Electronic Transactions on Numerical Analysis.
Volume 30, pp. 26-53, 2008.
Copyright © 2008, Kent State University.
ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

GEGENBAUER POLYNOMIALS AND SEMISEPARABLE MATRICES[*]

JENS KEINER[†]

Abstract. In this paper, we develop a new $\mathcal{O}(n \log n)$ algorithm for converting coefficients between expansions in different families of Gegenbauer polynomials up to a finite degree n. To this end, we show that the corresponding linear mapping is represented by the eigenvector matrix of an explicitly known diagonal plus upper triangular semiseparable matrix. The method is based on a new efficient algorithm for computing the eigendecomposition of such a matrix. Using fast summation techniques, the eigenvectors of an $n \times n$ matrix can be computed explicitly with $\mathcal{O}\left(n^2\right)$ arithmetic operations and the eigenvector matrix can be applied to an arbitrary vector at cost $\mathcal{O}\left(n \log n\right)$. All algorithms are accurate up to a prefixed accuracy ε. We provide brief numerical results.

Key words. Gegenbauer polynomials, polynomial transforms, semiseparable matrices, eigendecomposition, spectral divide-and-conquer methods

AMS subject classifications. 42C10, 42C20, 15A18, 15A23, 15A57, 65T50, 65Y20

1. Introduction.

The development and implementation of numerical algorithms for solving problems from mathematical physics has long evolved since the beginning of the computer age. With the availability of larger and larger computational resources, over the years, the demand for solving physical problems computationally has created a growing need for efficient and accurate algorithms. These involve the large class of special functions of mathematical physics which arise during mathematical formulation of the problem at hand. Standard references in this area are, for example, Nikiforov and Uvarov [32], Temme [41], and Olver [34].

The increased impact of numerical algorithms could not have become more clear since the publication of the celebrated FFT algorithm by Cooley and Tukey in [7]. The same method was used - with paper and pencil - already by Gauss in 1805 to estimate the trajectories of asteroids and has laid the foundation for a substantial part of everyday-use numerical algorithms. In the wake of the FFT algorithm, which in essence deals with sine and cosine functions, similar algorithms for more general function systems have drawn more and more attention. This also holds for Gegenbauer polynomials where sines and cosines (in this order) are included in the form of Chebyshev polynomials of second and first kind.

Gegenbauer polynomials are frequent companions in the mathematical treatment of a range of problems motivated from physics. Often-cited examples are Schrödinger-type equations which lead to differential equations of hypergeometric type; see [32, pp. 1]. Gegenbauer polynomials $C_n^{(\alpha)}$ with parameter $\alpha > 0$ and degree n form a complete set of orthogonal polynomial solutions to the more special Gegenbauer differential equation

$$\left(1-x^2\right) y'' - (2\alpha+1)xy' + n(n+2\alpha)y = 0 \quad (n \in \mathbb{N}_0,\, \alpha > -1/2, \alpha \neq 0,\, x \in [-1,1]).$$

Most textbooks on classical orthogonal polynomials mention Gegenbauer polynomials as a special case, notably Szegő [40], Chihara [6] and Nikiforov and Uvarov [32].

In this paper, we treat the problem of coefficient conversion between expansions in different families of Gegenbauer polynomials. It is a common task to evaluate, analyse or somehow process a function f which is known by its finite expansion in Gegenbauer polynomials $C_n^{(\alpha)}$

[*]Received March 6, 2007. Accepted for publication November 16, 2007. Published online on May 1, 2008. Recommended by V. Olshevsky.

[†]Institut für Mathematik, Universität zu Lübeck, Wallstraße 40, 23560 Lübeck, Germany (keiner@math.uni-luebeck.de).

for a certain fixed value of α, i.e.

$$f = \sum_{n=0}^{N} \mu_n C_n^{(\alpha)}.$$

It also happens frequently that this can be done most efficiently when the whole expansion could be converted into another one for a different parameter value β,

$$f = \sum_{n=0}^{N} \nu_n C_n^{(\beta)}.$$

A usual example is the case $\alpha > 0$ and $\beta = 0$ which corresponds to a conversion to Chebyshev polynomials of first kind. Expansions in these polynomials can be efficiently handled with FFT and NFFT techniques publicly available in software libraries; see [12, 25]. In most cases avoiding coefficient conversion results in prohibitively costly computations if no fast algorithms for the original form are available.

Many papers discussing coefficient conversion and related problems are available in the literature. Alpert and Rokhlin [2] consider the special case of converting between Legendre polynomials and Chebyshev polynomials of first kind. They develop an approximate $\mathcal{O}(n)$ algorithm based on matrix compression techniques. The method has been recently generalised to arbitrary families of Gegenbauer polynomials in [24] but the connection to semiseparable matrices established in this work is not revealed. It is, however, asymptotically faster. Potts, Steidl, and Tasche [37] and Potts [36] treat the evaluation of orthogonal polynomial expansions at Chebyshev- as well as at arbitrary nodes. Driscoll, Healy, Moore, and Rockmore [30, 9] earlier considered the 'transposed' transformation, but at Chebyshev nodes only. The algorithms are exact, need $\mathcal{O}(n \log^2 n)$ arithmetic operations, and solely depend on the related three-term recurrence relation. However, they can suffer from severe numerical instabilities unless stabilisation techniques are employed. Improved versions used in the context of fast spherical harmonics transforms appear in [8, 21] for special nodes and [27, 26] for arbitrary nodes. Suda and Takami [39] use similar concepts but a new stabilisation technique based on a nearly-optimal choice of interpolation nodes. Several other papers in the general context of structured matrices and orthogonal polynomials include Higham [22], Pan [35], Olshevsky and Pan [33], and many others.

In [38], Rokhlin and Tygert describe a new technique for converting between expansions in associated Legendre functions of different orders. Based on related differential equations, they discover that the linear mapping for coefficient conversion appears as the eigenvector matrix of an explicitly known symmetric diagonal plus semiseparable matrix. This allows for applying a well-known efficient algorithm by Chandrasekaran and Gu from [5] for computing the eigendecomposition of such a matrix. In combination with a variant of Greengard's and Rokhlin's fast multipole method [17] this yields a fast algorithm for computing spherical harmonics transforms. Our method is somewhat similar but focuses on Gegenbauer polynomials.

Semiseparable matrices themselves also appear in a range of fields in applied mathematics, such as physical problems that give rise to so-called oscillation matrices (Gantmakher and Krein [13]), statistical analysis with covariance matrices that turn out to be specially structured (Graybill [16]), and the discretisation of certain integral equations (Kailath [23]). In most cases, it is desirable to solve linear systems of equations involving a semiseparable matrix or to obtain its eigendecomposition. In [42], Vandebril gives a general introduction to semiseparable matrices with focus on the symmetric eigenvalue problem. Bella, Eidelman, Gohberg, Olshevsky, and Tyrtyshnikov have considered even more general classes of so-called quasiseparable matrices and give explicit and recursive formulae for eigenvectors, characteristic polynomials and other related quantities; see [11, 3]. They also derive fast algorithms exploiting structural properties and introduce new classes of orthogonal polynomials. This is done by generalising the concept of tridiagonal Jacoby matrices for classical orthogonal polynomials to quasiseparable matrices of order one. Instead, we use well-known properties of Gegenbauer polynomials and the structure of the eigendecomposition of a particular class of semiseparable matrices to establish a new connection to coefficient conversion between different families of Gegenbauer polynomials.

In [5], Chandrasekaran and Gu develop an efficient divide-and-conquer algorithm for computing the eigendecomposition of a symmetric block-diagonal plus semiseparable matrix. For an $n \times n$ matrix, they obtain an exact $\mathcal{O}\left(n^2\right)$ algorithm for computing only the eigenvalues and an exact $\mathcal{O}\left(n^3\right)$ algorithm for computing in addition also the eigenvectors. Using fast summation techniques, the algorithms can be accelerated to yield an $\mathcal{O}(n \log n)$ algorithm for computing only the eigenvalues, an $\mathcal{O}\left(n^2\right)$ algorithm for computing also the eigenvectors, and, moreover, an $\mathcal{O}\left(n \log n\right)$ algorithm for applying the eigenvector matrix, or optionally its inverse, to a vector. The accelerated algorithms are accurate up to a prefixed accuracy ε. The essential ingredient is a recursive splitting of the matrix in question into smaller matrices of the same type until the problem can be solved directly for each small matrix. The results are then efficiently combined to obtain the eigendecomposition of the whole matrix. This is a classical divide-and-conquer strategy. Mastronardi, Van Camp, and Van Barel devise similar algorithms in [29]. We develop a new efficient algorithm for computing the eigendecomposition of diagonal plus upper or lower triangular semiseparable matrices. They distinguish from the symmetric semiseparable case by that either their upper or their lower triangular part is zero. The procedure resembles the symmetric case but allows for some simplifications. To be applicable to the problem of coefficient conversion between expansions in Gegenbauer polynomials, we show that the corresponding linear mapping is the eigenvector matrix of an explicitly known diagonal plus upper triangular semiseparable matrix.

The structure of this paper is as follows: In Section 2, we introduce basic notation and definitions. In Section 3, we briefly survey Chandrasekaran's and Gu's divide-and conquer algorithm for computing the eigendecomposition of symmetric block-diagonal plus semiseparable matrices from [5] and develop a new strategy for diagonal plus upper or lower triangular semiseparable matrices. We comment briefly on accelerating the algorithms by fast summation techniques. In Section 4, we show that the linear mapping that converts between expansions in different families of Gegenbauer polynomials is the eigenvector matrix of an explicitly known upper triangular semiseparable matrix. We complement the theoretical findings by numerical results in Section 5.

2. Notational conventions and preliminaries.

In this section, we introduce basic notation and definitions. Scalars are displayed in normal face while vectors and matrices are displayed in boldface. We use capital letters to denote matrices and lower case letters for scalars or vectors. We denote by $\delta_{i,j}$ the usual Kronecker delta. Throughout this paper, all matrices have real entries and real eigenvalues. For a vector $\mathbf{x}$, we let $\operatorname{diag}(\mathbf{x})$ be the diagonal matrix with the entries of $\mathbf{x}$ on its diagonal. If $\mathbf{x}$ has length n and $0 \leq k \leq n$, then $\mathbf{x}_1$ denotes the subvector containing the first k elements of $\mathbf{x}$ and $\mathbf{x}_2$ is the subvector consisting of the last $n-k$ elements of $\mathbf{x}$. For a matrix $\mathbf{A}$, we define $\operatorname{diag}(\mathbf{A})$ to be the vector containing the diagonal entries of $\mathbf{A}$. Furthermore, $\operatorname{triu}(\mathbf{A})$ denotes the matrix that coincides with $\mathbf{A}$ strictly above the main diagonal while being zero elsewhere. Similarly, $\operatorname{tril}(\mathbf{A})$ is identical to $\mathbf{A}$ strictly below the main diagonal and zero elsewhere.

DEFINITION 2.1. *A matrix* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *is called* symmetric diagonal plus semiseparable[1] *or* $\mathcal{S}$-matrix *if it has the form*

$$\mathbf{A} = \operatorname{diag}(\mathbf{d}) + \operatorname{triu}\left(\mathbf{u}\,\mathbf{v}^{\mathrm{T}}\right) + \operatorname{tril}\left(\mathbf{v}\,\mathbf{u}^{\mathrm{T}}\right) \qquad (\mathbf{d},\mathbf{u},\mathbf{v} \in \mathbb{R}^n). \tag{2.1}$$

A generic example for an $\mathcal{S}$-matrix is

$$\mathbf{A} = \begin{pmatrix} d_1 & \mu_1\nu_2 & \mu_1\nu_3 & \dots & \mu_1\nu_n \\ \mu_1\nu_2 & d_2 & \mu_2\nu_3 & \dots & \mu_2\nu_n \\ \mu_1\nu_3 & \mu_2\nu_3 & d_3 & \dots & \mu_3\nu_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_1\nu_n & \mu_2\nu_n & \mu_3\nu_n & \dots & d_n \end{pmatrix}.$$

It consists of diagonal elements d_j and off-diagonal elements $\mu_j\nu_k$ which are part of rank-one matrices $\mathbf{u}\,\mathbf{v}^{\mathrm{T}}$ and $\mathbf{v}\,\mathbf{u}^{\mathrm{T}}$, respectively. The elements μ_n and ν_1 can be chosen arbitrarily since not referenced. A particular class of unsymmetric but similarly structured matrices is obtained by discarding either the lower or the upper triangular part from the defining equation (2.1).

DEFINITION 2.2. *A matrix* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *is called* diagonal plus upper triangular semiseparable *or* $\mathcal{U}$-matrix *if it has the form*

$$\mathbf{A} = \operatorname{diag}(\mathbf{d}) + \operatorname{triu}\left(\mathbf{u}\,\mathbf{v}^{\mathrm{T}}\right) \qquad (\mathbf{d},\mathbf{u},\mathbf{v} \in \mathbb{R}^n).$$

A matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$ *is called* diagonal plus lower triangular semiseparable *or* $\mathcal{L}$-matrix *if it can be represented as*

$$\mathbf{B} = \operatorname{diag}(\mathbf{d}) + \operatorname{tril}\left(\mathbf{v}\,\mathbf{u}^{\mathrm{T}}\right) \qquad (\mathbf{d},\mathbf{u},\mathbf{v} \in \mathbb{R}^n).$$

Both, $\mathcal{U}$- and $\mathcal{L}$-matrices, reveal their eigenvalues immediately which are just their respective diagonal entries d_j.

2.1. Eigenvalues and eigenvectors.

While being similar in structure, $\mathcal{S}$-, $\mathcal{U}$-, and $\mathcal{L}$-matrices differ with respect to the eigendecomposition. For an $\mathcal{S}$-matrix $\mathbf{A}$, the Spectral Theorem for symmetric matrices (see [15, Theorem 8.1.1, p. 393]) asserts an eigendecomposition $\mathbf{A} = \mathbf{Q}\,\mathbf{\Lambda}\,\mathbf{Q}^{\mathrm{T}}$ with an orthogonal eigenvector matrix $\mathbf{Q}$ and a real diagonal eigenvalue matrix $\mathbf{\Lambda}$. The eigenvectors of $\mathbf{A}$, i.e., the columns of $\mathbf{Q}$, are mutually orthogonal and therefore linearly independent. This holds even in the presence of multiple eigenvalues. Divide-and-conquer algorithms for computing the eigendecomposition of $\mathcal{S}$-matrices have been developed by Chandrasekaran and Gu in [5] and Mastronardi, Van Camp, and Van Barel in [29]. A deflation procedure from Bunch, Nielsen, and Sorensen in [4] allows for handling the case of multiple eigenvalues. Therefore, no further assumptions on the matrix have to be made beforehand.

Similar arguments for $\mathcal{U}$- and $\mathcal{L}$-matrices $\mathbf{A}$ do not hold. In general, unsymmetric matrices, even with real components, have complex eigenvalues. In our case, however, the eigenvalues coincide with the real entries $d_1, d_2, \dots, d_n$ on the main diagonal of $\mathbf{A}$ and hence are always real. But in the case of multiple eigenvalues, it might happen that $\mathbf{A}$ has degenerate eigenspaces. That is, the eigenspaces can have strictly lower dimension than the algebraic multiplicity of the corresponding eigenvalue. Eidelman, Gohberg, and Olshevsky ([11]) establish necessary conditions under which the eigenvalues are simple. A similar deflation

[1]Here and throughout the paper the term *semiseparable* is used as a convenient abbreviation to *generator representable semiseparable of semiseparability rank one*. Historically, the definition of semiseparability has been sometimes ambiguous; see [43].

procedure as for $\mathcal{S}$-matrices seems to be unknown yet. In the following, we therefore assume that all eigenvalues are distinct, i.e., $d_j \neq d_k$ whenever $j \neq k$, such that the eigenvectors are guaranteed to be linearly independent. The assumption will always be fulfilled in our application in Section 4.

