COMPUTING WITH EXPANSIONS IN
GEGENBAUER POLYNOMIALS

JENS KEINER*

Abstract. In this work, we develop fast algorithms for computations involving finite expansions in Gegenbauer polynomials. We describe a method to convert a linear combination of Gegenbauer polynomials up to degree $n$ into a representation in a different family of Gegenbauer polynomials with generally $\mathcal{O}(n \log(1/\varepsilon))$ arithmetic operations where $\varepsilon$ is a prescribed accuracy. Special cases where source or target polynomials are the Chebyshev polynomials of first kind are particularly important. In combination with (nonequispaced) discrete cosine transforms, we obtain efficient methods for the evaluation of an expansion at prescribed nodes and for the projection onto Gegenbauer polynomials from given function values, respectively.

AMS Subject Classification: 42C20, 65T50, 65Y20

Key words: Gegenbauer polynomials, polynomial transforms, fast algorithms

1. Introduction. The principal aim of this paper is to develop, implement and analyse an efficient algorithm for the conversion between finite expansions in different families of Gegenbauer polynomials. This can be combined with fast (nonequispaced) discrete cosine transforms (DCT, NDCT, [26, 8]) to evaluate such an expansion at prescribed nodes, or, conversely, to compute a projection onto Gegenbauer polynomials from given function values.

Gegenbauer polynomials have received much attention for their fundamental properties as well as for their use in applied mathematics. The described procedures are a key ingredient to the implementation of spectral and pseudo-spectral methods to solve certain types of differential equations (see e.g. [4]). Gegenbauer polynomials are a convenient basis for polynomial approximations since they are eigenfunctions of corresponding differential operators. For numerical methods, it is usually as convenient as efficient to convert between representations of a polynomial by expansion coefficients or by function values, respectively.

Our expansions will be of the form

$$f(t) = \sum_{j=0}^{n} \mu_j C_j^{(\alpha)}(t),$$

where $t \in [-1, 1]$ and $C_j^{(\alpha)}(t)$ is the degree-$j$ Gegenbauer polynomial with index $\alpha$ (We will specify the valid range for $\alpha$ in Section 3). Given this representation, an equivalent expansion

$$f(t) = \sum_{j=0}^{n} \nu_j C_j^{(\beta)}(t)$$

for a different index $\beta$ will be computed with generally $\mathcal{O}(n \log(1/\varepsilon))$ arithmetic operations. The method will be approximate and $\varepsilon$ is the desired accuracy. In special cases, the transformation will be computed exactly, and at a cost of $\mathcal{O}(n)$.

The algorithm can be used to efficiently evaluate a polynomial $f$, given through expansion coefficients $\mu_j$, at arbitrary distinct nodes $t_i \in [-1, 1], 1 \leq i \leq m$. First, an equivalent expansion in terms of Chebyshev polynomials of first kind $C_j^{(0)}(t) :=$

*School of Mathematics and Statistics, University of New South Wales, keiner@math.uniluebeck.de
Given nodes using a DCT or an NDCT. More succinctly, this reads

\[ f_i = f(t_i) = f(\cos(\theta_i)) = \sum_{j=0}^{n} \mu_j C_j^{(\alpha)}(\cos(\theta_i)) = \sum_{j=0}^{n} \nu_j T_j(\cos(\theta_i)) = \sum_{j=0}^{n} \nu_j \cos(j\theta_i). \]

The evaluation of the last sum is the DCT/NDCT step. A usual DCT can be used if the nodes \( t_i \) are the \( n + 2 \) extrema of \( T_{n+1}(t) \) in \([-1, 1]\) (type-I DCT) or the \( n + 1 \) zeros (often called the Chebyshev nodes) of \( T_{n+1}(t) \) in \((-1, 1)\) (type-II DCT). This adds \( O(n \log n) \) arithmetic operations to the cost. For other nodes, the approximate NDCT algorithm can be applied with \( O(n \log n + \log(1/\varepsilon) m) \) arithmetic operations. In either case, the second step is asymptotically more expensive than the first one. The whole procedure may be summarised as the matrix-vector product

\[
\mathbf{f} = \mathbf{P} \mathbf{u} = \mathbf{C} \mathbf{A} \mathbf{u} = \mathbf{C} \mathbf{v}, \tag{1.1}
\]

involving the Vandermonde-like matrix \( \mathbf{P} = (C_j^{(\alpha)}(t_i)) \in \mathbb{R}^{m \times (n+1)} \), the cosine matrix \( \mathbf{C} = (\cos(j\theta_i)) \in \mathbb{R}^{m \times (n+1)} \), the conversion matrix \( \mathbf{A} \in \mathbb{R}^{(n+1) \times (n+1)} \) (entries given in Section 3), and vectors \( \mathbf{f} = (f_j) \in \mathbb{R}^{m} \), \( \mathbf{u} = (\mu_j) \in \mathbb{R}^{n+1} \), and \( \mathbf{v} = (\nu_j) \in \mathbb{R}^{n+1} \).

The inverse problem consists in computing expansion coefficients \( \mathbf{u} \) from values \( \mathbf{f} \) drawn from a function \( f(t) \). If \( m = n \) and if \( f(t) \) is a degree-\( n \) polynomial, \( \mathbf{P} \) is nonsingular and solving the linear system \( \mathbf{P} \mathbf{u} = \mathbf{f} \) recovers \( \mathbf{u} \). Generally, we assume \( m \geq n \) and that a quadrature rule with weights \( w_i \) is known, so that each coefficient \( \mu_j \) is approximated by

\[
\mu_j = \sum_{i=1}^{m} w_i f_i C_j^{(\alpha)}(t_i). \tag{1.2}
\]

This corresponds to discretising the inner product integral

\[
\langle f, g \rangle := \int_{-1}^{1} g(t) h(t) (1 - t^2)^{\alpha-1/2} \, dt,
\]

for which the Gegenbauer polynomials \( C_j^{(\alpha)}(t) \) are orthogonal, so that exactness holds for any polynomials \( g(t) \) and \( h(t) \) of degree at most \( n \). Computing the sums (1.2) corresponds to the transposed product

\[
\mathbf{u} = \mathbf{P}^T \mathbf{W} \mathbf{f} = \mathbf{A}^T \mathbf{C}^T \mathbf{W} \mathbf{f} \tag{1.3}
\]

with the diagonal weight matrix \( \mathbf{W} = \text{diag}(w_i) \). A different approach is to first compute expansion coefficients \( \nu_j \) for Chebyshev polynomials \( T_j(t) \), i.e., \( \mathbf{v} = \mathbf{C}^T \mathbf{W} \mathbf{f} \). The matrix \( \mathbf{W} \) now contains weights for the case \( \alpha = 0 \) in (1.2). Plain DCTs are applicable for the same choices of nodes as before and correspond to Gauss quadrature of Chebyshev type with the Chebyshev nodes (type-III DCT, [1, p. 889]), or Gauss-Kronrod quadrature of Chebyshev type with the Chebyshev extrema (type-I DCT, [22]). Both quadrature rules use only \( n + 1 \) or \( n + 2 \) nodes, respectively, and have uniform weights (except for the endpoints of the second rule where the weights are halved). For other nodes, we can use the transposed NDCT algorithm. In a second step, the sought coefficients \( \mu_j \) are obtained by \( \mathbf{u} = \mathbf{A}^{-1} \mathbf{v} \). Thus, we compute

\[
\mathbf{u} = \mathbf{A}^{-1} \mathbf{C}^T \mathbf{W} \mathbf{f}. \tag{1.4}
\]
An $O(n \log^2 n)$ method for the multiplication with a general Vandermonde-like matrix $P$ as in (1.1) is described in [11]. Algorithms for the solution of linear systems involving a Vandermonde-like matrix at cost usually $O(n^2)$ exist (see [27] and references therein).

