# A NEW ALGORITHM FOR THE NONEQUISPACED FAST FOURIER TRANSFORM ON THE ROTATION GROUP

JENS KEINER[1] AND ANTJE VOLLRATH[2]

**Abstract.** We develop an approximate algorithm to efficiently calculate the discrete Fourier transform on the rotation group SO(3). Our method needs $\mathcal{O}\big(L^3 \log L \log(1/\varepsilon) + \log(1/\varepsilon)^3 Q\big)$ arithmetic operations for a degree-$L$ transform at $Q$ nodes free of choice and with desired accuracy $\varepsilon$. Our main contribution is a method that allows to replace finite expansions in Wigner-$d$ functions of arbitrary orders with those of low orders. It is based on new insights into the structure of related semiseparable matrices. This enables us to employ an established divide-and-conquer algorithm for symmetric semiseparable eigenproblems together with the fast multipole method to achieve an efficient algorithm.

*AMS Subject Classification: 42C10, 42C20.*
*Key words: rotation group, discrete Fourier transform, semiseparable matrices*

**1. Introduction.** The fast calculation of the discrete Fourier transform on the rotation group SO(3) is important for a number of applications. This includes texture analysis [10], protein-protein docking [16, 24], robot workspace generation [4], and magnetic resonance spectra calculation from powders [27].

For a degree-$L$ transform at $Q$ nodes the direct calculation of the discrete SO(3) Fourier transform requires $\mathcal{O}(L^3 Q)$ arithmetic operations.

Kostelec and Rockmore [15] have discussed an $\mathcal{O}(L^4)$ algorithm for $8L^3$ particular nodes. The acceleration is achieved by exploiting the tensor product character of the SO(3) basis functions and the particular choice of nodes; see also [19]. It is acknowledged that a variation of the Driscoll-Healy algorithm [5] could improve this to $\mathcal{O}(L^3 \log^2 L)$, but concerns about performance under practical conditions are noted.

A similar line of thought was nevertheless pursued by Potts, Prestin, and Vollrath [22] and combined, for the first time, with the nonequispaced fast Fourier transform (NFFT) [23] to an approximate $\mathcal{O}\big(L^3 \log^2 L + (1/\varepsilon)^3 Q\big)$ algorithm for $Q$ nodes free of choice, thus generalizing a similar algorithm for the discrete Fourier transform on the sphere $\mathbb{S}^2$ [14, 17].

The comparison of both schemes in [22] shows mixed results. For relatively large transform sizes (roughly $L > 1000$) the asymptotically faster algorithm outperforms the other. But unfortunately, issues with a stabilization scheme that must be employed for numerical stability appear to change the asymptotic cost. Therefore, it seems impossible, at least at this point, to clearly prefer one algorithm over the other. Also, for both methods, the enormous memory requirements have to be considered. It may be reasonable to attach as much importance to the memory question as to the theoretical arithmetic cost of the underlying algorithm.

We propose to replace the Driscoll-Healy-like method with a different algorithm. Our approach thereby generalizes results established by Rokhlin and Tygert [25] for the discrete Fourier transform on the sphere $\mathbb{S}^2$. The result is a new type of algorithm that also avoids the grave numerical problems mentioned above.

Our investigations are organized as follows. We start by collecting basic material about the rotation group SO(3) and the discrete SO(3) Fourier transform in Section 2. We then briefly outline our strategy for computing the transform.

[1]Institut für Mathematik, Universität zu Lübeck, keiner@math.uni-luebeck.de
[2]Institut Computational Mathematics, Technische Universität Braunschweig, vollrath@tu-bs.de

In Section 3, we present our main contribution, that is, a new efficient method to calculate a particular linear transformation that allows us to replace Wigner-$d$ functions of arbitrary orders with those of low orders. These functions arise with the decomposition of SO(3) basis functions.

Our results show that the matrix representation of this transformation is related to the eigendecomposition of certain semiseparable matrices. This enables us to employ a divide-and-conquer algorithm developed by Chandrasekaran an Gu [3] that we combine with the well-known fast multipole method (FMM) [9] to the desired fast algorithm.

We give a condensed but complete description of our entire fast SO(3) Fourier transform algorithm in Section 4. Finally, we compare in Section 5 our new method to the existing ones on the basis of some numerical results. Our conclusion is that the new method offers distinct advantages over previous approaches.

**1.1. Related Algorithms.** In addition to the previously mentioned examples, there exists a number of algorithms for the fast computation of the spherical harmonic transform which is closely related to the transforms of SO(3) basis functions. Another Driscoll-Healy-like method can be found in [28]. The main difference to the algorithm used in [17] is the way of coping with numerical instabilities. The latter uses discrete cosine transforms resulting in an $\mathcal{O}\big(L^2 \log^2 L \log(1/\varepsilon)\big)$ algorithm that is exact in exact arithmetic, but needs to employ a stabilization scheme apart from very small transform sizes. The former uses the fast multi- pole method (FMM) for approximate polynomial multiplication, thus providing some flexibility in the choice of internally needed interpolation nodes. The result is an approximate $\mathcal{O}\big(L^2 \log L \log(1/\varepsilon)\big)$ algorithm.

The method described in [30] is essentially based on the observation that the decomposed SO(3) basis functions are solutions to very similar differential equations. It shares many characteristics with the method we will discuss, but instead of semiseparable matrices this method deals with tridiagonal self-adjoint matrices. It enables an approximate $\mathcal{O}\big(L^2 \log L \log(1/\varepsilon)\big)$algorithm.

In [20] a method based on matrix compression is proposed that needs $\mathcal{O}(L^{5/2} \log L)$ arithmetic operations, or $\mathcal{O}(L^2 \log^2 L)$ for particular choice of $L$.

## 2. Preliminaries.

**2.1. Rotations in $\mathbb{R}^3$.** An orthogonal $3 \times 3$ matrix with unit determinant represents a linear mapping that is called a rotation in $\mathbb{R}^3$. The *special orthogonal group* SO(3) is the collection of these matrices,

$$\mathrm{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} : \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I},\ |\mathbf{R}| = 1\},$$

equipped with the usual group action, and neutral and inverse elements. One can conceive many different ways to characterize its structure. We follow Varshalovich, Moskalev, and Khersonski [33] with the Euler angle decomposition.

Definition 2.1. *The* Euler angle decomposition *of a rotation* $\mathbf{R} \in \mathrm{SO}(3)$ *is the representation*

$$\mathbf{R} = \mathbf{R}(\alpha, \beta, \gamma) = \mathbf{R}_Z(\alpha)\, \mathbf{R}_Y(\beta) \mathbf{R}_Z(\gamma),$$

*with angles* $\alpha, \gamma \in [0, 2\pi)$ *and* $\beta \in [0, \pi]$*, and the* $Y$*-axis and* $Z$*-axis rotation matrices*

$$\mathbf{R}_Y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}, \qquad \mathbf{R}_Z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

**2.2. Wigner-$D$ functions.** Any function $f$ defined on SO(3) can be represented in Euler angles, that is, we may write $f(\alpha,\beta,\gamma)$. The space of square integrable functions is denoted $\mathrm{L}^2\left(\mathrm{SO}(3)\right)$ and defined via the standard inner product

$$\langle f,g\rangle = \int_0^{2\pi}\int_0^{\pi}\int_0^{2\pi} f(\alpha,\beta,\gamma)\overline{g(\alpha,\beta,\gamma)}\,\sin\beta\,\mathrm{d}\gamma\,\mathrm{d}\beta\,\mathrm{d}\alpha.$$

A convenient orthogonal basis for $\mathrm{L}^2\big(\mathrm{SO}(3)\big)$ is given by the Wigner-$D$ functions

$$D_\ell^{m,n}(\alpha,\beta,\gamma) = \mathrm{e}^{-\mathrm{i}m\alpha}\,d_\ell^{m,n}(\cos\beta)\,\mathrm{e}^{-\mathrm{i}n\gamma}$$

which carry the degree $\ell$ and orders $m,n$ satisfying $\max\{|m|,|n|\}\le\ell$, and where $d_\ell^{m,n}$ are the Wigner-$d$ functions defined in Definition 2.3 below. With the usual Kronecker symbol $\delta_{j,k}=[j=k]$ we have

$$\langle D_\ell^{m,n}, D_{\ell'}^{m',n'}\rangle = \frac{8\pi^2}{2\ell+1}\delta_{\ell,\ell'}\delta_{m,m'}\delta_{n,n'},$$

hence, normalized Wigner-$D$ functions are given by $\tilde{D}_\ell^{m,n} = \frac{1}{2\pi}\left(\frac{2\ell+1}{2}\right)^{1/2} D_\ell^{m,n}$.

**2.3. Wigner-$d$ functions.** Before we describe Wigner-$d$ functions $d_\ell^{m,n}$, we introduce some symbols that we will conveniently use throughout the rest.

Definition 2.2. *For fixed orders $m$ and $n$ we define*

$$\begin{aligned} &\mu = |n-m|, && \nu = |n+m|, \\ &L_* = \max\{|m|,|n|\}, && s = \ell - L_*, \end{aligned} \qquad \varepsilon = \begin{cases} 1, & \text{if } m>n, \\ (-1)^{n-m}, & \text{if } m\le n. \end{cases}$$

We are now ready for our definition of Wigner-$d$ functions.

Definition 2.3. *Wigner-d functions $d_\ell^{m,n}$ are defined by*

$$d_\ell^{m,n}(x) = \varepsilon\left(\frac{s!(s+\mu+\nu)!}{(s+\mu)!(s+\nu)!}\right)^{1/2} 2^{-\frac{\mu+\nu}{2}}(1-x)^{\frac{\mu}{2}}(1+x)^{\frac{\nu}{2}}P_{\ell-L_*}^{(\mu,\nu)}(x), \tag{2.1}$$

*where $P_{\ell-L_*}^{(\mu,\nu)}(x)$ are the Jacobi polynomials.*

Note that $d_\ell^{m,n}$ is a polynomial of degree $\ell$ if $m+n$ is even. Otherwise, it is a polynomial of degree $\ell-1$ times a factor of $(1-x^2)^{1/2}$.

For fixed orders $m$ and $n$, the Wigner-$d$ functions represent an orthogonal system on the interval $[-1,1]$ and satisfy

$$\langle d_\ell^{m,n}, d_{\ell'}^{m,n}\rangle = \int_{-1}^{1} d_\ell^{m,n}(x)\,d_{\ell'}^{m,n}(x)\,\mathrm{d}x = \frac{2}{2\ell+1}\delta_{\ell,\ell'}. \tag{2.2}$$

The normalized flavours are therefore given by $\tilde{d}_\ell^{m,n} = \left(\frac{2\ell+1}{2}\right)^{1/2} d_\ell^{m,n}$. Wigner-$d$ functions also have a number of symmetries, for example,

$$d_\ell^{m,n} = (-1)^{n-m} d_\ell^{-m,-n} = (-1)^{n-m} d_\ell^{n,m} = d_\ell^{-n,-m}. \tag{2.3}$$

Another one is $\tilde{d}_\ell^{m,n}(\cos\beta) = \tilde{d}_\ell^{-m,n}\big(\cos(\pi-\beta)\big)$.

The single most important observation in our context is that Wigner-$d$ functions are solutions to the differential equation (see [34, p. 138])

$$(1-x^2)y'' - 2xy' + \left(\ell(\ell+1) - \frac{\mu^2}{2(1-x)} - \frac{\nu^2}{2(1+x)}\right) y = 0, \qquad y = d_\ell^{m,n}.$$

We may equivalently say that they are eigenfunctions of the differential operator

$$\mathcal{D}^{m,n}(y) = -(1-x^2)y'' + 2xy' + \left(\frac{\mu^2}{2(1-x)} + \frac{\nu^2}{2(1+x)}\right) y \tag{2.4}$$

to the eigenvalues $\ell(\ell+1)$.