3. Spectral divide-and-conquer algorithms. This section surveys Chandrasekaran's and Gu's spectral divide-and-conquer method from [5] for computing the eigendecomposition of $\mathcal{S}$-matrices. It is based on a dyadic decomposition strategy. In the divide phase, the problem is recursively split into smaller sub-problems of the same type until each smaller problem can be solved directly. In the conquer phase, solutions of smaller problems are successively combined to solutions for the next bigger problems, following the decomposition tree, until the sought solution of the original problem is obtained. We extend this concept to $\mathcal{U}$- and $\mathcal{L}$-matrices with the additional assumption on the simplicity of the eigenvalues.

3.1. Symmetric diagonal plus semiseparable matrices. The problem of computing the eigendecomposition of an $\mathcal{S}$-matrix $\mathbf{A} = \operatorname{diag}(\mathbf{d}) + \operatorname{triu}\left(\mathbf{u}\,\mathbf{v}^{\mathrm{T}}\right) + \operatorname{tril}\left(\mathbf{v}\,\mathbf{u}^{\mathrm{T}}\right) \in \mathbb{R}^n$ has been studied in [5]. We sketch the basic strategy here.

3.1.1. Divide phase. The main idea for a divide-and-conquer approach is the fact that an $\mathcal{S}$-matrix $\mathbf{A}$ admits the recursive representation

$$\mathbf{A} = \begin{pmatrix} \hat{\mathbf{A}}_1 & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{A}}_2 \end{pmatrix} + \rho\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}},$$

where $\rho = \pm 1$ is a free chosen scalar, $\mathbf{w} \in \mathbb{R}^n$ is a vector, and $\hat{\mathbf{A}}_1$ and $\hat{\mathbf{A}}_2$ are $\mathcal{S}$-matrices themselves. To check this, fix ρ and let $0 \leq k \leq n$ (usually, k is chosen as $k = \lfloor n/2 \rfloor$). Then we can write

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{u}_1\,\mathbf{v}_2^{\mathrm{T}} \\ \mathbf{v}_2\,\mathbf{u}_1^{\mathrm{T}} & \mathbf{A}_2 \end{pmatrix}$$

with the principal sub-matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ of $\mathbf{A}$ defined by

$$\mathbf{A}_j = \operatorname{diag}(\mathbf{d}_j) + \operatorname{triu}\left(\mathbf{u}_j\,\mathbf{v}_j^{\mathrm{T}}\right) + \operatorname{tril}\left(\mathbf{v}_j\,\mathbf{u}_j^{\mathrm{T}}\right) \qquad (j = 1, 2).$$

The rank-one matrices $\mathbf{u}_1\,\mathbf{v}_2^{\mathrm{T}}$ and $\mathbf{v}_2\,\mathbf{u}_1^{\mathrm{T}}$ account for the rest of the symmetric semiseparable part of $\mathbf{A}$. Finally, defining the vector $\mathbf{w} \in \mathbb{R}^n$ by

$$\mathbf{w} := \begin{pmatrix} \rho\,\mathbf{u}_1 \\ \mathbf{v}_2 \end{pmatrix},$$

we obtain the decomposition

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 - \rho\,\mathbf{u}_1\,\mathbf{u}_1^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 - \rho\,\mathbf{v}_2\,\mathbf{v}_2^{\mathrm{T}} \end{pmatrix} + \rho\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}} = \begin{pmatrix} \hat{\mathbf{A}}_1 & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{A}}_2 \end{pmatrix} + \rho\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}}.$$

Here, the newly defined matrices $\hat{\mathbf{A}}_1$ and $\hat{\mathbf{A}}_2$ with

$$\begin{aligned} \hat{\mathbf{A}}_1 := \mathbf{A}_1 - \rho\,\mathbf{u}_1\,\mathbf{u}_1^{\mathrm{T}} &= \operatorname{diag}\left(\mathbf{d}_1 - \rho\operatorname{diag}(\mathbf{u}_1\,\mathbf{u}_1^{\mathrm{T}})\right) + \operatorname{triu}\left(\mathbf{u}_1\,(\mathbf{v}_1 - \rho\,\mathbf{u}_1)^{\mathrm{T}}\right) \\ &\quad + \operatorname{tril}\left((\mathbf{v}_1 - \rho\,\mathbf{u}_1)\,\mathbf{u}_1^{\mathrm{T}}\right), \\ \hat{\mathbf{A}}_2 := \mathbf{A}_2 - \rho\,\mathbf{v}_2\,\mathbf{v}_2^{\mathrm{T}} &= \operatorname{diag}\left(\mathbf{d}_2 - \rho\operatorname{diag}(\mathbf{v}_2\,\mathbf{v}_2^{\mathrm{T}})\right) + \operatorname{triu}\left((\mathbf{u}_2 - \rho\,\mathbf{v}_2)\,\mathbf{v}_2^{\mathrm{T}}\right) \\ &\quad + \operatorname{tril}\left(\mathbf{v}_2\,(\mathbf{u}_2 - \rho\,\mathbf{v}_2)^{\mathrm{T}}\right) \end{aligned}$$

are $\mathcal{S}$-matrices themselves. They may be decomposed recursively in the same manner.

3.1.2. Conquer phase. Let ρ and k be defined as before. Suppose now that we have computed the eigendecompositions of two smaller $\mathcal{S}$-matrices $\hat{\mathbf{A}}_1$ and $\hat{\mathbf{A}}_2$ obtained from the decomposition of a bigger $\mathcal{S}$-matrix $\mathbf{A}$,

$$\hat{\mathbf{A}}_1 = \mathbf{Q}_1 \boldsymbol{\Lambda}_1 \mathbf{Q}_1^{\mathrm{T}}, \qquad \hat{\mathbf{A}}_2 = \mathbf{Q}_2 \boldsymbol{\Lambda}_2 \mathbf{Q}_2^{\mathrm{T}},$$

with the diagonal eigenvalue matrices $\boldsymbol{\Lambda}_1 := \mathrm{diag}\,(\boldsymbol{\lambda}_1)$, $\boldsymbol{\Lambda}_2 := \mathrm{diag}\,(\boldsymbol{\lambda}_2)$, and the orthogonal eigenvector matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$. For the matrix $\mathbf{A}$, this implies the representation

$$\mathbf{A} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix} \left(\boldsymbol{\Lambda} + \rho\, \mathbf{z}\, \mathbf{z}^{\mathrm{T}}\right) \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{\mathrm{T}}$$

with the matrix $\boldsymbol{\Lambda} := \mathrm{diag}\,(\boldsymbol{\lambda})$ and the vector

$$\mathbf{z} := \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{\mathrm{T}} \mathbf{w}.$$

To obtain the eigendecomposition of the matrix $\mathbf{A}$, let the eigendecomposition of the symmetric rank-one modified diagonal matrix $\boldsymbol{\Lambda} + \rho\, \mathbf{z}\, \mathbf{z}^{\mathrm{T}}$ be written as $\boldsymbol{\Lambda} + \rho\, \mathbf{z}\, \mathbf{z}^{\mathrm{T}} = \mathbf{U}\, \boldsymbol{\Omega}\, \mathbf{U}^{\mathrm{T}}$. Then the eigendecomposition $\mathbf{A} = \mathbf{Q}\, \boldsymbol{\Omega}\, \mathbf{Q}^{\mathrm{T}}$ is

$$\mathbf{A} = \underbrace{\left(\begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix} \mathbf{U}\right)}_{=\mathbf{Q}} \boldsymbol{\Omega} \underbrace{\left(\mathbf{U}^{\mathrm{T}} \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{\mathrm{T}}\right)}_{=\mathbf{Q}^{\mathrm{T}}}. \tag{3.1}$$

In essence, the eigenvector matrix $\mathbf{Q}$ of the matrix $\mathbf{A}$ can be written using the eigenvector matrices $\mathbf{Q}_1$, $\mathbf{Q}_2$ of two smaller $\mathcal{S}$-matrices times the eigenvector matrix $\mathbf{U}$ of a symmetric rank-one modified diagonal matrix. The eigenvalues of $\mathbf{A}$ in the diagonal matrix $\boldsymbol{\Omega}$ are the eigenvalues of the symmetric rank-one modified diagonal system.

In the original paper, Chandrasekaran and Gu provide a more elaborated description of the presented divide-and-conquer method showing that the explicit computation of new vectors in the decomposition phase can be avoided and that the vector $\mathbf{z}$ in the conquer phase can be computed on-line by using data from smaller sub-problems. The crucial part in implementing the divide-and-conquer strategy is the efficient computation of the eigendecomposition of the symmetric rank-one modified diagonal system in the conquer phase. This step is surveyed in the next subsection.

Without further optimisations, one obtains an $\mathcal{O}\left(n^2\right)$ algorithm for computing the eigenvalues only and an $\mathcal{O}\left(n^3\right)$ algorithm for computing also the eigenvectors. As mentioned in the next subsection, these algorithms can be accelerated to yield approximate $\mathcal{O}\left(n \log n\right)$ and $\mathcal{O}\left(n^2\right)$ algorithms, respectively, exploiting the matrix structure found in the symmetric rank-one modified diagonal eigenproblem; cf. [5].

3.2. Symmetric rank-one modification of the diagonal eigenproblem. This section briefly describes how to compute the eigendecomposition of a symmetric rank-one modified diagonal matrix $\mathbf{D} + \mathbf{z}\, \mathbf{z}^{\mathrm{T}}$. The problem of determining the eigendecomposition of such a matrix was first formulated by Golub in [14]. There, the eigenvalues are obtained as zeros of a secular equation and the computation of the eigenvectors by inverse iteration is proposed. In [4], Bunch, Nielsen, and Sorensen establish an explicit formula for computing the eigenvectors without inverse iteration and introduce a deflation procedure to handle the case of multiple eigenvalues. A perturbation analysis shows that the computation of the eigenvectors can, under some circumstances, be subject to severe instabilities so that they are possibly far from being numerically orthogonal. It was not before [18] that Gu and Eisenstat showed a stable way to compute numerically orthogonal eigenvectors close to the original ones. For the sake of simplicity, we assume in the following that all eigenvalues, hence diagonal entries, of $\mathbf{D}$ are distinct and that all entries of $\mathbf{z}$ are non-zero. If this is not the case, the deflation procedure in [4] leads to a reduced problem that enjoys the assumed properties. Moreover, we may assume that by permutations we have arranged for that the diagonal entries d_j are lined up in increasing order, i.e. $d_1 < d_2 < \ldots < d_n$. The following theorem, given without proof, restates results obtained in [14] and [4].

THEOREM 3.1. *Let $\mathbf{D}$ be a diagonal matrix with pairwise distinct entries $d_1 < d_2 < \ldots < d_n$, $\mathbf{z}$ be a vector with non-zero entries $z_1, z_2, \ldots, z_n$, and $\rho \neq 0$ be a scalar. Then for the symmetric rank-one modified diagonal matrix $\mathbf{D} + \rho\,\mathbf{z}\,\mathbf{z}^{\mathrm{T}}$ the following statements hold true:*

1. *The eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ have the interlacing property*

$$\begin{cases} d_1 < \lambda_1 < d_2 < \lambda_2 < \ldots < d_n < \lambda_n < d_n + \rho\,\mathbf{z}^{\mathrm{T}}\mathbf{z}, & \text{if } \rho > 0, \\ d_1 + \rho\,\mathbf{z}^{\mathrm{T}}\mathbf{z} < \lambda_1 < d_1 < \lambda_2 < d_2 < \ldots < \lambda_n < d_n, & \text{if } \rho < 0. \end{cases} \tag{3.2}$$

2. *The eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are solutions to the secular equation*

$$f(\lambda) := 1 + \rho \sum_{j=1}^{n} \frac{z_j^2}{d_j - \lambda} = 0. \tag{3.3}$$

3. *For each eigenvalue λ_j, a corresponding $\|\cdot\|_2$-normalised eigenvector $\mathbf{u}_j$ is given by*

$$\mathbf{u}_j = \left(\sum_{m=1}^{n} \frac{z_m^2}{(d_m - \lambda_j)^2} \right)^{-1/2} \cdot \left(\frac{z_1}{d_1 - \lambda_j}, \frac{z_2}{d_2 - \lambda_j}, \ldots, \frac{z_n}{d_n - \lambda_j} \right)^{\mathrm{T}}. \tag{3.4}$$

This theorem provides a way to compute the eigendecomposition of a symmetric rank-one modified diagonal matrix. The eigenvalues λ_j are bounded by the diagonal entries d_j of $\mathbf{D}$ and the inner product $\rho\,\mathbf{z}^{\mathrm{T}}\mathbf{z}$ in (3.2). Furthermore, the eigenvalues are the zeros of a rational function $f(\lambda)$ in (3.3). There exist fast iterative methods finding these zeros to machine precision; see for example [4]. The eigenvectors $\mathbf{u}_j$ have explicit expressions in terms of the entries of the vector $\mathbf{z}$, the diagonal entries d_j and the new eigenvalues λ_j as shows (3.4).

REMARK 3.2. The eigenvector matrix $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n) = (u_{j,k})_{i,j=1}^{n,n}$ is a Cauchy-like matrix with entries

$$u_{i,j} = z_i \cdot \frac{1}{d_i - \lambda_j} \cdot \left(\sum_{m=1}^{n} \frac{z_m^2}{(d_m - \lambda_j)^2} \right)^{-1/2}.$$

It is usually fully populated. Multiplying a vector by this matrix generally takes $\mathcal{O}(n^2)$ arithmetic operations. Using fast summation techniques, an acceleration to $\mathcal{O}(n)$ arithmetic operations can be achieved; see also Section 3.5. The resulting algorithm is then approximate, i.e. accurate up to a prefixed accuracy ε.

REMARK 3.3. The acceleration of matrix-vector multiplications involving Cauchy-like matrices allows for computing all eigenvalues of an $\mathcal{S}$-matrix with $\mathcal{O}(n \log n)$ arithmetic operations and for computing also the eigenvectors with $\mathcal{O}(n^2)$ arithmetic operations. This follows directly from the method described in Section 3.1. It is understood that the algorithms then also become approximate. By using that the eigenvector matrix $\mathbf{Q}$ in the eigendecomposition (3.1) is recursively made up of Cauchy-like matrices $\mathbf{U}$, one also obtains an efficient way to apply $\mathbf{Q}$ and $\mathbf{Q}^{\mathrm{T}} = \mathbf{Q}^{-1}$ to an arbitrary vector. In view of Theorem 3.1, only the entries of the vectors $\mathbf{z}$ and $\mathbf{d}$, and the eigenvalues λ_j which together fully determine each matrix $\mathbf{U}$ need to be stored. Since a single application of a matrix $\mathbf{U}$ or $\mathbf{U}^{\mathrm{T}}$ takes linear time $\mathcal{O}(n)$, this yields an $\mathcal{O}(n \log n)$ algorithm for applying either $\mathbf{Q}$ or $\mathbf{Q}^{\mathrm{T}}$ to an arbitrary vector.

3.3. Diagonal plus upper or lower triangular semiseparable matrices. In [11], Eidelman, Gohberg and Olshevsky give recursive expressions for characteristic polynomials and eigenvectors for quasiseparable matrices of order one which include $\mathcal{U}$- and $\mathcal{L}$-matrices. But as far as we know, the problem of computing the eigendecomposition of $\mathcal{U}$- and $\mathcal{L}$-matrices

$$\mathbf{A} = \operatorname{diag}(\mathbf{d}) + \operatorname{triu}\left(\mathbf{u}\,\mathbf{v}^{\mathrm{T}}\right), \qquad \mathbf{A} = \operatorname{diag}(\mathbf{d}) + \operatorname{tril}\left(\mathbf{v}\,\mathbf{u}^{\mathrm{T}}\right)$$

by divide-and-conquer methods has yet not been studied in the literature. It turns out that the special structure of $\mathcal{U}$- and $\mathcal{L}$-matrices allows for adopting an analogous divide-and-conquer strategy as for $\mathcal{S}$-matrices in Section 3.1. Moreover, the eigenvector matrices to be computed are also instantly inverted and the whole method is even easier: First, the decomposition into smaller sub-problems is done only by taking principal sub-matrices, i.e. avoiding explicit computations of new quantities right from the beginning. Second, the eigenvalues of all appearing matrices are known a priori, since they are triangular. We recall that we assume that all eigenvalues are distinct. In the following, we will restrict ourselves to the treatment of $\mathcal{U}$-matrices. In the case of $\mathcal{L}$-matrices a simple transpose reduces everything to applying the results for $\mathcal{U}$-matrices.