An algorithm with $O(n \log^2 n)$ operations was developed by Driscoll and Healy in [5, 6] to compute projections onto orthogonal polynomial families as in (1.3). The associated evaluation procedure in the style of (1.1) was described by Potts, Steidl, and Tasche in [25]. Both methods can be combined with the NDCT algorithm for arbitrary nodes at total cost $O(n \log^2 n + \log(1/\varepsilon)m)$ (see [24]). Both methods are applicable to general functions satisfying a three-term recurrence. Stabilisation techniques to partly overcome numerical defects are discussed in [13, 18] but could affect the cost bounds.

An older evaluation and projection method is the $O(n \log^2 n / \log \log n)$ algorithm from [23] improved to $O(n \log n)$ in [21] in the Legendre case. An $O(n \log n \log^3(1/\varepsilon))$ algorithm is mentioned in the technical report [29] but without numerical examples. A recent $O(n \log n \log(1/\varepsilon))$ scheme for conversions between Gegenbauer families was proposed in [16].

This paper focuses on the efficient application of the matrices $A$ and $A^{-1}$. Methods for the multiplication with the transposed matrix $A^T$ can be derived mechanically.

The key procedure described in this paper has been investigated by Alpert and Rokhlin in [2] for the transformation between Legendre polynomials ($\alpha = \frac{1}{2}$) and Chebyshev polynomials of first kind ($\beta = 0$), and vice versa. At the heart of the method is an approximation result for a certain function which allows to replace its occurrences in the definition of the involved matrices by a polynomial approximation. The rate of convergence of these approximations determines the accuracy and efficiency of the algorithm. However, the analysis in [2] lacks an ingredient to proof the observable convergence rate.

We extend the method of Alpert and Rokhlin to all families of Gegenbauer polynomials when the condition $|\alpha - \beta| < 1$ is satisfied. The theoretical basis are similar results for polynomial approximations. But our analysis also establishes a tight bound on the convergence rate which coincides with the numerically observed behaviour. Moreover, we state an exact algorithm for the case $|\alpha - \beta| \in \mathbb{N}$. We can then combine both methods for transformations between arbitrary Gegenbauer families. To our knowledge, this is the first linear time method for this type of transformation that operates directly on the expansion coefficients.

The plan of the paper is as follows. In Section 2 we state basic results from approximation theory which are key for the following investigations. In Section 3 we describe aspects of the linear transformation between different Gegenbauer polynomial families in detail. The relevant approximation results are given in Section 4. We first follow the lines of Alpert and Rokhlin in our more general case. By using a more elegant technique, we then obtain a tight convergence rate estimate. We briefly explain how the results can be used numerically. Section 5 describes the case $|\alpha - \beta| \in \mathbb{N}$ and how this can be combined with the method from Section 4 for arbitrary $\alpha$ and $\beta$. Finally, in Section 6 we conduct a range of numerical experiments to analyse our algorithm from a numerical point of view.

2. Preliminaries.

2.1. Chebyshev Approximation. A principal method used in this paper is to find a good approximation to a function defined on the interval $[-1, 1]$. A popular and efficient way is polynomial interpolation at the Chebyshev points.
Definition 2.1. For \( n \in \mathbb{N}_0 \) and \( 0 \leq i \leq n \) the degree-\( n \) Chebyshev points \( t_i \) and the associated Lagrange polynomials \( u_i(t) \) on the interval \([-1,1]\) are defined by

\[
t_i := \cos \left( \frac{(2i+1)\pi}{2(n+1)} \right), \quad u_i(t) := \prod_{j=0 \atop j \neq i}^n \frac{t - t_j}{t_i - t_j}. \tag{2.1}
\]

For functions on an arbitrary interval \([a,b]\), we define linear mappings, \( \phi \) and its inverse \( \phi^{-1} \), to transplant these functions from \([a,b]\) to \([-1,1]\) and vice versa by a linear change of variables.

Definition 2.2. Let \([a,b]\) be an arbitrary interval on the real line. Then the linear mappings \( \phi : \mathbb{C} \to \mathbb{C} \) and \( \phi^{-1} : \mathbb{C} \to \mathbb{C} \) are defined by

\[
\phi(t) := \frac{1}{2}(b - a)(t + 1) + a, \quad \phi^{-1}(x) := \frac{2x - a}{b - a} - 1.
\]

Note that \( \phi([-1,1]) = [a,b] \) and \( \phi^{-1}([a,b]) = [-1,1] \). Whenever we use \( \phi \) and \( \phi^{-1} \) it will be clear from the context what \( a \) and \( b \) actually are. We are now ready to introduce polynomial approximations based on interpolation at Chebyshev points.

Definition 2.3. Let \( f : [a,b] \to \mathbb{R} \). Then the degree-\( n \) Chebyshev approximation \( f_n \) to \( f \) is defined by

\[
f_n(x) := \sum_{i=0}^n f(\phi(t_i)) \ u_i(\phi^{-1}(x)).
\]

This type of Lagrangian interpolation and the error involved are well understood. The proof of the following Theorem is found in [3, p. 120].

Theorem 2.4. Let \( f : [a,b] \to \mathbb{R} \) be a function with \( n+1 \) continuous derivatives. Then

\[
\|f - f_n\|_\infty \leq \frac{2(b - a)^{n+1}}{4^{n+1}(n+1)!} \sup_{x \in [a,b]} |f^{(n+1)}(x)|. \tag{2.2}
\]

The maximum norm on the left hand side is taken over the interval \([a,b]\). Theorem 2.4 asserts exponential convergence in the maximum norm with increasing degree \( n \) provided that the function \( f \) is infinitely many times continuously differentiable and that a suitable bound for the \((n+1)\)st derivative of \( f \) can be found. A practical way to use Theorem 2.4 for a concrete function is provided by the following lemma. It is an immediate consequence of the Cauchy integral formula.

Lemma 2.5. Let \( z \in \mathbb{C} \) and let \( D \) be a closed disc around the point \( z \) with radius \( r \). If \( f : D \to \mathbb{C} \) is continuous on \( D \) and analytic in its interior, then

\[
|f^{(n)}(z)| \leq \frac{n!}{r^n} \sup_{\theta \in [0,2\pi]} |f(z + re^{i\theta})|. \tag{2.3}
\]

The last result reveals that knowledge about the function \( f \) in the complex plane can be useful although the approximation happens on the real line. Here is a further classical result for Chebyshev approximations for which the complex plane is rather essential.
Theorem 2.6. Let \( f : [a, b] \to \mathbb{R} \) be analytic in a neighbourhood of \([a, b]\). Then \( \|f - f_n\|_\infty = O(C^n) \) for some \( C < 1 \). In particular, if \( f \circ \phi \) is analytic in the ellipse with foci \( \pm 1 \) and semimajor and semiminor axis length \( K \geq 1 \) and \( k \geq 0 \), we may take \( C = 1/(K + k) \).

Proofs for this result are in [3, p. 121], [20, p. 297] and [4, p. 49]. In particular, if \( f \circ \phi \) has poles in the complex plane, then the rate of convergence is determined by the largest ellipse with foci \( \pm 1 \) in which the function is still analytic. Figure 2.1 shows several ellipses of this type. The following corollary explains how the exponential rate \( C \) can be calculated if the poles are known explicitly.

Corollary 2.7. Continuing the notation of Theorem 2.6, let \( (\mu_j, \eta_j) \in [0, \infty] \times [0, 2\pi), \ j = 1, 2, \ldots \) be elliptical coordinates so that \( z_j = \cosh(\mu_j + i \eta_j) \) are the poles of \( f \circ \phi \). If \( z^\ast \) corresponds to \( \mu^\ast := \min_j \mu_j \) then \( \|f - f_n\|_\infty = O(C^n) \) and \( C \) can be taken as

\[
C = \left(\alpha + \sqrt{\alpha^2 - 1}\right)^{-1}, \quad \alpha = \frac{1}{2} (|z^\ast + 1| + |z^\ast - 1|). \tag{2.4}
\]

Proof. Exponential convergence follows directly from Theorem 2.6. The rate \( C \) is obtained from the sum of the semimajor and semiminor axis lengths \( K = \cosh \mu^\ast \) and \( k = \sinh \mu^\ast \) of the convergence limiting ellipse,

\[
C = 1/(K + k) = 1/(\cosh \mu^\ast + \sinh \mu^\ast).
\]

Here, \( \cosh \mu^\ast \) is obtained from \( z^\ast \) by using the relation \( \cosh \mu^\ast = (r_1 + r_2)/2s \) with \( r_1 = |z^\ast + 1| \) and \( r_2 = |z^\ast - 1| \) the distances to the foci and \( 2s = 2 \) the distance between the foci. The semimajor axis length \( k = \sinh \mu^\ast \) is \( \sqrt{1 - \cosh^2 \mu^\ast} \).