**2.4. Wigner-$d$ functions of different orders.** At the heart of this text is a method to replace Wigner-$d$ functions $d_\ell^{m,n}$ of certain orders $m$ and $n$ with those of different orders $m'$ and $n'$. Our particular choice of $m'$ and $n'$ is given and justified in the following result.

LEMMA 2.4. *We denote by $\mathbb{D}_L^{m,n}$ the space spanned by the Wigner-d functions $d_\ell^{m,n}$ for $\ell = L_*, \dots, L$, and we define*

$$m' = \begin{cases} 0, & \text{if } m = n = 0, \\ 1, & \text{if } m = \pm n, \\ 0, & \text{else,} \end{cases} \qquad \text{and} \qquad n' = \begin{cases} 0, & \text{if } m = n = 0, \\ \pm 1, & \text{if } m = \pm n, \\ 2, & \text{if } m+n \text{ even,} \\ 1, & \text{if } m+n \text{ odd.} \end{cases}$$

*Then we have $\mathbb{D}_L^{m,n} \subseteq \mathbb{D}_L^{m',n'}$.*

*Proof.* If $m = n$, then $\mu = 0$, $\nu = |2n|$, and $m+n$ is an even number. The space $\mathbb{D}_L^{m,n}$ is spanned by Wigner-$d$ functions which are, in this case, proportional to the Jacobi polynomials

$$P_0^{(0,|n|)}, P_1^{(0,|n|)}, \dots, P_{L-|n|}^{(0,|n|)},$$

each multiplied by the factor $(1+x)^{|n|}$; cf. Definition 2.3. This is clearly a subspace of $\mathbb{D}_L^{1,1}$ which is spanned by the Jacobi polynomials

$$P_0^{(0,2)}, P_1^{(0,2)}, \dots, P_{L-1}^{(0,2)},$$

each multiplied by $(1+x)$. The reasoning in the remaining cases is entirely equivalent. ∎

Throughout the rest of this text, the orders $m'$ and $n'$ are always understood to depend on given orders $m$ and $n$.

**2.5. Nonequispaced discrete SO(3) Fourier transform.** We are now ready to describe the discrete Fourier transform on SO(3). Every function $f \in \mathrm{L}^2\big(\mathrm{SO}(3)\big)$ has a unique series expansion of the form

$$f = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \sum_{n=-\ell}^{\ell} \hat{f}_\ell^{m,n} \tilde{D}_\ell^{m,n},$$

where the Fourier coefficients $\hat{f}_\ell^{m,n}$ are given by the integrals

$$\hat{f}_\ell^{m,n} = \int_0^{2\pi} \int_0^{\pi} \int_0^{2\pi} f(\alpha,\beta,\gamma) \overline{\tilde{D}_\ell^{m,n}(\alpha,\beta,\gamma)} \sin\beta \, \mathrm{d}\gamma \, \mathrm{d}\beta \, \mathrm{d}\alpha. \tag{2.5}$$

For obvious reasons, we only deal with finite expansions of the form

$$f = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} \sum_{n=-\ell}^{\ell} \hat{f}_\ell^{m,n} \tilde{D}_\ell^{m,n}, \qquad \text{with } L \in \mathbb{N}_0. \tag{2.6}$$

To allow for a convenient notation, we introduce the index set

$$\mathcal{I}_L = \big\{(\ell, m, n) : \ell, |m|, |n| \in \mathbb{N}_0,\, \ell \le L,\, |m|, |n| \le l\big\}$$

which contains all admissible tupels $(\ell, m, n)$ that appear in (2.6). Similarly, we collect a number of $Q$ arbitrary nodes on SO(3) in the set

$$\mathcal{X}_Q = \{(\alpha_q, \beta_q, \gamma_q) : q = 1, \dots, Q\}.$$

Definition 2.5. *The* nonequispaced discrete SO(3) Fourier transform (NDSOFT) *is defined as the evaluation of the sums*

$$f(\alpha_q, \beta_q, \gamma_q) = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} \sum_{n=-\ell}^{\ell} \hat{f}_\ell^{m,n} \tilde{D}_\ell^{m,n}(\alpha_q, \beta_q, \gamma_q), \qquad q = 1, 2, \dots, Q, \tag{2.7}$$

*for given Fourier coefficients* $\hat{f}_\ell^{m,n}$ *and nodes* $(\alpha_q, \beta_q, \gamma_q)$.

In matrix-vector notation, we may write $\mathbf{f} = \mathbf{D}\,\hat{\mathbf{f}}$, with vectors $\mathbf{f} = \big(f(\alpha_q, \beta_q, \gamma_q)\big)$ and $\hat{\mathbf{f}} = \big(\hat{f}_\ell^{m,n}\big)$, and the matrix $\mathbf{D} = \big(\tilde{D}_\ell^{m,n}(\alpha_q, \beta_q, \gamma_q)\big)$.

The NDSOFT is a linear transformation, but is generally not invertible. However, we can recover the Fourier coefficients $\hat{f}_\ell^{m,n}$ from the function values $f(\alpha_q, \beta_q, \gamma_q)$ if a suitable quadrature rule with weights $w_q$ to discretize the integrals (2.5) is available. Then we could calculate the sums

$$\hat{f}_\ell^{m,n} = \sum_{q=1}^{Q} w_q\, f(\alpha_q, \beta_q, \gamma_q)\, \overline{\tilde{D}_\ell^{m,n}(\alpha_q, \beta_q, \gamma_q)}, \qquad (\ell, m, n) \in \mathcal{I}_L. \tag{2.8}$$

The *adjoint NDSOFT* is defined as the calculation of the product $\mathbf{D}^{\mathrm{H}}\,\mathbf{f}$, so we may identify (2.8) with the matrix-vector product

$$\hat{\mathbf{f}} = \mathbf{D}^{\mathrm{H}}\,\mathbf{W}\,\mathbf{f}, \qquad \text{with} \quad \mathbf{W} = \operatorname{diag}(w_q).$$

The direct calculation of the (adjoint) NDSOFT takes $\mathcal{O}(L^3 Q)$ operations. In the following, we develop an approximate $\mathcal{O}\big(L^3 \log L \log(1/\varepsilon) + (1/\varepsilon)^3 Q\big)$ algorithm to compute the same result within a prescribed accuracy $\varepsilon$ free of choice.

**2.6. Nonequispaced fast SO(3) Fourier transform.** This section is to outline our strategy for a fast approximate SO(3) Fourier transform algorithm. The chief idea to calculate the sums in (2.7) is to rewrite these as

$$f(\alpha, \beta, \gamma) = \sum_{\ell=-L}^{L} \sum_{m=-L}^{L} \sum_{n=-L}^{L} \hat{g}_\ell^{m,n} \mathrm{e}^{-\mathrm{i}m\alpha} \mathrm{e}^{-\mathrm{i}\ell\beta} \mathrm{e}^{-\mathrm{i}n\gamma}. \tag{2.9}$$

This is a plain three-dimensional Fourier sum and we can use the NFFT algorithm to evaluate it with $\mathcal{O}\big(L^3 \log L + (1/\varepsilon)^3 Q\big)$ operations; see [23]. The remaining question is: Can we calculate the coefficients $\hat{g}_\ell^{m,n}$ efficiently? We provide an affirmative answer through the following steps.

**First step.** We can rearrange (2.7) to

$$f(\alpha,\beta,\gamma) = \sum_{m=-L}^{L}\sum_{n=-L}^{L} \mathrm{e}^{-\mathrm{i}m\alpha}\,\mathrm{e}^{-\mathrm{i}n\gamma}\frac{1}{2\pi}\sum_{\ell=L_*}^{L} \hat{f}_\ell^{\,m,n}\tilde{d}_\ell^{\,m,n}(\cos\beta).$$

Lemma 2.4 allows us to replace the Wigner-$d$ functions in the inner sums with those of low orders. Setting $x = \cos\beta$, we arrive at

$$\frac{1}{2\pi}\sum_{\ell=L_*}^{L} \hat{f}_\ell^{\,m,n}\tilde{d}_\ell^{\,m,n}(x) = \frac{1}{2\pi}\sum_{\ell=L'_*}^{L} \bar{f}_\ell^{\,m,n}\tilde{d}_\ell^{\,m',n'}(x), \qquad m,n = -L,\dots,L. \tag{2.10}$$

Our task in Section 3 is to show that we can calculate the coefficients $\bar{f}_\ell^{\,m,n}$ from the coefficients $\hat{f}_\ell^{\,m,n}$ with no more than $\mathcal{O}\big(L^3\log L\log(1/\varepsilon)\big)$ arithmetic operations.

**Second step.** Clearly, the functions $\tilde{d}_\ell^{\,0,0}$, $\tilde{d}_\ell^{\,1,1}$, $\tilde{d}_\ell^{\,1,-1}$, and $\tilde{d}_\ell^{\,0,2}$ are all polynomials of degree $\ell$ while $\tilde{d}_\ell^{\,0,1}$ is a polynomial of degree $\ell-1$ multiplied by $(1-x^2)^{1/2}$. These are the only cases that remain in (2.10) after the first step. We rewrite the right-hand side for $m,n = -L,\dots,L$ using the Chebyshev polynomials of first kind $T_\ell(x)$,

$$\frac{1}{2\pi}\sum_{\ell=L'_*}^{L} \bar{f}_\ell^{\,m,n}\tilde{d}_\ell^{\,m',n'}(x) = \frac{1}{2\pi}\sum_{\ell=0}^{L-\chi} \tilde{f}_\ell^{\,m,n}(1-x^2)^{\chi/2}T_\ell(x), \tag{2.11}$$

where $\chi = [m+n \text{ odd}]$. We show in Section 4 how the coefficients $\tilde{f}_\ell^{\,m,n}$ may be computed from the coefficients $\bar{f}_\ell^{\,m,n}$ with $\mathcal{O}\big(L^3\log(1/\varepsilon)\big)$ arithmetic operations.

**Third step.** We are now able to replace the Chebyshev polynomials of first kind with complex exponentials,

$$\frac{1}{2\pi}\sum_{\ell=0}^{L-\chi} \tilde{f}_\ell^{\,m,n}(1-x^2)^{\chi/2}T_\ell(x) = \sum_{\ell=-L}^{L} \hat{g}_\ell^{\,m,n}\mathrm{e}^{-\mathrm{i}\ell\beta}, \qquad m,n = -L,\dots,L.$$

Section 4 explains how we can compute the coefficients $\hat{g}_\ell^{\,m,n}$ from the coefficients $\tilde{f}_\ell^{\,m,n}$ with $\mathcal{O}(L^3)$ arithmetic operations. The obtained form is now ready to be inserted into (2.7) to become (2.9).