3.3.1. Divide phase. For the decomposition, we start again with a representation of a $\mathcal{U}$-matrix $\mathbf{A} = \operatorname{diag}(\mathbf{d}) + \operatorname{triu}\left(\mathbf{u}\,\mathbf{v}^{\mathrm{T}}\right) \in \mathbb{R}^n$ by two smaller matrices of the same type, plus, new in this case, a particular unsymmetric rank-one modification,

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{pmatrix} + \tilde{\mathbf{u}}\,\tilde{\mathbf{v}}^{\mathrm{T}}.$$

As before we let $0 \le k \le n$ and define the principal submatrices $\mathbf{A}_1$ and $\mathbf{A}_2$ by

$$\mathbf{A}_j = \operatorname{diag}(\mathbf{d}_j) + \operatorname{triu}\left(\mathbf{u}_j\,\mathbf{v}_j^{\mathrm{T}}\right) \qquad (j = 1, 2).$$

The vectors $\tilde{\mathbf{u}} \in \mathbb{R}^n$ and $\tilde{\mathbf{v}} \in \mathbb{R}^n$ account for the upper right block of $\mathbf{A}$ and are given by

$$\tilde{\mathbf{u}} := \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{v}} := \begin{pmatrix} \mathbf{0} \\ \mathbf{v}_2 \end{pmatrix}.$$

It is not difficult to see that the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are $\mathcal{U}$-matrices. They can be decomposed recursively in the same style. Note that the entries of $\mathbf{d}$ are the eigenvalues of $\mathbf{A}$ and that the elements of $\mathbf{d}_1$ and $\mathbf{d}_2$ are the eigenvalues of the submatrices $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively. That is, the eigenvalues of $\mathbf{A}$ are the disjoint union of the eigenvalues of $\mathbf{A}_1$ and $\mathbf{A}_2$. In comparison to $\mathcal{S}$-matrices, the additional rank-one modification $\tilde{\mathbf{u}}\,\tilde{\mathbf{v}}^{\mathrm{T}}$ is unsymmetric, but of a particular type, since it modifies only the upper right $k \times (n-k)$ block of $\mathbf{A}$. The decomposition is almost trivial and does not require the computation of any new quantity.

3.3.2. Conquer phase. Let k be defined as before. Suppose that the eigendecomposition of two smaller matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ has already been computed. Since both matrices are upper triangular and have real eigenvalues, this reads

$$\mathbf{A}_1 = \mathbf{Q}_1\,\mathbf{D}_1\,\mathbf{Q}_1^{-1}, \qquad\qquad \mathbf{A}_2 = \mathbf{Q}_2\,\mathbf{D}_2\,\mathbf{Q}_2^{-1},$$

with $\mathbf{D}_1 := \operatorname{diag}(\mathbf{d}_1)$ and $\mathbf{D}_2 := \operatorname{diag}(\mathbf{d}_2)$. Notice that owing to the argument above, the eigenvalues contained in $\mathbf{d_1}$ and $\mathbf{d}_2$ are known a priori. Notice further that the eigenvector matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are not orthogonal but upper triangular. As we will see later, we can obtain the inverse matrices $\mathbf{Q}_1^{-1}$ and $\mathbf{Q}_2^{-1}$ without any extra computation directly from $\mathbf{Q}_1$ and $\mathbf{Q}_2$. Using the eigendecomposition of $\mathbf{A}_1$ and $\mathbf{A}_2$, we now obtain for $\mathbf{A}$ the representation

$$\mathbf{A} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix} \left(\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}\right) \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{-1}$$

with the vectors

$$\mathbf{w} := \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{-1} \tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{Q}_1^{-1}\,\mathbf{u}_1 \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{z} := \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{\mathrm{T}} \tilde{\mathbf{v}} = \begin{pmatrix} \mathbf{0} \\ \mathbf{Q}_2^{\mathrm{T}}\,\mathbf{v}_2 \end{pmatrix}. \tag{3.5}$$

The eigendecomposition of the unsymmetric rank-one modified diagonal matrix $\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}$ can be written as

$$\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}} = \mathbf{U}\,\mathbf{D}\,\mathbf{U}^{-1}. \tag{3.6}$$

The eigenvalues are the a priori known elements of the diagonal matrix $\mathbf{D} := \operatorname{diag}(\mathbf{d})$. Using this representation, we finally obtain the eigendecomposition of $\mathbf{A}$ as

$$\mathbf{A} = \underbrace{\left(\begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix} \mathbf{U}\right)}_{=:\mathbf{Q}} \mathbf{D} \underbrace{\left(\mathbf{U}^{-1} \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{-1}\right)}_{=\mathbf{Q}^{-1}}.$$

Let us check if the described conquer strategy can be carried out efficiently. To this end, fix a single conquer step that obtains the eigendecomposition of a matrix $\mathbf{A}$ using the eigendecompositions $\mathbf{A}_1 = \mathbf{Q}_1\,\mathbf{D}_1\,\mathbf{Q}_1^{-1}$ and $\mathbf{A}_2 = \mathbf{Q}_2\,\mathbf{D}_2\,\mathbf{Q}_2^{-1}$ of two smaller matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ obtained in the divide phase. Let us also assume that $\mathbf{A}$ itself has been obtained as the upper left principal submatrix $\bar{\mathbf{A}}_1$ or the lower right principal submatrix $\bar{\mathbf{A}}_2$ of a bigger $\mathcal{U}$-matrix $\bar{\mathbf{A}} = \operatorname{diag}(\bar{\mathbf{d}}) + \operatorname{triu}(\bar{\mathbf{u}}\,\bar{\mathbf{v}}^{\mathrm{T}})$.

The conquer procedure consists of two parts: the computation of the vectors $\mathbf{w}$ and $\mathbf{z}$ as in the defining equation (3.5), and second, the computation of the eigenvector matrix $\mathbf{U}$ of the modified diagonal system in the eigendecomposition (3.6). The latter problem will be dealt with in the next subsection. The former problem can be much simplified by using data from previous conquer steps. We apply an inductive idea and therefore have to distinguish between conquer steps using data from the bottom of the divide phase, and the remaining steps above. We can avoid explicitly computing the vectors $\mathbf{w}$ and $\mathbf{z}$ by formula (3.5) in every conquer step and are able to use an on-line updating technique instead.

Assume first that the eigenvector matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ and also their inverses $\mathbf{Q}_1^{-1}$ and $\mathbf{Q}_2^{-1}$ appearing in the conquer step have been computed explicitly by solving the smaller eigenproblems directly. In this case, we can use formula (3.5) to compute the vectors $\mathbf{w}$ and $\mathbf{z}$

directly. Let us also assume that we have solved the unsymmetric rank-one modified diagonal eigenproblem in (3.6). Then we also have $\mathbf{U}$ and $\mathbf{U}^{-1}$ in the equation $\mathbf{D}+\mathbf{w}\,\mathbf{z}^{\mathrm{T}} = \mathbf{U}\,\mathbf{D}\,\mathbf{U}^{-1}$. Now, in addition, we compute also the vectors

$$\tilde{\mathbf{w}} := \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{Q}_2^{-1}\,\mathbf{u}_2 \end{pmatrix},$$
$$\tilde{\mathbf{z}} := \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_1^{\mathrm{T}}\,\mathbf{v}_1 \\ \mathbf{0} \end{pmatrix}.$$

If now $\mathbf{A} = \bar{\mathbf{A}}_1$ with respect to the bigger matrix $\bar{\mathbf{A}}$, we can write

$$\mathbf{U}^{-1}\,(\mathbf{w}+\tilde{\mathbf{w}}) = \mathbf{U}^{-1} \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \bar{\mathbf{Q}}_1^{-1}\,\bar{\mathbf{u}}_1,$$

$$\mathbf{U}^{\mathrm{T}}\,(\mathbf{z}+\tilde{\mathbf{z}}) = \mathbf{U}^{\mathrm{T}} \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \bar{\mathbf{Q}}_1^{\mathrm{T}}\,\bar{\mathbf{v}}_1.$$

Similarly, if we have $\mathbf{A} = \bar{\mathbf{A}}_2$, computing the same quantities yields $\mathbf{U}^{-1}\,(\mathbf{w}+\tilde{\mathbf{w}}) = \bar{\mathbf{Q}}_2^{-1}\,\bar{\mathbf{u}}_2$ and $\mathbf{U}^{\mathrm{T}}\,(\mathbf{z}+\tilde{\mathbf{z}}) = \bar{\mathbf{Q}}_2^{\mathrm{T}}\,\bar{\mathbf{v}}_2$. This means that by performing the conquer steps for $\mathbf{A} = \bar{\mathbf{A}}_1$ and $\mathbf{A} = \bar{\mathbf{A}}_2$ in the manner just described, we have already computed the data needed in the conquer step for $\bar{\mathbf{A}}$ which would otherwise have to be computed by formula (3.5).

Suppose now that with respect to $\mathbf{A}$, the eigendecompositions of the submatrices $\mathbf{A}_1$, $\mathbf{A}_2$ have been obtained and that the matrices $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_1^{-1}$, and $\mathbf{Q}_2^{-1}$ have not been computed explicitly. But we assume that the vectors $\mathbf{Q}_1^{-1}\,\mathbf{u}_1$, $\mathbf{Q}_2^{-1}\,\mathbf{u}_2$, $\mathbf{Q}_1^{\mathrm{T}}\,\mathbf{v}_1$, and $\mathbf{Q}_2^{\mathrm{T}}\,\mathbf{v}_2$ have already been computed. This means nothing else than that we know the vectors $\mathbf{w}$, $\tilde{\mathbf{w}}$, $\mathbf{z}$, and $\tilde{\mathbf{z}}$. Now we can immediately proceed to solving the unsymmetric modified diagonal eigenproblem and obtain the eigenvector matrix $\mathbf{U}$ and its inverse $\mathbf{U}^{-1}$ in $\mathbf{D}+\mathbf{w}\,\mathbf{z}^{\mathrm{T}} = \mathbf{U}\,\mathbf{D}\,\mathbf{U}^{-1}$. Once solved this problem, we again compute the vectors $\mathbf{U}^{-1}\,(\mathbf{w}+\tilde{\mathbf{w}})$ and $\mathbf{U}^{\mathrm{T}}\,(\mathbf{z}+\tilde{\mathbf{z}})$ which can be used in the conquer step for $\bar{\mathbf{A}}$. This inductive idea can be used until we compute the eigendecomposition of the initial matrix where we can stop after solving the unsymmetric rank-one modified diagonal eigenproblem.

In essence, this procedure shows that in each conquer step, one only needs to solve the unsymmetric rank-one modified diagonal eigenproblem, and to multiply two vectors by the newly obtained matrices $\mathbf{U}^{-1}$ and $\mathbf{U}^{\mathrm{T}}$. Algorithms 1 and 2 provide a MATLAB implementation of this method where Algorithm 1 references Algorithm 2 to solve the unsymmetric modified diagonal eigenproblem. We summarise the whole divide-and-conquer method in the following theorem.

THEOREM 3.4. *Let* $\mathbf{A} = \mathrm{diag}\,(\mathbf{d}) + \mathrm{triu}\,\left(\mathbf{u}\,\mathbf{v}^{\mathrm{T}}\right) \in \mathbb{R}^{n\times n}$ *be a* $\mathcal{U}$*-matrix and* $0 \le k \le n$*. Then* $\mathbf{A}$ *can be written as*

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{pmatrix} + \tilde{\mathbf{u}}\,\tilde{\mathbf{v}}^{\mathrm{T}}, \qquad \mathbf{A}_j = \mathrm{diag}\,(\mathbf{d}_j) + \mathrm{triu}\,\left(\mathbf{u}_j\,\mathbf{v}_j^{\mathrm{T}}\right),$$
$$\tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{v}} = \begin{pmatrix} \mathbf{0} \\ \mathbf{v}_2 \end{pmatrix},$$

for $j = 1, 2$*. Given the eigendecompositions* $\mathbf{A}_1 = \mathbf{Q}_1\,\mathbf{D}_1\,\mathbf{Q}_1^{-1}$ *and* $\mathbf{A}_2 = \mathbf{Q}_2\,\mathbf{D}_2\,\mathbf{Q}_2^{-1}$*, and the matrix* $\mathbf{U}$ *from the unsymmetric rank-one modified diagonal eigenproblem*

$$\mathbf{D}+\mathbf{w}\,\mathbf{z}^{\mathrm{T}} = \mathbf{U}\,\mathbf{D}\,\mathbf{U}^{-1}, \quad \mathbf{D} = \mathrm{diag}\,(\mathbf{d}), \quad \mathbf{w} = \begin{pmatrix} \mathbf{Q}_1^{-1}\,\mathbf{u}_1 \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} \mathbf{0} \\ \mathbf{Q}_2^{\mathrm{T}}\,\mathbf{v}_2 \end{pmatrix},$$

the eigendecomposition of $\mathbf{A}$ *is*

$$\mathbf{A} = \mathbf{Q}\,\mathbf{D}\,\mathbf{Q}^{-1}, \qquad \mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{pmatrix} \mathbf{U}. \tag{3.7}$$

REMARK 3.5. Since the eigenvalues of a $\mathcal{U}$-matrix are known a priori, one is only interested in computing its eigenvectors. For an $n \times n$ matrix, Algorithm 1 computes all eigenvectors with $\mathcal{O}\left(n^3\right)$ arithmetic operations if in each conquer step all products involving matrices are computed directly. Explicit computation can be avoided, if one is only interested in applying the eigenvector matrix or its inverse to a vector. For a matrix $\mathbf{A}$ as in Theorem 3.4, it suffices to store all diagonal entries from $\mathbf{d}$ and the vectors $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{z}}$ appearing in each conquer step. Then one can apply either the eigenvector matrix $\mathbf{Q}$ or its inverse $\mathbf{Q}^{-1}$ by using the recursive definition (3.7) and computing the entries of each matrix $\mathbf{U}$ and $\mathbf{U}^{-1}$ on the fly. In the next subsection, we show that the particular structure found in the eigenvector matrix $\mathbf{U}$ from the particular unsymmetric rank-one modified diagonal eigenproblem allows for accelerating the algorithms. This will yield an $\mathcal{O}(n^2)$ algorithm for computing the eigenvectors explicitly and an $\mathcal{O}\left(n \log n\right)$ algorithm for applying the eigenvector matrix to an arbitrary vector. Again, these algorithms will be approximate up to a prefixed accuracy ε.

3.4. Particular unsymmetric rank-one modification of the diagonal eigenproblem. In this section, we study the eigendecomposition of a particular unsymmetric rank-one modified diagonal matrix $\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}} \in \mathbb{R}^n$ with $\mathbf{D} = \operatorname{diag}\left(\mathbf{d}\right)$ a diagonal matrix and vectors

$$\begin{aligned} \mathbf{d} &:= \left(d_1, d_2, \ldots, d_n\right), \\ \mathbf{w} &:= \left(w_1, w_2, \ldots, w_k, 0, 0, \ldots, 0\right)^{\mathrm{T}}, \\ \mathbf{z} &:= \left(0, 0, \ldots, 0, z_{k+1}, z_{k+2}, \ldots, z_n\right)^{\mathrm{T}}. \end{aligned}$$

We assume that the entries of $\mathbf{d}$ are distinct. As before, we let $0 \leq k \leq n$. The problem is different from the symmetric one treated in Section 3.2 since the rank-one modification $\mathbf{w}\,\mathbf{z}^{\mathrm{T}}$ to the diagonal matrix $\mathbf{D}$ is now unsymmetric. On the other hand, the special structure, i.e. the last $n-k$ entries of $\mathbf{w}$ and the first k entries in $\mathbf{z}$ being zero, yields a modification only in the upper right part of $\mathbf{D}$. This leaves everything on and below the main diagonal unchanged. The matrix $\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}$ has the form

$$\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}} = \begin{pmatrix} d_1 & 0 & \ldots & 0 & v_1 w_{k+1} & v_1 w_{k+2} & \ldots & v_1 w_n \\ 0 & d_2 & \ddots & \vdots & v_2 w_{k+1} & v_2 w_{k+2} & & v_2 w_n \\ \vdots & \ddots & \ddots & 0 & \vdots & \vdots & \ddots & \vdots \\ \vdots & & \ddots & d_k & v_k w_{k+1} & v_k w_{k+2} & \ldots & v_k w_n \\ \vdots & & & \ddots & d_{k+1} & 0 & \ldots & 0 \\ \vdots & & & & \ddots & d_{k+2} & \ddots & \vdots \\ \vdots & & & & & \ddots & \ddots & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0 & d_n \end{pmatrix}.$$

The facts about the eigendecomposition of $\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}$ are summarised in the following theorem.