Theorem 2.4 and Lemma 2.5 were used in [2]. This makes it easy for us to follow the lines therein. Theorem 2.6 opens a new way to estimate the actual rate of convergence \( C \). We shall postpone a discussion of this topic to Section 4.

2.2. The Function \( \Lambda \). In this section, we define and analyse a function \( \Lambda(z, \varepsilon) \) which will appear throughout much of the remainder of the paper.

Definition 2.8. For \( z \in \mathbb{C} \) and \( \varepsilon \in \mathbb{R} \), the function \( \Lambda(z, \varepsilon) \) is defined by

\[
\Lambda(z, \varepsilon) := \frac{\Gamma(z + \varepsilon)}{\Gamma(z + 1)}.
\]

Unless \( \varepsilon \) is an integer, \( \Lambda(z, \varepsilon) \) is a meromorphic function with simple poles at \( z = -\varepsilon, -(\varepsilon + 1), -(\varepsilon + 2), \ldots \) and zeros at \( z = -1, -2, -3, \ldots \) which is analytic in the half-plane \( \text{Re} \ z > -\varepsilon \). We will only need to evaluate \( \Lambda(z, \varepsilon) \) in its region of analyticity. Figure 2.2 shows \( \Lambda(z, \varepsilon) \) for \( \varepsilon \in (0, 1) \) which will be an important restriction in the analysis to follow. The case \( \varepsilon = \frac{1}{2} \), treated in [2], is shown as a thick black line.

Let us now establish upper and lower bounds on the modulus \( |\Lambda(z, \varepsilon)| \). We follow Alpert’s and Rokhlin’s rationale in a more general setting. First, we need the following lemma, known as Binet’s Log Gamma Formula, on the logarithm of the Gamma function \( \Gamma(z) \). A proof is found in [19, p. 11]. It enables us to prove the sought bounds in Lemma 2.10 below.

Lemma 2.9. Let \( z \in \mathbb{C}, \ \text{Re} \ z > 0 \). Then

\[
\ln \Gamma(z) = \left(z - \frac{1}{2}\right) \ln z - z + \frac{\ln 2\pi}{2} + I(z),
\]
where

\[ I(z) := \int_0^\infty \left( \frac{1}{e^t - 1} - \frac{1}{t} + \frac{1}{2} \right) \frac{1}{t} e^{-t z} \, dt. \]

Moreover, \(|I(z)| \leq \frac{1}{6|z|} \).

**Lemma 2.10.** Let \( z \in \mathbb{C}, \, \text{Re} \, z > \frac{1}{2} - \varepsilon \) with \( \varepsilon \in (0, 1) \). Then

\[ e^{-1/(6|\text{Re} \, z + \varepsilon|)} < |\Lambda(z, \varepsilon)| < \frac{e^{1/(6|\text{Re} \, z + \varepsilon|) + 1 - \varepsilon}}{|z + 1|^{1 - \varepsilon}}. \quad (2.5) \]

**Proof.** We prove the equivalent inequalities

\[ -\frac{1}{6|\text{Re} \, z + \varepsilon|} < \ln|\Lambda(z, \varepsilon)| + (1 - \varepsilon) \ln|z + 1| < \frac{1}{6|\text{Re} \, z + \varepsilon|} + 1 - \varepsilon. \]

Lemma 2.9 allows us to write the expression \( \ln|\Lambda(z, \varepsilon)| \) as

\[ \ln|\Lambda(z, \varepsilon)| = \text{Re}(F(z, \varepsilon)) + (1 - \varepsilon)(1 - \ln|z + 1|) + \text{Re}(Q(z, \varepsilon)), \]

where

\[ F(z, \varepsilon) := (z + \varepsilon - \frac{1}{2}) \ln \left( \frac{z + \varepsilon}{z + 1} \right), \quad Q(z, \varepsilon) := I(z + \varepsilon) - I(z + 1). \]

To take a closer look at \( Q(z, \varepsilon) \), note that \( 0 < 1 - e^{-t(1-\varepsilon)} < 1 \) and \( 0 < \left( \frac{1}{e^t - 1} - \frac{1}{t} + \frac{1}{2} \right) \frac{1}{t} \)

for every \( t > 0 \). The latter inequality is true since

\[ \left( \frac{1}{e^t - 1} - \frac{1}{t} + \frac{1}{2} \right) \frac{1}{t} = 2 \sum_{k=1}^{\infty} \frac{1}{t^2 + 4\pi^2 k^2} \]
as shown in [17, p. 378]. We conclude

\[
|\text{Re}(Q(z, \varepsilon))| \leq |Q(z, \varepsilon)| = |I(z + \varepsilon) - I(z + 1)|
\]

\[
= \left| \int_0^\infty \left( \frac{1}{e^t - 1} - \frac{1}{t} + \frac{1}{2} \right) \frac{1}{t} e^{-t(z+\varepsilon)} \left( 1 - e^{-t(1-\varepsilon)} \right) \, dt \right|
\]

\[
< \int_0^\infty \left( \frac{1}{e^t - 1} - \frac{1}{t} + \frac{1}{2} \right) \frac{1}{t} e^{-t(\text{Re} \, z + \varepsilon)} \, dt = I(\text{Re} \, z + \varepsilon) \leq \frac{1}{6|\text{Re} \, z + \varepsilon|}.
\]

Our second goal is now the estimate \(-1 + \varepsilon \leq \text{Re}(F(z, \varepsilon)) < 0\). First, we write

\[
\ln \left( \frac{z + \varepsilon}{z + 1} \right) = -\ln \left( \frac{z + 1 - \varepsilon}{z + \varepsilon} \right) = -\ln \left( 1 + \frac{1 - \varepsilon}{z + \varepsilon} \right) = -\int_C \frac{1}{1+w} \, dw,
\]

where \(C\) is the straight line connecting the origin with the point \(z_0 := \frac{-1-\varepsilon}{z+\varepsilon}\) in the complex plane. Since \(\text{Re} \, z_0 = (1-\varepsilon) \text{Re}(z+\varepsilon)/|z+\varepsilon|^2 > 0\), we estimate

\[
\left| (z + \varepsilon - \frac{1}{2}) \ln \left( \frac{z + \varepsilon}{z + 1} \right) \right| \leq |z + \varepsilon| \left| \int_C \frac{1}{1+w} \right| \, dw \leq |z + \varepsilon| \frac{1 - \varepsilon}{|z + \varepsilon|} = 1 - \varepsilon,
\]

where we have used \(|z + \varepsilon - \frac{1}{2}| \leq |z + \varepsilon|, \ (1 + w)^{-1} < 1\) and \(|C| = |z_0|\). It remains to show that the real part of \(F(z, \varepsilon)\) is negative. This follows from

\[
\text{Re}(F(z, \varepsilon)) = -\left( \text{Re}(z) + \varepsilon - \frac{1}{2} \right) \ln |1 + z_0| + \text{Im}(z) \text{arg}(1 + z_0),
\]

using \(\text{sign}(\text{arg} \, z) = -\text{sign}(\text{arg}(1 + z_0)). \quad \square\)

Under a moderate additional restriction, we get a simpler bound. For \(\varepsilon = \frac{1}{2}\), the next result is a slightly tighter version of Lemma 2.4 [2, p. 161].

