	4.151e−08	−0.194
`linverse`	4.011e−17	6.841e−07	4.725e−07	+0.130	5.630e−20	1.620e−10	1.487e−08	−1.226
`Pres_Poisson`	2.630e−16	2.762e−13	2.149e−08	+0.188	2.220e−16	2.016e−14	1.034e−08	−0.110
`Si5H12`	3.784e−17	1.168e−15	5.572e−09	+0.143	4.033e−17	4.424e−17	1.418e−09	−0.282
`brainpc2`	4.257e−16	5.896e−14	1.440e−08	+0.353	5.471e−16	8.890e−16	5.058e−09	−0.058
`rgg_n_2_15_s0`	3.296e−16	8.518e−14	4.252e−08	+0.241	2.776e−16	3.881e−16	1.284e−08	−0.170
`SiO`	6.592e−17	1.841e−15	2.054e−08	+0.359	5.605e−17	6.272e−17	2.775e−09	−0.293
`Andrews`	2.179e−17	4.955e−14	3.686e−08	+0.162	2.185e−17	4.729e−16	9.060e−09	−0.416
`Si34H36`	2.819e−17	5.895e−15	1.901e−08	+0.018	3.231e−17	1.617e−16	6.377e−09	−0.307
`fe_rotor`	6.549e−17	5.798e−13	5.138e−08	+0.117	6.679e−17	2.975e−15	1.595e−08	−0.327
`GraI-119k`	1.509e−16	2.326e−14	3.808e−08	−0.164	1.416e−16	1.653e−15	2.480e−08	−0.350

Table 5.22: *Butterfly orthogonalization, single sweep, intra-orthogonalization after each modification. Sort method:* ***worst-case residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***even*** *distribution of eigenpairs among* 64 *intervals of equal size.*

Comparing the numbers from Tables 5.22 and 5.23 with the respective numbers from Tables 5.14 and 5.15, we see that the values for the residual loss are exactly the same. Similarly, the values for residual and orthogonality are, wherever they are not just numerically irrelevant noise, identical. This result is somewhat counter-intuitive, as we expect vector blocks with large intra-orthogonalities not to be good orthogonality providers. On the other hand, the severity of all cases from the test set is attenuated since suspected duplicates are removed beforehand following Section 5.3.9. This leads to less disturbances during the several inter-orthogonalizations and, in turn, to less disturbances of intra-orthogonality.

Since repeated intra-orthogonalizations constitute a significant portion of the computations, being able to skip them in many cases increases the feasibility of the butterfly scheme. The intra-orthogonalization effort then may be reduced to once at the end of each sweep. In severe cases, it might be enough to add an

Name	locking				no locking			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`laser`	1.943e−16	8.673e−15	2.138e−09	+0.178	2.082e−16	2.984e−16	4.723e−10	−0.459
`SiH4`	5.768e−17	3.933e−14	1.300e−09	+0.166	3.990e−17	4.926e−14	3.819e−09	−0.329
`linverse`	4.688e−17	7.070e−12	3.829e−09	+0.046	3.123e−17	1.732e−14	1.585e−09	−0.267
`Pres_Poisson`	3.293e−16	6.238e−14	2.508e−08	+0.256	1.457e−16	6.979e−15	7.556e−09	−0.197
`Si5H12`	3.036e−17	5.660e−17	5.714e−09	+0.163	4.597e−17	5.139e−17	1.150e−09	−0.416
`brainpc2`	4.047e−16	2.909e−13	1.014e−08	+0.196	3.934e−16	2.588e−14	3.965e−09	−0.138
`rgg_n_2_15_s0`	2.884e−16	1.382e−13	4.481e−08	+0.268	3.626e−16	9.239e−15	1.748e−08	−0.112
`SiO`	4.445e−17	1.837e−15	1.957e−08	+0.340	5.551e−17	7.069e−17	3.041e−09	−0.393
`Andrews`	2.055e−17	4.183e−14	4.851e−08	+0.284	2.233e−17	2.571e−15	1.438e−08	−0.236
`Si34H36`	3.123e−17	5.205e−15	2.302e−08	+0.106	2.949e−17	2.050e−16	7.333e−09	−0.339
`fe_rotor`	5.855e−17	3.891e−13	5.185e−08	+0.118	7.112e−17	1.679e−14	1.814e−08	−0.177
`GraI-119k`	4.037e−16	4.695e−15	2.969e−08	−0.246	4.152e−16	3.793e−16	1.300e−10	−2.225

Table 5.23: *Butterfly orthogonalization, single sweep, intra-orthogonalization after each modification. Sort method:* ***worst-case residual****. Maximum residual, maximum orthogonalities, and residual loss for the matrix test set with* ***uneven*** *distribution of eigenpairs among* 64 *intervals of equal size.*

intra-orthogonalization phase after the first stage, since we expect nearest-neighbor interactions to apply the most severe disturbance.

5.3.15 Reduction of interaction distance

Examining the cases where the butterfly scheme performed the worst, the `linverse` and `SiH4` matrices, the main cause for the poor performance are large orthogonalities among nearest neighbors, as is to be expected. It is the also clear that large remaining orthogonalities after a butterfly sweep is caused by the same pairing; we could say that a butterfly sweep contracts large orthogonality towards closer neighbors. Consequentially, not all pairings need to be considered in following sweeps. For the sake of discussion, we will refer to intervals that are k-th neighbors as being k *hops* apart.

As we have seen already in Figure 4.32 from Section 4.5.3.1, the initial situation for the `linverse` matrix is dire: the maximum number of hops is required, if inter-orthogonality has to be brought close to machine precision. Figure 5.19 shows the orthogonalities computed from the `linverse` matrix after one sweep of the butterfly scheme with worst-case residual ordering. In all cases but the (even, no locking) data set, a second sweep would only have to consider nearest neighbors.

In order to optimize distances for following sweeps, if a full sweep is performed as first step, the computed orthogonalities during the orthogonalizations can be used to estimate new orthogonalities that would be achieved with this sweep, for example using the (pessimistic) method from Section 5.3.11. The basic assumption of equally bad orthogonalities among all vectors of all blocks is, however, way too pessimistic and the estimation cannot be used. Since we assume to know at least all

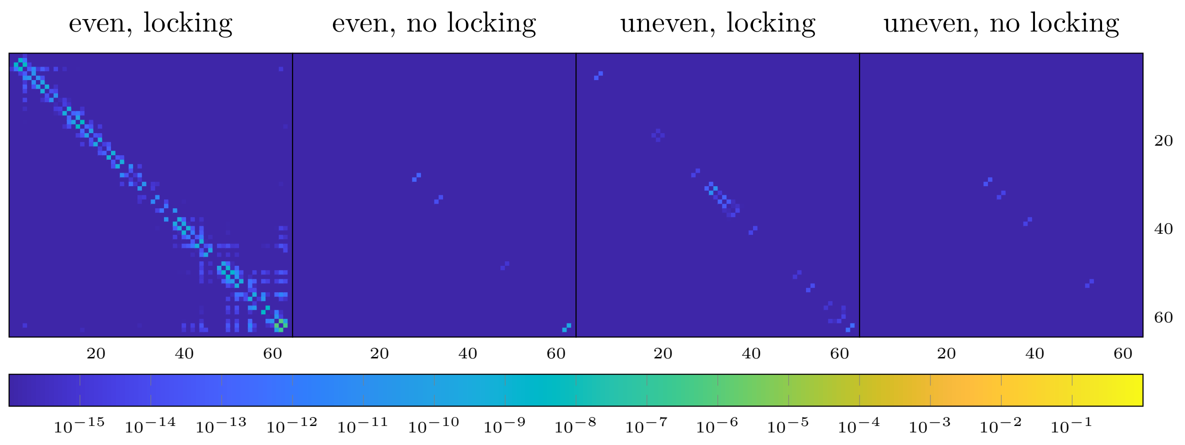


Figure 5.19: *Orthogonalities of the `linverse` matrix after one sweep. Shown are the worst orthogonalities of all possible interval combinations.*

maximum inter-orthogonalities among all intervals, a more precise estimation may be performed, based on the real orthogonalities between a given interval and all other intervals. We modify Equation (5.8) such that each vector block is orthogonalized against every other vector block. We obtain

$$\begin{aligned}V_j^H B V_i &= Z_j^H B Z_i - \sum_{\substack{k=1\\k\neq j}}^{p} Z_j^H B Z_k Z_k^H B Z_i - \sum_{\substack{k=1\\k\neq i}}^{p} Z_j^H B Z_k Z_k^H B Z_i \\ &\quad + \sum_{\substack{k=1\\k\neq j}}^{p} \sum_{\substack{\ell=1\\\ell\neq i}}^{p} Z_j^H B Z_k Z_k^H B Z_\ell Z_\ell^H B Z_i \\ &= - \sum_{\substack{k=1\\k\neq i,j}}^{p} Z_j^H B Z_k Z_k^H B Z_i + \sum_{\substack{k=1\\k\neq j}}^{p} \sum_{\substack{\ell=1\\\ell\neq i,k}}^{p} Z_j^H B Z_k Z_k^H B Z_\ell Z_\ell^H B Z_i,\end{aligned}$$

which we estimate, assuming that no values cancel, all Z_k are orthonormal, and every Gram matrix $Z_j^H B Z_i$ consists of a fixed Gram factor $\operatorname{orth}(Z_i, Z_j)$, as

$$\begin{aligned}&\sum_{\substack{k=1\\k\neq i,j}}^{p} m_k \operatorname{orth}(Z_j, Z_k) \operatorname{orth}(Z_k, Z_i) \\ &+ \sum_{\substack{k=1\\k\neq j}}^{p} \sum_{\substack{\ell=1\\\ell\neq i,k}}^{p} m_k m_l \operatorname{orth}(Z_j, Z_k) \operatorname{orth}(Z_k, Z_\ell) \operatorname{orth}(Z_\ell, Z_i), \qquad (5.10)\end{aligned}$$

where m_k is the number of vectors of the k-th vector block. Of course, in the several stages of a full butterfly sweep, we obtain orthogonalities of already modified vector blocks and we assume here that the inter-orthogonality among two intervals is not changed, in particular not improved, by orthogonalization of one of those two intervals with a third interval. Figure 5.20 shows the estimation for the four data

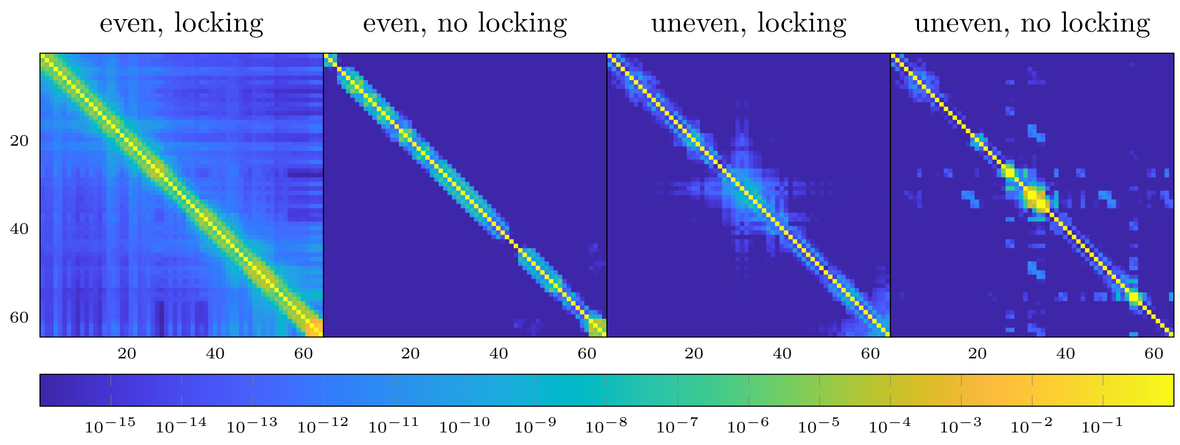


Figure 5.20: *Estimated orthogonalities of the `linverse` matrix for one sweep. Shown are the worst orthogonalities of all possible interval combinations.*

sets of the `linverse` matrix. This way, distance reduction could be automated and it would even be possible to not only skip interactions beyond a certain limit, but all interaction, near or far, if the prediction advises to exclude them. The results are not particularly close to the actually achieved result. In particular duplicate values in the (uneven, no locking) case skew the results and would have to be eliminated beforehand. This, however should normally be the case anyway. While some orthogonalities even increase, applying the estimator from Equation (5.10) multiple times eventually reduces all predicted residuals to machine precision. The required (estimated) number of sweeps also does not reflect actual results. A better estimator, maybe using additional information gathered during the first sweep, will have to be found in order to improve the reduction of hops for additional sweeps.

Ultimately, nearest-neighbor orthogonalization might be cheap enough to consider intermediate orthogonalization between iterations of the eigensolver again, enabling the characteristic gain in convergence speed (with properly chosen direction of orthogonalization) and simultaneous mitigation of the most significant source of large inter-orthogonalities could be worth the additional cost in terms of operations, communication, and synchronization points.

5.3.16 Large examples

For the two large cases from Section 4.1.2, only a single data set exists; calibration runs were performed with locking and even subdivision of the target interval into 32 sub-intervals of equal size. Since no spectral information beyond the density estimation is available, searchspace sizes were determined by adaptive increase, as described in Section 4.2.1.1. Due to its inefficiency in extremely dense spectral situations such as these, instead of a polynomial filter, a Cauchy filter of degree 16 was used. For comparison with the following orthogonalization test, the results are given in Table 5.24.

In order to also obtain results for the orthogonalization strategies applied to larger matrices with dense spectrum, we have tested block Gram Schmidt and a single sweep of the butterfly scheme, both with worst-case residual ordering, on the matrices `graph-1M` and `topi-1M` from Section 4.1.2. Cholesky QR has been used for reorthogonalization after the full sweep is completed; residual updates were not performed. The results are listed in Table 5.25. Since the initial inter-orthogonalities

Name	min res	max res	intra orth	inter orth
`graph-1M`	2.62e−12	6.77e−07	9.48e−11	5.68e−03
`topi-1M`	4.98e−11	1.67e−06	1.43e−04	1.61e−01

Table 5.24: *Final residuals and orthogonalities for the large examples. The minimum residual for the graphene case are outliers; most best residuals for the different intervals were in the order of* 10^{-9}. *For the topological insulator case, best residuals were, for the most part, spread out in the range from* 10^{-10} *to* $5 \cdot 10^{-9}$.