THEOREM 3.6. *Let* $\mathbf{D} \in \mathbb{R}^{n \times n}$ *be a diagonal matrix with pairwise distinct diagonal entries* $d_1, d_2, \ldots, d_n$ *and vectors*

$$\mathbf{w} := \left(w_1, w_2, \ldots, w_k, 0, 0, \ldots, 0\right)^{\mathrm{T}} \in \mathbb{R}^n, \quad \mathbf{z} := \left(0, 0, \ldots, 0, z_{k+1}, z_{k+2}, \ldots, z_n\right)^{\mathrm{T}} \in \mathbb{R}^n.$$

Then for the unsymmetric rank-one modified diagonal matrix $\mathbf{A} = \mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}$*, the following statements hold true:*

1. *The eigenvalues of* $\mathbf{D}$ *and* $\mathbf{A}$ *are* $d_1, d_2, \ldots, d_n$*.*
2. *The matrix* $\mathbf{A}$ *has the eigendecomposition* $\mathbf{A} = \mathbf{U}\,\mathbf{D}\,\mathbf{U}^{-1}$*. The eigenvector matrix* $\mathbf{U}$*, containing* $\|\cdot\|_2$*-normalised eigenvectors of* $\mathbf{A}$ *and its inverse* $\mathbf{U}^{-1}$ *have the form*

$$\mathbf{U} = \begin{pmatrix} \mathbf{I} & \mathbf{C}\,\tilde{\mathbf{D}} \\ \mathbf{0} & \tilde{\mathbf{D}} \end{pmatrix}, \qquad \mathbf{U}^{-1} = \begin{pmatrix} \mathbf{I} & -\mathbf{C} \\ \mathbf{0} & \tilde{\mathbf{D}}^{-1} \end{pmatrix}, \tag{3.8}$$

where $\mathbf{I}$ *is the* $k \times k$ *identity matrix,* $\tilde{\mathbf{D}} := \mathrm{diag}(\tilde{\mathbf{d}})$ *is an* $(n-k) \times (n-k)$ *diagonal matrix with non-zero entries* $\tilde{d}_{k+1}, \tilde{d}_{k+2}, \ldots, \tilde{d}_n$ *and* $\mathbf{C}$ *is a* $k \times (n-k)$ *matrix.*

3. *The components of the diagonal matrix* $\tilde{\mathbf{D}}$ *from (ii) are*

$$\tilde{d}_j = \left(1 + \sum_{i=1}^{k} \frac{w_i^2\, z_j^2}{(d_i - d_j)^2}\right)^{-1/2} > 0 \qquad (j = k+1, k+2, \ldots, n),$$

and the matrix $\mathbf{C}$ *is a Cauchy-like matrix* $\mathbf{C} = (c_{i,j})_{i=1,\, j=k+1}^{k,n}$ *whose components are*

$$c_{i,j} = \frac{w_i\, z_j}{d_i - d_j} \qquad (i = 1, 2, \ldots, k;\; j = k+1, k+2, \ldots, n).$$

Proof.

1. The matrix $\mathbf{D}$ is diagonal so that its entries $d_1, d_2, \ldots, d_n$ coincide with its eigenvalues. Since $\mathbf{A} = \mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}$ is an upper triangular matrix with the same diagonal elements as $\mathbf{D}$, this also holds for $\mathbf{A}$.
2. The matrix $\mathbf{A}$ is unsymmetric but has real eigenvalues as a consequence of (1). Therefore, the eigendecomposition of $\mathbf{A}$ reads $\mathbf{A} = \mathbf{U}\,\mathbf{D}\,\mathbf{U}^{-1}$ with $\mathbf{U}$ being the eigenvector matrix of $\mathbf{A}$. To show that $\mathbf{U}$ has the form

$$\mathbf{U} = \begin{pmatrix} \mathbf{I} & \mathbf{C}\,\tilde{\mathbf{D}} \\ \mathbf{0} & \tilde{\mathbf{D}} \end{pmatrix},$$

we first show that the first k unit vectors $\mathbf{e}_i \in \mathbb{R}^n$ are eigenvectors $\mathbf{u}_i := \mathbf{e}_i$ of $\mathbf{A}$ which correspond to the eigenvalues d_i. We have

$$\begin{aligned} \mathbf{A}\,\mathbf{e}_i &= \left(\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}\right)\mathbf{e}_i = \mathbf{D}\,\mathbf{e}_i + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}\,\mathbf{e}_i \\ &= d_i\,\mathbf{e}_i + \mathbf{u}\,\underbrace{(0, 0, \ldots, 0, z_{k+1}, z_{k+2}, \ldots, z_n)\,\mathbf{e}_i}_{=0} = d_i\,\mathbf{e}_i \quad (i = 1, 2, \ldots, k). \end{aligned}$$

This implies that the first k columns of $\mathbf{U}$, in MATLAB notation $\mathbf{U}\,(\,:, 1:k)$, have the form

$$\mathbf{U}\,(\,:, 1:k) = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k) = \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix}.$$

It remains to prove that for the rest of the eigenvectors $\mathbf{u}_{k+1}, \mathbf{u}_{k+2}, \ldots, \mathbf{u}_n$ corresponding to the eigenvalues $d_{k+1}, d_{k+2}, \ldots, d_n$, one has

$$\mathbf{U}\,(\,:, k+1:n) = (\mathbf{u}_{k+1}, \mathbf{u}_{k+2}, \ldots, \mathbf{u}_n) = \begin{pmatrix} \mathbf{C}\,\tilde{\mathbf{D}} \\ \tilde{\mathbf{D}} \end{pmatrix}.$$

It suffices to show that for the jth component of the eigenvector $\mathbf{u}_i$, denoted $(\mathbf{u}_i)_j$, we have $(\mathbf{u}_i)_j = \tilde{d}_j \delta_{i,j}$ for $i, j = k+1, k+2, \ldots, n$ and constants $\tilde{d}_j \neq 0$. If we fix i, we have on one hand $\mathbf{A}\,\mathbf{u}_i = d_i\,\mathbf{u}_i$, and on the other, the structure of $\mathbf{A}$ gives $\mathbf{A}\ \mathbf{u}_i = \left(\mathbf{D} + \mathbf{w}\,\mathbf{z}^{\mathrm{T}}\right)\mathbf{u}_i$. Combining these equations and using that $\mathbf{w}\,\mathbf{z}^{\mathrm{T}}$ is a matrix with zeros in rows $k+1, k+2, \ldots, n$, we get for the jth component the equations

$$d_i\,(\mathbf{u}_i)_j = d_j\ (\mathbf{u}_i)_j \quad (j = k+1, k+2, \ldots, n).$$

Since $d_i \neq d_j$ while $i \neq j$ by assumption, this gives the desired property by taking into account that $\tilde{d}_i := (\mathbf{u}_i)_i$ itself cannot be zero since this would render the matrix $\mathbf{U}$ singular. Finally, it is not difficult to verify the form of the inverse matrix $\mathbf{U}^{-1}$.

3. We first show that the upper right block of the eigenvector matrix $\mathbf{U}$ has the representation $\mathbf{C}\,\tilde{\mathbf{D}}$ with $\mathbf{C}$ a Cauchy-like matrix $\mathbf{C} = (c_{i,j})_{i=1,\,j=k+1}^{k,n}$ whose components are

$$c_{i,j} = \frac{w_i\,z_j}{d_i - d_j} \quad (i = 1, 2, \ldots, k;\ j = k+1, k+2, \ldots, n).$$

For fixed j, the eigenvector $\mathbf{u}_j$ and the eigenvalue d_j form an eigenpair of $\mathbf{A}$, i.e. $\mathbf{A}\,\mathbf{u}_j = d_j\,\mathbf{u}_j$, or equivalently, $\left(\mathbf{D} + \mathbf{w}\mathbf{z}^{\mathrm{T}}\right)\ \mathbf{u}_j = d_j\,\mathbf{u}_j$ for $j = 1, 2, \ldots, n$. By subtracting $d_j\,\mathbf{u}_j$ and $\mathbf{w}\,\mathbf{z}^{\mathrm{T}}\ \mathbf{u}_j$ on both sides and rearranging terms, we obtain

$$(\mathbf{D} - d_j\mathbf{I})\,\mathbf{u}_j = -\mathbf{w}\mathbf{u}_j^{\mathrm{T}}\mathbf{z}.$$

For the ith component in this equation, $1 \leq i \leq k$, we have

$$\left((\mathbf{D} - d_j\mathbf{I})\,\mathbf{u}_j\right)_i = (d_i - d_j)\,(\mathbf{u}_j)_i = -\left(\mathbf{w}\mathbf{u}_j^{\mathrm{T}}\mathbf{z}\right)_i.$$

As a consequence of the fact that the first k components of $\mathbf{z}$ are zero, and that $\mathbf{u}_j$'s last $(n-k)$ components, except $(\mathbf{u}_j)_j = \tilde{d}_j \neq 0$, are zero, we have $\mathbf{u}_j^{\mathrm{T}}\mathbf{z} = \tilde{d}_j z_j$. This gives $(d_i - d_j)\,(\mathbf{u}_j)_i = w_i \tilde{d}_j z_j$ and finally

$$(\mathbf{u}_j)_i = \frac{w_i\,z_j}{d_i - d_j}\tilde{d}_j.$$

We have now obtained the desired representation $\mathbf{U}(1:k, k+1:n) = \mathbf{C}\,\tilde{\mathbf{D}}$ for the upper right block of $\mathbf{U}$. To get an explicit expression for the values $\tilde{d}_j$, recall that each eigenvector $\mathbf{u}_j$ is normalised so that

$$\|\mathbf{u}_j\|_2^2 = \sum_{i=1}^{k}\left(\frac{w_i\,z_j}{d_i - d_j}\tilde{d}_j\right)^2 + \tilde{d}_j^2 = \left(\sum_{i=1}^{k}\left(\frac{w_i\,z_j}{d_i - d_j}\right)^2 + 1\right)\tilde{d}_j^2 = 1.$$

This implies

$$\tilde{d}_j = \left(1 + \sum_{i=1}^{k}\frac{w_i^2\,z_j^2}{(d_i - d_j)^2}\right)^{-1/2}. \ \square$$

REMARK 3.7. In some cases, it might be favourable to use a different normalisation of the eigenvectors by setting $\tilde{d}_j = 1$, $j = k+1, \ldots, n$. In this case, all diagonal elements of $\mathbf{U}$ are equal to one and the matrices take the simple form

$$\mathbf{U} = \begin{pmatrix} \mathbf{I} & \mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \qquad \mathbf{U}^{-1} = \begin{pmatrix} \mathbf{I} & -\mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

A MATLAB implementation which computes $\mathbf{U}$ as above is given in Algorithm 2. With respect to Algorithm 1 from Section 3.3, this ensures that in a conquer step (3.7), the diagonal elements of $\mathbf{Q}$ coincide with the diagonal elements of the smaller eigenvector matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$. Choosing the scaling in the initially and explicitly computed eigenvector matrices such that their diagonal elements are all equal to one, this forces the resulting matrices $\mathbf{Q}$ and $\mathbf{Q}^{-1}$ for the whole problem to contain only ones on their respective main diagonal. One might then apply a normalisation in $\mathbf{Q}$ and $\mathbf{Q}^{-1}$ based on the diagonal elements by an appropriate eigenvector scaling. This corresponds to a multiplication of the eigenvector matrix $\mathbf{Q}$ by a diagonal matrix $\tilde{\mathbf{D}}$ from the right and a multiplication of the inverse matrix $\mathbf{Q}^{-1}$ by the inverse diagonal matrix $\tilde{\mathbf{D}}^{-1}$ from the left.

REMARK 3.8. Similarly to $\mathcal{S}$-matrices, the particular structure of the matrices $\mathbf{U}$ and $\mathbf{U}^{-1}$ in (3.8) allows for using fast summation techniques. This directly yields an $\mathcal{O}\left(n^2\right)$ algorithm for computing all eigenvectors of a $\mathcal{U}$-matrix $\mathbf{A}$ explicitly. If one, however, is only interested in applying the eigenvector matrix to a vector, their explicit computation can be avoided. It suffices to compute and store all the vectors $\mathbf{w}$, $\tilde{\mathbf{w}}$, $\mathbf{z}$, and $\tilde{\mathbf{z}}$ as well as the eigenvalues $d_1, d_2, \ldots, d_n$. Then one can apply the matrices $\mathbf{Q}$ and $\mathbf{Q}^{-1}$ of $\mathbf{A}$ with $\mathcal{O}\left(n \log n\right)$ arithmetic operations. The accelerated algorithms are approximate up to a prefixed accuracy ε.

3.5. Acceleration by fast summation techniques. As mentioned in previous sections, Cauchy-like matrices allow for an accelerated matrix-vector multiplication. For an $n \times n$ Cauchy-like matrix $\mathbf{C}$, the cost can be reduced from $\mathcal{O}\left(n^2\right)$ to typically $\mathcal{O}(n)$ or $\mathcal{O}\left(n \log n\right)$. This is based on algorithms which were developed for fast evaluation of sums of the form

$$z_j = \sum_{i=1}^{N} \alpha_i K\left(\mathbf{x}_i, \mathbf{y}_j\right) \qquad \left(\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^n;\ M, N \in \mathbb{N}\right)$$

involving a multivariate kernel function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. In most cases, the dependence on the two variables $\mathbf{x}$ and $\mathbf{y}$ is such that the value of $K(\mathbf{x}, \mathbf{y})$ is a function of the distance $d(\mathbf{x}, \mathbf{y})$ given by some metric $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ on $\mathbb{R}^n$. Sums of this form arise in many fields in applied mathematics and gave rise to the *Fast Multipole Method (FMM)* introduced in [17]. Since then, many improved and alternative algorithms have been proposed; see for example [10, 44]. The concept of *hierarchical or $\mathcal{H}$-matrices* (Hackbusch [19] and Hackbusch, Khoromskij, and Sauter [20]) also resembles this technique. Recently, new kernel-independent fast multipole methods have been proposed in [28, 45] which rely entirely on numerical linear algebra. A second aspect that has to be taken into account when using fast summation techniques is that the arithmetic cost strongly depends on the distribution of the nodes $\mathbf{x}_i$ and $\mathbf{y}_j$. Under some assumptions on the distribution of the nodes, one is usually able to prove that the algorithm yields the desired linear arithmetic complexity bound $\mathcal{O}(n)$. A modification to the scheme that makes it retain linear complexity also for other unfavourable node distributions is given by Nabors, Korsmeyer, Leighton, and White in [31]. For our tests in Section 5, we implemented the method from [28] in MATLAB.