**3. Expansions in Wigner-$d$ functions of distinct orders.** In this section we fix the orders $m$ and $n$. Our primary goal is to analyze the relationship between the coefficients $\hat{f}_\ell^{\,m,n}$ and $\bar{f}_\ell^{\,m,n}$ in the formula

$$f = \sum_{\ell=L_*}^{L} \hat{f}_\ell^{\,m,n}\tilde{d}_\ell^{\,m,n} = \sum_{\ell=L'_*}^{L} \bar{f}_\ell^{\,m,n}\tilde{d}_\ell^{\,m',n'}.$$

More precisely, we seek to efficiently calculate the linear transformation $\bar{\mathbf{f}}^{m,n} = \mathbf{A}^{m,n}\,\hat{\mathbf{f}}^{m,n}$, represented by the matrix

$$\mathbf{A}^{m,n} = (a_{\ell,k}) = \Big(\langle \tilde{d}_\ell^{\,m',n'}, \tilde{d}_k^{\,m,n}\rangle\Big),$$

with the vectors $\hat{\mathbf{f}}^{m,n} = (\hat{f}_\ell^{\,m,n})$ and $\bar{\mathbf{f}}^{m,n} = (\bar{f}_\ell^{\,m,n})$. To simplify the following investigation, let us observe special cases and symmetries.

LEMMA 3.1. *We have* $\mathbf{A}^{m,n} = \mathbf{I}$ *whenever*

$$(m,n) \in \big\{(0,0), (\pm 1, \pm 1), (\pm 1, \mp 1), (0,1), (-1,0), (0, \pm 2), (\pm 2, 0), (2,0)\big\}. \tag{3.1}$$

*Furthermore, the following identities hold:*

$$\mathbf{A}^{m,n} = (-1)^{m-n}\mathbf{A}^{-m,-n}, \qquad \mathbf{A}^{m,n} = (-1)^{m-n}\mathbf{A}^{n,m},$$
$$\mathbf{A}^{m,n} = \operatorname{diag}\big((-1)^{\ell+m+\delta_{|m|,|n|}}\big)_{\ell=L'_*}^{L}\, \mathbf{A}^{-m,n}\, \operatorname{diag}\big((-1)^{\ell}\big)_{\ell=L_*}^{L}.$$

*Proof.* All statements are direct consequences of (2.3) and Lemma 2.4. ∎

Without loss of generality, we safely assume from now on that $0 \le n \le m \le L$ and that $m$ and $n$ do not fall into any of the simple cases in (3.1). The key idea for our analysis of the transformation represented by the matrix $\mathbf{A}^{m,n}$ is based on a related matrix $\mathbf{G}^{m,n}$ that we define as follows.

DEFINITION 3.2. *For the matrix* $\mathbf{A}^{m,n}$ *we define the matrix* $\mathbf{G}^{m,n} = (g_{\ell,k})$ *by*

$$g_{\ell,k} = \langle \tilde{d}_\ell^{\,m',n'}, \mathcal{D}^{m,n}(\tilde{d}_k^{\,m',n'})\rangle, \qquad \ell, k = L'_*, \dots, L.$$

The next result shows how the matrices $\mathbf{A}^{m,n}$ and $\mathbf{G}^{m,n}$ are related.

LEMMA 3.3. *Let* $\mathbf{a}_k = (a_{L'_*,k}, \dots, a_{L,k})^{\mathrm{T}}$ *with* $L_* \le k \le L$ *denote a column of the matrix* $\mathbf{A}^{m,n}$*. Then* $\mathbf{a}_k$ *is a normalized eigenvector of the matrix* $\mathbf{G}^{m,n}$ *to the eigenvalue* $k(k+1)$*.*

*Proof.* We recall that $\tilde{d}_k^{m,n} = \sum_{\ell=L'_*}^{L} a_{\ell,k}\tilde{d}_\ell^{\,m',n'}$. Then we have

$$(\mathbf{G}^{m,n}\,\mathbf{a}_k)_\ell = \sum_{j=L'_*}^{L} g_{\ell,j}a_{j,k} = \sum_{j=L'_*}^{L} \langle \tilde{d}_\ell^{\,m',n'}, \mathcal{D}^{m,n}(\tilde{d}_j^{\,m',n'})\rangle a_{j,k} = \langle \tilde{d}_\ell^{\,m',n'}, \mathcal{D}^{m,n}(\tilde{d}_k^{\,m,n})\rangle.$$

Now we can use that $\mathcal{D}^{m,n}(\tilde{d}_k^{m,n}) = k(k+1)\tilde{d}_k^{m,n}$ and work backwards to get

$$(\mathbf{G}^{m,n}\,\mathbf{a}_k)_\ell = k(k+1)\mathbf{a}_k.$$

The vectors $\mathbf{a}_k$ are normalized since the Wigner-$d$ functions for the order $m, n$ and $m', n'$ are normalized themselves. ∎

Our plan for the rest of this section is as follows. After calculating the entries of the matrix $\mathbf{G}^{m,n}$ explicitly, we will observe that it belongs to a particular class of structured matrices. This, as will be shown, can be exploited to obtain a fast approximate method to apply the desired matrix $\mathbf{A}^{m,n}$ to any vector.

**3.1. Explicit expressions for the matrix $\mathbf{G}^{m,n}$.** Our task is to calculate the entries of the matrix $\mathbf{G}^{m,n}$. To this end, we observe that the difference between the two differential operators $\mathcal{D}^{m,n}$ and $\mathcal{D}^{m',n'}$ has the form

$$\mathcal{D}^{m,n} - \mathcal{D}^{m',n'} = \mathcal{D}^{-} + \mathcal{D}^{+}, \quad \text{with} \quad \mathcal{D}^{-}(y) = \frac{\mu^2 - \mu'^2}{2(1-x)}y, \quad \mathcal{D}^{+}(y) = \frac{\nu^2 - \nu'^2}{2(1+x)}y.$$

We split the entries $g_{\ell,k}$ of the matrix $\mathbf{G}^{m,n}$ as follows,

$$g_{\ell,k} = \langle \tilde{d}_\ell^{\,m',n'}, \mathcal{D}^{m',n'}(\tilde{d}_k^{\,m',n'})\rangle + \langle \tilde{d}_\ell^{\,m',n'}, \mathcal{D}^{-}(\tilde{d}_k^{\,m',n'})\rangle + \langle \tilde{d}_\ell^{\,m',n'}, \mathcal{D}^{+}(\tilde{d}_k^{\,m',n'})\rangle,$$

and write $\mathbf{G}^{m,n} = \hat{\mathbf{D}} + \mathbf{S}^- + \mathbf{S}^+$, where

$$\begin{aligned}
\hat{\mathbf{D}} &= (\hat{d}_{\ell,k}), \quad d_{\ell,k} = \langle \mathcal{D}^{m',n'}(\tilde{d}_k^{m',n'}), \tilde{d}_\ell^{m',n'} \rangle = k(k+1)\delta_{\ell,k}, \\
\mathbf{S}^- &= (s^-_{\ell,k}), \quad s^-_{\ell,k} = \langle \mathcal{D}^{-}(\tilde{d}_k^{m',n'}), \tilde{d}_\ell^{m',n'} \rangle = \int_{-1}^{1} \tilde{d}_k^{m',n'}(x) \frac{\mu^2 - \mu'^2}{2(1-x)} \tilde{d}_\ell^{m',n'}(x)\, \mathrm{d}x, \\
\mathbf{S}^+ &= (s^+_{\ell,k}), \quad s^+_{\ell,k} = \langle \mathcal{D}^{+}(\tilde{d}_k^{m',n'}), \tilde{d}_\ell^{m',n'} \rangle = \int_{-1}^{1} \tilde{d}_k^{m',n'}(x) \frac{\nu^2 - \nu'^2}{2(1+x)} \tilde{d}_\ell^{m',n'}(x)\, \mathrm{d}x.
\end{aligned} \tag{3.2}$$

The next goal is to state explicit expressions for the entries $s^{\pm}_{\ell,k}$.

LEMMA 3.4. *For $\ell \le k$ the entries of the symmetric matrices $\mathbf{S}^-$ and $\mathbf{S}^+$ are*

$$s^-_{\ell,k} = \big(\mu^2 - \mu'^2\big) h_{\ell,k}, \qquad s^+_{\ell,k} = (-1)^{l+k} \big(\nu^2 - \nu'^2\big) h_{\ell,k},$$

$$h_{\ell,k} = \frac{\big((\ell+\frac{1}{2})(k+\frac{1}{2})\big)^{1/2}}{4} \begin{cases} \dfrac{\ell(\ell+1)}{k(k+1)} & \text{if } m = \pm n, \\ \left( \dfrac{(\ell-1)\ell(\ell+1)(\ell+2)}{(k-1)k(k+1)(k+2)} \right)^{1/2} & \text{if } m+n \text{ even}, \\ 2\left( \dfrac{\ell(\ell+1)}{k(k+1)} \right)^{1/2} & \text{if } m+n \text{ odd}. \end{cases}$$

*Proof.* We give the proof for the entries $s^-_{\ell,k}$. The procedure for $s^+_{\ell,k}$ is analogous. Our first aim is to fully expand the expression for $s^-_{\ell,k}$ using (2.3). We obtain

$$\begin{aligned}
s^-_{\ell,k} = \frac{\mu^2 - \mu'^2}{2^{\mu'+\nu'+1}} \left( \frac{(k-L'_*)!(k-L'_*+\mu'+\nu')!}{(k-L'_*+\mu')!(k-L'_*+\nu')!} \frac{(\ell-L'_*)!(\ell-L'_*+\mu'+\nu')!}{(\ell-L'_*+\mu')!(\ell-L'_*+\nu')!} \right)^{1/2} \\
\times \big((\ell+\tfrac{1}{2})(k+\tfrac{1}{2})\big)^{1/2} \int_{-1}^{1} (1-x)^{\mu'-1}(1+x)^{\nu'} P^{(\mu',\nu')}_{k-L'_*}(x) P^{(\mu',\nu')}_{\ell-L'_*}(x)\, \mathrm{d}x.
\end{aligned}$$

Next, we specialize the expression to the respective cases for the orders $m, n$, and hence $m', n'$. Note that for $m = n$, we have $s^-_{\ell,k} = 0$. In the remaining cases we have

$$\begin{aligned}
s^-_{\ell,k} =& \frac{1}{4}(\mu^2 - \mu'^2)\big((\ell+\tfrac{1}{2})(k+\tfrac{1}{2})\big)^{1/2} \\
&\times \begin{cases} \dfrac{1}{2} \displaystyle\int_{-1}^{1} (1-x) P^{(2,0)}_{k-1}(x) P^{(2,0)}_{\ell-1}(x)\, \mathrm{d}x & \text{if } n = -m, \\ \left( \dfrac{(k+1)(k+2)}{(k-1)k} \dfrac{(\ell+1)(\ell+2)}{(\ell-1)\ell} \right)^{1/2} & \\ \quad \times \dfrac{1}{8} \displaystyle\int_{-1}^{1} (1-x)(1+x)^2 P^{(2,2)}_{k-2}(x) P^{(2,2)}_{\ell-2}(x)\, \mathrm{d}x & \text{if } n+m \text{ even}, \\ \left( \dfrac{(k+1)}{k} \dfrac{(\ell+1)}{\ell} \right)^{1/2} \dfrac{1}{2} \displaystyle\int_{-1}^{1} (1+x) P^{(1,1)}_{k-1}(x) P^{(1,1)}_{\ell-1}(x)\, \mathrm{d}x & \text{if } n+m \text{ odd}. \end{cases}
\end{aligned}$$

We now derive explicit expressions for the three remaining integrals, where we can assume $\ell \le k$ without loss of generality.