Name	block-GS				butterfly(1)			
	intra orth	inter orth	max res	loss	intra orth	inter orth	max res	loss
`graph-1M`	3.331e−16	3.018e−16	3.623e−07	−0.272	2.637e−16	1.803e−07	3.623e−07	−0.272
`topi-1M`	2.152e−16	4.780e−15	1.978e−06	+0.074	1.657e−16	1.004e−06	1.978e−06	+0.074

Table 5.25: *Large a posteriori orthogonalization examples.*

were large, $5.68 \cdot 10^{-3}$ and $1.61 \cdot 10^{-1}$ for `graph-1M` and `topi-1M`, respectively, the butterfly approach does not achieve sufficiently well developed orthogonality in a single sweep. The initial inter-orthogonalities and the inter-orthogonalities after one sweep of the butterfly scheme are shown in Figure 5.21. The matrix `topi-1M` also is an extreme example of large intra-orthogonalities, caused by the locking mechanism; the separation criterion from Section 5.3.9 did not report any duplicates. This is

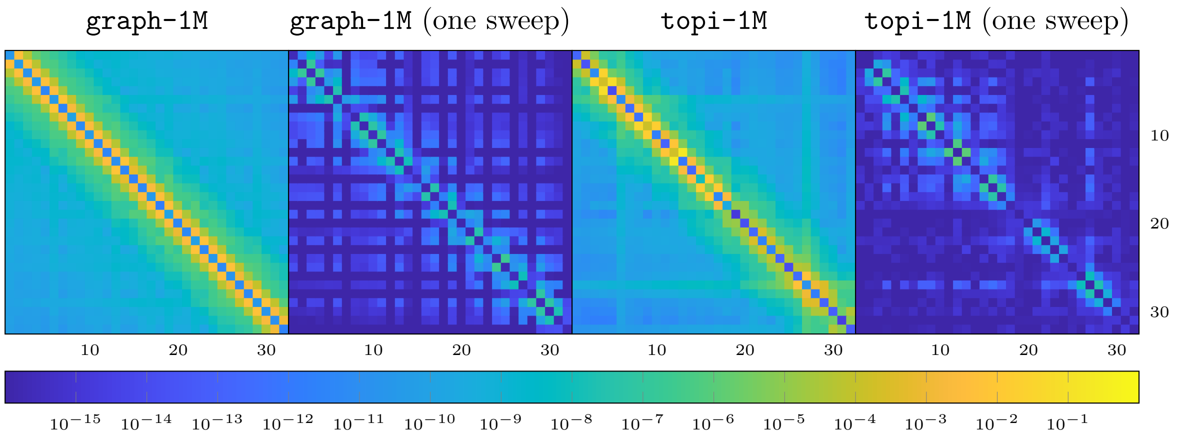


Figure 5.21: *Large matrix orthogonalities and estimations.*

confirmed by the successful orthogonalization using block Gram-Schmidt without much increase in residual.

5.3.17 Shifting scheme

The effect of using butterfly orthogonalization patterns and therefore unfinished vector blocks as orthogonality providers has similar implications as (block) weak Gram-Schmidt and we have to expect that applying the orthogonalization scheme more than once is required. Since the different vector blocks evolve over several stages and thus updated versions of the vectors are used, the result is an orthogonalization scheme that would best be described as *modified (block) weak Gram-Schmidt*, but without the implied mathematical equivalence (as was the case for classical Gram-Schmidt and modified Gram-Schmidt, Section 1.3.2).

An alternative implementation with effects comparable to block weak Gram-Schmidt is a shift pattern similar to Section 5.3.5 but with simultaneous shifts of all intervals as depicted in Figure 5.22. If computation and communication over-

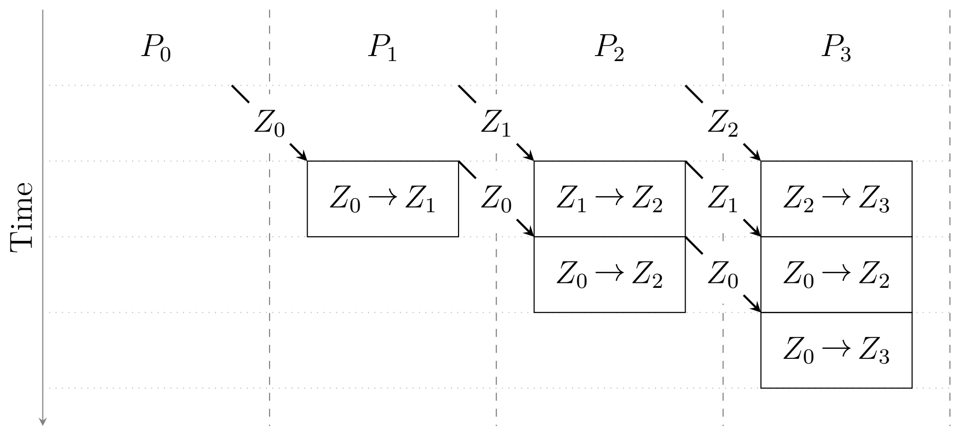


Figure 5.22: *Weak shift orthogonalization scheme.*

lap as shown here, updated vectors are never sent and the scheme is equivalent to weak Gram-Schmidt where all orthogonalizations are based on the original vectors and multiple iterations of this scheme are likely to be necessary. Similar to block Gram-Schmidt, different ordering for the intervals are possible. Waiting for updated vectors to pass them along immediately results in the block Gram-Schmidt shift scheme.

5.3.18 Application-specific requirements

Depending on the problem at hand, different requirements and possibilities for the way orthogonalization has to be or can be carried out may arise. This is mainly caused by differences in spectral composition and the number and position of eigenpairs that are to be computed. The two large matrices from Section 4.1.2 named `graph-1M` and `topi-1M`, for example, have distinct spectra which are representative of the respective problem type. In both cases, typically only few inner eigenpairs are sought and in both cases the spectral density in the center of the spectrum is comparably low. If the number of eigenpairs is low, interval-level parallelism is either not required or only few intervals are needed. The latter is predominantly the case for very large matrices with increased spectral density. In such cases, the inter-orthogonality between all intervals is likely to be affected (see for example the target intervals for the test set, Section 4.1.2 and Section 5.3.16, Figure 5.21; these intervals are not standard target intervals for this problem type). Since the overall orthogonalization effort for few intervals is lower, a full orthogonalization considering all possible interval pairs may be required. This scenario is covered by the orthogonalization schemes from before with little potential for optimization in terms of skipping certain steps based on interval distance or similar. All strategies introduced are more efficient with lower numbers of intervals.

The problem specification may differ from the one described above, though. Consider a case where the matrix size is limited, but large portions of the spectrum should be computed. In order to exploit large computation capacities and since handling large numbers of columns becomes increasingly inefficient, the subdivision into a large number of intervals seems more promising. In this case, the orthogonalization effort requires many steps to consider all interval pairs. With lower spectral density in combination with the possibly large distance among intervals that are not direct neighbors, however, it is more likely that orthogonalizations beyond a certain distance can be skipped altogether. To evaluate this condition was the goal of Section 4.5 and Section 4.6, but a reliable estimation of the orthogonality (without explicitly computing it) which is based solely on the residual and distance of Ritz values and which is not too pessimistic to deliver practical results is not available so far. If, however, an estimation of a minimum distance below of which orthogonalization is required can be made from the problem class alone, assuming similar behavior for all problems of a certain class, new possibilities to perform a more effective orthogonalization open up.

Consider a sequence of k intervals $\mathcal{I}_1, \ldots, \mathcal{I}_k$. If we can consider a certain distance between two intervals $\mathcal{I}_i$ and $\mathcal{I}_j$, i.e., the smallest distance between pairs of Ritz values from different intervals, safe such that no orthogonalization is required beyond this distance, many interactions between distant intervals can now be skipped. Similarly, if the requirements of orthogonality are not that strict and a certain number of orders of magnitude above machine precision is acceptable, interactions can be skipped in the same way. In more strictly ordered schemes such as block Gram-Schmidt, the corresponding orthogonalizations can be omitted, but the overall scheme (Section 5.3.5) has to be completed for all intervals. In particular a shift-based implementation requires passing along received blocks of vectors. For a more flexible scheme, such as the butterfly pattern (Section 5.3.10) which works in order of increasing distance, the complete process can be stopped after a few steps, as soon as the safe distance is surpassed for all interactions of a step. Since butterfly schemes are to be considered more expensive due to multiple sweeps needed, they have the potential to be more effective when only orthogonalizations among close neighbors have to be performed.

Experiment 5.7 — Distance-based restrictions of orthogonalization
We apply the butterfly orthogonalization pattern to all data sets from the test set, using worst-case residual ordering and three sweeps to assure that the minimum possible orthogonality can be reached. The maximum distance up to which an orthogonalization can take place is restricted to different orders of magnitude. Table 5.26 lists the results for distances 10^{-6} to 10^{-1} and the (even, locking) data set. The remaining data sets are covered by Tables 5.27 to 5.29.

Name	10^{-6}		10^{-5}		10^{-4}		10^{-3}		10^{-2}		10^{-1}	
	orthog.	h	orthog.	h	orthog.	h	orthog.	h	orthog.	h	orthog.	h
`laser`	2.74e−06	0	2.74e−06	0	1.86e−06	1	5.68e−10	2	3.08e−14	8	1.63e−15	29
`SiH4`	7.20e−05	1	7.20e−05	1	7.20e−05	1	1.47e−05	1	6.85e−15	6	1.24e−15	44
`linverse`	2.93e−03	1	2.43e−03	1	6.43e−04	1	1.11e−04	5	3.67e−10	17	1.45e−17	63
`Pres_Poisson`	7.23e−05	1	2.98e−05	1	1.17e−05	1	7.75e−09	3	3.05e−15	18	3.05e−16	63
`Si5H12`	3.25e−06	0	3.25e−06	1	3.25e−06	1	5.69e−10	1	3.97e−16	6	2.99e−16	55
`brainpc2`	7.82e−05	1	2.92e−05	1	1.11e−05	1	3.13e−10	4	2.39e−15	22	1.94e−16	63
`rgg_n_2_15_s0`	5.53e−05	1	5.53e−05	1	3.76e−09	1	9.88e−10	2	1.45e−15	13	7.04e−17	63
`SiO`	9.71e−06	1	9.71e−06	1	1.92e−07	1	1.92e−07	1	3.93e−14	6	2.73e−16	54
`Andrews`	2.29e−05	1	2.29e−05	1	1.13e−09	1	1.13e−09	1	1.84e−16	8	1.45e−17	63
`Si34H36`	1.14e−05	1	1.14e−05	1	3.62e−06	1	2.66e−10	1	2.01e−16	7	1.76e−16	58
`fe_rotor`	6.67e−05	1	3.63e−05	1	1.58e−08	1	3.90e−12	2	3.15e−16	14	1.99e−17	63
`GraI-119k`	2.89e−05	0	1.43e−05	1	1.06e−08	1	1.06e−08	2	6.58e−15	8	9.02e−16	51

Table 5.26: *Inter-orthogonalities for different maximum orthogonalization distances for the (even, locking) data set and restricted orthogonalization distances. Also shown is the number of hops (h) required to reach the distance limit.*

With a fixed distance, the maximum number of hops is unknown until after a sweep has finished. It might happen that a number of hops would cover larger distances, but orthogonalization would be rejected due to the distance limit, see, e.g., Table 5.26, line five. Here, one hop covers distances 10^{-5} to 10^{-3} (columns two to four), but in columns two and three, orthogonalizations have been rejected such that the potentially possible orthogonality (column four) for a single hop is not reached. Since the gain can be quite high (about four orders of magnitude in line five), it is advisable to compute interval distances in a preparation step to determine the required number of hops in order to obey the distance limit and consequentially removing the distance limit from the base orthogonalization phase in favor of a hop limit. The number of hops does not differ between data sets with and without locking, but achieved orthogonalities are better, confirming previous results. For uneven distribution of eigenpairs per interval, the number of hops to reach certain levels of orthogonality is reduced due to larger gaps between intervals.