**Corollary 2.11.** Under the assumptions of Lemma 2.10 and if further \(\text{Re} \, z \geq \frac{1}{2}\), it is true that

\[
\frac{e^{\frac{1}{2} - \varepsilon}}{|z+1|^{1-\varepsilon}} < |\Lambda(z, \varepsilon)| < \frac{e^{\frac{1}{2} + 1-\varepsilon}}{|z+1|^{1-\varepsilon}}. \tag{2.6}
\]

3. **Gegenbauer Polynomials.** Generally, a family of orthogonal polynomials \(p_0, p_1, p_2, \ldots\) on a finite real interval \([a, b] \subset \mathbb{R}\) is a set of polynomials of degrees 0, 1, 2, \ldots which are orthogonal with respect to an inner product of the form

\[
\langle f, g \rangle := \int_a^b f(t)g(t) \, w(t) \, dt,
\]

with a continuous and nonnegative weight function \(w(t)\) on \([a, b]\). Gegenbauer polynomials \(C_j^{(\alpha)}(t)\) with index \(\alpha > -1/2\), \(\alpha \neq 0\) are Jacobi polynomials with a particular normalisation,

\[
C_j^{(\alpha)}(t) := \frac{\Gamma \left( \alpha + \frac{1}{2} \right) \Gamma(j + 2\alpha)}{\Gamma(2\alpha) \Gamma \left( j + \alpha + \frac{1}{2} \right)} P_j^{(\alpha-1/2, \alpha-1/2)}(t).
\]

This definition follows the conventions from Szegő’s book [28] who uses the term **ultraspherical polynomials**. Gegenbauer polynomials are orthogonal over the interval
[−1, 1] with respect to the weight function w(t) = (1 − t^2)^{α−1/2}. Note that the definition does not allow α to be zero. Nevertheless, the limits

\[
\lim_{\alpha \to 0} C^{(α)}_0(t) = T_0(t), \quad \lim_{\alpha \to 0} jC^{(α)}_j(t) = T_j(t) (j > 0)
\]  

(3.1)
exist for every \( t \in [−1, 1] \). To avoid a confusing notation, we define the polynomials \( C^{(α)}_j(t) \) to be the Chebyshev polynomials of first kind \( T_j(t) \) as obtained by the limits.

3.1. Connection Between Families of Gegenbauer Polynomials. This section describes how an expansion in one family of Gegenbauer polynomials can be expressed in a different family of Gegenbauer polynomials. Let \( f : [−1, 1] \to \mathbb{R} \) be a function with a finite Gegenbauer expansion of the form

\[
f(t) = \sum_{j=0}^{n} \mu_j C^{(α)}_j(t).
\]

Then there exists also an expansion of the form

\[
f(t) = \sum_{j=0}^{n} \nu_j C^{(β)}_j(t)
\]

for any different index \( β \). Moreover, there exists a matrix \( A \in \mathbb{R}^{(n+1) \times (n+1)} \) such that the vectors \( u := (\mu_j)^T \in \mathbb{R}^{n+1} \) and \( v := (\nu_j)^T \in \mathbb{R}^{n+1} \) are related by the equation

\[
v = Au.
\]

The following result gives explicit expressions for the entries of the matrix \( A \) for different choices of \( α \) and \( β \).

**Lemma 3.1.** Let \( α, β > -\frac{1}{2} \), \( α \neq β \), and the matrix \( A = (a_{j,k}) \) be defined as before. Then the coefficients \( a_{j,k} \) are

\[
a_{j,k} = \begin{cases} 0, & \text{if } k < j \text{ or } k - j \text{ is odd}, \\ 1, & \text{if } k = j = 0, \\ \end{cases}
\]

and in all other cases

1. if \( α < β < α + 1 \) and defining \( ε = β - α \),

\[
a_{j,k} = \frac{\lambda_{α,β}}{(k+1−ε)(k−1+α)} \Lambda \left( \frac{k−j}{2}, 1−ε \right) \Lambda \left( \frac{k+j}{2} + β, 1−ε \right),
\]

(3.3)

2. if \( α < β \) and \( ε = β - α \in \mathbb{N} \),

\[
a_{j,k} = \begin{cases} 0, & \text{if } j < k − 2ε, \\ (-1)^{\frac{ε−j}{2}} \lambda_{α,β} \left( \frac{ε}{k−j} \right) \Lambda \left( \frac{k+j}{2} + β, −ε \right), & \text{else}, \\ \end{cases}
\]

(3.4)

3. in all other cases\(^3\) and defining \( ε = α - β \),

\[
a_{j,k} = \lambda_{α,β} \Lambda \left( \frac{k−j}{2}, ε \right) \Lambda \left( \frac{k+j}{2} + β, ε \right),
\]

(3.5)

\(^3\)These are \( β < α \) and \( α < β \), the latter in combination with \( β - α > 1 \) and \( β - α \notin \mathbb{N} \).
where

\[ \tilde{\lambda}_{\alpha,\beta} := \begin{cases} \frac{1}{2} \Gamma(\beta)(j + \beta)k, & \text{if } \alpha = 0, \\ \frac{2 - \delta_j}{\Gamma(\alpha)}, & \text{if } \beta = 0, \\ \frac{\Gamma(\beta)(j + \beta)}{\Gamma(\alpha)}, & \text{else}, \end{cases} \]

and

\[ \lambda_{\alpha,\beta} := \frac{\tilde{\lambda}_{\alpha,\beta}}{\Gamma(\alpha - \beta)}. \]

**Proof.** First, (3.2) is an immediate consequence of orthogonality and the fact that Gegenbauer polynomials of even degree are even and Gegenbauer polynomials of odd degree are odd. Furthermore, the degree-zero polynomial is always constant one. The rest of the expressions derives from the formula

\[ a_{j,k} = \frac{\Gamma(\beta)(j + \beta)\Gamma\left(\frac{k-\beta}{2} + \alpha - \beta\right)\Gamma\left(\frac{k+j}{2} + \alpha\right)}{\Gamma(\alpha)\Gamma\left(\frac{k-\beta}{2} + 1\right)\Gamma\left(\frac{k+j}{2} + \beta + 1\right)} \]  

(3.6)

which can be found in [28, p. 99]. The expression is potentially undefined if \( \alpha = 0 \) or \( \beta = 0 \) or whenever \( \beta - \alpha \) is a positive integer. For (i) and (iii) and if \( \alpha = 0 \) or \( \beta = 0 \), we keep in mind that the polynomials \( C_j^{(0)}(t) \) or \( C_k^{(0)}(t) \) are given by the limits in (3.1) and multiply the expressions for \( a_{j,k} \) by the appropriate factors if necessary. We can then take the limit \( \lim_{\alpha \to 0^+} \) or \( \lim_{\beta \to 0^+} \) in (3.6), respectively. In case (ii), when \( \beta - \alpha \) is a positive integer, say \( \ell \), we note that

\[ \lim_{\ell' \to \ell} \frac{\Gamma\left(\frac{k-\beta}{2} - \ell'\right)}{\Gamma(-\ell')} = \begin{cases} (-1)^{\frac{k+\beta}{2}} \frac{\ell!}{(\ell - \frac{k-\beta}{2})!}, & \text{if } \frac{k-\beta}{2} \leq \ell, \\ 0, & \text{else}. \end{cases} \]

Fixing one of \( \alpha \) or \( \beta \) and taking the limit with respect to the other parameter in (3.6) yields the expression for the coefficients \( a_{j,k} \). 

Figure 3.1 gives a visual impression of the matrix \( A \) when \( \alpha \) and \( \beta \) are relatively close together. Importantly, the entries \( a_{j,k} \) of the matrix \( A \) can be regarded as samples of a bivariate function \( A(x,y) \) of two continuous arguments at positive integer combinations \( x = j, \ y = k \). This will be the key for the investigations in the next section.

4. **An Efficient Scheme for the Case** \( |\alpha - \beta| < 1 \). Let in this section \( \alpha \) and \( \beta \) be such that \( |\alpha - \beta| < 1 \). The matrix \( A \) as defined in the previous section has in this case the property that its entries do not vary "too much" among rows and columns. More precisely, if the entries \( a_{j,k} \) are viewed as samples of a function \( A(x,y) \) \( (x = j, \ y = k) \) then this function is smooth whenever \( y - x \) is far enough from zero. The purpose of this section is to quantify this behaviour and to briefly explain how it can be exploited numerically.