We start with the case when $n+m$ is odd. Our main ingredient is the relation

$$\int P^{(\alpha,\beta)}_\ell(x)\, \mathrm{d}x = \frac{2}{\ell+\alpha+\beta} P^{(\alpha-1,\beta-1)}_{\ell+1}(x).$$

Then we can calculate the respective integral with partial integration,

$$\frac{1}{2}\int_{-1}^{1}(1+x)P_{k-1}^{(1,1)}(x)P_{\ell-1}^{(1,1)}(x)\,\mathrm{d}x = \frac{1}{\ell+1}\left[(1+x)P_{k-1}^{(1,1)}(x)P_{\ell}^{(0,0)}(x)\right]_{-1}^{1} = \frac{2k}{\ell+1}.$$

The next case, $m+n$ even, is slightly more complicated but can be handled by applying partial integration twice,

$$\begin{aligned}
&\frac{1}{8}\int_{-1}^{1}(1-x)(1+x)^2P_{k-2}^{(2,2)}(x)P_{\ell-2}^{(2,2)}(x)\,\mathrm{d}x\\
=&\frac{1}{8}\int_{-1}^{1}(-x^3-x^2+x+1)P_{\ell-2}^{(2,2)}(x)P_{k-2}^{(2,2)}(x)\,\mathrm{d}x\\
=&\frac{-1}{4(k+2)}\Bigg(\int_{-1}^{1}(-3x^2-2x+1)P_{\ell-2}^{(2,2)}(x)P_{k-1}^{(1,1)}(x)\mathrm{d}x\\
&\qquad\qquad+\int_{-1}^{1}(-x^3-x^2+x+1)\frac{(\ell+3)}{2}P_{\ell-3}^{(3,3)}P_{k-1}^{(1,1)}(x)\,\mathrm{d}x\Bigg)\\
=&\frac{-1}{2(k+1)(k+2)}\left[(-3x^2-2x+1)P_{\ell-2}^{(2,2)}(x)P_{k}^{(0,0)}(x)\right]_{-1}^{1}\\
=&\frac{2}{(k+1)(k+2)}P_{\ell-2}^{(2,2)}(1)P_{k}^{(0,0)}(1)\\
=&\frac{(\ell-1)\ell}{(k+1)(k+2)}.
\end{aligned}$$

Now for the harder part, that is, the case $n=-m$. In the following, we calculate the corresponding integral $I_{\ell,k}$ by double induction over $\ell$ and $k$ to

$$I_{\ell,k} = \frac{1}{2}\int_{-1}^{1}(1-x)P_{k-1}^{(2,0)}(x)P_{\ell-1}^{(2,0)}(x)\,\mathrm{d}x = \frac{\ell(\ell+1)}{k(k+1)}. \tag{3.3}$$

The base case $\ell=1$ for the outer induction over $\ell$ is resolved by the formula [8, p. 228, 7.391, 2.],

$$I_{1,k} = \frac{1}{2}\int_{-1}^{1}(1-x)P_{k-1}^{(2,0)}(x)\,\mathrm{d}x = \frac{2}{k(k+1)}, \qquad k=1,2,\ldots.$$

For the inductive step, we fix $\ell\geq 2$ and assume that (3.3) is true for $\ell$ replaced by $\ell-1$ and all $k\geq\ell-1$. We now calculate the integral $I_{\ell,k}$ by the inner induction over $k$. The first base case is given by formula [8, p. 228, 7.391, 5.],

$$I_{\ell,\ell} = \frac{1}{2}\int_{-1}^{1}(1-x)\left(P_{\ell-1}^{(2,0)}(x)\right)^2\,\mathrm{d}x = 1.$$

The second base case is the integral $I_{\ell,\ell+1}$. Using the recursive formula [29, p. 71],

$$P_{\ell}^{(2,0)}(x) = \left(\frac{2\ell+1}{\ell^2(\ell+2)}+\frac{(\ell+1)(2\ell+1)}{\ell(\ell+2)}x\right)P_{\ell-1}^{(2,0)}(x) - \frac{(\ell-1)(\ell+1)^2}{\ell^2(\ell+2)}P_{\ell-2}^{(2,0)}(x), \tag{3.4}$$

we obtain

$$\begin{aligned} I_{\ell,\ell+1} &= \frac{1}{2}\int_{-1}^{1}(1-x)P_{\ell}^{(2,0)}(x)P_{\ell-1}^{(2,0)}(x)\,\mathrm{d}x \\ &= \frac{2\ell+1}{\ell^2(\ell+2)}\bigg(\frac{1}{2}\int_{-1}^{1}(1-x)\left(P_{\ell-1}^{(2,0)}(x)\right)^2\,\mathrm{d}x \\ &\quad + \ell(\ell+1)\frac{1}{2}\int_{-1}^{1}(1-x)x\left(P_{\ell-1}^{(2,0)}(x)\right)^2\,\mathrm{d}x \\ &\quad - \frac{(\ell-1)(\ell+1)^2}{2\ell+1}\frac{1}{2}\int_{-1}^{1}(1-x)P_{\ell-2}^{(2,0)}(x)P_{\ell-1}^{(2,0)}(x)\,\mathrm{d}x\bigg). \end{aligned} \tag{3.5}$$

The first and the third integral are identical to $I_{\ell,\ell}$ and $I_{\ell-1,\ell}$, respectively, and the second one may be rearranged to

$$\frac{1}{2}\int_{-1}^{1}(1-x)x\left(P_{\ell-1}^{(2,0)}(x)\right)^2\,\mathrm{d}x = I_{\ell,\ell} - \frac{1}{2}\int_{-1}^{1}(1-x)^2\left(P_{\ell-1}^{(2,0)}(x)\right)^2\,\mathrm{d}x.$$

For the integral on the right-hand side the following formula holds,

$$\int_{-1}^{1}(1-x)^2\left(P_{\ell-1}^{(2,0)}(x)\right)^2\,\mathrm{d}x = \frac{8}{2\ell+1}.$$

Now, all quantities in (3.5) are known and we arrive at

$$I_{\ell,\ell+1} = \frac{2\ell+1}{\ell^2(\ell+2)}\left(I_{\ell,\ell} + \frac{4\ell(\ell+1)}{2\ell+1} - \frac{(\ell-1)(\ell+1)^2}{2\ell+1}\frac{1}{2}I_{\ell-1,\ell}\right) = \frac{\ell}{\ell+2}.$$

To complete our proof, we work out the inductive step for arbitrary but fixed $k \geq \ell+2$. Let us assume that (3.3) is valid if we replace $k$ by $k-1$ or $k-2$. With the help of the recursive formula (3.4) we finally obtain

$$\begin{aligned} I_{\ell,k} &= \frac{1}{2}\int_{-1}^{1}(1-x)P_{k-1}^{(2,0)}(x)P_{\ell-1}^{(2,0)}(x)\,\mathrm{d}x \\ &= \frac{1}{2(k-1)^2(k+1)}\bigg((2k-1)\int_{-1}^{1}(1-x)P_{k-2}^{(2,0)}(x)P_{\ell-1}^{(2,0)}(x)\,\mathrm{d}x \\ &\qquad + (k-1)k(2k-1)\int_{-1}^{1}(1-x)xP_{k-2}^{(2,0)}(x)P_{\ell-1}^{(2,0)}(x)\,\mathrm{d}x \\ &\qquad - (k-2)k^2\int_{-1}^{1}(1-x)P_{k-3}^{(2,0)}(x)P_{\ell-1}^{(2,0)}(x)\,\mathrm{d}x\bigg) \\ &= \frac{2(2k-1)I_{\ell,k-1} + 2(k-1)k(2k-1)I_{\ell,k-1} - 2(k-2)k^2I_{\ell,k-2}}{2(k-1)^2(k+1)} = \frac{\ell(\ell+1)}{k(k+1)}. \end{aligned}$$

□

We are now endowed with explicit expressions for the entries of the matrix $\mathbf{G}^{m,n}$ that has its eigenvectors, up to scaling, identical to the columns of the matrix $\mathbf{A}^{m,n}$. This puts us in a position to observe that the matrix $\mathbf{G}^{m,n}$ belongs to the class of *semiseparable matrices* that is defined as follows.

DEFINITION 3.5. *An $n \times n$ matrix $\mathbf{G}$ is called* diagonal plus symmetric $(R)$-generator representable semiseparable *if there exist vectors $\mathbf{d}, \mathbf{u}_r, \mathbf{v}_r \in \mathbb{R}^n$, $r =$*

$1, \ldots, R$, *such that*

$$\mathbf{G} = \operatorname{diag}(\mathbf{d}) + \sum_{r=1}^{R} \Big( \operatorname{triu}\left(\mathbf{u}_r\, \mathbf{v}_r^{\mathrm{T}}\right) + \operatorname{tril}\left(\mathbf{v}_r\, \mathbf{u}_r^{\mathrm{T}}\right) \Big).$$

Here, $\operatorname{diag}(\mathbf{d})$ denotes the diagonal matrix that contains the entries of the vector $\mathbf{d}$. Furthermore, $\operatorname{triu}\left(\mathbf{u}_r\, \mathbf{v}_r^{\mathrm{T}}\right)$ and $\operatorname{tril}\left(\mathbf{v}_r\, \mathbf{u}_r^{\mathrm{T}}\right)$ are the strictly upper and lower triangular parts of the rank-one matrices $\mathbf{u}_r\, \mathbf{v}_r^{\mathrm{T}}$ and $\mathbf{v}_r\, \mathbf{u}_r^{\mathrm{T}}$, respectively; see [31]. Clearly, the matrix $\mathbf{G}^{m,n}$ falls into this category.

COROLLARY 3.6. *The matrix* $\mathbf{G}^{m,n}$ *is diagonal plus symmetric* (2)*-generator representable semiseparable,*

$$\mathbf{G} = \operatorname{diag}(\mathbf{d}) + \operatorname{triu}\left(\mathbf{u}\, \mathbf{v}^{\mathrm{T}}\right) + \operatorname{tril}\left(\mathbf{v}\, \mathbf{u}^{\mathrm{T}}\right) + \operatorname{triu}\left(\mathbf{x}\, \mathbf{y}^{\mathrm{T}}\right) + \operatorname{tril}\left(\mathbf{y}\, \mathbf{x}^{\mathrm{T}}\right), \tag{3.6}$$

*with the vectors* $\mathbf{d} = \left(d_\ell\right)$, $\mathbf{u} = (u_\ell)$, $\mathbf{v} = (v_\ell)$, $\mathbf{x} = (x_\ell)$, *and* $\mathbf{y} = (y_\ell)$ *for* $\ell = L'_*, \ldots, L$, *where*

$$\begin{aligned}
&d_\ell = \ell(\ell+1) + \left(\mu^2 - \mu'^2 + \nu^2 - \nu'^2\right) \frac{\gamma^2(2\ell+1)}{8},\\
&u_\ell = w_\ell h_\ell, \quad v_\ell = w_\ell h_\ell^{-1}, \qquad x_\ell = z_\ell h_\ell, \quad y_\ell = z_\ell h_\ell^{-1},\\
&w_\ell = \frac{\gamma}{2}\big((\mu^2 - \mu'^2)(\ell + \tfrac{1}{2})\big)^{1/2}, \qquad z_\ell = (-1)^\ell \frac{\gamma}{2}\big((\nu^2 - \nu'^2)(\ell + \tfrac{1}{2})\big)^{1/2},\\
&h_\ell = \begin{cases} \ell(\ell+1), & \text{if } m = \pm n,\\ \big((\ell-1)\ell(\ell+1)(\ell+2)\big)^{1/2}, & \text{if } m+n \text{ even},,\\ \big(\ell(\ell+1)\big)^{1/2}, & \text{if } m+n \text{ odd}, \end{cases} \quad \gamma = \begin{cases} \sqrt{2}, & \text{if } m+n \text{ odd},\\ 1, & \text{else}. \end{cases}
\end{aligned}$$

REMARK 3.7. *It is not entirely surprising that the results in this section can be generalized to all classical associated functions. A detailed description of the general case is found in [12].*

**3.2. A fast algorithm for the symmetric semiseparable eigenproblem.** We need a fast algorithm to apply the eigenvector matrix of the matrix $\mathbf{G}^{m,n}$ to a vector. In the following we therefore give a brief description of a minor extension of a method developed by Chandrasekaran and Gu [3]. For specific details concerning the implementation, we refer the reader to the original manuscript.