Not shown in the table, but worth noting is the invariance of residual loss, which remains roughly the same for different distances and number of hops, as long as at

Name	10^{-6}		10^{-5}		10^{-4}		10^{-3}		10^{-2}		10^{-1}	
	orthog.	h	orthog.	h	orthog.	h	orthog.	h	orthog.	h	orthog.	h
`laser`	2.30e−06	0	2.30e−06	0	8.63e−07	1	2.28e−11	2	1.51e−15	8	1.52e−15	29
`SiH4`	3.07e−05	1	3.07e−05	1	3.07e−05	1	1.13e−05	1	3.18e−15	6	1.24e−15	44
`linverse`	9.22e−04	1	2.03e−04	1	2.16e−06	1	2.17e−09	5	3.29e−14	17	2.08e−17	63
`Pres_Poisson`	3.71e−05	1	3.45e−05	1	3.44e−06	1	1.56e−10	3	2.05e−15	18	3.58e−16	63
`Si5H12`	1.41e−06	0	1.41e−06	1	9.94e−07	1	2.77e−10	1	4.12e−16	6	2.76e−16	55
`brainpc2`	3.60e−05	1	7.16e−06	1	3.64e−06	1	2.74e−11	4	3.18e−15	22	1.98e−16	63
`rgg_n_2_15_s0`	2.38e−05	1	2.38e−05	1	7.84e−11	1	1.81e−11	2	7.29e−16	13	6.94e−17	63
`SiO`	4.02e−06	1	4.02e−06	1	8.05e−10	1	8.05e−10	1	3.53e−16	6	3.17e−16	54
`Andrews`	2.05e−05	1	2.05e−05	1	6.51e−11	1	6.51e−11	1	1.95e−16	8	1.39e−17	63
`Si34H36`	6.60e−06	1	5.67e−06	1	2.75e−07	1	1.42e−11	1	2.13e−16	7	1.54e−16	58
`fe_rotor`	3.86e−05	1	3.09e−05	1	1.16e−10	1	3.29e−14	2	3.30e−16	14	2.43e−17	63
`GraI-119k`	2.89e−05	0	1.43e−05	1	3.99e−09	1	2.09e−10	2	7.57e−15	8	9.13e−16	51

Table 5.27: *Inter-orthogonalities for different maximum orthogonalization distances for the (even, no locking) data set and restricted orthogonalization distances. Also shown is the number of hops (h) required to reach the distance limit.*

Name	10^{-6}		10^{-5}		10^{-4}		10^{-3}		10^{-2}		10^{-1}	
	orthog.	h	orthog.	h	orthog.	h	orthog.	h	orthog.	h	orthog.	h
`laser`	6.83e−07	0	6.83e−07	0	6.83e−07	1	8.74e−08	1	6.01e−10	3	7.08e−16	28
`SiH4`	1.47e−06	0	1.47e−06	0	1.47e−06	1	3.86e−07	1	1.34e−14	5	1.04e−15	43
`linverse`	6.91e−05	0	1.10e−05	1	2.56e−06	1	2.36e−06	1	8.35e−11	10	1.26e−17	63
`Pres_Poisson`	3.61e−05	0	3.61e−05	1	1.40e−05	1	9.42e−09	1	3.67e−15	10	2.94e−16	63
`Si5H12`	2.96e−06	0	2.96e−06	0	2.33e−06	1	5.65e−11	1	3.53e−16	6	2.95e−16	55
`brainpc2`	2.89e−05	0	2.89e−05	1	1.38e−05	1	3.78e−10	2	3.35e−15	13	1.95e−16	63
`rgg_n_2_15_s0`	5.18e−05	0	5.18e−05	1	4.86e−09	1	5.65e−12	2	6.21e−16	12	9.71e−17	63
`SiO`	9.38e−06	0	9.38e−06	1	5.78e−06	1	1.66e−08	1	3.19e−13	6	2.63e−16	53
`Andrews`	2.41e−05	0	2.41e−05	1	1.19e−09	1	1.19e−09	1	1.98e−16	8	1.34e−17	63
`Si34H36`	1.35e−05	0	1.35e−05	1	6.25e−06	1	5.01e−10	1	1.91e−16	6	1.43e−16	58
`fe_rotor`	4.83e−05	1	4.83e−05	1	1.03e−08	1	3.02e−12	2	2.90e−16	12	1.69e−17	63
`GraI-119k`	2.36e−05	0	2.36e−05	1	1.48e−09	1	1.48e−09	1	6.18e−15	5	7.43e−16	47

Table 5.28: *Inter-orthogonalities for different maximum orthogonalization distances for the (uneven, locking) data set and restricted orthogonalization distances. Also shown is the number of hops (h) required to reach the distance limit.*

least nearest neighbors were orthogonalized. This is a confirmation for what has been stated before: most residual disturbances occur due to nearest neighbor interactions.

Experiment 5.8 — Distance and orthogonality

To gain a more fine grained understanding on how certain distances relate to the computed orthogonalities, we plot orthogonalities over distances in Figure 5.24 for selected matrices from the test set. For comparison, the orthogonality matrices are displayed in Figure 5.23.

Name	10^{-6}		10^{-5}		10^{-4}		10^{-3}		10^{-2}		10^{-1}	
	orthog.	*h*	orthog.	*h*	orthog.	*h*	orthog.	*h*	orthog.	*h*	orthog.	*h*
`laser`	7.28e−07	0	7.28e−07	0	7.28e−07	1	8.71e−08	1	1.56e−10	3	7.55e−16	28
`SiH4`	1.90e−05	0	1.90e−05	0	1.90e−05	1	2.96e−06	1	4.70e−14	5	1.10e−15	43
`linverse`	6.59e−05	1	1.83e−06	1	1.69e−06	1	1.46e−06	1	6.55e−14	10	6.10e−18	63
`Pres_Poisson`	3.33e−05	0	3.33e−05	1	1.39e−05	1	1.30e−09	1	2.12e−15	10	2.92e−16	63
`Si5H12`	2.05e−06	0	2.05e−06	0	1.12e−06	1	5.82e−12	1	4.57e−16	6	3.15e−16	55
`brainpc2`	1.35e−05	0	1.35e−05	1	3.62e−06	1	4.80e−11	2	2.68e−15	13	3.06e−16	63
`rgg_n_2_15_s0`	4.70e−05	0	4.70e−05	1	4.45e−10	1	4.77e−14	2	6.46e−16	12	8.67e−17	63
`SiO`	3.18e−06	0	3.18e−06	1	2.02e−06	1	5.30e−11	1	3.31e−16	6	3.06e−16	53
`Andrews`	2.16e−05	0	2.16e−05	1	1.47e−10	1	1.47e−10	1	1.91e−16	8	1.37e−17	63
`Si34H36`	8.81e−06	0	8.81e−06	1	6.74e−06	1	2.14e−11	1	1.90e−16	6	1.44e−16	58
`fe_rotor`	2.48e−05	1	2.25e−05	1	6.44e−10	1	1.04e−14	2	2.92e−16	12	2.30e−17	63
`GraI-119k`	5.46e−06	0	3.85e−07	1	8.25e−14	1	8.25e−14	1	6.16e−15	5	7.22e−16	47

Table 5.29: *Inter-orthogonalities for different maximum orthogonalization distances for the (uneven, no locking) data set and restricted orthogonalization distances. Also shown is the number of hops (h) required to reach the distance limit.*

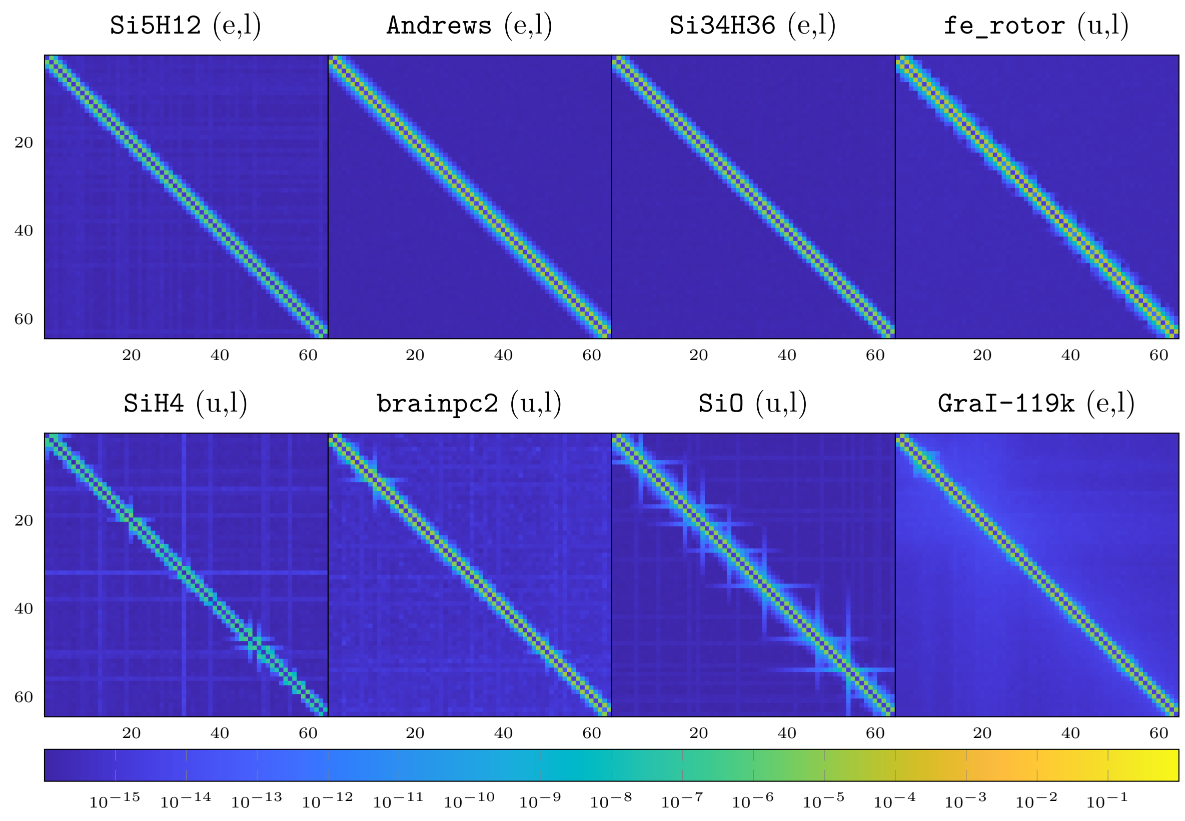


Figure 5.23: *Orthogonalities for some matrices from the test set. Matrices from the (even, locking) set are marked as (e,l); matrices from the (uneven, locking) set are marked (u,l).*

The top row of Figure 5.23 shows cases where distance based restrictions can lead to reasonable levels of overall inter-orthogonality. If the orthogonality between intervals that are a larger number of hops apart fades towards machine precision, and if generally no artifacts in terms of increased orthogonality appear for such interval pairs, assuming that the information of this distance limit is somehow known, reasonable orthogonality could be achieved, c.f. Tables 5.26 to 5.29. The visualization in Figure 5.24 gives an idea of how different distance limits would affect overall orthogonality. Drawing a vertical line at any distance in any of the plots, whatever

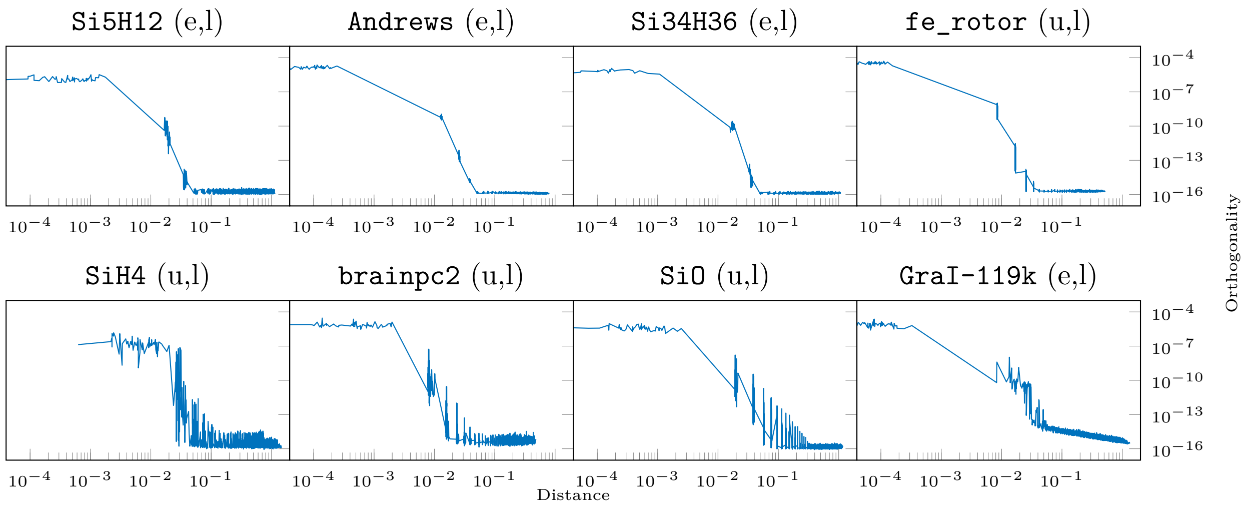


Figure 5.24: *Orthogonalities plotted over distances for the matrices from Figure 5.23.*

remains as maximum orthogonality to the right of this line gives the final maximum inter-orthogonality for this particular case.

For many cases (there are exceptions), orthogonality seems to not improve significantly beyond a distance of 10^{-2}. While the orthogonality is not in the order of machine precision, it might be sufficient in practice. Orthogonalization schemes that rely on distance to sort intervals into classes, such that intervals from the same class are far enough apart to compute them without orthogonalization and subsequently orthogonalize against those known final vectors when computing the next class not only becomes increasingly expensive for larger number of classes but also requires a reliable criterion for a safe distance, which, as we have seen, may only ensure moderate levels of orthogonality. We therefore do not consider it here. In general, the difficulty of acquiring the essential information of which distance is at least required for orthogonalization to become unnecessary remains.

5.3.19 Performance

The analysis of the performance and scaling behavior of the orthogonalization patterns and interval-parallelization in general depends on many factors and shall only be touched upon here briefly. If, with the increase of the number of intervals, also the number of computed eigenpairs increases while matrix size remains the same, interval parallelism should scale perfectly. Despite the system size remaining the same, we would classify this scenario as weak scaling, in comparison to increasing the number of intervals while the number of computed eigenpairs remains the same, which we could consider strong scaling. In the latter case, performance largely depends on whether the applied projector scales well for reduced numbers of right-hand sides, ignoring the overhead introduced by the Rayleigh-Ritz procedure. Since this is no quantity we care to explore here, the analysis is restricted to the a posteriori orthogonalization phase, which we always should consider as requiring a significant portion of the total computation time.

Experiment 5.9 — Orthogonalization performance
Using the `graph-1M` matrix, we test the following two scenarios on an increasing number of processes $p = 4, 8, 16, 32, 64$.

- *A constant total number of vectors* $m_t = 2\,048$ *is computed in* $k = p/2$ *intervals, changing the number of vectors per process with the number of used processes* $m = 1\,024, 512, 256, 128, 64$ *for each test.*
- *An increasing total number of vectors* $m_t = 32p$ *is computed in* $k = p/2$ *intervals, keeping the number of vectors per interval* $m = 64$ *constant.*

Both scenarios were tested with block Gram-Schmidt and one full sweep of the butterfly scheme. For reestablishment of intra-orthogonality, two passes of weak Gram-Schmidt were used. For both block Gram-Schmidt and the butterfly pattern, this happens only once per interval at the end of the orthogonalization phase. Figure 5.25 shows the results.