ETNA
Kent State University
http://etna.math.kent.edu

Algorithm 1 Divide-and-Conquer Algorithm for the Eigendecomposition of $\mathcal{U}$-Matrices

```
function Q = eigu(d,u,v,w)
% Eigenvectors of U-matrices
d = d(:); u = u(:); v = v(:);
if (length(d) == 1)
  Q = w(1); return;
end
% Divide phase
n = length(d); t = ceil(log2(n)); ns = zeros(1,pow2(t));
ns(1)= n;
for i = 1:t
  for j = 2^(i-1):-1:1
    ns(2*j) = floor(ns(j)/2); ns(2*j-1) = ceil(ns(j)/2);
  end
end
% Conquer phase
wt = zeros(n,1); zt = zeros(n,1); Q = zeros(n);
for i = t:-1:1
  offset = 1;
  for j = 1:2^(i-1)
    j1 = 2*j-1; j2 = 2*j; l1 = ns(j1); l2 = ns(j2);
    n1 = offset + l1 - 1; n2 = n1 + l2; i1 = offset:n1;
    i2 = (n1+1):n2; ind = [i1,i2];
    if i == t
      A1 = umatrix(d(i1),u(i1),v(i1));
      A2 = umatrix(d(i2),u(i2),v(i2));
      [Q(i1,i1),Dt] = eig(A1); [Q(i2,i2),Dt] = eig(A2);
      wt(ind) = [Q(i1,i1)\u(i1);Q(i2,i2)\u(i2)];
      zt(ind) = [Q(i1,i1)’*v(i1);Q(i2,i2)’*v(i2)];
    end
    len = l1 +l2; C = eigdpwz(d(ind),wt(i1),zt(i2));
    Q(i1,i2) = Q(i1,i1)*C;
    if (i > 1 && l1 > 0 && l2 > 0)
      wt(i1) = wt(i1) - C*wt(i2); zt(i2) = C’*zt(i1) + zt(i2);
    end
    offset = offset + len; ns(j) = len;
  end
end
% Normalisation
if (exist(’w’,’var’)==false)
  for i=1:n Q(:,i) = Q(:,i)./norm(Q(:,i)); end
else
  for i=1:n Q(:,i) = w(i)*Q(:,i); end
end
```

Algorithm 2 Algorithm for the Particular Unsymmetric Modified Diagonal Eigenproblem

```
function [C] = eigdpwz(d,w,z)
% Eigenvectors of diagonal matrix plus upper right
% rank-one modification.
d = d(:); w = w(:); z = z(:);
n = length(d); n2 = length(w); i1 = 1:n2; i2 = (n2+1):n;
[D1,D2] = meshgrid(d(i2),d(i1)); [W,Z] = meshgrid(z,w);
C = (W.*Z)./(D1-D2);
```

4. Fast Gegenbauer polynomial transforms.

In this section, we show that the conversion between different expansions in terms of Gegenbauer polynomials corresponds to the eigenvector matrix of an explicitly known $\mathcal{U}$-matrix. The connection is new and is a direct consequence of well-known properties of Gegenbauer polynomials. It should be noted that the concept is so general that it admits generalisations to other classes of orthogonal polynomials and functions if the same or similar properties are available. Sections 4.1 to 4.3 restate basic definitions and properties of orthogonal polynomials found in the standard references [6, 40, 1]. In Section 4.4, we first show that the sought transformation can be represented as the eigenvector matrix to a certain matrix which is constructed from the differential equations of the two involved families of Gegenbauer polynomials. Explicit expressions for the entries of this matrix reveal that it is actually a $\mathcal{U}$-matrix. We also show how to scale the eigenvectors correctly.

4.1. Orthogonal polynomials.

All polynomials in this work are defined on a finite real domain and have real coefficients. A sequence of orthogonal polynomials $f_0, f_1, \ldots$ on a finite real domain $[a, b] \subset \mathbb{R}$ is a sequence in which each polynomial $f_n = k_n x^n + k'_n x^{n-1} + \ldots$ has precisely degree n, i.e. $k_n \neq 0$, and is orthogonal in the Hilbert space of polynomials $\mathbb{P}$ equipped with inner product $\langle \cdot , \cdot \rangle$ and induced norm $\|\cdot\|$,

$$\langle f, g\rangle := \int_a^b f(x)g(x)\, w(x)\, \mathrm{d}x, \qquad \|f\| := \sqrt{\langle f, f\rangle} = \left(\int_a^b (f(x))^2\ w(x)\ \mathrm{d}x\right)^{1/2}.$$

We write $\langle f_n, f_m\rangle = h_n \delta_{n,m}$ with the usual Kronecker delta and

$$h_n := \langle f_n, f_n\rangle = \|f_n\|^2 > 0.$$

The restriction of $\mathbb{P}$ to the space of polynomials of degree at most n is denoted $\mathbb{P}_n$. The function $w(x)$ in the integrals is a suitable weight function which is strictly positive inside the interval, possibly zero or singular at the endpoints and for which the integral against any polynomial is finite. If the sequence $f_0, f_1, \ldots$ is infinite, it comprises a basis for the infinite dimensional space $\mathbb{P}$ and is uniquely determined up to a constant factor in each polynomial. For the sake of concise notation, we add $f_{-1}(x) := 0$ with $h_{-1} := 1$ to the sequence of polynomials. Every sequence of orthogonal polynomials has a three-term recurrence formula

$$f_{n+1}(x) = (\tilde{a}_n x + \tilde{b}_n)\, f_n(x) - \tilde{c}_n\, f_{n-1}(x) \quad (n \geq 0,\ \tilde{a}_n \neq 0).$$

After rearranging terms, one verifies the equivalent expression

$$x f_n(x) = a_n\, f_{n-1}(x) - b_n\, f_n(x) + c_n f_{n+1}(x) \tag{4.1}$$

with the coefficients $a_n := \tilde{c}_n/\tilde{a}_n$, $b_n := \tilde{b}_n/\tilde{a}_n$, and $c_n := 1/\tilde{a}_n$. It is not difficult to see that the leading coefficient k_n of the polynomial f_n is $k_n = \tilde{a}_{n-1} \cdot \ldots \cdot \tilde{a}_0 \cdot k_0$ for $n \geq 1$, where k_0 is given by $k_0 = (\frac{1}{h_0}\int_a^b w(x)\ \mathrm{d}x)^{-1/2}$.

4.2. Orthogonal polynomials from hypergeometric differential equations. The family of classical orthogonal polynomials arises from the differential equation of hypergeometric type,

$$\sigma(x)y''(x) + \tau(x)y'(x) + \lambda y(x) = 0, \tag{4.2}$$

where $\sigma \in \mathbb{P}_2$, $\tau \in \mathbb{P}_1$, and $\lambda \in \mathbb{R}$. We define an associated differential operator $\mathcal{D}$ by

$$\mathcal{D}(y) := -\sigma y'' - \tau y'$$

which implies that the solutions y of (4.2) are eigenfunctions of $\mathcal{D}$ to the eigenvalues λ, i.e., $\mathcal{D}(y) = \lambda y$. The differential equation has singular solutions unless the parameter λ takes certain values. For a sequence of values $\lambda_0, \lambda_1, \ldots$ there exist non-singular solutions constituting an infinite sequence of orthogonal polynomials f_n if certain conditions are met. In particular, if the polynomial σ is quadratic and has distinct roots that enclose the root of the linear polynomial τ and if the leading terms of σ and τ have the same sign, the polynomials f_n are the Jacobi-like polynomials. They have a closed interval of orthogonality.

4.3. Gegenbauer polynomials. By an affine linear transformation in the domain and an appropriate normalisation, Jacobi-like polynomials are standardised into the Jacobi polynomials $P_n^{(\alpha,\beta)}$. They carry two parameters $\alpha, \beta > -1$ and are orthogonal on the interval $[-1, 1]$. Gegenbauer polynomials $C_n^{(\alpha)}$ with a single parameter $\alpha > -1/2$ distinct from zero are Jacobi polynomials with a different normalisation,

$$C_n^{(\alpha)}(x) := \frac{\Gamma(\alpha+\frac{1}{2})\Gamma(n+2\alpha)}{\Gamma(2\alpha)\Gamma(n+\alpha+\frac{1}{2})} P_n^{(\alpha-\frac{1}{2},\alpha-\frac{1}{2})}(x), \quad h_n = \frac{2^{1-2\alpha}\pi\Gamma(n+2\alpha)}{(n+\alpha)(\Gamma(\alpha))^2\Gamma(n+1)}.$$

The associated inner product is denoted $\langle\,\cdot\,,\,\cdot\,\rangle_{(\alpha)}$. Gegenbauer polynomials are solutions to the Gegenbauer differential equation

$$\left(1-x^2\right) y''(x) - (2\alpha+1)xy'(x) + n(n+2\alpha)y(x) = 0 \tag{4.3}$$

with the corresponding differential operator $\mathcal{D}^{(\alpha)}(y) := -\left(1-x^2\right) y'' + (2\alpha+1)xy'$ which has eigenvalues $\lambda_n^{(\alpha)} := n(n+2\alpha)$. The three-term recurrence formulae read

$$C_{n+1}^{(\alpha)}(x) = \left(\tilde{a}_n x + \tilde{b}_n\right) C_n^{(\alpha)}(x) - \tilde{c}_n C_{n-1}^{(\alpha)}(x), \tag{4.4}$$

with $\tilde{a}_n = 2(n+\alpha)/(n+1)$, $\tilde{b}_n = 0$, and $\tilde{c}_0 = 0$, $\tilde{c}_n = (n+2\alpha-1)/(n+1)$, and

$$xC_n^{(\alpha)} = a_n\, C_{n-1}^{(\alpha)}(x) + b_n\, C_n^{(\alpha)}(x) + c_n\, C_{n+1}^{(\alpha)}(x), \tag{4.5}$$

where $a_0 = 0$, $a_n = 1/2 + (\alpha-1)/(2n+2\alpha)$, $b_n = 0$, and $c_n = (n+1)/(2n+2\alpha)$. A consequence of the coefficients b_n being zero is that the polynomial $C_n^{(\alpha)}(x)$ is an even function if n is even, and that it is odd otherwise. For $n \geq 1$, the Gegenbauer polynomials satisfy the derivative identity

$$\frac{\mathrm{d}}{\mathrm{d}x} C_n^{(\alpha)}(x) = 2(n-1+\alpha)C_{n-1}^{(\alpha)}(x) + \frac{\mathrm{d}}{\mathrm{d}x} C_{n-2}^{(\alpha)}(x) \tag{4.6}$$

which by repetitive application implies

$$\frac{\mathrm{d}}{\mathrm{d}x} C_n^{(\alpha)}(x) = \sum_{k=0}^{\lfloor (n-1)/2 \rfloor} 2(2k+\chi+\alpha)C_{2k+\chi}^{(\alpha)}(x) \quad (\chi := [n \text{ is even}]). \tag{4.7}$$

ETNA
Kent State University
http://etna.math.kent.edu

Here, an expression of the form $[expr]$ evaluates to one if *expr* is true, and to zero otherwise. The last equality allows to express the derivative of a Gegenbauer polynomial through a sum of Gegenbauer polynomials of smaller degree.

REMARK 4.1. The fact that Gegenbauer polynomials are not defined for $\alpha = 0$ is due to the particular normalisation. Nevertheless, the limit $\lim_{\alpha\to 0} C_n^{\alpha}(x)/\alpha = \frac{2}{n}T_n(x)$ exists for all x and $n > 0$, where $T_n(x) := \cos(n \arccos x)$ are the Chebyshev polynomials of first kind. This motivates the definition

$$C_0^{(0)}(x) := 1, \qquad C_n^{(0)}(x) := \frac{2}{n}T_n(x), \qquad h_n = \begin{cases} \pi, & \text{if } n = 0, \\ 2\pi/n^2, & \text{if } n > 0. \end{cases}$$

The differential equation (4.3) and the subsequent definition of $\mathcal{D}^{(\alpha)}$ and $\lambda_n^{(\alpha)}$ remain valid for $\alpha = 0$. The constants in the three-term recurrence formulae (4.4) and (4.5) change to

$$\tilde{a}_n = \begin{cases} 2, & \text{if } n = 0, \\ 2n/(n+1), & \text{if } n \geq 1, \end{cases} \qquad \tilde{b}_n = 0, \quad \tilde{c}_n = \begin{cases} 0, & \text{if } n = 0, \\ 1, & \text{if } n = 1, \\ (n-1)/(n+1), & \text{if } n \geq 2, \end{cases}$$

$$a_n = \begin{cases} 0, & \text{if } n = 0, \\ 1, & \text{if } n = 1, \\ (n-1)/(2n), & \text{if } n \geq 2, \end{cases} \qquad b_n = 0, \quad c_n = \begin{cases} 1/2, & \text{if } n = 0, \\ (n+1)/(2n), & \text{if } n \geq 1. \end{cases}$$

The derivative identities (4.6) and (4.7), now for general $\alpha > -1/2$, remain valid with a slight modification,

$$\frac{\mathrm{d}}{\mathrm{d}x}C_n^{(\alpha)}(x) = \gamma_{n-1}^{(\alpha)}C_{n-1}^{(\alpha)}(x) + \frac{\mathrm{d}}{\mathrm{d}x}C_{n-2}^{(\alpha)}(x) = \sum_{k=0}^{\lfloor (n-1)/2 \rfloor} \gamma_{2k+\chi}^{(\alpha)}C_{2k+\chi}^{(\alpha)}(x), \tag{4.8}$$

where we let $\gamma_n^{(\alpha)} := 2(n + \alpha + \delta_{\alpha,0}\delta_{n,0})$.

REMARK 4.2. The most important constants for Gegenbauer polynomials are summarised in Table 5.2 in the Appendix. In the following, we use a superscript of the form $^{(\alpha)}$ or $^{(\beta)}$ for the quantities h_n, λ_n, a_n, and c_n whenever necessary for clarity.

4.4. Expansions in different families of Gegenbauer polynomials. We are now ready to turn to the conversion of expansion coefficients between different families of Gegenbauer polynomials. Let $\alpha, \beta > -1/2$ be distinct and fixed throughout this section and let $C_n^{(\alpha)}$ and $C_n^{(\beta)}$ be the corresponding Gegenbauer polynomials. We assume that a polynomial $f \in \mathbb{P}_N$ with N a natural number has been expanded into the orthogonal sum

$$f = \sum_{n=0}^{N} \mu_n C_n^{(\alpha)}$$

with known coefficients μ_n. We want to compute the coefficients ν_n in the expansion

$$f = \sum_{n=0}^{N} \nu_n C_n^{(\beta)}$$

involving the Gegenbauer polynomials for β. To begin, we define the linear mapping that realises the sought linear transformation from μ_n to ν_n by the associated matrix.

DEFINITION 4.3. *Let $N \in \mathbb{N}$ and $N_1 := \lfloor N/2 \rfloor$, $N_2 := \lfloor (N-1)/2 \rfloor$. For $m, n \geq 0$, we define the coefficients $\phi_{n,m} \in \mathbb{R}$ by*

$$\phi_{n,m} := \langle C_m^{(\alpha)}, C_n^{(\beta)} / h_n^{(\beta)} \rangle_{(\beta)}.$$

They are the components of the $(N+1) \times (N+1)$ Gram matrix

$$\mathbf{\Phi} := (\phi_{n,m})_{n,m=0}^{N} \in \mathbb{R}^{(N+1)\times(N+1)}.$$

Furthermore, we define the matrices

$$\mathbf{\Phi}_e := (\phi_{2n,2m})_{n,m=0}^{N_1} \in \mathbb{R}^{(N_1+1)\times(N_1+1)}, \quad \mathbf{\Phi}_o := (\phi_{2n+1,2m+1})_{n,m=0}^{N_2} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$$

containing only the components in mutually even rows and columns, and mutually odd rows and columns of $\mathbf{\Phi}$, respectively.

The following two lemmas show that the matrices $\mathbf{\Phi}$, $\mathbf{\Phi}_{\mathrm{e}}$, and $\mathbf{\Phi}_{\mathrm{o}}$ can be used to transform the coefficients μ_n into the coefficients ν_n.