4.1. **Smoothness of the Function** \( A \). If the function \( A(x,y) \) is smooth enough, it should be well approximated by a Chebyshev approximation. This will be established in this subsection. To handle the smooth regions, we need the following definition of well-separateness.

---

\(^{2}\text{The result is actually due to Gegenbauer ([10]). A proof for } \beta \text{ > } \alpha \text{ is in [15, pp. 139]}\)
**Definition 4.1.** A square \( S \subset \mathbb{R} \times \mathbb{R} \) defined by the formula \( S = [x_0, x_0 + c] \times [y_0, y_0 + c] \) with \( c > 0 \) is said to be well-separated if \( y_0 - x_0 \geq 2c \).

We prove now a generalised form of Theorem 3.2 from [2, p. 164] which Alpert and Rokhlin used as the principal tool to analyse Chebyshev approximations to \( A(x, y) \) for \( \alpha = \frac{1}{2} \) and \( \beta = 0 \), and vice versa. We prove a similar result for general \( \alpha = \varepsilon \in (0, 1) \) and \( \beta = 0 \), and vice versa. The assertion is that under these conditions, the function \( A(x, y) \) can be well approximated by a Chebyshev approximation in \( x \) or \( y \) on any well-separated square \( S \) independent of its size \( c \).

**Theorem 4.2.** Let \( S = [x_0, x_0 + c] \times [y_0, y_0 + c] \) with \( c \geq 2 \) be a well-separated square, \((x, y) \in S\), and denote by \( A_n(\cdot, y) \) and \( A_n(x, \cdot) \) the degree-\( n \) Chebyshev approximations to \( A(\cdot, y) \) and \( A(x, \cdot) \) on \([x_0, x_0 + c] \) and \([y_0, y_0 + c]\), respectively. If \( \alpha = \varepsilon \in (0, 1) \) and \( \beta = 0 \), then

\[
\|A(\cdot, y) - A_n(\cdot, y)\|_\infty \leq \frac{8e^{\frac{4}{3}+2(1-\varepsilon)}}{3n+1} \|A(x, y)\|,
\]

\[
\|A(x, \cdot) - A_n(x, \cdot)\|_\infty \leq \frac{8e^{\frac{4}{3}+2(1-\varepsilon)}}{3n+1} \|A(x, y)\|.
\]

Similarly, if \( \alpha = 0 \) and \( \beta = \varepsilon \in (0, 1) \), then

\[
\|A(\cdot, y) - A_n(\cdot, y)\|_\infty \leq \frac{128e^{\frac{4}{3}+2(1-\varepsilon)}}{3n+1} \|A(x, y)\|_{x + \varepsilon},
\]

\[
\|A(x, \cdot) - A_n(x, \cdot)\|_\infty \leq \frac{224e^{\frac{4}{3}+2(1-\varepsilon)}}{3n+1} \|A(x, y)\|_{x + \varepsilon}.
\]
Proof. We prove only the first estimate since the reasoning for the other cases is almost identical. Similar to the lines in [2], we establish the inequalities

\[
\|A(\cdot, y) - A_n(\cdot, y)\|_\infty \leq \frac{32e^{\frac{4}{3}+2(1-\epsilon)}}{3^{n+1}} (\Gamma(\epsilon))^2 \left(\frac{y-x+2}{y+x+2}\right)^{1-\epsilon},
\]

(4.1)

\[
|A(x, y)| \geq \frac{4e^{-\frac{2}{3}}}{(\Gamma(\epsilon))^2} \left(\frac{y-x+2}{y+x+2}\right)^{1-\epsilon},
\]

(4.2)

from which the estimate follows directly. To show (4.1), let \((x, y) \in S\) and \(\theta \in [0, 2\pi)\). By the defining equation (3.5) and the estimate (2.6) we get

\[
\left|A \left( x + \frac{3}{4}(y-x)e^{i\theta}, y \right) \right| \leq \frac{2}{(\Gamma(\epsilon))^2} \left| \Lambda \left( \frac{y-x}{2} - \frac{3}{8}(y-x)e^{i\theta}, \epsilon \right) \right| \left| \Lambda \left( \frac{y+x}{2} + \frac{3}{8}(y-x)e^{i\theta}, \epsilon \right) \right| \leq \frac{e^{\frac{4}{3}+2(1-\epsilon)}}{(\Gamma(\epsilon))^2} \left(\frac{((y-x)/8+1)((y+7x)/8+1)}{((y-x)/8+1)((y+7x)/8+1)}\right)^{1-\epsilon}.
\]

(4.3)

The Cauchy-Integral estimate (2.3) and noting that \(y-x \geq c\) then give

\[
\left| \partial^{n+1}A(\cdot, y) \right| \leq \frac{(n+1)!}{(3\epsilon/4)^{n+1}} \frac{16e^{\frac{4}{3}+2(1-\epsilon)}}{(\Gamma(\epsilon))^2} \left(\frac{y-x+2}{y+x+2}\right)^{1-\epsilon}.
\]

(4.4)

Now (4.1) follows by combining (4.4) with (2.2). Finally, estimate (4.2) is also a consequence of (3.5) in junction with (2.6),

\[
|A(x, y)| = \frac{2}{(\Gamma(\epsilon))^2} \left| \Lambda \left( \frac{y-x}{2}, \epsilon \right) \right| \left| \Lambda \left( \frac{y+x}{2}, \epsilon \right) \right| \geq \frac{2e^{-\frac{2}{3}}}{(\Gamma(\epsilon))^2} \left(\frac{y-x+2}{y+x+2}\right)^{1-\epsilon}.
\]

\[
\square
\]

Similar to Theorem 4.2, proofs can be established for the general case \(|\alpha - \beta| < 1\) yielding the same convergence rate estimate of \(3^{-(n+1)}\) but with slightly different constants. We omit the proofs as the concrete estimates would be of marginal interest.

Are the bounds of Theorem 4.2 the best we can get? Probably not as numerical results indicate\(^3\). Here is a new theorem giving the optimal\(^4\) rate for \(|\alpha - \beta| < 1\).

**Theorem 4.3.** Let \(S = [x_0, x_0 + c] \times [y_0, y_0 + c] \) with \(c > 0\) be a well-separated square, \((x, y) \in S\), and \(|\alpha - \beta| < 1\). Then

\[
\|A(\cdot, y) - A_n(\cdot, y)\|_\infty = O(3^{n}\sqrt{8}^{-n}),
\]

\[
\|A(x, \cdot) - A_n(x, \cdot)\|_\infty = O(3^{n}\sqrt{8}^{-n}).
\]

\[
\text{Proof.}\] We start with the case \(\alpha = \varepsilon \in (0, 1)\) and \(\beta = 0\). The plan is to invoke Theorem 2.6 for \(A(\cdot, y)\) on \([x_0, x_0 + c]\) and \(A(x, \cdot)\) on \([y_0, y_0 + c]\). Fix now an arbitrary

\(^3\)Here are Rokhlin and Alpert ([2, p. 165, Remark 3.3.] on their theorem: ” Estimates (3.1)-(3.4) in Theorem 3.2 are quite pessimistic. Numerical experiments indicate that the errors in those estimates all decay approximately as \(5^{-k}\), as opposed to \(3^{-k}\).”