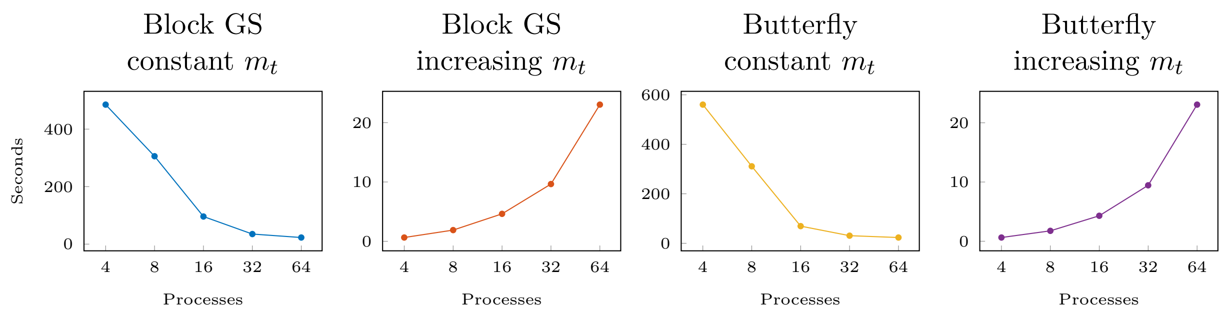


Figure 5.25: *Scaling of orthogonalization schemes. The tests were performed on the Emmy HPC cluster at Friedrich-Alexander-Universität Erlangen-Nürnberg.*

In all scenarios, the number of processes holding one interval or one vector block is constant. This way, the results do not contain the scaling behavior of the inner products in terms of process number, but they do contain the scaling behavior in terms of block size, i.e., the number of columns for each block, which changes with the number of intervals in the first scenario. The plots one and three of Figure 5.25 reflect the decreasing number of vectors per inner product and the decreasing communication volume. In contrast, plots two and four show the behavior with constant communication volume and increasing number of communication steps. The number of inner products increases likewise, but the number of vectors for each inner product remains the same. It is difficult to say if communication or inner products are the dominant factor in these tests, and in practice both factors are always intertwined.

For comparison, the scaling behavior of the inner products with respect to process number is shown in Figure 5.26, where a series of tests with two intervals and constant number of 512 vectors, resulting in five inner products, was conducted.

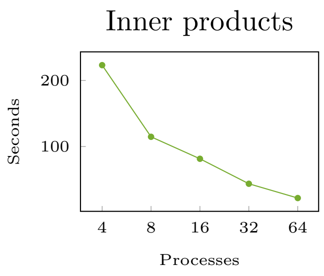


Figure 5.26: *Scaling of inner products. The tests were performed on the Emmy HPC cluster at Friedrich-Alexander-Universität Erlangen-Nürnberg.*

Overall, the scaling of block Gram-Schmidt and the butterfly pattern is comparable, in particular for lower numbers of vectors or larger number of processes; the total number of inner products and messages sent is identical for both orthogonalization patterns.

5.4 Upshot

With all the different variations of a posteriori orthogonalization examined in this chapter, the final verdict remains simple and somewhat sobering. A purely a posteriori orthogonalization approach works reasonably well and if better residuals are produced by the eigensolver, even if only for some vectors of each interval, the orthogonalization procedure is likely to even improve the overall result in terms of residual while still producing a fully orthogonal set of eigenvectors. Intermediate residual updates and reestablishment of intra-orthogonality can be omitted without significant impact on the result but with considerable improvement of performance. Indeed, a low residual seems to be significantly more important to this end. More sophisticated strategies of orthogonalization order have little to no benefit compared to a simple worst-case residual based ordering. Unordered orthogonalization (the butterfly scheme) has similar implications on the orthogonality as weak Gram-Schmidt and is likely to require more than one sweep, making it more expensive than block

Gram-Schmidt if communication does not outweigh computation. If orthogonalization distance can be limited, however, the butterfly scheme might be more efficient if few stages (or hops) are required. The exclusion of certain interactions between intervals is not easily decided without actually computing the orthogonalities, at which point the orthogonalization can be performed anyway since the expensive inner product has already been computed. Considering the cost effectiveness of a one-shot orthogonalization, a prophylactic safety net of prescribing a residual of about half an order of magnitude lower than otherwise specified for the eigensolver to reach, such that it can be sacrificed for orthogonality, is not a bad deal.

Conclusion and outlook

Conclusion

In this thesis, we have introduced a collection of the most typically used filters for spectral projection. Filters from electronic filter design have, to the best of the author's knowledge, not been used in this field, at least not prominently. We apologize for any missing references. The more complex types of these filters are more versatile and allow for more fine-tuning of parameters. More importantly, some of the common filters are special cases of filters from electronic filter design, namely the Cauchy-midpoint filter which is the Butterworth filter while the Zolotarev filter is a special case of elliptic filter (despite the different derivation).

We have confirmed the convergence theorem for subspace iteration algorithms, in particular using the filters introduced previously, and related it to many effects emerging in practical runs of such algorithms. We further have applied the convergence theorem to the estimated filter values obtained from the second Rayleigh quotient; this is not new per se, but the choice of subspace basis in this case allows for a more direct understanding of why the filter values evolve as observed. In particular the at first glance surprising correctness of filter values for the so-called Ritz phantoms could be explained.

Based on these considerations, we further have described many density-related effects on convergence: the emergence of Ritz phantoms and effects of densely clustered eigenpairs and filter values. We also shed some (faint) light on the difficulty of linear system arising from rational approximations of the filtering functions for iterative linear system solvers.

Based on the evolution of the residual and its corner cases, we have postulated the necessity of a stagnation-based termination criterion with details on how to handle the several possible conditions under which other termination criteria might fail. We have also highlighted the importance of a correct eigencount estimation to complement the stagnation-based criterion.

In order to obtain estimations for the achievable inter-orthogonality in a multi-interval context, we first analyzed residual progression and achievable residual under different conditions since the residual of the result is sure to have an impact on inter-orthogonality. We have tested the precision of previously used bounds and derived possible bounds for the inter-orthogonality from them. The tested bounds for inter-orthogonality are pessimistic but at least some of them, based on residual and spectral gap, seem not to break even in difficult situations where the residual that has been reached is still very large.

The ultimate goal of the identification of residual-based bounds for inter-orthogonality is the exclusion of single orthogonalization operations in a full orthogonalization scheme where otherwise all possible interval combinations have to be considered. To this end, analyzing subspace overlap, i.e., shared directions among the sub-spaces of adjacent intervals, seemed promising but ultimately only works if eigenpairs are allowed to reach saturation (or possibly a sufficiently low residual). The condition cannot be used reliably in practice. An analysis of the evolution of inter-orthogonality under different conditions revealed a clear connection to the behavior of the residual, but formulating a more precise estimation of the orthogonality that can be expected at the end of the eigensolver phase is not easily possible.

Expecting an expensive full-fledged orthogonalization phase, we argue for the futility of intermediate orthogonalizations using unfinished vectors and in favor of a purely a posteriori orthogonalization phase where we now aim at minimizing a possibly negative effect on the previously achieved residual. To this end, we have tested many of the better known conventional orthogonalization methods, with respect to their numerical stability and their influence on residual of the affected eigenpairs. We further explored inherently iterative orthogonalization approaches via what we have called weak Gram-Schmidt, and which also may serve as cheap post-processing step for other algorithms to improve results in case of ill-conditioned input matrices.

We then have extended these considerations to blocks of vectors, exploring the effectiveness of different orthogonalization schemes using static and dynamic orderings, block Gram-Schmidt and the butterfly pattern as well as possible optimizations, such as waiving reestablishment of intra-orthogonality in-between orthogonalization stages and residual updates using intermediate vectors, always keeping track of the loss in residual. As a result, the inevitability of overlap between adjacent intervals in order to eliminate the possibility to miss eigenpairs has become apparent and a reliable method to remove duplicates under these circumstances has been presented. The success of the method is limited by the achieved residual and the ability to separate eigenpairs based on it.

A short portion of this work was devoted to give an overview over the accompanying software framework named BEAST. Over time, BEAST has evolved into a feature-rich extensive iterative spectral projection eigensolver software framework, capable of solving very large scale sparse Hermitian generalized eigenproblems on state-of-the-art hybrid parallel high performance supercomputers making use of many improvements in terms of performance, reliability and robustness. Although it has not found much coverage here, many of the aspects described herein, and more, were integral to its implementation.

Finally, we want to mention the identification of the assumed filter representation for the Sakurai-Sugiura method (SSM). Even if the filter form could just be derived experimentally, we expect it to shed more light on the convergence behavior of the SSM in an iterative context.

Outlook

All the filters presented here can be considered somewhat conservative; they approximate the window function. More unconventional filters or employing rational function fitting may allow to move the poles away from the spectrum in order to make the use of iterative linear system solver more feasible. The identification of a suitable iterative linear solver remains an important task, though advancements in these directions have been made elsewhere. The field of electronic filter design has only been scratched here and may offer further filters, methodology, and improvements.

For the estimated filter values obtained from the second Rayleigh quotient a way to estimate the residual of the secondary implicit Rayleigh-Ritz process is desirable to judge the accuracy of filter values and eigencount. So far, an additional full application of the approximate projector is required. It might be worthwhile trying to append at least some vectors of the respective subspace basis to the conventional right-hand sides to this end. Better understanding of the convergence of the Sakurai-Sugiura method, based on the new assumed filter representation, may allow for the identification of thresholds for eigencount estimations akin to what has been established for the filters discussed here.

Many aspects introduced here leave potential for further optimization. The ordering of intervals, for example, plays a role in the impact of orthogonalization on the residual. Certainly, an ordering that improves the residual in all cases is impossible; further improvements, however, cannot be ruled out. The termination criterion introduced here also is certainly not the final word on termination criteria.

An important aspect of multi-interval computations is the assessment of workload distribution. The KPM serves as possible source for this information, but is limited to standard eigenproblems or requires many linear solves involving the matrix B. Other approaches require linear system solves of the same kind as emerges from the core algorithm. A system that can globally self-adjust after one iteration, based on the obtained preliminary information, may be feasible.

Finally, the BEAST software will be improved further, using some of the insight gained here. Technical improvements will be made, allowing multiple prides to run the algorithms in order to increase potential for parallelism and use the available memory more efficiently. The framework already offers functionality that may benefit other algorithms that compute eigenpairs and not necessarily do so in similar fashion. The subdivision into and distribution of sub-intervals, for example, is essentially separated from the algorithmic core and may be used by algorithms which support multi-subset computation in principle. Methods that compute eigenpairs close to a specific shift, like Jacobi-Davidson or other shift-invert strategies, are primary candidates and may thus also profit from methods for duplicate removal. This brings us to another core functionality of BEAST that may be employed elsewhere: the establishment of overall orthogonality for independently computed sets of eigenpairs in a post processing stage. Since this is a pure a posteriori method, further interaction between eigensolver and orthogonalization stage is not necessary. Relaying certain information from solver to orthogonalization stage can, of course, improve results.

Appendix A

Algorithms for elliptic functions

A.1 Computation of Jacobi's elliptic functions

Algorithms for computing Jacobi's elliptic functions are based on transformations of the defining incomplete elliptic integrals to smaller or larger moduli, which is to allow the application of approximations that are only valid under certain conditions. The following approximations exist and are easily computed [AS74]. Let $\kappa \approx 0$, then

$$\operatorname{sn}(\nu, \kappa) \approx \sin(\nu), \quad \operatorname{cn}(\nu, \kappa) \approx \cos(\nu), \quad \text{and} \quad \operatorname{dn}(\nu, \kappa) \approx 1.$$

Similarly, let $\kappa \approx 1$. Then

$$\operatorname{sn}(\nu, \kappa) \approx \tanh(\nu), \quad \operatorname{cn}(\nu, \kappa) \approx \operatorname{sech}(\nu), \quad \text{and} \quad \operatorname{dn}(\nu, \kappa) \approx \operatorname{sech}(\nu).$$

Transforming the problem such that these approximations become applicable allows for the computation of Jacobi's elliptic functions. A transformation for increasing or decreasing the modulus κ is introduced as *Landen transformations* in [AS74]. A distinct alternative transformation is given in [Olv+10] and we will adopt the distinction here. Recall the elliptic integral of the first kind from Equation (2.6), this time in angular form,

$$F(\kappa \setminus \phi) = \int_0^{\phi} \frac{1}{\sqrt{1 - \kappa^2 \sin^2(\theta)}}\, d\theta.$$

It is

$$F(\kappa \setminus \phi) = \frac{2}{1+\kappa} F\left(\frac{2\sqrt{\kappa}}{1+\kappa} \setminus \psi\right) \quad \text{if} \quad \sin(2\psi - \phi) = \kappa \sin\phi.$$

This is the *ascending Landen transformation*. Similarly, the *descending Landen transformation* is given as

$$F(\kappa \setminus \phi) = \frac{1}{1+\kappa'} F\left(\frac{1-\kappa'}{1+\kappa'} \setminus \psi\right) \quad \text{if} \quad \tan(\psi - \phi) = \kappa' \tan\phi.$$

The following very similar transformations are attributed to Gauss and hence are called *ascending* and *descending Gauss transformation*, respectively. It is

$$F(\kappa \setminus \phi) = \frac{1}{1+\kappa}\, F\left(\frac{2\sqrt{\kappa}}{1+\kappa} \setminus \psi\right) \quad \text{if} \quad \sin\psi = \frac{(1+\kappa)\sin\phi}{1+\kappa\sin^2\phi}$$

and

$$F(\kappa \setminus \phi) = \frac{2}{1+\kappa'}\, F\left(\frac{1-\kappa'}{1+\kappa'} \setminus \psi\right) \quad \text{if} \quad \sin\psi = \frac{(1+\kappa')\sin\phi}{1+\sqrt{1-\kappa^2\sin^2\phi}}.$$

It is easy to confirm that the new moduli are indeed inverses of each other,

$$\frac{2}{1+\kappa'} = 1 + \frac{1-\kappa'}{1+\kappa'}$$

and thus

$$\frac{2\sqrt{\frac{1-\kappa'}{1+\kappa'}}}{1+\frac{1-\kappa'}{1+\kappa'}} = \frac{(1+\kappa')\sqrt{1-\kappa'}}{\sqrt{1+\kappa'}} = \sqrt{(1+\kappa')(1-\kappa')} = \kappa.$$

All of the above offers in total four ways for the computation of $\operatorname{sn}(\nu,\kappa) = \sin\phi$. We decide for methods reducing the modulus since the approximations involves simpler trigonometric functions and because they are more common in the literature, see, e.g., [Ken05; Wac00]. Algorithms based on Landen transformations also require the manual differentiation between the principal branch of inverse trigonometric functions and their actual value [Wac00]. We will therefore employ the Gauss transformations instead, similar to [Orf05].