The algorithm is based on the divide-and-conquer paradigm. In the divide phase, the matrix at hand is split into successively smaller matrices of the same type until we can afford to compute their eigendecompositions with a standard algorithm. In the conquer phase, the eigendecompositions of smaller matrices are combined to the desired eigendecomposition. At this point, we will also see how we can apply the eigenvector matrix to a vector efficiently.

**3.2.1. Divide phase.** Given a symmetric (2)-generator representable semiseparable matrix

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

we would like to write this in the form of several smaller matrices. This can be done in the following way. We take $\rho_1, \rho_2 = \pm 1$ to be freely chosen. Then we split each of

the vectors $\mathbf{d}$, $\mathbf{u}$, $\mathbf{v}$, $\mathbf{x}$, and $\mathbf{y}$ into 2 vectors with the first $\lfloor n/2 \rfloor$ components in the first vector and the remaining components in the second vector. That is, we define $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{y}_1$, $\mathbf{y}_2$, two additional vectors $\mathbf{w}_1$, $\mathbf{w}_2$, and write $\mathbf{G}$ such that

$$\mathbf{d} = \begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{pmatrix}, \ \mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix}, \ \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix}, \ \mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix},$$

$$\mathbf{G} = \begin{pmatrix} \hat{\mathbf{G}}_1 & 0 \\ 0 & \hat{\mathbf{G}}_2 \end{pmatrix} + \rho_1 \, \mathbf{w}_1 \, \mathbf{w}_1^{\mathrm{T}} + \rho_2 \, \mathbf{w}_2 \, \mathbf{w}_2^{\mathrm{T}}, \qquad \mathbf{w}_1 = \begin{pmatrix} \rho_1 \, \mathbf{u}_1 \\ \mathbf{v}_2 \end{pmatrix}, \ \mathbf{w}_2 = \begin{pmatrix} \rho_2 \, \mathbf{x}_1 \\ \mathbf{y}_2 \end{pmatrix},$$

where $\hat{\mathbf{G}}_1$ and $\hat{\mathbf{G}}_2$ are defined to the symmetric (2)-generator representable semiseparable matrices

$$\begin{aligned} \hat{\mathbf{G}}_1 = {} & \operatorname{diag}\left(\mathbf{d}_1 - \rho_1 \operatorname{diag}\left(\mathbf{u}_1 \mathbf{u}_1^{\mathrm{T}}\right) - \rho_2 \operatorname{diag}\left(\mathbf{x}_1 \mathbf{x}_1^{\mathrm{T}}\right)\right) \\ & + \operatorname{triu}\left(\mathbf{u}_1 (\mathbf{v}_1 - \rho_1 \mathbf{u}_1)^{\mathrm{T}}\right) + \operatorname{tril}\left((\mathbf{v}_1 - \rho_1 \mathbf{u}_1)\mathbf{u}_1^{\mathrm{T}}\right) \\ & + \operatorname{triu}\left(\mathbf{x}_1 (\mathbf{y}_1 - \rho_2 \mathbf{x}_1)^{\mathrm{T}}\right) + \operatorname{tril}\left((\mathbf{y}_1 - \rho_2 \mathbf{x}_1)\mathbf{x}_1^{\mathrm{T}}\right) \\ \hat{\mathbf{G}}_2 = {} & \operatorname{diag}\left(\mathbf{d_2} - \rho_1 \operatorname{diag}\left(\mathbf{v}_2 \mathbf{v}_2^{\mathrm{T}}\right) - \rho_2 \operatorname{diag}\left(\mathbf{y}_2 \mathbf{y}_2^{\mathrm{T}}\right)\right) \\ & + \operatorname{triu}\left((\mathbf{u}_2 - \rho_1 \mathbf{v}_2)\mathbf{v}_2^{\mathrm{T}}\right) + \operatorname{tril}\left(\mathbf{v}_2 (\mathbf{u}_2 - \rho_1 \mathbf{v}_2)^{\mathrm{T}}\right) \\ & + \operatorname{triu}\left((\mathbf{x}_2 - \rho_2 \mathbf{y}_2)\mathbf{y}_2^{\mathrm{T}}\right) + \operatorname{tril}\left(\mathbf{y}_2 (\mathbf{x}_2 - \rho_2 \mathbf{y}_2)^{\mathrm{T}}\right). \end{aligned}$$

This result can be easily verified. We may now decompose the matrices $\hat{\mathbf{G}}_1$ and $\hat{\mathbf{G}}_2$ into a similar pattern.

**3.2.2. Conquer phase.** In the conquer phase we regroup the subproblems into larger problems. Suppose that two symmetric (2)-generator representable semiseparable matrices $\hat{\mathbf{G}}_1$ and $\hat{\mathbf{G}}_2$, obtained from the divide phase, have the eigendecomposition

$$\hat{\mathbf{G}}_1 = \mathbf{V}_1 \mathbf{\Lambda}_1 \mathbf{V}_1^{\mathrm{T}} \quad \text{and} \quad \hat{\mathbf{G}}_2 = \mathbf{V}_2 \mathbf{\Lambda}_2 \mathbf{V}_2^{\mathrm{T}},$$

with diagonal eigenvalue matrices $\mathbf{\Lambda}_1$ and $\mathbf{\Lambda}_2$ and orthogonal eigenvector matrices $\mathbf{V}_1$ and $\mathbf{V}_2$. Then $\mathbf{G}$ has the representation

$$\mathbf{G} = \begin{pmatrix} \mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2 \end{pmatrix} \left(\mathbf{\Lambda} + \rho_1 \, \mathbf{z}_1 \mathbf{z}_1^{\mathrm{T}} + \rho_2 \, \mathbf{z}_2 \mathbf{z}_2^{\mathrm{T}}\right) \begin{pmatrix} \mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2 \end{pmatrix}^{\mathrm{T}},$$

where

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_1 & 0 \\ 0 & \mathbf{\Lambda}_2 \end{pmatrix}, \quad \mathbf{z}_1 = \begin{pmatrix} \mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2 \end{pmatrix}^{\mathrm{T}} \mathbf{w}_1, \quad \mathbf{z}_2 = \begin{pmatrix} \mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2 \end{pmatrix}^{\mathrm{T}} \mathbf{w}_2,$$

and where $\rho_1, \rho_2$ are defined as they were for the divide phase. Now suppose that we can efficiently compute the eigendecomposition of the symmetric rank-two modified diagonal matrix $\mathbf{\Lambda} + \rho_1 \, \mathbf{z}_1 \mathbf{z}_1^{\mathrm{T}} + \rho_2 \, \mathbf{z}_2 \mathbf{z}_2^{\mathrm{T}}$, which is written as

$$\mathbf{\Lambda} + \rho_1 \, \mathbf{z}_1 \mathbf{z}_1^{\mathrm{T}} + \rho_2 \, \mathbf{z}_2 \mathbf{z}_2^{\mathrm{T}} = \mathbf{U} \Omega \mathbf{U}^{\mathrm{T}}.$$

We can then write the eigendecomposition of $\mathbf{G}$ as

$$\mathbf{G} = \begin{pmatrix} \mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2 \end{pmatrix} \mathbf{U} \, \mathbf{\Omega} \, \mathbf{U}^{\mathrm{T}} \begin{pmatrix} \mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2 \end{pmatrix}^{\mathrm{T}} = \mathbf{V} \, \mathbf{\Omega} \, \mathbf{V}^{\mathrm{T}}, \quad \text{with} \quad \mathbf{V} = \begin{pmatrix} \mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2 \end{pmatrix} \mathbf{U}.$$

The critical step is the calculation of the eigendecomposition of the rank-two modified diagonal matrix $\mathbf{\Lambda} + \rho_1\,\mathbf{z}_1\mathbf{z}_1^{\mathrm{T}} + \rho_2\,\mathbf{z}_2\mathbf{z}_2^{\mathrm{T}}$ and the efficient application of the eigenvector matrix $\mathbf{U}$. Important for us is that the application of the matrix $\mathbf{U}$ can be accelerated by the fast multipole method (FMM) [9]. This way, we achieve an algorithm that applies the eigenvector matrix $\mathbf{V}$ to any vector with $\mathcal{O}\big(L\log L\log(1/\varepsilon)\big)$ arithmetic operations. For the details, we refer the reader to [3] and the references therein.

**4. Detailed description of the algorithm.** In this section, we offer a more detailed description of our nonequispaced fast SO(3) Fourier transform algorithm.

**4.1. First step.** We start with given Fourier coefficients $\hat{f}_\ell^{\,m,n}$ in

$$\begin{aligned} f(\alpha,\beta,\gamma) &= \sum_{\ell=0}^{L}\sum_{m=-\ell}^{\ell}\sum_{n=-\ell}^{\ell}\hat{f}_\ell^{\,m,n}\tilde{D}_\ell^{m,n}(\alpha,\beta,\gamma) \\ &= \frac{1}{2\pi}\sum_{m=-L}^{L}\sum_{n=-L}^{L}\mathrm{e}^{-\mathrm{i}m\alpha}\,\mathrm{e}^{-\mathrm{i}n\gamma}\sum_{\ell=L_*}^{L}\hat{f}_\ell^{\,m,n}\tilde{d}_\ell^{m,n}(\cos\beta). \end{aligned}$$

We combine the results from Section 3.1 and the algorithm described in Section 3.2 to compute the coefficients $\bar{f}_\ell^{\,m,n}$ from the coefficients $\hat{f}_\ell^{\,m,n}$ in

$$f(\alpha,\beta,\gamma) = \frac{1}{2\pi}\sum_{m=-L}^{L}\sum_{n=-L}^{L}\mathrm{e}^{-\mathrm{i}m\alpha}\,\mathrm{e}^{-\mathrm{i}n\gamma}\sum_{\ell=L'_*}^{L}\bar{f}_\ell^{\,m,n}\tilde{d}_\ell^{m',n'}(\cos\beta).$$

We can represent this step in matrix-vector notation by

$$\bar{\mathbf{f}}^{\,m,n} = \mathbf{A}^{m,n}\,\hat{\mathbf{f}}^{\,m,n}, \qquad m,n=-L,\ldots,L. \tag{4.1}$$

For a fixed pair of orders $m,n$ we need $\mathcal{O}\big(L\log L\log(1/\varepsilon)\big)$ arithmetic operations to compute the transformation. This totals up to $\mathcal{O}\big(L^3\log L\log(1/\varepsilon)\big)$ operations for the whole step.