\(^4\)The rate is optimal in the sense that a better rate cannot hold uniformly for all admissible choices of \(\alpha, \beta, x_0, y_0\) and \(c\).
y in \([y_0, y_0 + \varepsilon]\). It follows from the defining equation (3.5) that \(A(\cdot, y)\) has poles at 

\[x = \pm (y + 2(j + \varepsilon))\] for \(j \in \mathbb{N}_0\). Consequently, \(g(\cdot) := A(\phi(\cdot), y) : [-1, 1] \to \mathbb{R}\) has poles at

\[
z = \phi^{-1}(y + 2(j + \varepsilon)) = 4\frac{j + \varepsilon}{c} + 2\frac{y - x_0}{c} - 1 > 3,
\]

\[
z = \phi^{-1}(-y - 2(j + \varepsilon)) = -4\frac{j + \varepsilon}{c} - 2\frac{y + x_0}{c} - 1 < -5
\]
on the real line. The first bound is, for which we have used \(y - x_0 \geq 2c\), is sharp since \(y = y_0\) is possible and \(c\) is arbitrary. Therefore, the point \(z = 3\) is a "worst-case" estimate for the convergence limiting pole of \(g\). From (2.4), the convergence rate \(C\) of Chebyshev approximations to \(g\) is then determined as

\[C = 3 + \sqrt{8} \approx 5.83.\]

Similar considerations establish the same result for the function \(A(x, \cdot)\). Remaining cases \(0 < \beta < \alpha < \beta + 1\) and \(0 \leq \alpha < \beta < \alpha + 1\) are handled analogously. □

Note that Theorem 4.3 does not assert that the multiplicative constant in the \(O\)-notation is independent of \(c\). But this can be seen as follows: In [20, p. 295] and [3, p. 121] it is noted that a bound for the constant in the \(O\)-notation can be obtained by bounding the function \(g(\cdot) = A(\cdot, y)\) on the convergence limiting ellipse. In the proof of Theorem 4.3, this is the one with semimajor and semiminor axis lengths \(K = 3\), \(k = \sqrt{8}\). By the maximum modulus principle, it is sufficient to find a bound on the circle \(|z| = 3\) in the complex plane. If \(\alpha = \varepsilon \in (0, 1)\) and \(\beta = 0\), and using (2.5), we can estimate for \(c > 1\) that

\[|A(\phi(3e^{i\theta}), y)| = \frac{2}{(\Gamma(\varepsilon))^{2}} \left| \Lambda \left( \frac{y - x_0}{2} - \varepsilon \frac{3e^{i\theta}}{4}, \varepsilon \right) \right| \left| \Lambda \left( \frac{y + x_0}{2} + \frac{c}{4} (3e^{i\theta}), \varepsilon \right) \right| \leq \frac{2^{2-\varepsilon}}{(\Gamma(\varepsilon))^{2}} e^{\frac{1}{4} + \frac{1}{2\pi} + 2(1-\varepsilon)}.
\]

The unknown constant in the \(O\)-notation can thus be bounded regardless of \(c\). As \(\varepsilon\) goes to zero, our estimate blows up dramatically. But as numerical results indicate, see Figure 4.1(a) for an example, this does not happen in practice to the unknown constant. But we do not have a proof for that. Again, this can be generalised to arbitrary \(\alpha\) and \(\beta\) satisfying \(|\alpha - \beta| < 1\).

### 4.2. Fast Approximate Matrix-Vector Multiplication.

The obtained results on Chebyshev approximations to the function \(A(x, y)\) can be exploited to devise a fast algorithm for applying the matrix \(A\) to a vector. The method we will review here is essentially the one in [2]. It is applicable in our more general case "as is". The basic principles of this method remain untouched for which we keep the description brief. All details can be found in the original manuscript. The key principle is that the smoothness of the function \(A(x, y)\) on well-separated squares enables us to replace it by a Chebyshev approximation. Suppose that \(A' = (a_{j,k})\) with \(j = j_0, \ldots, j_0 + c\) and \(k = k_0, \ldots, k_0 + c\) is a submatrix of \(A\) so that the square \(S = [j_0, j_0 + c] \times [k_0, k_0 + c]\) is well-separated. Let us also assume for a moment that the matrix was fully populated. The standard way to compute the matrix-vector product \(y = A'x\) with vectors \(x = (x_0, \ldots, x_c)^T\), \(y = (y_0, \ldots, y_c)^T\) is to evaluate the sums

\[y_j = \sum_{k=0}^{c} a_{j_0+j,k_0+k}x_k = \sum_{k=0}^{c} A(j_0 + j, k_0 + k)x_k\]

(4.5)
which takes $O(c^2)$ operations. Using however that the function $A(x, y)$ can be well approximated by a Chebyshev approximation of degree, say, $p$ in $x$ and $y$, i.e.,

$$A(x, y) \approx \sum_{r=0}^{c} \sum_{s=0}^{c} A(\phi_1(t_r), \phi_2(t_s)) u_r(\phi_1^{-1}(x)) u_s(\phi_2^{-1}(y)),$$

we get

$$y_j \approx \sum_{k=0}^{c} \sum_{r=0}^{c} \sum_{s=0}^{c} A(\phi_1(t_r), \phi_2(t_s)) u_r(\phi_1^{-1}(j_0 + j)) u_s(\phi_2^{-1}(k_0 + k)) x_k.$$

$$= \sum_{r=0}^{c} \left( u_r(\phi_1^{-1}(j_0 + j)) \left( \sum_{s=0}^{c} A(\phi_1(t_r), \phi_2(t_s)) \left( \sum_{k=0}^{c} u_s(\phi_2^{-1}(k_0 + k)) x_k \right) \right) \right) .$$

The triple sum should be evaluated in the order indicated by the brackets. This takes $O(p^3)$ operations for the innermost sum. The cost for the remaining sums is $O(p^2)$ and $O(pc)$, respectively. In total and if $p$ is fixed the cost is $O(c)$. The sacrifice is that the result $y_j$ is only obtained approximately. This is acceptable since the error can be controlled by the choice of the degree $p$. To apply this principle to the whole matrix $A$, a suitable decomposition into well-separated squares as visualised in Figure 4.1(b) is needed. The region close to the diagonal which is not covered by well-separated squares is handled directly as in (4.5). Implementation-wise, the method can be interpreted as a variant of the fast multipole method (FMM) in [12] where the nodes are positive integers and pairs of nodes $j$ and $k$ only interact if $j < k$ and $j$ and $k$ have the same parity. In total, the method needs $O(n \log(1/\varepsilon))$ arithmetic operations for our $(n + 1) \times (n + 1)$ matrix $A$. Further details can be found in [2] and [7].

5. Transforms For Arbitrary Indices. In this brief section, we describe how transformations between two Gegenbauer families with arbitrary indices $\alpha$ and $\beta$ can be computed. To this end, we split the transformation into two steps. Let $\beta'$ be the number between $\alpha$ and $\beta$ so that $|\alpha - \beta'|$ is an integer and as large as possible (e.g. for $\alpha = \frac{1}{2}$ and $\beta = 3$, we would have $\beta' = \frac{5}{2}$). We can compute the transformation from $\alpha$ to $\beta$ by transforming first from $\alpha$ to $\beta'$ and then from $\beta'$ to $\beta$. The choice of $\beta'$ implies $|\beta' - \beta| < 1$ and the results of the last section enable us to handle the transformation from $\beta'$ to $\beta$ efficiently. What is left is to transform from $\alpha$ to $\beta'$.

To explain this case, it is convenient to divide the corresponding matrix $A$ into odd and even components. So let $A_e := (a_{2j,2j})$ and $A_o := (a_{2j+1,2k+1})$ denote the matrices that contain the entries of $A$ with mutually even and odd indices, respectively. This decomposition eliminates all entries $a_{j,k}$ which are zero by parity of $j$ and $k$ (cf. (3.2)). So odd and even components can be treated separate from each other.

Let us start with the case $\alpha < \beta'$. Case (ii) of Lemma 3.1 shows that the matrix $A$ is upper triangular with not more than $2(\beta - \alpha)$ nonzero diagonals. It is not difficult to see that $A_e$ and $A_o$ are each upper triangular with $(\beta - \alpha)$ nonzero diagonals. Applying the matrix $A$ to a vector reduces to two multiplications with the banded upper triangular matrices $A_e$ and $A_o$. This is straightforward an $O(n)$ process.

The other case $\alpha > \beta'$ can be handled as follows. Suppose that $A$ and its two submatrices $A_e$ and $A_o$ correspond to the inverse transformation from $\beta'$ to $\alpha$ so that they are of the same banded structure before. Since our goal is to compute the transformation from $\alpha$ to $\beta'$ we have to solve two quadratic linear systems involving $A_e$ and $A_o$, respectively. Since both matrices are upper triangular and banded, this can be done again straightforward by a simple back substitution at cost $O(n)$. 