Let the modulus κ be decreased in a sequence of moduli $\kappa = \kappa_0, \dots, \kappa_\eta \approx 0$ such that after η steps $F(\kappa_\eta \setminus \phi) \approx F(0 \setminus \phi) = \phi$. This involves repeated descending Gauss transformation of ν, say $\nu = \nu_0, \dots, \nu_\eta \approx \phi$ as

$$\nu_{j+1} = \frac{1}{1+\kappa_{j+1}}\nu_j \quad \text{or} \quad \nu_\eta = \nu_0 \prod_{j=1}^{\eta} \frac{1}{1+\kappa_j} \quad \text{where} \quad \kappa_{j+1} = \frac{1-\kappa'_j}{1+\kappa'_j}. \tag{A.1}$$

It is then $\phi_\eta \approx \nu_\eta$ for modulus κ_η. A sequence of ascending Gauss transformations for $\phi = \phi_0, \dots, \phi_\eta \approx \nu_\eta$ gives the desired result $\operatorname{sn}(\nu,\kappa) = \sin\phi$ via

$$\sin\phi_{j-1} = \frac{(1+\kappa_j)\sin\phi_j}{1+\kappa_j\sin^2\phi_j}.$$

It should be noted that transformation steps involving κ_η can of course be skipped since $\nu_\eta \approx \nu_{\eta-1}$ and $\sin\phi_\eta \approx \sin\phi_{\eta-1}$. The functions $\operatorname{cn}(\nu,\kappa)$ and $\operatorname{dn}(\nu,\kappa)$ are readily available from the computed $\operatorname{sn}(\nu,\kappa)$ such that no separate algorithm is required.

A.1.1 The Algebraic-Geometric Mean

While the algorithm above is complete, the strong relation between the transformations of elliptic integrals outlined before and the sequence of algebraic and geometric means of two numbers is worth mentioning, seeing that most implementations rely on this relation [Ken05; Wac00]. Given two numbers a_0 and b_0, the sequences

$$a_j = \frac{1}{2}(a_{j-1} + b_{j-1}) \quad \text{and} \quad b_j = \sqrt{a_{j-1}b_{j-1}}$$

converge rapidly [AS74]. The limit, denoted $\mathrm{AGM}(a_0, b_0)$, is known as the *algebraic-geometric mean* of a_0 and b_0. Its relation to elliptic transformations was discovered by Gauss. It is easy to see that[1]

$$\mathrm{AGM}(a,b) = \mathrm{AGM}\left(\frac{a+b}{2}, \sqrt{ab}\right) = \frac{a+b}{2}\,\mathrm{AGM}\left(1, \frac{2\sqrt{ab}}{a+b}\right).$$

We see that b_j/a_j always yields the next larger modulus in accordance with the ascending transformations if the scheme was started with $a_0 = 1$ and $b_0 = \kappa$,

$$\mathrm{AGM}(1,\kappa) = \frac{1+\kappa}{2}\,\mathrm{AGM}\left(1, \frac{2\sqrt{\kappa}}{1+\kappa}\right).$$

If κ increases, its complementary modulus κ' decreases accordingly,

$$\kappa' = \sqrt{1 - \left(\frac{2\sqrt{\kappa}}{1+\kappa}\right)^2} = \sqrt{\frac{(1+\kappa)^2 - 4\kappa}{(1+\kappa)^2}} = \sqrt{\frac{1-2k+\kappa^2}{(1+\kappa)^2}} = \frac{1-\kappa}{1+\kappa}.$$

In order to obtain decreasing moduli, we therefore start the scheme with $a_0 = 1$ and $b_0 = \kappa'$ such that κ decreases in the process. Let $\kappa'_0, \ldots, \kappa'_\eta$ be the sequence of increasing complementary moduli and $\kappa_0, \ldots, \kappa_\eta$ the corresponding sequence of decreasing moduli. Then, with $\kappa'_n \approx 1$,

$$a_\eta = \mathrm{AGM}(1,\kappa'_0) = \frac{1+\kappa'_0}{2}\,\mathrm{AGM}(1,\kappa'_1) = \prod_{j=0}^{\eta-1} \frac{1+\kappa'_j}{2}\,\mathrm{AGM}\big(1,\kappa'_\eta\big) \approx \prod_{j=0}^{\eta-1} \frac{1+\kappa'_j}{2}.$$

We recognize this as the transformation factor in Equation (A.1) since

$$\frac{1+\kappa'_j}{2} = \frac{1}{1 + \frac{1-\kappa'_j}{1+\kappa'_j}} = \frac{1}{1+\kappa_{j+1}}.$$

We may obtain the next modulus κ in every step as $(a_{j-1} - b_{j-1})/2a_j$ since

$$\frac{a_{j-1} - b_{j-1}}{2a_j} = \frac{a_{j-1} - b_{j-1}}{a_{j-1} + b_{j-1}} = \frac{1-\kappa'_{j-1}}{1+\kappa'_{j-1}} = \kappa_j.$$

The procedure can be extended to apply to complete elliptic integrals. We know that

$$F\left(\kappa_\eta \setminus \frac{\pi}{2}\right) = K(\kappa_\eta) \approx \frac{\pi}{2}$$

[1] Generally $\mathrm{AGM}(ca, cb) = c\,\mathrm{AGM}(a,b)$ since $\frac{1}{2}(ca_{j-1} + cb_{j-1}) = ca_j$ and $\sqrt{c^2 a_{j-1}b_{j-1}} = cb_j$.

and the application of ascending Gauss transformations yields

$$K(\kappa_0) \approx \frac{\pi}{2} \prod_{i=1}^{\eta} (1+\kappa_j) = \frac{\pi}{2a_\eta}.$$

Therefore the complete elliptic integral of modulus κ can conveniently be computed as a byproduct of the evaluation of $\mathrm{sn}(\nu,\kappa)$. Due to this circumstance and having the arguments to elliptic functions often formulated in terms of multiples of the period K, e.g., [MLB; Orf05], an alternative definition of $\mathrm{sn}(\nu,\kappa)$ that implies multiplication of the argument with K, $\omega = \mathrm{sn}(\nu K,\kappa) = \mathrm{sn}_\mathrm{K}(\nu,\kappa)$, is convenient and often allows to perform computations without the explicit knowledge of K. The modification can be introduced into the computation as multiplication of ν_η with $\pi/2$. The procedure is terminated in step η if $\kappa_\eta \approx 0$ respectively $\kappa'_\eta \approx 1$ or $a_\eta \approx b_\eta$. In a numerical sense this refers to introducing a limit for the deviation, conventionally chosen as machine epsilon. We summarize the procedure derived so far in Algorithm A.1. The result is similar to the one found in [MLB], which is based on [Orf05] but does not use the algebraic-geometric mean. Proper implementation of this algorithm requires only

Input: Argument ν, modulus κ, tolerance t
Output: $\omega \approx \mathrm{sn}_K(\nu,\kappa) = \mathrm{sn}(\nu K,\kappa)$, $K \approx K(\kappa)$

```
1: $a_0 \leftarrow 1$
2: $b_0 \leftarrow \sqrt{1-\kappa^2}$                          ▷ $\kappa'$
3: $\kappa_0 \leftarrow \kappa$
4: for $j = 1, \ldots$ do
5:     $a_j \leftarrow \frac{1}{2}(a_{j-1} + b_{j-1})$
6:     $b_j \leftarrow \sqrt{a_{j-1} b_{j-1}}$
7:     $\kappa_j \leftarrow \frac{a_{j-1} - b_{j-1}}{2a_j}$
8:     if $|a_j - b_j| \le t$ then break
9: $K \leftarrow \frac{\pi}{2a_j}$
10: $\omega_j \leftarrow \sin\left(\frac{\pi}{2} a_j \nu\right)$              ▷ implicit $K$
11: for $\ell = j, \ldots, 1$ do
12:     $\omega_{\ell-1} \leftarrow \frac{(1+\kappa_\ell)\omega_\ell}{1+\kappa_\ell \omega_\ell^2}$
```

Algorithm A.1: *Jacobi elliptic function* sn_K.

the storage of the sequence $\kappa_0, \ldots, \kappa_j$. The expression of κ_j in terms of a_{j-1} and b_{j-1} allows for a recursive implementation that does not require to store the sequence of moduli [Ken05]. With

$$1 + \kappa_{\eta-j} = 1 + \frac{a_{j-1} - b_{j-1}}{a_{j-1} + b_{j-1}} = \frac{2a_{j-1}}{a_{j-1} + b_{j-1}}$$

the ascending Gauss transformation for the angle becomes

$$\sin\phi_j = \frac{2a_{j-1}\sin\phi_{j-1}}{a_{j-1}+b_{j-1}+(a_{j-1}-b_{j-1})\sin^2\phi_{j-1}}.$$

A.2 Computation of inverse elliptic functions

An algorithm for the computation of $\mathrm{sn}^{-1}(\omega,\kappa)$ is outlined in [AS74] and refined as well as described in more detail in [Wac00]. By definition, elliptic functions are inverses of incomplete elliptic integrals. The computation of inverse elliptic functions therefore merely amounts to the computation of incomplete elliptic integrals, which again relies on the transformations described in Section A.1.

Given is now $\omega = \mathrm{sn}(\nu,\kappa) = \sin\phi$ and ν is sought. Let again the sequence of descending moduli be $\kappa = \kappa_0,\dots,\kappa_\eta \approx 0$ and corresponding ascending complementary moduli $\kappa' = \kappa'_0,\dots,\kappa'_\eta \approx 1$. We obtain a sequence $\omega = \omega_0,\dots,\omega_\eta$ via descending Gauss transformations, which we may also express in terms of only moduli or complementary moduli,

$$\omega_{j+1} = \frac{\left(1+\kappa'_j\right)\omega_j}{1+\sqrt{1-\kappa_j^2\omega_j^2}} = \frac{2}{1+\kappa_{j+1}}\,\frac{\omega_j}{1+\sqrt{1-\kappa_j^2\omega_j^2}} = \frac{\left(1+\kappa'_j\right)\omega_j}{1+\sqrt{1-\left(1-\kappa_j'^2\right)\omega_j^2}}.$$

Then $\omega_\eta = \sin\phi_\eta = \sin\nu_\eta$ or $\nu_\eta = \sin^{-1}\omega_\eta$. Using ascending Gauss transformations gives the sequence $\nu_\eta,\dots,\nu_0 = \nu$,

$$\nu_{j-1} = (1+\kappa_j)\nu_j = \nu_\eta\prod_{j=1}^{\eta}(1+\kappa_j) = \frac{1}{a_\eta}\nu_\eta.$$

The implicit multiplication of the argument ν with K such that $\nu = 1/K\,\mathrm{sn}^{-1}(\omega,\kappa) = \mathrm{sn_K}^{-1}(\omega,\kappa)$ may again be included by letting $\nu_\eta = 2/\pi\sin^{-1}\omega_\eta$ with the same reasoning as before. As was the case for Algorithm A.1, the algorithm for $\mathrm{sn_K}^{-1}$ can be based on the algebraic-geometric mean. It is summarized in Algorithm A.2. The inverse $\mathrm{cd_K}^{-1}(\omega,\kappa)$ is then found as $1-\mathrm{sn_K}^{-1}(\omega,\kappa)$. Note that the implicit division by K cancels the multiplication of $\sin^{-1}\omega_\eta$ with a_η. Otherwise, the last line of the algorithm reads $\nu \leftarrow a_\eta\sin^{-1}\omega_\eta$. The implementation of Algorithm A.2 does not require storing any of the sequences involved.

This procedure also clarifies the connection between the elliptic rational function and Chebyshev polynomials. For $\kappa \to 0$ it is $\mathrm{sn}(\nu,\kappa) \approx \sin\nu$ and consequentially $\mathrm{cd}(\nu,\kappa) = \mathrm{sn}(\nu+K,\kappa) \approx \sin(\nu+K)$. At the same time $K \approx \pi/2$, such that $\mathrm{cd}(\nu,\kappa) \approx \sin(\nu+\pi/2) = \cos\nu$. Similarly, $\mathrm{cd}^{-1}(\omega,\kappa) = K - \mathrm{sn}^{-1}(\omega,\kappa) \approx \pi/2 - \sin^{-1}\omega = \cos^{-1}\omega$. Therefore, if $\kappa \to 0$ and $\kappa_1 \to 0$, the elliptic rational function becomes a Chebyshev polynomial. The resulting algorithm is again similar to the one found in [MLB; Orf05], but uses the algebraic-geometric mean and is based on the computation of sn instead of cd.

```
Input: Argument $\omega$, modulus $\kappa$, tolerance $t$
Output: $\nu \approx \frac{1}{K}\,\mathrm{sn}^{-1}(\omega,\kappa) = \mathrm{sn_K}^{-1}(\omega,\kappa),\ K \approx K(\kappa)$
 1: $a_0 \leftarrow 1$
 2: $b_0 \leftarrow \sqrt{1-\kappa^2}$                                ▷ $\kappa'$
 3: $\kappa'_0 \leftarrow b_0$
 4: $\omega_0 \leftarrow \omega$
 5: for $j = 1, \ldots$ do
 6:     $a_j \leftarrow \frac{1}{2}(a_{j-1} + b_{j-1})$
 7:     $b_j \leftarrow \sqrt{a_{j-1} b_{j-1}}$
 8:     $\omega_j \leftarrow \dfrac{\left(1+\kappa'_{j-1}\right)\omega_{j-1}}{1+\sqrt{1-\left(1-\kappa'^2_{j-1}\right)\omega^2_{j-1}}}$
 9:     $\kappa'_j \leftarrow \dfrac{b_j}{a_j}$
10:     if $\left|1-\kappa'_j\right| \le t$ then break
11: $K \leftarrow \dfrac{\pi}{2a_j}$
12: $\nu \leftarrow \frac{2}{\pi}\sin^{-1}\omega_j$                       ▷ implicit $K$
```

Algorithm A.2: *Inverse elliptic function* $\mathrm{sn_K}^{-1}$.