LEMMA 4.4. *Let $N \in \mathbb{N}$ and $N_1 := \lfloor N/2 \rfloor$, $N_2 := \lfloor (N-1)/2 \rfloor$. The coefficients ν_n can be obtained from the coefficients μ_n by the matrix-vector product $\boldsymbol{\nu} = \mathbf{\Phi}\,\boldsymbol{\mu}$ with the vectors $\boldsymbol{\mu} = (\mu_0, \mu_1, \ldots, \mu_N)^{\mathrm{T}}$, $\boldsymbol{\nu} = (\nu_0, \nu_1, \ldots, \nu_N)^{\mathrm{T}} \in \mathbb{R}^{N+1}$.*

Proof. The nth component $(\mathbf{\Phi}\,\boldsymbol{\mu})_n$ of the vector $\mathbf{\Phi}\,\boldsymbol{\mu}$ is

$$\begin{aligned}(\mathbf{\Phi}\,\boldsymbol{\mu})_n &= \sum_{m=0}^{N} \phi_{n,m}\mu_m = \sum_{m=0}^{N} \langle C_m^{(\alpha)}, C_n^{(\beta)}/h_n^{(\beta)} \rangle_{(\beta)} \mu_m = \langle \sum_{m=0}^{N} \mu_m C_m^{(\alpha)}, C_n^{(\beta)}/h_n^{(\beta)} \rangle_{(\beta)} \\ &= \langle f, C_n^{(\beta)}/h_n^{(\beta)} \rangle_{(\beta)} = \nu_n. \quad \square\end{aligned}$$

LEMMA 4.5. *Let $N \in \mathbb{N}$ and $N_1 := \lfloor N/2 \rfloor$, $N_2 := \lfloor (N-1)/2 \rfloor$. The coefficients ν_n can be obtained from the coefficients μ_n by computing the matrix-vector products $\boldsymbol{\nu}_e = \mathbf{\Phi}_e\,\boldsymbol{\mu}_e$ and $\boldsymbol{\nu}_o = \mathbf{\Phi}_o\,\boldsymbol{\mu}_o$, with the vectors*

$$\begin{aligned}\boldsymbol{\mu}_e &= (\mu_0, \mu_2, \ldots, \mu_{2N_1})^{\mathrm{T}} \in \mathbb{R}^{N_1+1}, & \boldsymbol{\mu}_o &= (\mu_1, \mu_3, \ldots, \mu_{2N_2+1})^{\mathrm{T}} \in \mathbb{R}^{N_2+1}, \\ \boldsymbol{\nu}_e &= (\nu_0, \nu_2, \ldots, \nu_{2N_1})^{\mathrm{T}} \in \mathbb{R}^{N_1+1}, & \boldsymbol{\nu}_o &= (\nu_1, \nu_3, \ldots, \nu_{2N_2+1})^{\mathrm{T}} \in \mathbb{R}^{N_2+1}.\end{aligned}$$

Proof. This follows by invoking Lemma 4.4 and by taking into account that $\phi_{m,n} = 0$ holds when n is even and m is odd and vice versa. □

We have now reduced the problem of determining the coefficients ν_n from the coefficients μ_n to the computation of matrix-vector products. For this method to be efficient, we need a fast way to apply the matrices $\mathbf{\Phi}_{\mathrm{e}}$, $\mathbf{\Phi}_{\mathrm{o}}$ to a vector. For this purpose, we introduce three new matrices related to $\mathbf{\Phi}$, $\mathbf{\Phi}_{\mathrm{e}}$, and $\mathbf{\Phi}_{\mathrm{o}}$.

DEFINITION 4.6. *Let $N \in \mathbb{N}$ and $N_1 := \lfloor N/2 \rfloor$, $N_2 := \lfloor (N-1)/2 \rfloor$. We define the matrix $\mathbf{G} = (g_{n,m})_{n,m=0}^{N}$ by its components*

$$g_{n,m} := \langle \mathcal{D}^{(\alpha)}(C_m^{(\beta)}), C_n^{(\beta)}/h_n^{(\beta)} \rangle_{(\beta)}.$$

Furthermore, we let

$$\mathbf{G}_e := (g_{2n,2m})_{n,m=0}^{N_1} \in \mathbb{R}^{(N_1+1)\times(N_1+1)}, \quad \mathbf{G}_o := (g_{2n+1,2m+1})_{n,m=0}^{N_2} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$$

be the matrices containing only the components in mutually even rows and columns, and mutually odd rows and columns of $\mathbf{G}$, respectively.

We are now in a position to show that the matrices $\mathbf{\Phi}$ and $\mathbf{\Phi}_{\mathrm{e}}$, $\mathbf{\Phi}_{\mathrm{o}}$ contain the eigenvectors of the matrices $\mathbf{G}$ and $\mathbf{G}_{\mathrm{e}}$, $\mathbf{G}_{\mathrm{o}}$, respectively.

LEMMA 4.7. *Let $N \in \mathbb{N}$, $0 \le m \le N$, and the matrices $\mathbf{\Phi}$ and $\mathbf{G}$ be defined as in Definitions 4.3 and 4.6. Moreover, let $\boldsymbol{\phi}_m := (\phi_{0,m}, \phi_{1,m}, \ldots, \phi_{N,m})^{\mathrm{T}} \in \mathbb{R}^{N+1}$ be the $(m+1)$th column of the matrix $\mathbf{\Phi}$. Then the vector $\boldsymbol{\phi}_m$ is an eigenvector of $\mathbf{G}$ to the eigenvalue $\lambda_m^{(\alpha)}$.*

Proof. We recall that $C_m^{(\alpha)} = \sum_{n=0}^{N} \phi_{n,m} C_n^{(\beta)}$ and write $(\mathbf{G}\,\boldsymbol{\phi}_m)_n$ for the nth component of the matrix-vector product $\mathbf{G}\,\boldsymbol{\phi}_m$. We obtain the result by a straightforward computation,

$$\begin{aligned}
(\mathbf{G}\,\boldsymbol{\phi}_m)_n &= \sum_{j=0}^{N} g_{n,j}\phi_{j,m} & &= \sum_{j=0}^{N} \langle \mathcal{D}^{(\alpha)}(C_j^{(\beta)}), C_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)}\phi_{j,m} \\
&= \langle \mathcal{D}^{(\alpha)}(\sum_{j=0}^{N}\phi_{j,m}C_j^{(\beta)}), C_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)} & &= \langle \mathcal{D}^{(\alpha)}(C_m^{(\alpha)}), C_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)} \\
&= \langle \lambda_m^{(\alpha)} C_m^{(\alpha)}, C_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)} & &= \langle \lambda_m^{(\alpha)} \sum_{j=0}^{N}\phi_{m,j}C_j^{(\beta)}, C_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)} \\
&= \lambda_m^{(\alpha)} \sum_{j=0}^{N}\langle C_j^{(\beta)}, C_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)}\phi_{j,m} & &= \lambda_m^{(\alpha)}\phi_{n,m} \\
&= \lambda_m^{(\alpha)} (\boldsymbol{\phi}_m)_n . \qquad \square
\end{aligned}$$

The following corollary breaks the result down to the matrices $\mathbf{\Phi}_{\mathrm{e}}$, $\mathbf{\Phi}_{\mathrm{o}}$. It is a immediate consequence of the preceding theorem and the fact that $\phi_{m,n} = 0$ holds when n is even and m is odd and vice versa.

COROLLARY 4.8. *Let $N \in \mathbb{N}$, $N_1 := \lfloor N/2 \rfloor$, $N_2 := \lfloor (N-1)/2 \rfloor$, and the matrices $\mathbf{\Phi}_e$, $\mathbf{\Phi}_o$ and $\mathbf{G}_e$, $\mathbf{G}_o$ be defined as in Definitions 4.3 and 4.6, respectively. For $0 \le m_1 \le N_1$, we write*

$$\boldsymbol{\phi}_{m_1}^{e} := (\phi_{0,2m_1}, \phi_{2,2m_1}, \ldots, \phi_{2N_1,2m_1})^{\mathrm{T}} \in \mathbb{R}^{N_1+1}$$

for the (m_1+1)th column of $\mathbf{\Phi}_e$, and for $0 \le m_2 \le N_2$ we let

$$\boldsymbol{\phi}_{m_2}^{o} := (\phi_{1,2m_2+1}, \phi_{3,2m_2+1}, \ldots, \phi_{2N_2+1,2m_2+1})^{\mathrm{T}} \in \mathbb{R}^{N_2+1}$$

be the (m_2+1)th column of $\mathbf{\Phi}_o$. Then $\boldsymbol{\phi}_{m_1}^{e}$ is an eigenvector of $\mathbf{G}_e$ to the eigenvalue $\lambda_{2m_1}^{(\alpha)}$ and $\boldsymbol{\phi}_{m_2}^{o}$ is an eigenvector of $\mathbf{G}_o$ to the eigenvalue $\lambda_{2m_2+1}^{(\alpha)}$.

REMARK 4.9. It is clear that the matrix $\mathbf{G}$, and consequently $\mathbf{G}_{\mathrm{e}}$ and $\mathbf{G}_{\mathrm{o}}$ as well, has simple eigenvalues $\lambda_m^{(\alpha)} = m(m+2\alpha)$ which is a requirement for the application of Algorithm 1 from Section 3.

Knowing that the columns of the matrices $\mathbf{\Phi}_{\mathrm{e}}$ and $\mathbf{\Phi}_{\mathrm{o}}$ are the eigenvectors of the matrices $\mathbf{G}_{\mathrm{e}}$ and $\mathbf{G}_{\mathrm{o}}$ yet does not yield a fast method to apply $\mathbf{\Phi}_{\mathrm{e}}$ and $\mathbf{\Phi}_{\mathrm{o}}$ to a vector. But we are now ready to show that $\mathbf{G}_{\mathrm{e}}$ and $\mathbf{G}_{\mathrm{o}}$ are actually $\mathcal{U}$-matrices. We are also able to give explicit expressions for the entries. This is established in the following theorem.

THEOREM 4.10. *Let $N \in \mathbb{N}$, $N_1 := \lfloor N/2 \rfloor$, $N_2 := \lfloor (N-1)/2 \rfloor$, and the matrices and $\mathbf{G}_e$, $\mathbf{G}_o$ be defined as in Definition 4.6. Then the matrices $\mathbf{G}_e$ and $\mathbf{G}_o$ are $\mathcal{U}$-matrices.*

Proof. We will treat the matrices $\mathbf{G}_{\mathrm{e}}$ and $\mathbf{G}_{\mathrm{o}}$ separately. The main idea of the proof is to use the similarity of the differential operators $\mathcal{D}^{(\alpha)}$ and $\mathcal{D}^{(\beta)}$. This resemblance is reflected in the differential operator

$$\mathcal{D}^{\Delta} := \mathcal{D}^{(\alpha)} - \mathcal{D}^{(\beta)} = 2\,(\alpha-\beta)\,x\frac{\mathrm{d}}{\mathrm{d}x}$$

which differentiates its argument once and then multiplies the result by x and the constant $2\,(\alpha-\beta)$.

Let us first turn to the matrix $\mathbf{G}_{\mathrm{e}}$. Our first goal is to split $\mathbf{G}_{\mathrm{e}}$ into a sum of two matrices each of which we will examine separately. Let now $0 \le n, m \le N_1$. For an entry $g_{2n,2m}$ of $\mathbf{G}_{\mathrm{e}}$ we can write

$$\begin{aligned} g_{2n,2m} &= \langle \mathcal{D}^{(\alpha)}(C_{2m}^{(\beta)}), C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)} \\ &= \langle \mathcal{D}^{(\beta)}(C_{2m}^{(\beta)}), C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)} + \langle \mathcal{D}^{\Delta}(C_{2m}^{(\beta)}), C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)} \end{aligned}$$

which lets us define the two matrices $\mathbf{D}_{\mathrm{e}} := (d_{2n,2m})_{n,m=0}^{N_1}$ and $\mathbf{S}_{\mathrm{e}} := (s_{2n,2m})_{n,m=0}^{N_1}$ with entries

$$d_{2n,2m} := \langle \mathcal{D}^{(\beta)}(C_{2m}^{(\beta)}), C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)}, \qquad s_{2n,2m} := \langle \mathcal{D}^{\Delta}(C_{2m}^{(\beta)}), C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)}.$$

It is clear that $\mathbf{G}_{\mathrm{e}} = \mathbf{D}_{\mathrm{e}} + \mathbf{S}_{\mathrm{e}}$. First, we establish that $\mathbf{D}_{\mathrm{e}}$ is a diagonal matrix. To check this, note that $C_{2m}^{(\beta)}$ is an eigenfunction of the differential operator $\mathcal{D}^{(\beta)}$ to the eigenvalue $\lambda_{2m}^{(\beta)}$ for which we obtain

$$d_{2n,2m} = \langle \mathcal{D}^{(\beta)}(C_{2m}^{(\beta)}), C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)} = \langle \lambda_{2m}^{(\beta)} C_{2m}^{(\beta)}, C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)} = \lambda_{2m}^{(\beta)}\delta_{2m,2n}.$$

With the definition $\mathbf{d}_{\mathrm{e}} := (\lambda_{2m}^{(\beta)})_{m=0}^{N_1}$, we can write $\mathbf{D}_{\mathrm{e}} = \mathrm{diag}(\mathbf{d}_{\mathrm{e}})$.

Now for the harder part: We show that the matrix $\mathbf{S}_{\mathrm{e}}$ is a $\mathcal{U}$-matrix. We invoke the differentiation formula (4.8) and the recurrence relation (4.1) (cf. Table 5.2) to study the effect of the differential operator $\mathcal{D}^{\Delta}$ on a Gegenbauer polynomial $C_{2m}^{(\beta)}$. In a first step this yields

$$\begin{aligned} \mathcal{D}^{\Delta}(C_{2m}^{(\beta)}) &= 2(\alpha-\beta)x \cdot \sum_{k=0}^{m-1} \gamma_{2k+1}^{(\beta)} C_{2k+1}^{(\beta)}(x) \\ &= 2(\alpha-\beta) \cdot \sum_{k=0}^{m-1} \gamma_{2k+1}^{(\beta)} \left(x\, C_{2k+1}^{(\beta)}(x)\right) \\ &= 2(\alpha-\beta) \cdot \sum_{k=0}^{m-1} \gamma_{2k+1}^{(\beta)} \left(a_{2k+1}^{(\beta)}\, C_{2k}^{(\beta)}(x) + c_{2k+1}^{(\beta)}\, C_{2k+2}^{(\beta)}(x)\right). \end{aligned}$$

Using this result, we are now ready to write down an explicit expression for the entries $s_{2n,2m}$ of the matrix $\mathbf{S}_{\mathrm{e}}$. It remains to take the results of the previous display and use them to obtain a simple and explicit expression for $s_{2n,2m}$. We arrive at

$$\begin{aligned} s_{2n,2m} &= \langle \mathcal{D}^{\Delta}(C_{2m}^{(\beta)}), C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)} \\ &= \langle 2(\alpha-\beta) \cdot \sum_{k=0}^{m-1} \gamma_{2k+1}^{(\beta)} \left(a_{2k+1}^{(\beta)}\, C_{2k}^{(\beta)}(x) + c_{2k+1}^{(\beta)}\, C_{2k+2}^{(\beta)}(x)\right), C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)} \\ &= 2(\alpha-\beta) \cdot \sum_{k=0}^{m-1} \gamma_{2k+1}^{(\beta)} \left(a_{2k+1}^{(\beta)}\, \langle C_{2k}^{(\beta)}, C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)} + c_{2k+1}^{(\beta)}\, \langle C_{2k+2}^{(\beta)}, C_{2n}^{(\beta)}/h_{2n}^{(\beta)}\rangle_{(\beta)}\right) \\ &= 2(\alpha-\beta) \cdot \begin{cases} 0, & \text{if } m = 0, \\ \gamma_1^{(\beta)} a_1^{(\beta)}, & \text{if } m > 0 \text{ and } n = 0, \\ 0, & \text{if } m > 0 \text{ and } m < n, \\ \gamma_{2n-1}^{(\beta)} c_{2n-1}^{(\beta)}, & \text{if } m > 0 \text{ and } m = n, \\ \gamma_{2n+1}^{(\beta)} a_{2n+1}^{(\beta)} + \gamma_{2n-1}^{(\beta)} c_{2n-1}^{(\beta)}, & \text{if } m > 0 \text{ and } m > n. \end{cases} \end{aligned}$$

This was the crucial step. We finish the proof for $\mathbf{S}_\mathrm{e}$ by writing the matrix in a more convenient form revealing that it belongs to the class of $\mathcal{U}$-matrices. Define the vectors $\mathbf{u}_\mathrm{e} := (u_{2n})_{n=0}^{N_1} \in \mathbb{R}^{N_1+1}$, $\mathbf{v}_\mathrm{e} := (v_{2n})_{n=0}^{N_1} \in \mathbb{R}^{N_1+1}$, and $\tilde{\mathbf{d}}_\mathrm{e} := (\tilde{d}_{2n})_{n=0}^{N_1} \in \mathbb{R}^{N_1+1}$ by

$$u_{2n} := 2(\alpha-\beta)\cdot\begin{cases}\gamma_1^{(\beta)} a_1^{(\beta)}, & \text{if } n=0,\\[2ex] \gamma_{2n+1}^{(\beta)} a_{2n+1}^{(\beta)} + \gamma_{2n-1}^{(\beta)} c_{2n-1}^{(\beta)}, & \text{if } n>0,\end{cases}$$

$$v_{2n} := \begin{cases}0, & \text{if } n=0,\\ 1, & \text{if } n>0,\end{cases}$$

$$\tilde{d}_{2n} := 2(\alpha-\beta)\cdot\begin{cases}0, & \text{if } n=0,\\ \gamma_{2n-1}^{(\beta)} c_{2n-1}^{(\beta)}, & \text{if } n>0.\end{cases}$$

Then we can write $\mathbf{S}_\mathrm{e}$ as a $\mathcal{U}$-matrix by $\mathbf{S}_\mathrm{e} = \operatorname{diag}(\mathbf{d}_\mathrm{e}) + \operatorname{triu}(\mathbf{u}_\mathrm{e}\,\mathbf{v}_\mathrm{e}^\mathrm{T})$. As as consequence, we obtain

$$\mathbf{G}_\mathrm{e} = \operatorname{diag}(\mathbf{d}_\mathrm{e} + \tilde{\mathbf{d}}_\mathrm{e}) + \operatorname{triu}(\mathbf{u}_\mathrm{e}\,\mathbf{v}_\mathrm{e}^\mathrm{T}).$$

Note that by Lemma 4.11 we already know that the entries of $\operatorname{diag}(\mathbf{d}_\mathrm{e} + \tilde{\mathbf{d}}_\mathrm{e})$ are the distinct values $\lambda_{2n}^{(\alpha)}$ for $n = 0,\ldots,N_1$. The matrix entries are proportional to the difference $\alpha-\beta$ but otherwise not dependent on α.