13
In both cases, the number of operations depends linearly on $|\alpha - \beta'|$. Since this value is usually small compared to $n$, we omit this influence in our $O$-notation.

6. Implementation and Numerical Results. The methods described in Sections 4 and 5 have been implemented in C and tested on a variety of problems. All tests were conducted on an Intel Pentium IV 2.80GHz system with 512KB second level cache and 1GB of RAM in double precision arithmetic. To compare with the transform algorithm from [25], we used the NFFT 3.0.3 library which also implements the NDCT algorithm. Since this library depends on the FFTW library [9], we also compiled and linked FFTW 3.1.2. We generally the Intel C++ Compiler 10.0 with compiler options -O3 -ansi-alias -xN. For time measurements we employed FFTW's cycle counters interface which allows us to measure very small time spans accurately. Since cycle counter values are in arbitrary units and differ from platform to platform, we determined the approximate number of cycles per second by pausing the system two seconds and measuring the number of elapsed cycles.

To evaluate the accuracy of results obtained, we used the relative 1-norm $E_1$ and the relative maximum norm $E_\infty$,

$$E_1 = \frac{\|y^* - y\|_1}{\|y^*\|_1}, \quad E_\infty = \frac{\|y^* - y\|_\infty}{\|y^*\|_\infty},$$

(6.1)
on result vectors $y^*$ and $y$ containing computed coefficients. The vector $y^*$ stands for reference values which we computed in extended precision arithmetic\(^5\) on the same platform applying every transform matrix directly. Error figures were computed against these values using results obtained in double precision.

If not stated otherwise, we always drew input coefficients from a uniform quasi-random distribution in $[-1, 1]$ using the standard C function `drand48()`.

The algorithms tested were

1. direct application of the matrix $A$ with online evaluation of the entries,
2. direct application of the matrix $A$ with precomputed entries,
3. the general fast algorithm as explained in Section 5,
4. a direct method using Clenshaw’s algorithm (where directly applicable),
5. the fast polynomial transform from [25] (where directly applicable).

The first variant is generally an $O(n^2)$ method. If $|\alpha - \beta|$ is an integer it becomes $O(n)$ as described in Section 5. This method has the advantage that it requires only a constant amount of memory. It is relatively slow since the evaluation of the matrix entries is somewhat expensive\(^6\).

The second variant works like the first, except that all matrix entries are computed and stored in advance. This is faster than the first method if it is applied multiple times for the same matrix $A$. Since the precomputation is done only once, this part is not accounted for in the execution time. A drawback is that the precomputed values have to be stored in memory which leads to memory requirements of the same order as the number of arithmetic operations, either $O(n)$ or $O(n^2)$. In the latter case, this became prohibitively much for tested transform sizes larger than $n = 16,384$. The large amount of memory needed for this method is also an obstacle if several transforms for different combinations of $\alpha$ and $\beta$ have to be computed.

The third is our fast method with $O(n \log(1/\varepsilon))$ arithmetic operations. Unless otherwise stated, we used degree-18 Chebyshev expansions and $c = 255$ as size for the smallest boxes in the decomposition of the matrix $A$ (cf. Subsection 4.2). This combination was chosen to maximise accuracy and speed on our test system.

The fourth algorithm was used for comparison in cases where it can be applied directly ($\beta = 0$). This method uses Clenshaw’s algorithm to evaluate the input expansion at the Chebyshev points. From these values, Chebyshev coefficients are computed using a type-III DCT. This method always uses $O(n^2)$ arithmetic operations but needs only $O(n)$ extra memory to store the intermediate result.

Finally, the fifth method is an $O(n \log^2 n)$ algorithm directly applicable if $\beta = 0$. It needs $O(n \log n)$ memory of precomputed data. The threshold parameter (see [18]) which influences the numerical accuracy was chosen to be 1,000. Lower values did not improve the accuracy.

The last two methods are implemented in complex arithmetic in the NFFT library for which they have a theoretical factor-2 disadvantage compared to our real-arithmetic implementation of the other methods.

### 6.1. Conversion Between Gegenbauer Families

We tested the algorithms for the conversion between different families of Gegenbauer polynomials. To this end,

\(^5\)This is a 96bit data type with approximately 18 significant decimal figures which is roughly three more than in double arithmetic.

\(^6\)Every entry can be evaluated in constant time but the sheer number of different cases in Lemma (3.1) and the appearance of the Gamma function in the definition make this nevertheless a computationally slow process. Due to software engineering considerations, we did not make the attempt to unfold different execution paths which would likely lead to a measurable speedup.
The fourth and fifth variant are not only slower but also suffer from deteriorating even the variant with precomputation is already slower for the smallest tested size.

The results are given in Tables 6.1, 6.2 and 6.3. Generally, our fast method out-

Table 6.1: Test results for integer distances $|\alpha - \beta| \in \mathbb{N}$. Shown are the relative error $E_\infty$ and the execution time $t$ in seconds for different methods. The algorithms tested were the direct matrix application with online evaluation ($E_{\infty}^{d}$) and precomputation ($E_{\infty}^{f}$), the Clenshaw-based method ($E_{\infty}^{\text{cl}}$) and the fast discrete polynomial transform ($E_{\infty}^{\text{ft}}$). Note that the Clenshaw-based method and the fast polynomial transform make no use of sparsity of $A$.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$n$</th>
<th>$E_{\infty}^{d}$</th>
<th>$E_{\infty}^{f}$</th>
<th>$E_{\infty}^{\text{cl}}$</th>
<th>$E_{\infty}^{\text{ft}}$</th>
<th>$t^d$</th>
<th>$t^f$</th>
<th>$t^{\text{cl}}$</th>
<th>$t^{\text{ft}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.54</td>
<td>-0.46</td>
<td>1,000</td>
<td>9.7E-16</td>
<td>9.7E-16</td>
<td>9.7E-16</td>
<td>9.7E-16</td>
<td>1.2E-03</td>
<td>7.8E-06</td>
<td>1.2E-02</td>
<td>1.9E-04</td>
</tr>
<tr>
<td>3.00</td>
<td>6.00</td>
<td>1,000</td>
<td>7.0E-16</td>
<td>7.0E-16</td>
<td>7.0E-16</td>
<td>7.0E-16</td>
<td>1.2E-01</td>
<td>3.3E-03</td>
<td>1.2E-01</td>
<td>3.3E-03</td>
</tr>
<tr>
<td>3.50</td>
<td>1.00</td>
<td>1,000</td>
<td>2.1E-15</td>
<td>2.1E-15</td>
<td>2.1E-15</td>
<td>2.1E-15</td>
<td>1.2E-05</td>
<td>2.6E-06</td>
<td>1.5E+00</td>
<td>7.4E-02</td>
</tr>
<tr>
<td>4.00</td>
<td>-0.30</td>
<td>1,000</td>
<td>3.1E-15</td>
<td>3.1E-15</td>
<td>3.1E-15</td>
<td>3.1E-15</td>
<td>2.1E-02</td>
<td>4.4E+00</td>
<td>2.1E+02</td>
<td>5.0E+00</td>
</tr>
</tbody>
</table>

We computed a series of transformations with different combinations for the indices $\alpha$ and $\beta$. To see how the different methods compare, we conducted separate tests for the cases $|\alpha - \beta| < 1$, $|\alpha - \beta| \in \mathbb{N}$, and $|\alpha - \beta| > 1$ with $|\alpha - \beta| \not\in \mathbb{N}$. Where directly applicable, we also used algorithms 4 and 5 for comparison.

The results are given in Tables 6.1, 6.2 and 6.3. Generally, our fast method outperforms any other tested method while maintaining full accuracy. The method has the blessing property that it explicitly allows to trade accuracy for speed in a well-controlled way. The less accuracy is required, the bigger the speed advantage becomes. The direct matrix application with online evaluation is slow for obvious reasons and even the variant with precomputation is already slower for the smallest tested size. The fourth and fifth variant are not only slower but also suffer from deteriorating accuracy as the polynomial degree increases.