Index

orthogonality provider, 213
orthogonality recipient, 213

alpha, 100
Arnoldi method, 32

B-orthogonal, 4
B-singular value, 196
back end library, 96
band, 39
BEAST, 95
Butterworth filter, 71

canonical angle, 175
Cauchy filter, 50
Cauer filter, 71
characteristic function, 69
characteristic polynomial, 3
Chebyshev filter, type I, 71
Chebyshev filter, type II, 71
Chebyshev polynomial, 42
Cholesky decomposition, 195
cluster, 115
coalition, 98
companion matrix, 86
contour integration, 50
convergence, 121
convergence reference value, 122
CRAFT, 110
critical entry, 236
critical overlap, 178
cutoff frequency, 70

dampening, 26
decay, 115
defective, 10
definite pair, 6
degeneracy, 3
diagonalizable, 9
disjoint, 178
dominance, 7, 19, 116
drop rate, 115
duplicate, 234

e-notation, 119
eigenbasis, 9
eigencount, 116
eigendecomposition, 8
eigenpair, 3
eigenproblem, 3
eigenproblem, generalized, 3
eigenproblem, standard, 3
eigenspace, 3
eigenvalue, 3
eigenvector, 3
Elliptic filter, 71
elliptic integral, complete, 58
elliptic integral, incomplete, 58
elliptic rational function, 78
ELPA, 96
ELPA-AEO, 96
ESSEX, 96

faction, 98
FEAST, 95
filtering function, 26
floating point, 160
following, 101

gain, 38
gap, 173, 182
Gauss-Legendre quadrature, 51
Gaussian quadrature, 51
GHOST, 96
Gibbs phenomenon, 45
Gram factor, 201
Gram-Schmidt, 16

Gram-Schmidt, modified, 17
Gram-Schmidt, rearranged, 17

half-point, 72
Hankel matrix, 86
harmonic Rayleigh-Ritz, 22
Hermitian, 6
hop, 250
Horner's method, 93
Householder reflection, 192
hpd, 6

idempotent, 23
indicator entry, 238
inter-orthogonality, 174
interaction, 174
intra-orthogonality, 174
invariant subspace, 18
inverse iteration, 7
iterate, 7
iteration, 7
iterative Gram-Schmidt, 200

Jacobi elliptic function, 58

kick-off residual, 166
Krylov subspace, 32

Lanczos method, 33
left inverse, 5
locality, 179

machine precision, 160
matrix function, 25
matrix pencil, 3
metric, 28
midpoint rule, 52
modulus, 58, 78
moment, 85
multiplicity, 3
multiplicity, algebraic, 3
multiplicity, geometric, 3

Newton-Cotes formula, closed, 51
Newton-Cotes formula, open, 51
norm, matrix, 5
norm, vector, 5

occupation, 115
offending entry, 236
orthogonal, 4
orthogonal iteration, 20, 31
orthogonal projection, 15
orthogonal projector, 23
orthogonality, 4, 113, 173
orthogonality matrix, 236
orthogonalization, 16
orthonormal, 5
orthonormalization, 16
overlap, 177

pack, 99
passband, 38
PHIST, 96
positive definite, 6
positive semi-definite, 6
power iteration, 7
pride, 97
problematic entry, 236
projector, 23
projector, approximate, 26
projector, reduced representation, 23

QR algorithm, 30
QR decomposition, 30, 191
QR decomposition, thin, 191
quadrature, 51
quarter period, 78
quasi-eigenspace, 145
quasi-multiplicity, 145

rank deficiency, 128
Rayleigh quotient, 20
rayleigh quotient, 14
rayleigh quotient iteration, 14
rebound, 150
residual, 29
residual loss, 231
residue, 84
resolvent, 50
ripple factor, 69

Ritz pair, 20
Ritz phantom, 20, 126, 130, 139, 140
Ritz values, 20
Ritz vector, 20
Ritz vector, primitive, 20

saturation, 115, 167
saturation residual, 115
scalar product, 4
Schur form, 10
searchspace, 114
searchspace basis, 114
searchspace set, 114
searchspace size, 114
searchspace vectors, 114
selection function, 56
shift-inverse iteration, 7
shifted iteration, 8
similar matrices, 9
similarity transform, 9
simultaneous iteration, 20
singular value, 196
singular value decomposition, 196
slot, 114, 122, 139, 144, 145
smoothing kernel, 45
SNC, 171
spectral projection, 2
spectral projector, 24
spectral range, 47
spectrum, 3, 26
SPPEXA, 96
stopband, 38
subspace iteration, 20

tamer, 99
target searchspace, 177
theta function, 82
transition band, 39
trapezoidal rule, 51
TSQR, 197

underrepresented, 144
unitary, 5

Vandermonde matrix, 87

weak Gram-Schmidt, 201
window function, 25
worst-case orthogonality, 174, 186, 208
worst-case residual, 167, 176, 208

List of Algorithms

1.1 Generic power iteration. 7
1.2 Gram-Schmidt orthonormalization (classical and modified) 16
1.3 Modified Gram-Schmidt orthonormalization, reordered. 17
1.4 Power subspace iteration, standard form. 21
1.5 Power subspace iteration, generalized form. 22
1.6 Orthogonal iteration 31
1.7 The Arnoldi algorithm. 32

5.1 Block modified Gram-Schmidt orthonormalization, reordered. 222

A.1 Jacobi elliptic functions. 270
A.2 Inverse elliptic functions. 272

List of Experiments

4.1 Test set calibration . . . 119
4.2 Residual drop rates . . . 122
4.3 On-the-fly increase of searchspace size . . . 124
4.4 Residual drop rates and searchspace size . . . 126
4.5 Slot swapping . . . 129
4.6 Filter induced rank deficiency . . . 130
4.7 Effective multi-moment filter . . . 135
4.8 Movement of Ritz phantoms . . . 140
4.9 Ritz phantom disturbances . . . 141
4.10 Convergence of filter values . . . 143
4.11 Convergence of clustered values with under-representation . . . 146
4.12 Clustered super convergence . . . 149
4.13 Projector disturbances . . . 161
4.14 Influence of Ritz phantoms . . . 163
4.15 Mixed precision . . . 164
4.16 Effect of spectral density on kick-off residual . . . 166
4.17 Variation in spectral position . . . 168
4.18 Variation in spectral range . . . 169
4.19 Interval gap and searchspace occupation (ideal filter) . . . 176
4.20 Interval gap and searchspace occupation (non-ideal) . . . 178
4.21 Overlap and ideal orthogonality . . . 179
4.22 Overlap-based exclusion . . . 181
4.23 Quality of orthogonality bounds . . . 184
4.24 Evolution of orthogonality under different conditions . . . 185
4.25 Evolution of orthogonality with increasing distance . . . 186

5.1 Influence of local reorthogonalization on the residual . . . 208
5.2 Intermediate orthogonalization . . . 214
5.3 Intermediate orthogonalization with uneven convergence . . . 217
5.4 Residual disturbances of continuous orthogonalizations . . . 218
5.5 Single orthogonalization . . . 219
5.6 Influence of inter-orthogonalization on the residual . . . 228
5.7 Distance-based restrictions of orthogonalization . . . 256
5.8 Distance and orthogonality . . . 257
5.9 Orthogonalization performance . . . 260

List of Figures

1.1 Effect of applying a three-dimensional matrix pencil. 9
1.2 Complex spectral inversion. 14
1.3 Effect of applying a three-dimensional projector. 24
1.4 Window function and complex filter function. 25
1.5 Eigenvalue algorithms overview. 34

2.1 Filter properties. 39
2.2 Chebyshev filtering functions with smoothing kernels. 46
2.3 Example of a Chebyshev polynomial based filter. 47
2.4 Discrete Chebyshev filtering functions. 49
2.5 Example of a Cauchy filter. 56
2.6 Cauchy filter with different integration schemes. 57
2.7 Visualization of the Möbius transformation. 61
2.8 Example of a Zolotarev filter. 69
2.9 Filter design parameters. 70
2.10 Example of a Butterworth filter. 74
2.11 Example of a type-I Chebyshev filter. 76
2.12 Example of a type-II Chebyshev filter. 77
2.13 Example of an elliptic filter. 81
2.14 Spectral filtering functions for the moments 0 to 3, absolute value. . 91

3.1 Subdivision into block rows. 97
3.2 Subdivision into blocks of right-hand sides. 97
3.3 Individually computable poles. 98
3.4 Exemplary interval subdivision. 99
3.5 Process group hierarchy. 100
3.6 Simplified dynamic interval parallelism protocol. 102
3.7 Block-wise transformation of banded matrix. 106
3.8 Reduction to smaller system. 106
3.9 Scaling behavior of the banded solver. 108

4.1 Low resolution density of states for graph-1M and topi-1M. 120
4.2 Residual drop rates. 123
4.3 Residual history and drop rates for searchspace size increase. 125
4.4 Residual drop rate by searchspace size. 127
4.5 Slot swapping. 129
4.6 Development of the rank of the searchspace set. 130
4.7 SSM effective filters for the midpoint rule. 137

4.8 SSM effective filters for the Gauss-Legendre rule. 137
4.9 SSM moments (absolute values). 138
4.10 Paths of Ritz phantoms. 141
4.11 Disturbance of Ritz values by nearby Ritz phantoms. 142
4.12 Convergence of approximate filter values. 144
4.13 Convergence of clustered values with under-representation. 146
4.14 Convergence of clustered values with under-representation. 147
4.15 Convergence of clustered values with full representation. 149
4.16 Filter stopband deviations. 151
4.17 Comparison of filters of degree 16. 156
4.18 Filter parameter modification. 156
4.19 Difficulty of an exemplary linear system. 158
4.20 Difficulty of linear systems. 159
4.21 Difficulty of linear systems (zoom). 159
4.22 Residual barriers. 162
4.23 Filter disturbances. 163
4.24 Phantom disturbances of terminal residuals. 164
4.25 Mixed precision results. 165
4.26 Kick-off residual for different spectral ranges and densities. 167
4.27 Saturation residual for variations of the spectral center. 168
4.28 Saturation residual for variations of the spectral range. 169
4.29 Gap influence for different searchspace combinations. 177
4.30 Gap influence for different searchspace combinations. 178
4.31 Influence of different types of overlap on inter-orthogonality. 180
4.32 Orthogonalities produced by the linverse matrix. 183
4.33 Interval distances of the `linverse` matrix. 183
4.34 Orthogonality bounds for `linverse`(even). 184
4.35 Inter-orthogonality of two independently iterated intervals. 186
4.36 Inter-orthogonality of two intervals with increasing distance. 187
4.37 Inter-orthogonality of two intervals with decreasing gap. 187

5.1 Orthogonalization schemes. 190
5.2 TSQR binary tree scheme with four processes. 198
5.3 Single step of the TSQR sequential scheme. 199
5.4 Minimum required initial orthogonality for weak Gram-Schmidt. . . 202
5.5 Gram factor improvement of weak Gram-Schmidt. 203
5.6 Minimum required orthogonality to reach machine precision. 204
5.7 Orthogonality improvement of weak Gram-Schmidt. 205
5.8 The effect of local (intra-)reorthogonalization. 209
5.9 Orthogonality with and without intermediate orthogonalization. . . 216
5.10 Effects of orthogonalization between intervals with different progress. 218
5.11 Evolution of orthogonality with single orthogonalization. 220
5.12 Block Gram-Schmidt shift scheme. 223
5.13 Block Gram-Schmidt broadcast scheme. 224

5.14 The effect of orthogonalization on eigenvector angles. 226
5.15 Inter-orthogonalization. 229
5.16 Selected inter-orthogonalities of the `SiH4` matrix. 237
5.17 Duplicate detection: orthogonality patterns. 238
5.18 Butterfly orthogonalization pattern. 241
5.19 Orthogonalities of the linverse matrix after one sweep. 251
5.20 Estimated orthogonalities of the linverse matrix for one sweep. . . . 252
5.21 Large matrix orthogonalities and estimations. 253
5.22 Weak shift orthogonalization scheme. 254
5.23 Orthogonalities for some matrices from the test set. 258
5.24 Orthogonalities plotted over distances. 259
5.25 Scaling of orthogonalization schemes. 260
5.26 Scaling of inner products. 261

List of Tables

1.1 Power iteration flavors and methods of application. 8
1.2 Matrix pencils of power iteration flavors. 13

2.1 Smoothing kernels. 45
2.2 Filter design parameters. 70

4.1 Standard eigenproblem test set “The Dirty Dozen”. 118
4.2 Final residuals for the four data sets of the matrix test set. 120
4.3 Cauchy filter deviations and resulting transition bands. 151
4.4 Best residual drop rates. 153
4.5 Worst residual drop rates. 154
4.6 Minimum pole distances to the real axis. 154
4.7 Subspace overlap evaluation. 181

5.1 Method name abbreviations. 209
5.2 Exclusion criteria and stability bounds. 211
5.3 Serial and parallel cost of orthogonalization algorithms. 212
5.4 Maximum residual and orthogonalities for the test set (part I). . . 230
5.5 Maximum residual and orthogonalities for the test set (part II). . . 231
5.6 Block Gram-Schmidt orthogonalization (native, even). 232
5.7 Block Gram-Schmidt orthogonalization (native, uneven). 232
5.8 Block Gram-Schmidt orthogonalization (worst-case res., even). . . 233
5.9 Block Gram-Schmidt orthogonalization (worst-case res., uneven). . 233
5.10 Block Gram-Schmidt orthogonalization (average res., even). 234
5.11 Block Gram-Schmidt orthogonalization (average res., uneven). . . 234
5.12 Direct comparison of sorting methods (even). 235
5.13 Direct comparison of sorting methods (uneven). 235
5.14 Butterfly orthogonalization (worst-case residual, even). 244
5.15 Butterfly orthogonalization (worst-case residual, uneven). 244
5.16 Butterfly orthogonalization (boundary res., single vector, even). . . 245
5.17 Butterfly orthogonalization (boundary res., single vector, uneven). 246
5.18 Butterfly orthogonalization (boundary res., five vectors, even). . . 246
5.19 Butterfly orthogonalization (boundary res., five vectors, uneven). . 247
5.20 Butterfly orthogonalization (residual update, even). 248
5.21 Butterfly orthogonalization (residual update, uneven). 248
5.22 Butterfly orthogonalization (intra-orthogonalization, even). 249
5.23 Butterfly orthogonalization (intra-orthogonalization, uneven). . . . 250
5.24 Final residuals and orthogonalities for the large examples. 253

5.25 Large a posteriori orthogonalization examples. 253
5.26 Maximum orthogonalization distances (even, locking). 256
5.27 Maximum orthogonalization distances (even, no locking). 257
5.28 Maximum orthogonalization distances (uneven, locking). 257
5.29 Maximum orthogonalization distances (uneven, no locking). 258

Bibliography

[Agr+] N. Agrawal et al. *Boost Math Toolkit.* Version 2.6.0. URL: https://www.boost.org.