**4.2. Second step.** We now replace the Wigner-$d$ functions $\tilde{d}_\ell^{\,m',n'}$ with the Chebyshev polynomials of first kind $T_\ell$ to obtain

$$f(\alpha,\beta,\gamma) = \frac{1}{2\pi}\sum_{m=-L}^{L}\sum_{n=-L}^{L}\mathrm{e}^{-\mathrm{i}m\alpha}\,\mathrm{e}^{-\mathrm{i}n\gamma}\sum_{\ell=0}^{L-\chi}\tilde{f}_\ell^{\,m,n}(1-x^2)^{\chi/2}T_\ell(x).$$

We may write this as

$$\tilde{\mathbf{f}}^{\,m,n} = \mathbf{B}^{m,n}\,\bar{\mathbf{f}}^{\,m,n}, \qquad m,n=-L,\ldots,L. \tag{4.2}$$

To obtain the representation of the matrix $\mathbf{B}^{m,n}$, we need to use a number of identities. With the defining equation (2.1) and formulae (22.7.15), (22.7.15) found in [1, p. 782], we can expand the functions $\tilde{d}_\ell^{\,m',n'}$ for all remaining cases for $m'$ and $n'$ into Jacobi polynomials as follows,

$$\begin{aligned} &\tilde{d}_\ell^{0,0} = b_\ell^{0,0}P_\ell^{(0,0)}, \\ &\tilde{d}_\ell^{0,1} = \big(1-x^2\big)^{1/2}b_\ell^{0,1}P_{\ell-1}^{(1,1)}, \qquad \tilde{d}_\ell^{0,2} = b_{\ell,\ell}^{0,2}P_\ell^{(1,1)} + b_{\ell-2,\ell}^{0,2}P_{\ell-2}^{(1,1)}, \\ &\tilde{d}_\ell^{1,1} = b_{\ell,\ell}^{1,1}P_\ell^{(0,1)} + b_{\ell-1,\ell}^{1,1}P_{\ell-1}^{(0,1)}, \quad \tilde{d}_\ell^{1,-1} = b_{\ell,\ell}^{1,-1}P_\ell^{(1,0)} + b_{\ell-1,\ell}^{1,-1}P_{\ell-1}^{(1,0)}, \end{aligned} \tag{4.3}$$

where

$$b_\ell^{0,0} = \left(\frac{2\ell+1}{2}\right)^{1/2}, \qquad b_\ell^{0,1} = -\frac{1}{2}\left(\frac{(2\ell+1)(\ell+1)}{2\ell}\right)^{1/2},$$

$$b_{k,\ell}^{1,1} = \big(4\ell+2\big)^{-1/2}\begin{cases}\ell, & \text{if } k=\ell,\\ \ell+1, & \text{if } k=\ell-1,\end{cases} \qquad b_{k,\ell}^{1,-1} = (-1)^{k-l} b_{k,\ell}^{1,1},$$

$$b_{k,\ell}^{0,2} = \frac{1}{2}\left(\frac{(\ell+1)(\ell+2)}{(\ell-1)\ell(4\ell+2)}\right)^{1/2}\begin{cases}-\dfrac{\ell(\ell-1)}{(\ell+1)}, & \text{if } k=\ell,\\ \ell, & \text{if } k=\ell-2.\end{cases}$$

Next, we want to replace the Jacobi polynomials $P_\ell^{(1,0)}$, $P_\ell^{(0,1)}$, and $P_\ell^{(1,1)}$ with the Legendre polynomials $P_\ell^{(0,0)}$. For this, we use the formulae

$$P_\ell^{(0,1)} = \sum_{j=0}^{\ell} c_{\ell,j}^{0,1} P_j^{(0,0)}, \qquad P_\ell^{(1,0)} = \sum_{j=0}^{\ell} c_{\ell,j}^{1,0} P_j^{(0,0)}, \qquad P_\ell^{(1,1)} = \sum_{j=0}^{\ell} c_{j,\ell}^{1,1} P_j^{(0,0)},$$

with

$$c_{\ell,j}^{0,1} = (-1)^{j+\ell}\frac{2j+1}{\ell+1}, \qquad c_{\ell,j}^{1,0} = \frac{2j+1}{\ell+1}, \qquad c_{j,\ell}^{1,1} = \begin{cases}0, & \text{if } j+\ell \text{ odd},\\ \dfrac{2(2j+1)}{\ell+2}, & \text{else.}\end{cases}$$

Finally, the Legendre polynomials $P_\ell^{(0,0)}$ may be replaced with the Chebyshev polynomials of first kind via the formula

$$P_\ell^{(0,0)} = \sum_{j=0}^{\ell} d_{j,\ell}^{0,0} T_j, \qquad d_{j,\ell}^{0,0} = \frac{2-\delta_{j,0}}{\pi}\frac{\Gamma\left(\frac{\ell-j}{2}+\frac{1}{2}\right)}{\Gamma\left(\frac{\ell-j}{2}+1\right)}\frac{\Gamma\left(\frac{\ell+j}{2}+\frac{1}{2}\right)}{\Gamma\left(\frac{\ell+j}{2}+1\right)}.$$

We may now define the matrices

$$\begin{aligned}&\mathbf{B}^{0,0}\in\mathbb{R}^{(L+1)\times(L+1)}, && \mathbf{B}^{0,1}\in\mathbb{R}^{L\times L}, && \mathbf{B}^{0,2}\in\mathbb{R}^{(L+1)\times(L-1)},\\ &\mathbf{B}^{1,1}\in\mathbb{R}^{(L+1)\times L}, && \mathbf{B}^{1,-1}\in\mathbb{R}^{(L+1)\times L}, && \mathbf{C}^{0,1}\in\mathbb{R}^{(L+1)\times(L+1)},\\ &\mathbf{C}^{1,0}\in\mathbb{R}^{(L+1)\times(L+1)}, && \mathbf{C}^{1,1}\in\mathbb{R}^{(L+1)\times(L+1)}, && \mathbf{D}^{0,0}\in\mathbb{R}^{(L+1)\times(L+1)},\end{aligned}$$

that have the form

$$\mathbf{B}^{0,0} = \begin{pmatrix} b_0^{0,0} & & \\ & b_1^{0,0} & \\ & & \ddots \end{pmatrix}, \quad \mathbf{B}^{0,1} = \begin{pmatrix} b_1^{0,1} & & \\ & b_2^{0,1} & \\ & & \ddots \end{pmatrix}, \quad \mathbf{B}^{0,2} = \begin{pmatrix} b_{0,2}^{0,2} & & \\ 0 & b_{1,3}^{0,2} & \\ b_{2,2}^{0,2} & 0 & \ddots \\ & b_{3,3}^{0,2} & \ddots \\ & & \ddots \end{pmatrix},$$

$$\mathbf{B}^{1,1} = \begin{pmatrix} b_{0,1}^{1,1} & & \\ b_{1,1}^{1,1} & b_{1,2}^{1,1} & \\ & b_{2,2}^{1,1} & \ddots \\ & & \ddots \end{pmatrix}, \mathbf{B}^{1,-1} = \begin{pmatrix} b_{0,1}^{1,-1} & & \\ b_{1,1}^{1,-1} & b_{1,2}^{1,-1} & \\ & b_{2,2}^{1,-1} & \ddots \\ & & \ddots \end{pmatrix}, \mathbf{C}^{0,1} = \begin{pmatrix} e_{0,0}^{0,1} & \cdots & e_{0,L}^{0,1} \\ & \ddots & \vdots \\ & & e_{L,L}^{0,1} \end{pmatrix},$$

$$\mathbf{C}^{1,0} = \begin{pmatrix} c_{0,0}^{1,0} & \cdots & c_{0,L}^{1,0} \\ & \ddots & \vdots \\ & & c_{L,L}^{1,0} \end{pmatrix}, \mathbf{C}^{1,1} = \begin{pmatrix} c_{0,0}^{1,1} & \cdots & c_{0,L}^{1,1} \\ & \ddots & \vdots \\ & & c_{L,L}^{1,1} \end{pmatrix}, \mathbf{D}^{0,0} = \begin{pmatrix} c_{0,0}^{0,0} & \cdots & c_{0,L}^{0,0} \\ & \ddots & \vdots \\ & & c_{L,L}^{0,0} \end{pmatrix},$$

which we combine in the respective cases for $m'$ and $n'$ to

$$\mathbf{C}^{m,n} = \begin{cases} \mathbf{D}^{0,0}\,\mathbf{B}^{0,0}, & \text{if } m' = n' = 0, \\ \mathbf{D}^{0,0}\,\mathbf{C}^{1,1}\,\mathbf{B}^{0,1}, & \text{if } m' = 0,\ n' = 1, \\ \mathbf{D}^{0,0}\,\mathbf{C}^{1,1}\,\mathbf{B}^{0,2}, & \text{if } m' = 0,\ n' = 2, \\ \mathbf{D}^{0,0}\,\mathbf{C}^{0,1}\,\mathbf{B}^{1,1}, & \text{if } m' = 1,\ n' = 1, \\ \mathbf{D}^{0,0}\,\mathbf{C}^{1,0}\,\mathbf{B}^{1,-1}, & \text{if } m' = 1,\ n' = -1. \end{cases}$$

The diagonal and bidiagonal matrices $\mathbf{B}^{0,0}$, $\mathbf{B}^{0,1}$, $\mathbf{B}^{0,2}$, $\mathbf{B}^{1,1}$, and $\mathbf{B}^{1,-1}$ clearly may be applied to a vector with $\mathcal{O}(L)$ arithmetic operations. The same holds for the matrices $\mathbf{C}^{0,1}$, $\mathbf{C}^{1,0}$, and $\mathbf{C}^{1,1}$ owing to the separability of the entries with respect to the indices $\ell$ and $j$; see [32, p. 98]. For the matrix $\mathbf{D}^{0,0}$ there is an approximate FMM-based algorithm that applies the matrix to any vector with $\mathcal{O}\big(L\log(1/\varepsilon)\big)$ arithmetic operations; see [2, 11]. In total, we can compute this step with $\mathcal{O}\big(L^3\log(1/\varepsilon)\big)$ arithmetic operations.

**4.3. Third step.** We have arrived at the form

$$f(\alpha,\beta,\gamma) = \frac{1}{2\pi}\sum_{m=-L}^{L}\sum_{n=-L}^{L} \mathrm{e}^{-\mathrm{i}m\alpha}\,\mathrm{e}^{-\mathrm{i}n\gamma}\sum_{\ell=0}^{L-\chi}\tilde{f}_\ell^{\,m,n}(1-x^2)^{\chi/2}T_\ell(x).$$

Our goal is to rewrite this as

$$f(\alpha,\beta,\gamma) = \sum_{m=-L}^{L}\sum_{n=-L}^{L} \mathrm{e}^{-\mathrm{i}m\alpha}\,\mathrm{e}^{-\mathrm{i}n\gamma}\sum_{\ell=-L}^{L}\hat{g}_\ell^{\,m,n}\mathrm{e}^{-\mathrm{i}\ell\beta}. \tag{4.4}$$

To this end, we note that

$$\frac{1}{2\pi}T_\ell(x) = \frac{1}{2\pi}T_\ell(\cos\beta)\cos(\ell\beta) = \frac{1}{4\pi}\big(\mathrm{e}^{\mathrm{i}\ell\beta} + \mathrm{e}^{-\mathrm{i}\ell\beta}\big)$$

and

$$\begin{aligned}\frac{1}{2\pi}(1-x^2)^{1/2}\,T_\ell(x) &= \frac{1}{2\pi}\sin(\beta)\;T_\ell(\cos\beta) \\ &= -\frac{\mathrm{i}}{8\pi}\left(\mathrm{e}^{\mathrm{i}(\ell+1)\beta} - \mathrm{e}^{\mathrm{i}(\ell-1)\beta} + \mathrm{e}^{-\mathrm{i}(\ell-1)\beta} - \mathrm{e}^{-\mathrm{i}(\ell+1)\beta}\right).\end{aligned}$$

Consequently, we define the matrices $\mathbf{E}^{m,n} \in \mathbb{C}^{(2L+1)\times(L-\chi+1)}$ by

$$\mathbf{E}^{m,n} \begin{cases} \mathbf{E}^0, & \text{if } m+n \text{ even}, \\ \mathbf{E}^1, & \text{if } m+n \text{ odd}, \end{cases} \qquad \mathbf{E}^0 = \frac{1}{4\pi} \begin{pmatrix} & & 2 & & \\ & 1 & & 1 & \\ \iddots & & & & \ddots \end{pmatrix}^{\mathrm{T}},$$

$$\mathbf{E}^1 = \frac{\mathrm{i}}{8\pi} \begin{pmatrix} & & & -2 & 0 & 2 & & & \\ & & -1 & 0 & 0 & 0 & 1 & & \\ & -1 & 0 & 1 & & -1 & 0 & 1 & \\ \iddots & \iddots & \iddots & & & & \ddots & \ddots & \ddots \end{pmatrix}^{\mathrm{T}}.$$

Then this step can be represented in matrix-vector notation as

$$\hat{\mathbf{g}}^{m,n} = \mathbf{E}^{m,n}\, \tilde{\mathbf{f}}^{\,m,n}, \qquad m,n = -L,\dots,L. \tag{4.5}$$

Clearly, we may compute this step with $\mathcal{O}(L^3)$ operations.