6.2. Timings. To illustrate the differences in execution time more clearly, we chose $\alpha = 0.5$, $\beta = 0$ and computed transforms for degrees $n = 2, 4, 8, 16, \ldots, 65536$. Results are displayed in Figure 6.1(a). The asymptotically slowest methods are the direct matrix application (first and second method) and the variant based on Clenshaw’s algorithm (fourth method). The asymptotically faster fifth variant is slower than the Clenshaw-method for sizes up to $n = 128$ and slower than the direct matrix application with precomputation (second method) up to $n = 2048$. The latter method latter method is the fastest of all up to degree $n = 512$ when the fast third method takes over. Note that due to our choice of $c = 255$ for the size of the smallest boxes in the decomposition of the matrix $A$, this algorithm is virtually identical to the direct precomputation method up to the break-even point.

6.3. Accuracy of the FMM-like Method. We have shown in Section 4 that the function $A(x, y)$ is well approximated by Chebyshev expansions in certain regions. With this example, we verify that the expected behaviour with respect to increasing
Table 6.2: Test results for the case $|\alpha - \beta| < 1$. Shown are the relative error $E_\infty$ and the execution time $t$ in seconds for different methods. The algorithms tested were the direct matrix application with online evaluation ($E_\infty^d t^d$) and precomputation ($E_\infty^p t^p$), the fast method from Section 4 ($E_\infty^f t^f$), the Clenshaw-based method ($E_\infty^{f,\text{dpt}} t^{f,\text{dpt}}$) and the fast discrete polynomial transform ($E_\infty^{\text{dpt}} t^{\text{dpt}}$).

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$n$</th>
<th>$E_\infty^d$</th>
<th>$E_\infty^f$</th>
<th>$E_\infty^{f,\text{dpt}}$</th>
<th>$E_\infty^{\text{dpt}}$</th>
<th>$t^d$</th>
<th>$t^f$</th>
<th>$t^{f,\text{dpt}}$</th>
<th>$t^{\text{dpt}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-0.17$</td>
<td>$-0.49$</td>
<td>1000</td>
<td>1.5E-15</td>
<td>1.2E-15</td>
<td>1.1E-04</td>
<td>1.0E-03</td>
<td>4.8E-04</td>
<td>6.6E-03</td>
<td>6.6E-03</td>
<td>6.6E-03</td>
</tr>
<tr>
<td>$0.38$</td>
<td>$-0.49$</td>
<td>1000</td>
<td>1.7E-14</td>
<td>6.9E-15</td>
<td>1.0E-03</td>
<td>1.0E-03</td>
<td>5.2E-04</td>
<td>6.6E-03</td>
<td>6.6E-03</td>
<td>6.6E-03</td>
</tr>
<tr>
<td>$0.50$</td>
<td>$0.00$</td>
<td>1000</td>
<td>3.4E-15</td>
<td>1.5E-15</td>
<td>1.1E-11</td>
<td>3.8E-10</td>
<td>4.8E-09</td>
<td>6.6E-03</td>
<td>6.6E-03</td>
<td>6.6E-03</td>
</tr>
<tr>
<td>$0.50$</td>
<td>$0.37$</td>
<td>1000</td>
<td>2.6E-15</td>
<td>1.6E-15</td>
<td>1.4E-01</td>
<td>1.7E-01</td>
<td>2.5E-03</td>
<td>2.5E-03</td>
<td>2.5E-03</td>
<td>2.5E-03</td>
</tr>
<tr>
<td>$0.00$</td>
<td>$0.50$</td>
<td>1000</td>
<td>1.3E-15</td>
<td>1.3E-15</td>
<td>1.2E-02</td>
<td>1.2E-02</td>
<td>4.2E+00</td>
<td>4.2E+00</td>
<td>4.2E+00</td>
<td>4.2E+00</td>
</tr>
</tbody>
</table>

Table 6.3: Test results for the case $|\alpha - \beta| > 1$ and $|\alpha - \beta| \not\in \mathbb{N}$. Shown are the relative error $E_\infty$ and the execution time $t$ in seconds for different methods. The algorithms tested were the direct matrix application with online evaluation ($E_\infty^d t^d$) and precomputation ($E_\infty^p t^p$), the fast algorithm from Section 5 ($E_\infty^f t^f$), the Clenshaw-based method ($E_\infty^{f,\text{dpt}} t^{f,\text{dpt}}$) and the fast discrete polynomial transform ($E_\infty^{\text{dpt}} t^{\text{dpt}}$).

degree of Chebyshev expansions is indeed reflected in the algorithm from Section 4. To do this, we chose $\beta = 0$ and computed for the cases $\alpha = 0.01$, $\alpha = 0.5$, and $\alpha = 0.99$ transforms of size $n = 100,000$ for increasing degrees $p = 0, 1, \ldots, 30$ of the Chebyshev expansions used in the algorithm. Figure 6.1(b) shows the result.

6.4. Combined Projection and Evaluation. This example is particularly important for the evaluation of polynomial expansions at prescribed nodes and the projection onto Gegenbauer polynomials from function values. Computing first a pro-
projection onto Gegenbauer polynomials with index $\alpha$ by interpolation at the Chebyshev nodes, and second, evaluating the projection back at the same nodes should recover the initial values.

As test functions from which function values were drawn, we chose $f_1(t) := |t - 0.1|^2$ and $f_2(t) := |t - 0.1|$. These functions are in $C^0([-1, 1])$ and $C^1([-1, 1])$, respectively, which makes the decay of their Chebyshev expansions to be only of polynomial order with respect to the degree $n$. We made this choice to be sure that we do not compute with expansion coefficients which essentially vanish for yet moderate degrees. Table 6.4 shows the errors $E_1$ and $E_\infty$ from (6.1) of the recovered function values at the Chebyshev nodes against the original values for increasing degree $n$ and different Gegenbauer indices $\alpha$. All computations here use the exact algorithms as described in Section 5 in combination with DCTs.

It can be seen clearly that the larger $\alpha$ is the more inaccurate the results become for both functions $f_1$ and $f_2$. Comparing these two, the function $f_2$ with more slowly decreasing expansion coefficients leads to worse results at the same combination of degree $n$ and index $\alpha$. A difference between the two error measures $E_1$ and $E_\infty$ is also observed as the former yields generally smaller values than the latter. The general behaviour can be explained with severe cancellations in the numerical computations as well as an intrinsic ill-conditioning of the involved matrices. While techniques to combat cancellations exist (see e.g. [14]), the ill-conditioning can still cause large
errors if the input has already undergone some rounding in previous computations. A detailed analysis of this issue is behind the scope of this paper. Nevertheless, the computations are accurate enough in a reasonably large range for \( \alpha \) and \( n \).

7. Conclusions. In this paper, we have developed a general method to convert between expansions in different families of Gegenbauer polynomials with indices \( \alpha \) and \( \beta \), respectively. Based on a previous approximate algorithm which converts Legendre polynomials to Chebyshev polynomials of first kind, we have described an approximate method for the general case \(|\alpha - \beta| < 1\). The new and best possible convergence rate estimate shows that the error made by the method decreases exponentially as \((3 + \sqrt{8})^{-n} \approx 5.83^{-n}\) for increasing degree \( n \) of the internally used Chebyshev approximations. If \( \alpha \) and \( \beta \) are such that \(|\alpha - \beta| \in \mathbb{N}\) we have described two straightforward methods which use well-known facts from linear algebra. Combining both cases, we have obtained a method for arbitrary \( \alpha \) and \( \beta \).

Several numerical experiments show that the described methods are faster and more accurate than the polynomial transform from [25]. They also show that our approximate method is capable of performing as accurate as exact algorithms while also allowing to trade accuracy against speed in a well-controlled way. For practical situations, it seems nevertheless necessary to avoid computations for a large difference in the indices \( \alpha \) and \( \beta \) where computations can suffer from a significant loss of accuracy due to inherent properties of the involved matrices.

Acknowledgements. We acknowledge the kind support of the German Academic Exchange Service DAAD and the University of New South Wales. We also like to thank Prof. Daniel Potts and Prof. Jürgen Prestin for many useful comments that have improved the paper.
REFERENCES