[Agu+20] E. Agullo et al. *MUMPS: a parallel sparse direct solver.* Version 5.3.1. 04/2020. URL: http://mumps.enseeiht.fr.

[Ahl79] L. V. Ahlfors. *Complex Analysis.* 3rd. McGraw-Hill, Inc., 1979.

[Akh56] N. I. Akhiezer. *Theory of Approximation.* 2nd. New York, USA: Frederick Ungar Publishing Co., 1956.

[Akh90] N. I. Akhiezer. *Elements of the Theory of Elliptic Functions.* Vol. 79. Translations of Mathematical Monographs. Providence, Rhode Island, United States: American Mathematical Society, 1990.

[Alv+14] A. Alvermann et al. „ESSEX: Equipping Sparse Solvers for Exascale". In: *Euro-Par 2014: Parallel Processing Workshops.* Ed. by L. Lopes et al. Cham: Springer International Publishing, 2014, pp. 577–588.

[Alv+19] A. Alvermann et al. „Benefits from using mixed precision computations in the ELPA-AEO and ESSEX-II eigensolver projects". In: *Jpn. J. Ind. Appl. Math.* 36 (2 04/2019), pp. 699–717.

[Ant79] A. Antoniou. *Digital Filters: Analysis and Design.* New York, USA: McGraw-Hill, 1979.

[Arn51] W. E. Arnoldi. „The principle of minimized iteration in the solution of the matrix eigenvalue problem". In: *Qart. Appl. Math.* 9 (1951), pp. 17–29.

[AS74] M. Abramowitz and I. A. Stegun, eds. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* 9th. New York, NY: Dover Publications, Inc., 1974.

[Bin+02] D. Bindel et al. „On computing givens rotations reliably and efficiently". In: *ACM Trans. Math. Software* 28.2 (2002), pp. 206–238.

[Blu+09] V. Blum et al. „Ab initio molecular simulations with numeric atom-centered orbitals". In: *Comput. Phys. Commun.* 180.11 (2009), pp. 2175–2196.

[Cha87] T. F. Chan. „Rank revealing QR factorizations". In: *Linear Algebra Appl.* 88–89 (1987), pp. 67–82.

[CS82] M. P. Carpentier and A. F. D. Santos. „Solution of equations involving analytic functions". In: *J. Comput. Phys.* 45 (1982), pp. 210–220.

[Dan+76] J. W. Daniel et al. „Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization“. In: *Math. Comput.* 30 (10/1976), pp. 772–795.

[Dan74] R. W. Daniels. *Approximation Methods for Electronic Filter Design.* New York, USA: McGraw-Hill, 1974.

[Dav75] E. R. Davidson. „The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices“. In: *J. Comput. Phys.* 17.1 (1975), pp. 87–94.

[Dem+12] J. Demmel et al. „Communication-optimal Parallel and Sequential QR and LU Factorizations“. In: *SIAM J. Sci. Comput.* 34.1 (02/2012), A206–A239.

[DH11] T. A. Davis and Y. Hu. „The University of Florida Sparse Matrix Collection“. In: *ACM Trans. Math. Software* 38.1 (2011), 1:1–1:25.

[Dir58] P. A. M. Dirac. *Principles Of Quantum Mechanics.* 4th. Oxford, England: Oxford University Press, 1958.

[DK70] C. Davis and W. M. Kahan. „The Rotation of Eigenvectors by a Perturbation. III“. In: *SIAM J. Numer. Anal.* 7.1 (1970), pp. 1–46.

[DL67] L. M. Delves and J. N. Lyness. „A numerical method for locating the zeros of an analytic function“. In: *Math. Comput.* 21 (1967), pp. 543–560.

[DPS16] E. Di Napoli, E. Polizzi, and Y. Saad. „Efficient estimation of eigenvalue counts in an interval“. In: *Numer. Linear Algebra Appl.* 23.4 (2016), pp. 674–692.

[DR84] P. J. Davis and P. Rabinowitz. *Methods of numerical integration.* 2nd. Orlando, FL: Academic Press, 1984.

[Els+99] U. Elsner et al. „The Anderson Model of Localization: A Challenge for Modern Eigenvalue Methods“. In: *SIAM J. Sci. Comput.* 20.6 (1999), pp. 2089–2102.

[Eri+96] K. Eriksson et al. *Computational Differential Equations.* 2nd. Cambridge, UK: Cambridge University Press, 1996.

[FS12] H.-r. Fang and Y. Saad. „A Filtered Lanczos Procedure for Extreme and Interior Eigenvalue Problems“. In: *SIAM J. Sci. Comput.* 34.4 (08/2012), A2220–A2246.

[FTS10] Y. Futamura, H. Tadano, and T. Sakurai. „Parallel stochastic estimation method of eigenvalue distribution“. In: *JSIAM Letters* 2 (2010), pp. 127–130.

[Fuk+14] T. Fukaya et al. „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.* 11/2014, pp. 31–38.

[Fuk+20] T. Fukaya et al. „Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices“. In: *SIAM J. Sci. Comput.* 42.1 (2020), A477–A503.

[Fut+17] Y. Futamura et al. *Reducing linear system size with moment based methods in the BEAST framework.* Talk at RIKEN LSPANC Meeting, 2017. 2017.

[Fut+18] Y. Futamura et al. *Using the Moment to Reduce Linear System Size.* Poster at SIAM Conference on Parallel Processing for Scientific Computing. 2018.

[Fut+19] Y. Futamura et al. *Contour Integration and Moments for the Solution of Large Eigenproblems.* Poster at SIAM Computer Science and Engineering 2019. 2019.

[Gal+09] M. Galassi et al. *GNU Scientific Library Reference Manual.* 3rd. Godalming, Surrey, UK: Network Theory Ltd., 2009.

[Gal+14] M. Galgon et al. „Improving robustness of the FEAST algorithm and solving eigenvalue problems from graphene nanoribbons“. In: *Proc. Appl. Math. Mech.* 14.1 (2014), pp. 821–822.

[Gal+15] M. Galgon et al. „On the parallel iterative solution of linear systems arising in the FEAST algorithm for computing inner eigenvalues“. In: *Parallel Comput.* 49 (2015), pp. 153–163.

[Gal+17] M. Galgon et al. „Improved Coefficients for Polynomial Filtering in ESSEX“. In: *Eigenvalue Problems: Algorithms, Software and Applications in Petascale Computing.* Vol. 117. Lecture Notes in Computational Science and Engineering. Springer, 01/2017, pp. 63–79.

[GG05] D. Gordon and R. Gordon. „Component-Averaged Row Projections: A Robust, Block-Parallel Scheme for Sparse Linear Systems“. In: *SIAM J. Sci. Comput.* 27.3 (2005), pp. 1092–1117.

[GG10] D. Gordon and R. Gordon. „CARP-CG: A robust and efficient parallel solver for linear systems, applied to strongly convection dominated PDEs“. In: *Parallel Comput.* 36 (09/2010), pp. 495–515.

[GHL18a] M. Galgon, S. Huber, and B. Lang. *BEAST: a framework for interior eigenvalue problems.* Poster at International High Performance Computing Summer School 2018. 2018.

[GHL18b] M. Galgon, S. Huber, and B. Lang. *Large-scale testing of the BEAST-P eigensolver.* CoSaS 2018 International Symposium on Computational Science at Scale. 2018.

[GHL18c] M. Galgon, S. Huber, and B. Lang. *Mixed precision in a large iterative parallel eigensolver framework: BEAST.* Poster at EPASA 2018 International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing. 2018.

[GHL18d] M. Galgon, S. Huber, and B. Lang. „Mixed precision in subspace iteration-based eigensolvers“. In: *Proc. Appl. Math. Mech.* 18.1 (12/2018), e201800334.

[GHL18e] M. Galgon, S. Huber, and B. Lang. *Recent developments and results for the BEAST eigensolver.* Talk at 149th R-CCS Cafe. 2018.

[GHL18f] M. Galgon, S. Huber, and B. Lang. *Using mixed precision in iterative eigensolvers.* Talk at GAMM Annual Meeting 2018. 2018.

[Ghy+15] P. Ghysels et al. „An Efficient Multicore Implementation of a Novel HSS-Structured Multifrontal Solver Using Randomized Sampling“. In: *SIAM J. Sci. Comput.* 38 (02/2015), pp. 358–384.

[Ghy+18] P. Ghysels et al. *STRUMPACK – STRUctured Matrices PACKage.* Version 3.1.0. 10/2018. URL: https://portal.nersc.gov/project/sparse/strumpack.

[Gir+05] L. Giraud et al. „Rounding error analysis of the classical Gram-Schmidt orthogonalization process“. In: *Numer. Math.* 101.1 (07/2005), pp. 87–100.

[GKL11] M. Galgon, L. Krämer, and B. Lang. „The FEAST algorithm for large eigenvalue problems“. In: *Proc. Appl. Math. Mech.* 11.1 (2011), pp. 747–748.

[GKL12] M. Galgon, L. Krämer, and B. Lang. „Counting eigenvalues and improving the integration in the FEAST algorithm“. Preprint BUW-IMACM 12/22, http://www.imacm.uni-wuppertal.de/imacm/research/preprints.html. 2012.

[GKL18] M. Galgon, L. Krämer, and B. Lang. „Improving projection-based eigensolvers via adaptive techniques“. In: *Numer. Linear Algebra Appl.* 25.1 (2018), e2124.

[GMP18] B. Gavin, A. Międlar, and E. Polizzi. „FEAST eigensolver for nonlinear eigenvalue problems“. In: *J. Comput. Sci.* 27 (2018), pp. 107–117.

[GNU] GNU Project. *GNU Scientific Library.* Version 2.5. URL: https://www.gnu.org/software/gsl.

[Gol91] D. Goldberg. „What Every Computer Scientist Should Know About Floating-point Arithmetic“. In: *ACM Comput. Surv.* 23.1 (03/1991), pp. 5–48.

[Güt+15] S. Güttel et al. „Zolotarev quadrature rules and load balancing for the FEAST eigensolver“. In: *SIAM J. Sci. Comput.* 37.4 (2015), A2100–A2122.

[GV13] G. H. Golub and C. F. Van Loan. *Matrix Computations.* 4th. Baltimore, MD: Johns Hopkins University Press, 2013.

[GW69] G. H. Golub and J. H. Welsch. „Calculation of Gauss Quadrature Rules“. In: *Math. Comput.* 23.106 (1969), pp. 221–230.

[Hig02] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* 2nd. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002.

[Hig08] N. J. Higham. *Functions of Matrices: Theory and Computation.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008.

[HJ13] R. A. Horn and C. R. Johnson. *Matrix Analysis: Second Edition.* 2nd. Cambridge, UK: Cambridge University Press, 2013.

[HJ88] R. W. Hockney and C. R. Jesshope. *Parallel Computers 2: Architecture, Programming and Algorithms.* 2nd. Bristol, UK: IOP Publishing Ltd., 1988.

[HK10] M. Z. Hasan and C. L. Kane. „Colloquium: Topological insulators“. In: *Rev. Mod. Phys.* 82 (4 11/2010), pp. 3045–3067.

[Hoe11] M. Hoemmen. „A communication-avoiding, hybrid-parallel, rank-revealing orthogonalization method“. In: *Proceedings - 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011.* Anchorage, AK, USA: IEEE Computer Society, 06/2011, pp. 966–977.

[Hof89] W. Hoffmann. „Iterative algorithms for Gram-Schmidt orthogonalization“. In: *Computing* 41.4 (12/1989), pp. 335–348.

[HS52] M. R. Hestenes and E. Stiefel. „Methods of Conjugate Gradients for Solving Linear Systems“. In: *J. Res. Nat. Bur. Stand.* 49 (1952), pp. 409–435.

[Hub+] S. Huber et al. „Flexible subspace iteration with moments for an effective contour-integration based eigensolver“. In preparation.

[IDS16] A. Imakura, L. Du, and T. Sakurai. „Error bounds of Rayleigh–Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems“. In: *Numer. Algorithms* 71.1 (2016), pp. 103–120.

[IS10] T. Ikegami and T. Sakurai. „Contour Integral Eigensolver for Non-Hermitian Systems: A Rayleigh-Ritz-Type Approach“. In: *Taiwan. J. Math.* 14.3A (2010), pp. 825–837.

[ISN10] T. Ikegami, T. Sakurai, and U. Nagashima. „A filter diagonalization for generalized eigenvalue problems based on the Sakurai–Sugiura projection method“. In: *J. Comput. Appl. Math.* 233 (2010), pp. 1927–1936.

[Jac46] C. G. J. Jacobi. „Über ein leichtes Verfahren die in der Theorie der Säcularstörungen vorkommenden Gleichungen numerisch aufzulösen.“ In: *J. Reine Angew. Math. (Crelle's Journal)* (1846), pp. 51–94.

[Kac37] S. Kaczmarz. „Angenäherte Auflösung von Systemen linearer Gleichungen“. In: *Bull. Acad. Polon. Sci. Lett.* A35 (1937), pp. 355–357.

[KE+] M. Kreutzer, D. Ernst, et al. *GHOST - General, Hybrid and Optimized Sparse Toolkit.* URL: `https://bitbucket.org/essex/ghost`.

[Ken05] A. D. Kennedy. „Fast Evaluation of Zolotarev Coefficients“. In: *QCD and Numerical Analysis III.* Vol. 47. Lecture Notes in Computational Science and Engineering. Springer, 2005, pp. 169–189.

[Knu97] D. Knuth. *Seminumerical Algorithms.* Vol. 2. The Art of Computer Programming. Boston, USA: Addison-Wesley, 1997.

[Krä+13] L. Krämer et al. „Dissecting the FEAST algorithm for generalized eigenproblems“. In: *J. Comput. Appl. Math.* 244 (2013), pp. 1–9.