**4.4. Fourth step.** We invoke the NFFT algorithm on the three-dimensional Fourier sum (4.4). The cost is identical to that of a three-dimensional FFT plus a linear term for the $Q$ nodes, that is, $\mathcal{O}\big(L^3 \log L + \log(1/\varepsilon)^3 Q\big)$. We represent this step as the matrix-vector product

$$\mathbf{f} = \mathbf{F}\,\hat{\mathbf{g}}.$$

**4.5. Summary.** Our realization of the nonequispaced discrete SO(3) Fourier transform algorithm may be summarized as the calculation of the product

$$\mathbf{f} = \mathbf{D}\,\hat{\mathbf{f}} = \mathbf{F} \cdot \mathbf{E} \cdot \mathbf{B} \cdot \mathbf{A} \cdot \hat{\mathbf{f}},$$

where we collect all matrices $\mathbf{A}^{m,n}$, $\mathbf{B}^{m,n}$, and $\mathbf{E}^{m,n}$ for $m,n = -L,\dots,L$ in the block diagonal matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{E}$. Note that these do not depend on the nodes $\mathcal{X}$, but only on the expansion degree $L$. With the same algorithms, we may also compute the adjoint product $\mathbf{D}^{\mathrm{H}}\mathbf{f} = \mathbf{A}^{\mathrm{T}} \cdot \mathbf{B}^{\mathrm{T}} \cdot \mathbf{E}^{\mathrm{H}} \cdot \mathbf{F}^{\mathrm{H}} \cdot \mathbf{f}$ at identical cost.

In total, we need $\mathcal{O}\big(L^3 \log L \log(1/\varepsilon) + \log(1/\varepsilon)^3 Q\big)$ arithmetic operations. This compares favorably with $\mathcal{O}(L^4)$ for the algorithm from [15] (which requires particular nodes) and $\mathcal{O}\big(L^3 \log^2 L + \log(1/\varepsilon)^3 Q\big)$ for the algorithm from [22] (which seems to have higher cost in practice).

**5. Numerical Tests.** We have described an efficient mechanism for computing discrete SO(3) Fourier transformations that is based on an efficient method to transform an expansion in Wigner-$d$ functions of arbitrary orders $m$ and $n$ into one with well-defined low orders $m'$ and $n'$.

The main goal of this section is to test this algorithm under realistic conditions and to weigh the results against those of others. Not all of the tested algorithms are similarly efficient, but can serve as reference points with respect to numerical stability and performance. For the comparison, we selected a total of four algorithms that are described below.

Ranking the available algorithms amounts to more than just comparing timing and error figures. The different algorithms realize slightly different transformations, but can essentially serve the same purpose as part of a whole SO(3) Fourier transform algorithm. They are also distinguished by other properties such as the amount of work for precomputation or advantages offered for certain corner cases. Therefore, we also describe the specific advantages and disadvantages of each method.

**5.1. Test Environment.** All algorithms were implemented in double precision C on an Intel Core 2 Duo 2.16 GHz MacBook Air with 4GB RAM running Mac OS X 10.6.7. We have used Apple's `llvm-gcc-4.2` compiler with the optimization flags `-O3 -fomit-frame-pointer -malign-double -ffast-math -mtune=core2 -march=core2`. Time spans were measured accurately using the CPU cycle counters interface from the popular FFTW library [7]. In addition to our own code, we used the software libraries FFTW 3.2.1 [7], NFFT 3.1.4 [13], MPACK [21], and MPFR [6] that we compiled with the same settings.

**5.2. Test Description.** Since not all of the tested algorithms calculate exactly the same transformation, they cannot be compared directly next to each other. Even if that were the case, it would still be a formidable task to calculate the exact and correct output for each given input to be able to assess the accuracy. Therefore, we resort to a more feasible method that, at the same time, is also closer to the actual application in mind and thus allows us to define a standardized way to compare all algorithms. The test consists in evaluating a linear combination of Wigner-$d$ functions for a given pair of orders $m$, $n$ and with uniformly random expansion coefficients, i.e.,

$$f(x) = \sum_{\ell=L^*}^{L} \hat{f}_\ell^{m,n} \tilde{d}_\ell^{m,n}(x),$$

where $x$ sweeps across $Q = 1000$ uniformly random nodes in the interval $[-1, 1]$.

First, we assure ourselves that computation of the correct output (within the bounds of double precision) is feasible. From [26], we know that the Clenshaw algorithm satisfies the error bound

$$\big|f(x) - \tilde{f}(x)\big| \leq \epsilon \sum_{\ell=L^*}^{L} \big|\hat{f}_\ell^{m,n}\big|\big|\tilde{d}_\ell^{m,n}(x)\big|,$$

where $f(x)$ is the exact value and $\tilde{f}(x)$ is the computed value. Furthermore, [18] asserts the bound

$$\max_{x\in[-1,1]} \big|\tilde{d}_\ell^{0,n}(x)\big| < \frac{2^{5/4}}{\pi^{3/4}} \frac{\sqrt{2\ell+1}}{2n^{1/4}}.$$

While we have no general proof at hand, our numerical results strongly indicate that a bound for Wigner-$d$ functions of arbitrary order is

$$\max_{x\in[-1,1]} \big|\tilde{d}_\ell^{m,n}(x)\big| < \frac{11}{20}\sqrt{2\ell+1}$$

which implies the error bound

$$\big|f(x) - \tilde{f}(x)\big| \leq \epsilon \frac{11}{20} \sum_{\ell=L^*}^{L} \sqrt{2\ell+1}\big|\hat{f}_\ell^{m,n}\big|. \tag{5.1}$$

By using the Clenshaw algorithm together with the MPFR library in quadruple precision, i.e., $\epsilon \approx 10^{-31}$, we thus make sure that we can compute reference values that can be considered exact in double precision. Also, the MPFR library offers a large exponent range that ensures that intermediate calculated numbers remain representable to full precision.

The bound (5.1) also motivates the definition of the error

$$E(f) := \frac{20}{11} \frac{\max_{q=1,\dots,Q}\{f(x_q) - \tilde{f}(x_q)\}}{\sum_{\ell=L^*}^{L} \sqrt{2\ell+1}|\hat{f}_\ell^{m,n}|} \tag{5.2}$$

to assess the accuracy achieved by any of the tested algorithms in double precision.

For each of the algorithms under consideration, our test procedure starts with a given expansion in Wigner-$d$ functions with random expansion coefficients. Since all methods realize a change of basis to replace the given Wigner-$d$ expansion with a new one in a different set of functions, we proceed by applying these transformations. Finally, using the Clenshaw algorithm for the new set of functions, we evaluate the resulting expansion at the desired random nodes and compute the error $E$ as in (5.2). For a single algorithm, we repeated this procedure for different sizes $L = 2, 4, 8, 16, 32, \dots$ and all different combinations of orders $m$, $n$ taken from the set

$$\big\{(m,n) : m = \lfloor (i/4)L \rfloor,\, n = \lfloor (j/4)L \rfloor,\, i = 0, 1, \dots, 4,\, j = 0, 1, \dots, i\big\}.$$

**5.3. Tested Algorithms.** For the comparison, we selected a total of four algorithms described in the following.

**5.3.1. DPT.** The discrete polynomial transform (DPT) can be used to transform an expansion in Wigner-$d$ functions into an expansion in Chebyshev polynomials of first kind. Depending on the orders $m$ and $n$, the Wigner-$d$ functions may contain a common fractional power of $(1 \pm x)^{1/2}$ or $(1 - x^2)^{1/2}$ that cannot be represented by the target polynomials. Therefore, this factor is formally pulled out and applied at a later stage.

The method consists in applying the Clenshaw algorithm to evaluate the source expansion at Chebyshev nodes in the interval $[-1, 1]$. From there, the expansion coefficients with respect to Chebyshev polynomials can be computed with a discrete cosine transform.

The method requires $\mathcal{O}(L^2)$ arithmetic operations for the actual transformation and $\mathcal{O}(L)$ for a moderate precomputation stage. The Clenshaw algorithm is provably backward stable (see [26]) and thus, while not very efficient, interesting as a reference point with respect to numerical stability. It must be noted though that backward stability does not prevent calculations from leaving the range of floating point numbers representable in the numerical format of choice — visible in the actual numerical results below.

**5.3.2. FPT.** The fast polynomials transform (FPT) is a more efficient variant of the DPT. The principal idea is that consecutive steps in the Clenshaw algorithm may be combined to a single one that can be realized by a fast polynomial multiplication in the Chebyshev basis. The acceleration is achieved through fast discrete cosine transforms. By using a tree-like scheme of arranging the successively larger polynomial multiplications, the resulting algorithm achieves an arithmetic cost of $\mathcal{O}(L \log^2 L)$. However, numerical evidence has shown that this method is often unstable even for moderate sizes and thus requires a stabilization technique. This affords additional calculations that likely increase the actual cost. Therefore, $\mathcal{O}(L \log^2 L)$ must be only considered a lower bound.

Besides these issues, the FPT algorithm inherits the same numerical problem from the Clenshaw algorithm, that is that the numerical range of numbers may not be large enough to carry out the computation for any expansion in Wigner-$d$ functions exceeding a certain degree $L$. During our calculations, we observed exactly this behavior

which renders the FPT unusable, save for some special cases, for transformations with $L$ above a defined bound.

**5.3.3. LAPACK.** Our main result, as presented in Section 3 is that any expansion in Wigner-$d$ functions can be transformed into an equivalent one from a small set of well-defined low orders $m'$ and $n'$. To this end, we have constructed a semiseparable matrix $\mathbf{G}$ whose eigenvectors to certain known eigenvalues form the matrix that represents the desired linear transformation.

While we have also described how the structure of the matrix can be used to accelerate the computation, there is also the option of applying a standard algorithm (e.g. provided by the LAPACK software library) to compute the full eigendecomposition of $\mathbf{G}$ explicitly. Then, the respective transformation matrix may be applied directly to compute the desired result. We chose this algorithm as it allows for an insightful comparison with the Clenshaw algorithm and the new and asymptotically faster divide-and-conquer method.

Of course, in general, pre-computation as well as the actual transformation have a relatively high arithmetic cost, namely $\mathcal{O}(L^3)$ and $\mathcal{O}(L^2)$. On the other hand, standard routines to compute the eigendecomposition of $\mathbf{G}$ have proven robust and the direct application of the transformation matrix eliminates the approximate nature that characterizes the divide-and-conquer method. For special cases like $L^* = L$, the transformation (but not precomputation) is actually realized at cost $\mathcal{O}(L)$ since the transformation matrix collapses to a single column.