For the rest of the proof, it remains to repeat the whole procedure for the matrix $\mathbf{G}_\mathrm{o}$. As before, we first establish that $\mathbf{G}_\mathrm{o}$ can be written as the sum of two matrices. In view of the entries $g_{2n+1,2m+1}$ of $\mathbf{G}_\mathrm{o}$,

$$\begin{aligned} g_{2n+1,2m+1} &= \langle \mathcal{D}^{(\alpha)}(C_{2m+1}^{(\beta)}), C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)}\\ &= \langle \mathcal{D}^{(\beta)}(C_{2m+1}^{(\beta)}), C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)} + \langle \mathcal{D}^{\Delta}(C_{2m+1}^{(\beta)}), C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)},\end{aligned}$$

we define the matrices $\mathbf{D}_\mathrm{o} := (d_{2n+1,2m+1})_{n,m=0}^{N_2}$ and $\mathbf{S}_\mathrm{o} := (s_{2n+1,2m+1})_{n,m=0}^{N_2}$ with entries

$$\begin{aligned} d_{2n+1,2m+1} &:= \langle \mathcal{D}^{(\beta)}(C_{2m+1}^{(\beta)}), C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)},\\ s_{2n+1,2m+1} &:= \langle \mathcal{D}^{\Delta}(C_{2m+1}^{(\beta)}), C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)}.\end{aligned}$$

Again, it is straightforward to show that $\mathbf{D}_\mathrm{o}$ is a diagonal matrix since

$$\begin{aligned} d_{2n+1,2m+1} &= \langle \mathcal{D}^{(\beta)}(C_{2m+1}^{(\beta)}), C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)} = \langle \lambda_{2m+1}^{(\beta)} C_{2m+1}^{(\beta)}, C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)}\\ &= \lambda_{2m+1}^{(\beta)}\delta_{2m+1,2n+1}.\end{aligned}$$

With the definition $\mathbf{d}_\mathrm{o} := (\lambda_{2m+1}^{(\beta)})_{m=0}^{N_2}$, we can write $\mathbf{D}_\mathrm{o} = \operatorname{diag}(\mathbf{d}_\mathrm{o})$. The more involved part is again the treatment of the matrix $\mathbf{S}_\mathrm{o}$. With the same ideas as before, we use the differentiation formula (4.8) to get

$$\begin{aligned} \mathcal{D}^{(\Delta)}(C_{2m+1}^{(\beta)}) &= 2(\alpha-\beta)x\cdot\sum_{k=0}^{m}\gamma_{2k}^{(\beta)} C_{2k}^{(\beta)}(x)\\ &= 2(\alpha-\beta)\cdot\sum_{k=0}^{m}\gamma_{2k}^{(\beta)}\left(x\,C_{2k}^{(\beta)}(x)\right)\\ &= 2(\alpha-\beta)\cdot\sum_{k=0}^{m}\gamma_{2k}^{(\beta)}\left(a_{2k}^{(\beta)}\,C_{2k-1}^{(\beta)}(x) + c_{2k}^{(\beta)}\,C_{2k+1}^{(\beta)}(x)\right).\end{aligned}$$

This now allows to obtain an explicit expression for the entries $s_{2n+1,2m+1}$ of $\mathbf{S}_{\mathrm{o}}$, i.e.

$$\begin{aligned} s_{2n+1,2m+1} &= \langle \mathcal{D}^{\Delta}(C_{2m+1}^{(\beta)}), C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)} \\ &= \langle 2(\alpha-\beta)\cdot \sum_{k=0}^{m} \gamma_{2k}^{(\beta)} \left(a_{2k}^{(\beta)}\, C_{2k-1}^{(\beta)}(x) + c_{2k}^{(\beta)}\, C_{2k+1}^{(\beta)}(x)\right), C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)} \\ &= 2(\alpha-\beta)\cdot \sum_{k=0}^{m} \gamma_{2k}^{(\beta)} \left(a_{2k}^{(\beta)}\, \langle C_{2k-1}^{(\beta)}, C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)} \right. \\ &\qquad\qquad\qquad\qquad \left. + c_{2k}^{(\beta)}\, \langle C_{2k+1}^{(\beta)}, C_{2n+1}^{(\beta)}/h_{2n+1}^{(\beta)}\rangle_{(\beta)}\right) \\ &= 2(\alpha-\beta)\cdot \begin{cases} 0, & \text{if } m < n, \\ \gamma_{2n}^{(\beta)} c_{2n}^{(\beta)}, & \text{if } m = n, \\ \gamma_{2n+2}^{(\beta)} a_{2n+2}^{(\beta)} + \gamma_{2n}^{(\beta)} c_{2n}^{(\beta)}, & \text{if } m > n. \end{cases} \end{aligned}$$

Finally, with the definition of the vectors $\mathbf{u}_{\mathrm{o}} := (u_{2n+1})_{n=0}^{N_2} \in \mathbb{R}^{N_2+1}$, $\mathbf{v}_{\mathrm{o}} := (v_{2n+1})_{n=0}^{N_2} \in \mathbb{R}^{N_2+1}$, and $\tilde{\mathbf{d}}_{\mathrm{o}} := (\tilde{d}_{2n+1})_{n=0}^{N_2} \in \mathbb{R}^{N_2+1}$ by

$$\begin{aligned} u_{2n+1} &:= 2(\alpha-\beta)\cdot \left(\gamma_{2n+2}^{(\beta)} a_{2n+2}^{(\beta)} + \gamma_{2n}^{(\beta)} c_{2n}^{(\beta)}\right), \\ v_{2n+1} &:= 1, \\ \tilde{d}_{2n+1} &:= 2(\alpha-\beta)\cdot \left(\gamma_{2n}^{(\beta)} c_{2n}^{(\beta)}\right), \end{aligned}$$

we can write $\mathbf{S}_{\mathrm{o}}$ as a $\mathcal{U}$-matrix by $\mathbf{S}_{\mathrm{o}} = \operatorname{diag}(\mathbf{d}_{\mathrm{o}}) + \operatorname{triu}(\mathbf{u}_{\mathrm{o}}\,\mathbf{v}_{\mathrm{o}}^{\mathrm{T}})$. In the end, we obtain

$$\mathbf{G}_{\mathrm{o}} = \operatorname{diag}(\mathbf{d}_{\mathrm{o}} + \tilde{\mathbf{d}}_{\mathrm{o}}) + \operatorname{triu}(\mathbf{u}_{\mathrm{o}}\,\mathbf{v}_{\mathrm{o}}^{\mathrm{T}}).$$

As before, the diagonal entries of the matrix $\operatorname{diag}(\mathbf{d}_{\mathrm{o}})$ are the values $\lambda_{2n+1}^{(\alpha)}$ for $n = 0, \ldots, N_2$. □

Now, we are almost ready to use the algorithm developed in Section 3.3 to apply the eigenvector matrices of the $\mathcal{U}$-matrices $\mathbf{G}_{\mathrm{e}}$ and $\mathbf{G}_{\mathrm{o}}$ to an arbitrary vector. Since these are the matrices $\mathbf{\Phi}_{\mathrm{e}}$ and $\mathbf{\Phi}_{\mathrm{o}}$, respectively, we then obtain a fast method for realising the conversion between different families of Gegenbauer polynomials. The last missing part is the correct scaling of the eigenvectors. As we have explained in Remark 3.7, it is possible to set up Algorithm 1 for computing the eigendecomposition of a $\mathcal{U}$-matrix so that the eigenvectors are scaled properly based on known diagonal elements of the eigenvector matrix. The following lemma gives an explicit expression for these diagonal entries.

LEMMA 4.11. *Let $N \in \mathbb{N}$, $N_1 := \lfloor N/2 \rfloor$, $N_2 := \lfloor (N-1)/2 \rfloor$, and the matrices $\mathbf{\Phi}_e$ and $\mathbf{\Phi}_o$ be defined as in Definitions 4.3. Then the diagonal entries $\phi_{n,n}$ for $n = 0, 1, \ldots, N$ are*

$$\phi_{n,n} = \frac{k_n^{(\alpha)}}{k_n^{(\beta)}}.$$

Proof. By Definition 4.3, we have $\phi_{n,n} = \langle C_n^{(\alpha)}, C_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)}$. Since the polynomial $C_n^{(\alpha)}$ takes the form $C_n^{(\alpha)} = k_n^{(\alpha)} \cdot x^n + \ldots$ and since the polynomial $C_n^{(\beta)}$ is orthogonal to every polynomial of strictly smaller degree, we have

$$\phi_{n,n} = k_n^{(\alpha)} \langle x^n, \tilde{C}_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)} = \frac{k_n^{(\alpha)}}{k_n^{(\beta)}} \langle C_n^{(\beta)}, C_n^{(\beta)}/h_n^{(\beta)}\rangle_{(\beta)} = \frac{k_n^{(\alpha)}}{k_n^{(\beta)}}. \quad □$$

5. Numerical tests.

We present brief numerical tests of the conversion principle explained in Section 4.4. All tests have been conducted on an Intel Pentium 4 computer with 3.2 GHz. As reference we used MATHEMATICA 5.2 to compute the entries of the matrices $\mathbf{\Phi}$ explicitly and to evaluate matrix-vector products $\boldsymbol{\nu} = \mathbf{\Phi}\,\boldsymbol{\mu}$ to 32 decimal digits of accuracy. The entries of the input vectors $\boldsymbol{\mu}$ were drawn from a uniform random distribution in the complex square $[-\frac{1}{2}, \frac{1}{2}] + \mathrm{i}[-\frac{1}{2}, \frac{1}{2}]$.

The polynomial transform scheme has been implemented in MATLAB in double precision following Algorithms 1 and 2 but split into a precomputation and an evaluation part. In the former, the entries of the vectors $\mathbf{d}$, $\mathbf{u}$, and $\mathbf{v}$ for the matrices $\mathbf{G}_{\mathrm{e}}$ and $\mathbf{G}_{\mathrm{o}}$ and all quantities necessary to apply the corresponding eigenvector matrices are computed. In the evaluation part, these matrices are used to realise the coefficient conversion. We have implemented three variants of the algorithm: In the first, all Cauchy-like matrices are computed and stored explicitly. In the second, only the vectors to form these matrices are computed and stored while matrix-vector multiplication is realised by computing the entries online. The last variant differs from the second in that it additionally uses the fast multipole method to apply the Cauchy-like matrices – during precomputation as well as for evaluation . To achieve numerical stability, it was necessary to add the possibility, to split transformations for $|\alpha - \beta|$ greater than a certain maximum stepsize s into multiple transformations with smaller stepsizes at most s. The algorithm retains the proposed complexity but the constant hiding in the $\mathcal{O}$ notation now grows linearly with $|\alpha - \beta|$. As error measure, we used the relative $\|\cdot\|_2$ error

$$E := \frac{\|\boldsymbol{\nu} - \tilde{\boldsymbol{\nu}}\|_2}{\|\boldsymbol{\nu}\|_2},$$

where $\boldsymbol{\nu}$ denotes the reference coefficients from MATHEMATICA 5.2 and $\tilde{\boldsymbol{\nu}}$ are the quantities computed by the MATLAB implementation.

5.1. The influence of the stepsize s.

We study the influence of the maximum stepsize s on the accuracy of the algorithm. Figure 5.1 shows the error E for a transformation of length $n = 4096$ with $\alpha = 0$ and $\beta = 10$ for increasing threshold s. The value $s = 2$ means, for example, that 5 single transformations are actually computed, namely $\alpha = 0 \to 2 \to 4 \to 6 \to 8 \to 10 = \beta$. The figure shows that for values of s larger than 2 the accuracy quickly drops regardless which variant of the algorithm is used. For moderate values, however, the transformation can be computed to machine precision in the error measure E. To see why this happens, one has to take analytical properties of the coefficient mapping into account. As explicit expressions for the entries of the transform matrix $\mathbf{\Phi}$ exist (see [40] or [24]), one verifies that for large values $|\alpha - \beta|$, the mapping is no longer smooth in the sense that the entries of the involved matrices do no longer vary smoothly among rows and columns. Our conjecture is that this structural property does not allow for accurate algorithms that do not depend on $|\alpha - \beta|$ in any way.

5.2. Errors and timings.

To allow for accurate computations, we fix $s = 1.0$ and compute transforms for sizes $n = 128, 256, 512, 1024, 2048, 4096$ and a range of different combinations for α and β. We include several special cases, for example $\alpha = 1$, $\beta = 0$, as well as transforms between non-integer α and β. We compare the accuracy and runtimes of the three variants of the algorithm. The results are averaged over a number of transforms for each combination of parameters. Table 5.1 shows the results. The following points should be noted here:

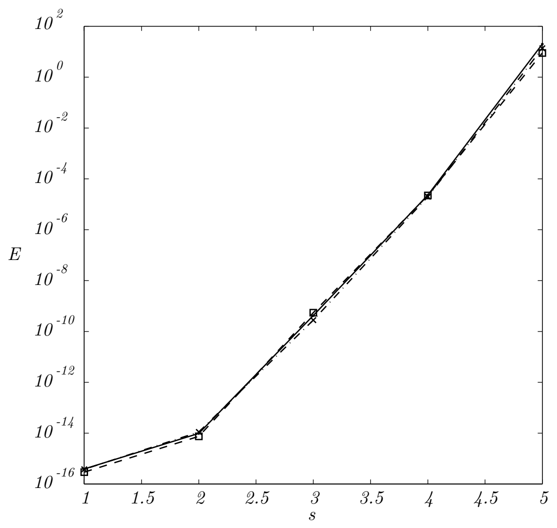

FIG. 5.1. *Influence of the stepsize s. The figure shows the error E for $\alpha = 0$, $\beta = 10$, transform length $n = 4096$ and increasing values of s using the three variants of the algorithm (solid = precomputed matrices, dash-dotted = direct computation of matrix-vector products, dashed = FMM accelerated computation of matrix-vector products). The curves are virtually indistinguishable. The results were averaged over 10 transforms.*

- The variant accelerated by the fast multipole method uses a MATLAB implementation of the method described in [28] while the slower variant with online evaluation of matrix entries makes use of a direct matrix-vector multiplication compiled as an external Fortran routine. A compiled version of the fast multipole method would certainly lead to a speedup for the accelerated algorithm which would lower the break-even point with respect to computation time significantly. Our method is yet faster than the direct method in our examples.
- The accelerated algorithm does not yet beat the algorithm with precomputed matrices. Again, more sophistication in the implementation would make the asymptotic advantage of the accelerated algorithm appear earlier. For large transforms or multiple transforms with different combinations of α and β, another aspect in favour of the accelerated algorithm is that precomputation of all matrices could otherwise exhaust memory resources quickly.
- In terms of arithmetic complexity, the method developed in this paper has been recently superseded by a generalisation of the $\mathcal{O}(n)$ method in [2], which only worked for Legendre polynomials and Chebyshev polynomials of first kind, to Gegenbauer polynomials in [24]. Also, the C implementation tested there is much more efficient than our MATLAB code. On the other hand, that method does not reveal the link to semiseparable matrices in the context of Gegenbauer polynomials.