[Krä14] L. Krämer. „Integration based solvers for standard and generalized Hermitian eigenvalue problems“. `http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:hbz:468-20140701-112141-6`. PhD thesis. Bergische Universität Wuppertal, 2014.

[Kre+16] M. Kreutzer et al. „Performance engineering and energy efficiency of building blocks for large, sparse eigenvalue computations on heterogeneous supercomputers“. In: *Software for Exascale Computing-SPPEXA 2013-2015.* Springer, 2016, pp. 317–338.

[Kre+17] M. Kreutzer et al. „GHOST: building blocks for high performance sparse linear algebra on heterogeneous systems“. In: *Int. J. Parallel Program.* (2017), pp. 1–27.

[Kre98] R. Kress. *Numerical Analysis.* Vol. 181. Graduate Texts in Mathematics. Springer Netherlands, 1998.

[Kry31] A. N. Krylov. „On the numerical solution of the equation by which the frequency of small oscillations is determined in technical problems (in Russian)“. In: *Izv. Akad. Nauk SSSR Ser. Fiz.-Mat.* 4 (1931), pp. 491–539.

[KSB99] P. Kravanja, T. Sakurai, and M. V. Barel. „On locating clusters of zeros of analytic functions“. In: *BIT Numer. Math.* 39 (1999), pp. 646–682.

[KSS13] A. M. Klinvex, F. Saied, and A. H. Sameh. „Parallel implementations of the trace minimization scheme TraceMIN for the sparse symmetric eigenvalue problem“. In: *Comput. Math. Appl.* 65.3 (2013), pp. 460–468.

[KU98] A. R. Krommer and C. W. Ueberhuber. *Computational Integration.* Philadelphia, PA: SIAM, 1998.

[Lan50] C. Lanczos. „An iteration method for the solution of the eigenvalue problem of linear differential and integral operators“. In: *J. Res. Nat. Bur. Stand.* 45.4 (1950), pp. 255–282.

[Lau12] S. E. Laux. „Solving complex band structure problems with the FEAST eigenvalue algorithm“. In: *Phys. Rev. B* 86 (2012), pp. 075–103.

[Li+16] R. Li et al. „A Thick-Restart Lanczos Algorithm with Polynomial Filtering for Hermitian Eigenvalue Problems“. In: *SIAM J. Sci. Comput.* 38.4 (08/2016), A2512–A2534.

[LS84] D. H. Lawrie and A. H. Sameh. „The Computation and Communication Complexity of a Parallel Banded System Solver“. In: *ACM Trans. Math. Software* 10.2 (05/1984), pp. 185–195.

[LSY16] L. Lin, Y. Saad, and C. Yang. „Approximating Spectral Densities of Large Matrices“. In: *SIAM Rev.* 58.1 (02/2016), pp. 34–65.

[LT05] M. D. Lutovac and D. V. Tošić. „Elliptic Rational Functions“. In: *The Mathematica Journal* 9.3 (2005).

[LTE01] M. D. Lutovac, D. V. Tošić, and B. L. Evans. *Filter Design for Signal Processing using MATLAB® and Mathematica®*. New Jersey, USA: Prentice Hall, 2001.

[LY17] Y. Li and H. Yang. „Interior Eigensolver for Sparse Hermitian Definite Matrices Based on Zolotarev's Functions“. http://arxiv.org/abs/1701.08935 [math.NA], version 3. 01/2017.

[Mar+14] A. Marek et al. „The ELPA Library: Scalable Parallel Eigenvalue Solutions for Electronic Structure Theory and Computational Science“. In: *J. Phys.: Condens. Matter* 26.21 (05/2014), p. 213201.

[Meu+17] A. Meurer et al. „SymPy: symbolic computing in Python“. In: *PeerJ Comput. Sci.* 3 (2017), e103. URL: https://www.sympy.org.

[MLB] The MathWorks, Inc. *MATLAB® and DSP System Toolbox^{TM}*. Version R2018a. Natick, Massachusetts, US. URL: https://www.mathworks.com.

[MPI15] MPI Forum. *MPI: A Message-Passing Interface Standard. Version 3.1.* available at: https://www.mpi-forum.org/docs (Mar. 2020). 06/2015.

[Mul06] J.-M. Muller. *Elementary Functions—Algorithms and Implementation*. 2nd. Boston–Basel–Berlin: Birkhäuser, 2006.

[Nak12] Y. Nakatsukasa. „The $\tan\theta$ theorem with relaxed conditions“. In: *Linear Algebra Appl.* 436.5 (2012), pp. 1528–1534.

[Net+09] A. H. C. Neto et al. „The electronic properties of graphene“. In: *Rev. Mod. Phys.* 81 (1 01/2009), pp. 109–162.

[Neu71] J. von Neumann. *Mathematical foundations of quantum mechanics*. 6th. New Jersey, USA: Princeton University Press, 1971.

[Neu85] A. Neumaier. „Residual Inverse Iteration for the Nonlinear Eigenvalue Problem“. In: *SIAM J. Numer. Anal.* 22.5 (1985), pp. 914–923.

[Oli+00] S. Oliveira et al. „Analysis of different partitioning schemes for parallel Gram-Schmidt algorithms“. In: *Parallel Algorithms Appl.* 14.4 (2000), pp. 293–320.

[Olv+10] F. W. J. Olver et al., eds. *NIST Handbook of Mathematical Functions.* 1st. Cambridge, UK: Cambridge University Press, 2010.

[Ope18] OpenMP Architecture Review Board. *OpenMP Application Program Interface Version 5.0.* available at: https://www.openmp.org/specifications (Mar. 2020). 11/2018.

[Orf05] S. J. Orfanidis. „High-Order Digital Parametric Equalizer Design“. In: *J. Audio Eng. Soc.* 53 (2005), pp. 1026–1046.

[Pag+98] L. Page et al. *The PageRank Citation Ranking: Bringing Order to the Web.* Tech. rep. Stanford InfoLab, 1998.

[Par80] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall Series in Computational Mathematics. Englewood-Cliffs, NJ: Prentice-Hall, 1980.

[Pie+16] A. Pieper et al. „High-performance implementation of Chebyshev filter diagonalization for interior eigenvalue computations“. In: *J. Comput. Phys.* 325 (2016), pp. 226–243.

[PK15] E. Polizzi and J. Kestyn. *FEAST Eigenvalue Solver v3.0 User Guide.* 2015. arXiv: 1203.4031v3 [cs.MS].

[Pol09] E. Polizzi. „Density-matrix-based algorithm for solving eigenvalue problems“. In: *Phys. Rev. B* 79 (2009), pp. 115–112.

[Pol13] E. Polizzi. *A High-Performance Numerical Library for Solving Eigenvalue Problems: FEAST Solver v2.0 User's Guide.* 2013. arXiv: 1203.4031v2 [cs.MS].

[Pol20] E. Polizzi. *FEAST Eigenvalue Solver v4.0 User Guide.* 2020. arXiv: 2002.04807 [cs.MS].

[PP87] P. P. Petrushev and V. A. Popov. *Rational Approximation of Real Functions.* Vol. 28. Encyclopedia of Mathematics and its Applications. Cambridge, UK: Cambridge University Press, 1987.

[Pre+92] W. H. Press et al. *Numerical Recipes in FORTRAN 77: The Art of Scientific Computing.* 2nd. Cambridge, UK: Cambridge University Press, 1992.

[PS06] E. Polizzi and A. H. Sameh. „A parallel hybrid banded system solver: the SPIKE algorithm“. In: *Parallel Comput.* 32.2 (2006), pp. 177–194.

[Ray99] J. W. Strutt (3rd Baron Rayleigh). „On the calculation of the frequency of vibration of a system in its gravest mode, with an example from hydrodynamics“. In: *Philos. Mag. Series 5* 47.289 (1899), pp. 566–572.

[Rit09] W. Ritz. „Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik“. In: *J. Reine Angew. Math.* 135 (1909), pp. 1–61.

[Rou+16] F.-H. Rouet et al. „A Distributed-Memory Package for Dense Hierarchically Semi-Separable Matrix Computations Using Randomization“. In: *ACM Trans. Math. Software* 42.4 (06/2016).

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems.* 2nd. Philadelphia, PA: SIAM, 2003.

[Saa11] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* 2nd. Philadelphia, PA: SIAM, 2011.

[Sak+19] T. Sakurai et al. „Scalable Eigen-Analysis Engine for Large-Scale Eigenvalue Problems“. In: *Advanced Software Technologies for Post-Peta Scale Computing: The Japanese Post-Peta CREST Research Project.* Ed. by M. Sato. Singapore: Springer Singapore, 2019, pp. 37–57.

[Sha+19] F. Shahzad et al. „CRAFT: A library for easier application-level Checkpoint/Restart and Automatic Fault Tolerance“. In: *IEEE Trans. Parallel Distrib. Syst.* 30 (3 03/2019), pp. 501–514.

[Sin06] S. Singer. „Indefinite QR Factorization“. In: *BIT Numer. Math.* 46.1 (03/2006), pp. 141–161.

[SK78] A. H. Sameh and D. J. Kuck. „On Stable Parallel Linear System Solvers“. In: *J. ACM* 25.1 (01/1978), pp. 81–91.

[SPS18] B. S. Spring, E. Polizzi, and A. H. Sameh. „A Feature Complete SPIKE Banded Algorithm and Solver“. In: *CoRR* abs/1811.03559 (2018). arXiv: 1811.03559.

[SS03a] T. Sakurai and H. Sugiura. „A projection method for generalized eigenvalue problems“. In: *J. Comput. Appl. Math.* 159 (2003), pp. 119–128.

[SS03b] S. Singer and S. Singer. „Rounding error and perturbation bounds for the symplectic QR factorization“. In: *Linear Algebra Appl.* 358.1 (2003), pp. 255–279.

[SS08] S. Singer and S. Singer. „Orthosymmetric block reflectors“. In: *Linear Algebra Appl.* 429.5 (2008), pp. 1354–1385.

[SS90] G. W. Stewart and J. Sun. *Matrix Perturbation Theory.* San Diego, CA: Academic Press, 1990.

[ST07] T. Sakurai and H. Tadano. „CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems“. In: *Hokkaido Math. J.* 36 (2007), pp. 745–757.

[Ste01] G. W. Stewart. *Matrix Algorithms.* Vol. II, Eigensystems. Philadelphia, PA: SIAM, 2001.

[Ste08] G. W. Stewart. „Block Gram-Schmidt Orthogonalization“. In: *SIAM J. Sci. Comput.* 31 (01/2008), pp. 761–775.

[Ste98] G. W. Stewart. *Matrix Algorithms.* Vol. I, Basic decompositions. Philadelphia, PA: SIAM, 1998.

[SV00] G. L. G. Sleijpen and H. A. van der Vorst. „A Jacobi–Davidson iteration method for linear eigenvalue problems“. In: *SIAM Rev.* 42.2 (2000), pp. 267–293.

[SW06] A. Stathopoulos and K. Wu. „A Block Orthogonalization Procedure with Constant Synchronization Requirements“. In: *SIAM J. Sci. Comput.* 23.6 (06/2006), pp. 2165–2182.

[SW82] A. H. Sameh and J. A. Wisniewski. „A Trace Minimization Algorithm for the Generalized Eigenvalue Problem“. In: *SIAM J. Numer. Anal.* 19.6 (1982), pp. 1243–1259.

[TB97] L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra.* Philadelphia, PA: SIAM, 1997.

[Thi+16] J. Thies et al. „Towards an exascale enabled sparse solver repository“. In: *Software for Exascale Computing-SPPEXA 2013-2015.* Springer, 2016, pp. 295–316.

[Thi+19] J. Thies et al. „PHIST: a Pipelined, Hybrid-parallel Iterative Solver Toolkit“. In: *ACM Trans. Math. Software* (01/2019).

[Tim37] S. Timoshenko. *Vibration Problems in Engineering.* 2nd. New York, USA: D. Van Nostrand Company, Inc., 1937.

[TP] P. T. P. Tang and E. Polizzi. „Subspace Iteration with Approximate Spectral Projection“. http://arxiv.org/abs/1302.0432 [math.NA], version 3.

[TR+] J. Thies, M. Röhrig-Zöllner, et al. *PHIST - a Pipelined Hybrid Parallel Iterative Solver Toolkit.* URL: https://bitbucket.org/essex/phist.

[Via12] G. Viaud. „The FEAST algorithm for generalised eigenvalue problems“. MA thesis. University of Oxford, 2012.

[Wac00] E. L. Wachspress. „Evaluating Elliptic Functions and Their Inverses“. In: *Comput. Math. Appl.* 39 (2000), pp. 131–136.

[Wat07] D. S. Watkins. *The Matrix Eigenvalue Problem.* Philadelphia, PA: SIAM, 2007.

[Wei+06] A. Weiße et al. „The Kernel Polynomial Method“. In: *Rev. Mod. Phys.* 78 (1 2006), pp. 275–306.

[Wei02] J. A. C. Weideman. „Numerical Integration of Periodic Functions: A Few Examples“. In: *Am. Math. Mon.* 109.1 (2002), pp. 21–36.

[Wel+20] G. Wellein et al. „Equipping Sparse Solvers for Exascale“. In: *Software for Exascale Computing - SPPEXA 2016-2019.* Lecture Notes on Computational Science and Engineering. Springer, 01/2020.

[Wil65] J. H. Wilkinson. *The algebraic eigenvalue problem.* Oxford, UK: Clarendon Press, 1965.

[XLS18] Y. Xi, R. Li, and Y. Saad. „Fast Computation of Spectral Densities for Generalized Eigenvalue Problems“. In: *SIAM J. Sci. Comput.* 40.4 (08/2018), A2749–A2773.

[XS16] Y. Xi and Y. Saad. „Computing Partial Spectra with Least-Squares Rational Filters“. In: *SIAM J. Sci. Comput.* 38.5 (09/2016), A3020–A3045.

[Yam+15] Y. Yamamoto et al. „Roundoff error analysis of the Cholesky QR2 algorithm“. In: *Electron. Trans. Numer. Anal.* 44 (01/2015), pp. 306–326.

[Zol77] Y. I. Zolotarev. „Application of elliptic functions to questions of functions deviating least and most from zero (in Russian)“. In: *Zap. Imp. Akad. Nauk St. Petersburg* 30 (1877), pp. 1–59.