Conclusion. This paper combines a new algorithm for computing the eigendecomposition of upper and lower triangular diagonal plus semiseparable matrices with a new method to develop fast algorithms for transformations between different families of Gegenbauer polynomials. The former algorithm resembles existing algorithms for symmetric diagonal plus semiseparable matrices and extends the concepts to triangular matrices. The new approach to transforms between different systems of Gegenbauer polynomials presented in Section 4 relies only on very basic and well-known properties of these orthogonal polynomials. We have shown that coefficient conversion is intimately connected to an eigenproblem for upper trian-

ETNA
Kent State University
http://etna.math.kent.edu

TABLE 5.1

Error and computation time comparison. The table shows the error E and the computation time t in seconds for a range of different combinations α and β and different transform lengths n. Superscripts indicate the algorithm used, namely, fully precomputed matrices (e), direct evaluation of matrix-vector products (d) and FMM-accelerated evaluation of matrix-vector products (f). For the accelerated algorithm, we chose the parameters of our FMM implementation so as to keep the error smaller than roughly 10^{-10}. All results were averaged over m transformations. Similar errors for the direct and the FMM-accelerated algorithm at small transform sizes show that the matrix-vector multiplication is still dominated by direct evaluation of matrix entries in the FMM-accelerated version.

α	β	n	m	E_2^{e}	E_2^{d}	E_2^{f}	t^{e}	t^{d}	t^{f}
		1024	50	6.2E−16	6.9E−16	6.9E−16	1.5E−02	2.5E−02	2.0E−02
0.0	0.5	2048	25	7.1E−16	7.8E−16	7.8E−16	2.8E−02	9.4E−02	8.1E−02
		4096	10	7.9E−16	8.5E−16	1.0E−12	7.0E−02	3.8E−01	1.5E−01
		1024	50	2.4E−16	3.7E−16	3.7E−16	2.4E−02	4.3E−02	3.5E−02
0.0	1.5	2048	25	2.0E−16	4.3E−16	4.3E−16	5.5E−02	1.8E−01	1.2E−01
		4096	10	2.1E−16	2.9E−16	5.8E−15	1.3E−01	7.5E−01	2.4E−01
		1024	50	3.0E−16	2.1E−16	2.1E−16	6.1E−02	1.0E−01	7.2E−02
0.0	5.0	2048	25	3.3E−16	2.8E−16	2.8E−16	1.4E−01	4.3E−01	2.1E−01
		4096	10	2.6E−16	2.2E−16	2.2E−16	3.5E−01	1.8E+00	4.5E−01
		1024	50	8.2E−16	4.5E−16	4.5E−16	1.2E−01	2.0E−01	1.4E−01
0.0	10.0	2048	25	3.2E−16	2.4E−16	2.4E−16	2.7E−01	8.7E−01	4.3E−01
		4096	10	3.8E−16	3.7E−16	2.9E−16	7.1E−01	3.6E+00	9.2E−01
		1024	50	5.4E−16	6.3E−16	6.3E−16	1.2E−02	2.2E−02	2.0E−02
0.5	0.0	2048	25	3.9E−15	3.9E−15	3.9E−15	2.7E−02	9.4E−02	8.1E−02
		4096	10	5.4E−15	5.5E−15	1.1E−11	7.1E−01	3.8E−01	1.5E−01
		1024	50	3.6E−15	3.5E−15	3.5E−15	2.4E−02	4.4E−02	4.1E−02
1.5	0.0	2048	25	3.5E−15	3.4E−15	3.4E−15	5.5E−02	1.8E−01	1.6E−01
		4096	10	5.6E−15	5.3E−15	1.1E−10	1.4E−01	7.8E−01	3.0E−01
		1024	50	2.1E−15	1.9E−15	1.9E−15	6.1E−02	1.0E−01	9.0E−02
0.5	0.0	2048	25	2.1E−15	2.1E−15	2.1E−15	1.3E−01	4.6E−01	3.7E−01
		4096	10	2.6E−15	2.3E−15	1.3E−11	3.5E−01	1.9E+00	7.9E−01
		1024	50	3.6E−16	4.1E−16	4.1E−16	8.6E−02	1.5E−01	1.3E−01
π	π^2	2048	25	4.2E−16	3.8E−16	3.6E−16	1.9E−01	6.4E−01	4.8E−01
		4096	10	4.3E−16	5.1E−16	5.1E−16	4.9E−01	2.6E+00	1.0E+00
		1024	50	9.6E−16	1.2E−15	1.2E−15	6.1E−02	1.1E−01	1.0E−01
5.0	0.0	2048	25	6.4E−15	6.5E−15	6.5E−15	1.3E−01	4.7E−01	4.1E−01
		4096	10	2.3E−15	2.0E−15	2.7E−11	3.5E−01	1.9E+00	7.7E−01
		1024	50	5.8E−13	5.8E−13	5.8E−13	6.1E−02	1.1E−01	1.0E−01
6.9	2.4	2048	25	1.3E−12	1.3E−12	1.3E−12	1.3E−01	4.7E−01	4.1E−01
		4096	10	4.4E−12	4.4E−12	5.6E−11	3.5E−01	1.9E+00	8.2E−01
		1024	50	2.3E−14	2.4E−14	2.4E−14	8.6E−02	1.5E−01	1.4E−01
π^2	π	2048	25	5.7E−14	5.7E−14	5.7E−14	1.9E−01	6.5E−01	5.8E−01
		4096	10	4.1E−14	4.1E−14	1.8E−13	4.9E−01	2.7E+00	1.1E+00
		1024	50	1.5E−15	1.3E−15	1.3E−15	1.2E−01	2.2E−01	2.1E−01
10.0	0.0	2048	25	2.3E−15	2.3E−15	2.2E−15	2.7E−01	9.4E−01	8.3E−01
		4096	10	2.1E−15	2.7E−15	2.0E−12	7.0E−01	3.8E+00	1.5E+00

gular diagonal plus semiseparable matrices. The concept might allow for generalisations to other types of orthogonal polynomials, like the entirety of Jacobi polynomials or Hermite and Laguerre polynomials. We have shown that the proposed algorithm can be implemented in a way such that high accuracy (in the error measure we used) is achieved. Specific implementations issues like the split into multiple transforms, the FMM acceleration of the precomputation part or the benefit from specifically tailored FMM implementations have only briefly been addressed. As said before, implementation-wise, the method described and tested in [24] is now to be considered more efficient. But that does not render the theoretical findings in this paper any less interesting.

Appendix. Explicit expressions for constants needed to set up the matrices $\mathbf{G}_\mathrm{e}$ and $\mathbf{G}_\mathrm{o}$ are given below in Table 5.2.

TABLE 5.2
Values of the most important constants for Gegenbauer polynomials.

	$\alpha = 0$	$\alpha > -1/2, \alpha \neq 0$
h_n	π, if $n = 0$, $2\pi/n^2$, if $n > 0$.	$\dfrac{2^{1-2\alpha}\pi\Gamma(n+2\alpha)}{(n+\alpha)(\Gamma(\alpha))^2\Gamma(n+1)}$
λ_n	$n(n+2\alpha)$	$n(n+2\alpha)$
a_n	0, if $n = 0$, 1, if $n = 1$, $(n-1)/(2n)$, if $n \geq 2$.	0, if $n = 0$, $1/2 + (\alpha-1)/(2n+2\alpha)$, if $n \geq 1$.
b_n	0	0
c_n	$1/2$, if $n = 0$, $(n+1)/(2n)$, if $n \geq 1$.	$(n+1)/(2n+2\alpha)$

Acknowledgements. We acknowledge the helpful comments and suggestions by the anonymous referee. We also thank Prof. Jürgen Prestin and Prof. Daniel Potts for valuable discussions and contributions to improvements of this paper.

REFERENCES

[1] M. ABRAMOWITZ AND I. A. STEGUN, eds., *Handbook of Mathematical Functions*, National Bureau of Standards, Washington, D.C., 1972.

[2] B. K. ALPERT AND V. ROKHLIN, *A fast algorithm for the evaluation of Legendre expansions*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 158–179.

[3] T. BELLA, Y. EIDELMAN, I. GOHBERG, V. OLSHEVSKY, AND E. TYRTYSHNIKOV, *Fast inversion of Hessenberg-quasiseparable-Vandermonde matrices and resulting recurrence relations and characterizations*, preprint, 2006.

[4] J. R. BUNCH, C. P. NIELSEN, AND D. C. SORENSEN, *Rank-one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

[5] S. CHANDRASEKARAN AND M. GU, *A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices*, Numer. Math., 96 (2004), pp. 723–731.

[6] T. CHIHARA, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.

[7] J. W. COOLEY AND J. W. TUKEY, *An algorithm for machine calculation of complex Fourier series*, Math. Comput., 19 (1965), pp. 297–301.

[8] J. R. DRISCOLL AND D. HEALY, *Computing Fourier transforms and convolutions on the 2–sphere*, Adv. in Appl. Math., 15 (1994), pp. 202–250.

[9] J. R. DRISCOLL, D. HEALY, AND D. ROCKMORE, *Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs*, SIAM J. Comput., 26 (1996), pp. 1066–1099.

[10] A. DUTT, M. GU, AND V. ROKHLIN, *Fast algorithms for polynomial interpolation, integration and differentiation*, SIAM J. Numer. Anal., 33 (1996), pp. 1689–1711.

[11] Y. EIDELMAN, I. GOHBERG, AND V. OLSHEVSKY, *Eigenstructure of order-one-quasiseparable matrices. Three-term and two-term recurrence relations*, Linear Algebra Appl., 405 (2005), pp. 1–40.

[12] M. FRIGO AND S. G. JOHNSON, *FFTW, C subroutine library*. http://www.fftw.org

[13] F. R. GANTMAKHER AND M. G. KREIN, *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems*, AMS Chelsea Publishing, New York, 2002.

[14] G. H. GOLUB, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.

[15] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Second ed., The Johns Hopkins University Press, Baltimore, 1993.

[16] F. A. GRAYBILL, *Matrices with Applications in Statistics*, Second ed., The Wadsworth Statistics/Probability Series, Duxbury Press, Belmont, 1983.
[17] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[18] M. GU AND S. C. EISENSTAT, *A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1266–1276.
[19] W. HACKBUSCH, *A sparse matrix arithmetic based on $\mathcal{H}$–matrices, Part I: introduction to $\mathcal{H}$–matrices*, Computing, 62 (1999), pp. 89–108.
[20] W. HACKBUSCH, B. N. KHOROMSKIJ, AND S. A. SAUTER, *On $\mathcal{H}^2$-matrices*, in Lectures on Applied Mathematics, H.-J. Bungartz, R. H. W. Hoppe, and C. Zenger, eds., Springer, Berlin, 2000, pp. 9–29.
[21] D. HEALY, P. KOSTELEC, S. MOORE, AND D. ROCKMORE, *FFTs for the 2-sphere - improvements and variations*, J. Fourier Anal. Appl., 9 (2003), pp. 341–385.
[22] N. J. HIGHAM, *Fast Solution of Vandermonde-Like Systems Involving Orthogonal Polynomials*, IMA J. Numer. Anal., 8 (1988), pp. 473–486.
[23] T. KAILATH, *Fredholm resolvents, Wiener-Hopf equations, and Riccati differential equations*, IEEE Trans. Inform. Theory, 15 (1969), pp. 665–672.
[24] J. KEINER, *Computing with Expansions in Gegenbauer Polynomials*, Tech. Rep. AMR07/10, University of New South Wales, 2007.
[25] J. KEINER, S. KUNIS, AND D. POTTS, *NFFT 3.0, C subroutine library*, 2006. http://www.tu-chemnitz.de/~potts/nfft
[26] J. KEINER AND D. POTTS, *Fast evaluation of quadrature formulae on the sphere*, Math. Comput., 77 (2008), pp. 397–419.
[27] S. KUNIS AND D. POTTS, *Fast spherical Fourier algorithms*, J. Comput. Appl. Math., 161 (2003), pp. 75–98.
[28] P. MARTINSSON AND V. ROKHLIN, *An accelerated kernel-independent fast multipole method in one dimension*, SIAM J. Sci. Comput., 29 (2007), pp. 1160–1178.
[29] N. MASTRONARDI, E. V. CAMP, AND M. V. BAREL, *Divide and conquer algorithms for computing the eigendecomposition of symmetric diagonal-plus-semiseparable matrices*, Numer. Algorithms, 39 (2005), pp. 379–398.
[30] S. S. B. MOORE, D. HEALY, AND D. ROCKMORE, *Symmetry stabilization for fast discrete monomial transforms and polynomial evaluation*, Linear Algebra Appl., 192 (1993), pp. 249–299.
[31] K. NABORS, F. T. KORSMEYER, F. T. LEIGHTON, AND J. WHITE, *Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory*, SIAM J. Sci. Comput., 15 (1994), pp. 713–735.
[32] A. F. NIKIFOROV AND V. B. UVAROV, *Special Functions of Mathematical Physics*, Birkhäuser, Basel, 1988.
[33] V. OLSHEVSKY AND V. Y. PAN, *Polynomial and rational evaluation and interpolation (with structured matrices)*, in Automata, Languages and Programming–26th International Colloquium, ICALP '99, Prague, Czech Republic, July 11–15, 1999, J. Wiedermann, P. van Emde Boas, and M. Nielsen, eds., Lecture Notes in Comput. Sci., Vol. 1644, Springer, Berlin, 1999, pp. 585–594.
[34] F. W. J. OLVER, *Asymptotics and Special Functions*, Academic Press, New York, 1974.
[35] V. PAN, *Fast evaluation and interpolation at the Chebyshev set of points*, Appl. Math. Lett., 2 (1989), pp. 255–258.
[36] D. POTTS, *Fast algorithms for discrete polynomial transforms on arbitrary grids*, Linear Algebra Appl., 366 (2003), pp. 353–370.
[37] D. POTTS, G. STEIDL, AND M. TASCHE, *Fast algorithms for discrete polynomial transforms*, Math. Comput., 67 (1998), pp. 1577–1590.
[38] V. ROKHLIN AND M. TYGERT, *Fast algorithms for spherical harmonic expansions*, SIAM J. Sci. Comput., 27 (2006), pp. 1903–1928.
[39] R. SUDA AND M. TAKAMI, *A fast spherical harmonics transform algorithm*, Math. Comput., 71 (2002), pp. 703–715.
[40] G. SZEGŐ, *Orthogonal Polynomials*, Fourth ed., Amer. Math. Soc., Providence, 1975.
[41] N. M. TEMME, *Special Functions*, John Wiley & Sons, Inc., New York, 1996.
[42] R. VANDEBRIL, *Semiseparable Matrices and the Symmetric Eigenvalue Problem*, PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, 2004.
[43] R. VANDEBRIL, M. V. BAREL, G. GOLUB, AND N. MASTRONARDI, *A bibliography on semiseparable matrices*, Calcolo, 42 (2005), pp. 249–270.
[44] N. YARVIN AND V. ROKHLIN, *An improved fast multipole algorithm for potential fields on the line*, SIAM J. Numer. Anal., 36 (1999), pp. 629–666.
[45] L. YING, G. BIROS, AND D. ZORIN, *A kernel-independent adaptive fast multipole method in two and three dimensions*, J. Comput. Physics, 196 (2004), pp. 591–626.