Generally, the LAPACK method provides numerical stability close to the optimum, but the rapidly growing precomputation and transformation time, as well as the substantial memory requirements, make this method suitable only for moderately sized problems.

**5.3.4. DAQ.** This is our new divide-and-conquer method that computes the same transformation as the LAPACK method, but differs from it by exploiting the structure offered by the matrix $\mathbf{G}$ to obtain a more efficient way of applying the transformation matrix with the help of the FMM. In Section 3.2 we have described how the structure of $\mathbf{G}$ may be exploited for an approximate $\mathcal{O}\big(L \log L \log(1/\epsilon)\big)$ implementation that needs $\mathcal{O}\big(L^2\big)$ operations for pre-computation[1]. Due to the nature of the algorithm, the work needed for a single transformation does not vary significantly with the choice of the orders $m$ and $n$.

An interesting aspect is that the FMM is an approximate algorithm that can nevertheless be tuned to achieve almost full accuracy. Our tests convincingly show that this is the case for the desired transformation. On the negative side, the divide-and-conquer algorithm is relatively complicated to implement and has many places where an all too straightforward implementation can quickly jeopardize the numerical stability of the whole method. In our implementation, we actually found it necessary to use the MPFR library with almost quadruple precision for pre-computation to achieve satisfactory results. A negative side effect of increasing the precision there is that this stage now also takes substantially longer than in double precision (about a factor of 100). On the positive side, the extra precision is not needed during the actual transformation which we thus implemented in double precision.

**5.4. Discussion.** Figures 5.1 to 5.4 show error ($E(f)$) and time ($t$) figures for the four tested algorithms. The figures produce an overlay of the curves corresponding

[1] An acceleration of the pre-computation stage to $\mathcal{O}\big(L \log L \log(1/\epsilon)\big)$ is possible with the FMM. We have not implemented this.

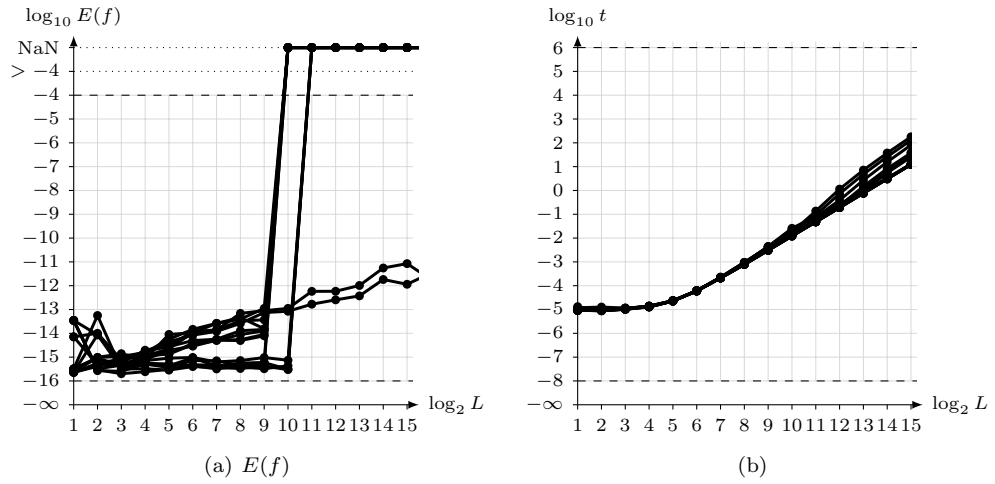


FIG. 5.1. *DPT: Error $E(f)$ and transformation time $t$ in seconds for different sizes of $L$.*

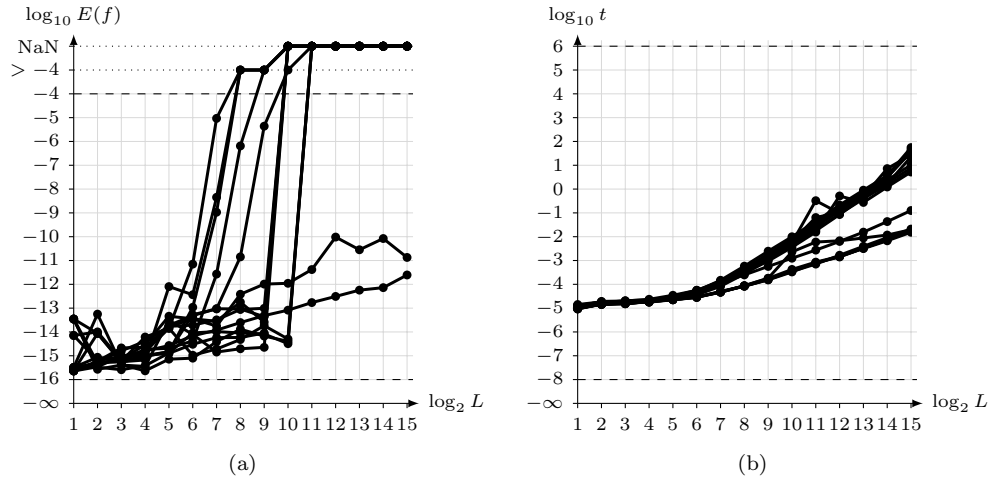


FIG. 5.2. *FPT: Error $E(f)$ and transformation time $t$ in seconds for different sizes of $L$.*

to the different combinations of orders $m$ and $n$.

The DPT algorithm (Figure 5.1), anyway asymptotically slower than the more efficient methods, produces NaN values for most scenarios with transform size $L > 1024$. The effect can be explained with the nature of the Clenshaw algorithm that, applied to Wigner-$d$ functions, leads to intermediate numerical values that leave the range representable in double precision whenever $L$ is just large enough — save for corner cases like $m = n = 0$ and $m = n = L$. In conclusion this algorithm is neither very efficient, numerically stable or otherwise suitable as a reference point.

The FPT algorithm (Figure 5.2) has a behavior similar to the DPT algorithm, but with a few differences. First, it is interesting to note that a substantial number

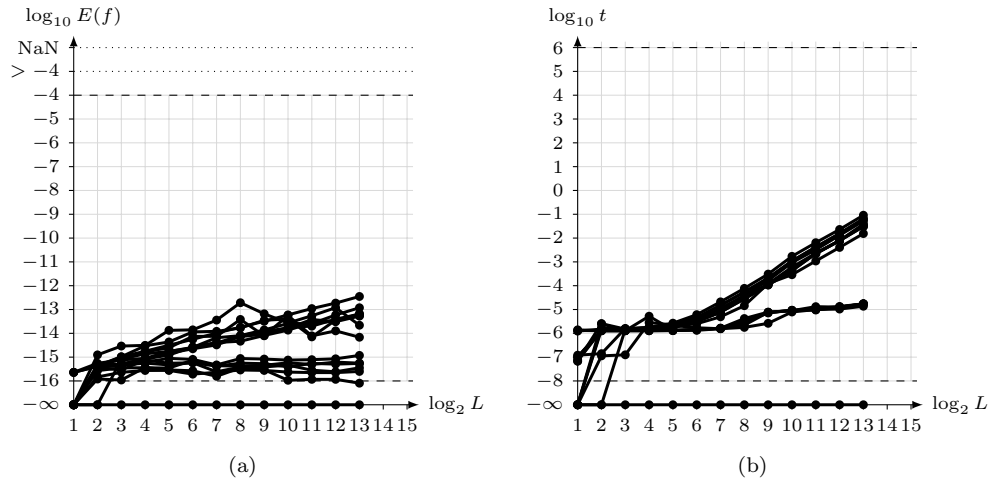


FIG. 5.3. *LAPACK: Error $E(f)$ and transformation time $t$ in seconds for different sizes of $L$.*

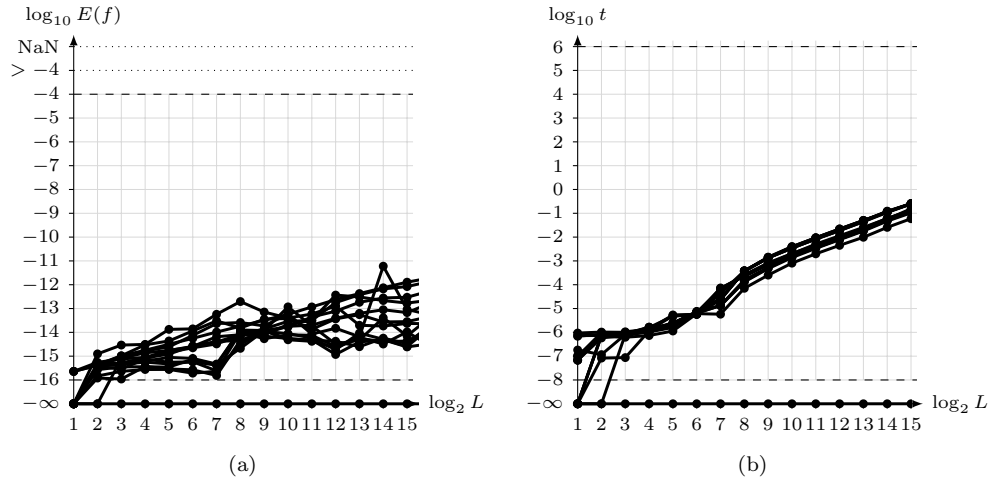


FIG. 5.4. *DAQ: Error $E(f)$ and transformation time $t$ in seconds for different sizes of $L$.*

of the tested scenarios shows no substantial asymptotical advantage in speed over the DPT algorithm — while some others do. The only explanation is that the employed stabilization mechanism implies too many extra computations to be able to attain the $\mathcal{O}(L \log^2 L)$ bound anymore. Instead, the observed cost seems to be close to $\mathcal{O}(L^2)$ for the problematic cases. Second, the FPT algorithm inherits the numerical issues causes by the Clenshaw algorithm and shows an even worse behavior since a degradation in accuracy can already be observed for $L < 512$. Similar to the DPT, this algorithm must be considered problematic for its numerical issues and the undesirable numerical cost.

The LAPACK algorithm (Figure 5.3) justifies its position as reference algorithm

in our series of tests. It shows the expected relatively high arithmetic cost $\mathcal{O}(L^2)$ (for a few corner cases: $\mathcal{O}(L)$) but achieves satisfactory numerical accuracy for our error measure over the tested scenarios.

Finally, the DAQ algorithm (Figure 5.4) shows a very similar profile with respect to numerical accuracy when compared to the LAPACK algorithm while achieving the claimed asymptotic cost of $\mathcal{O}\big(L \log L \log(1/\varepsilon)\big)$. This makes this method the algorithm of choice among the tested methods, especially for any transform size $L \geq 1024$.

In conclusion, we have developed, implemented, and tested a new efficient algorithm for FFT-like transforms involving Wigner-$d$ functions that marks a substantial improvement over methods known so far.

**References.**


[1] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions.* National Bureau of Standards, Washington, DC, USA, 1972.

[2] B. K. Alpert and V. Rokhlin. A fast algorithm for the evaluation of Legendre expansions. *SIAM J. Sci. Stat. Comput.*, 12:158 – 179, 1991.

[3] S. Chandrasekaran and M. Gu. A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices. *Numer. Math.*, 96:723 – 731, 2004.

[4] G. S. Chirikjian and A. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis: with Emphasis on Rotation and Motion Groups.* CRC Press, Boca Raton, FL, USA, 2001.

[5] J. R. Driscoll and D. Healy. Computing Fourier transforms and convolutions on the 2–sphere. *Adv. in Appl. Math.*, 15:202 – 250, 1994.

[6] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. Mpfr: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33, June 2007.

[7] M. Frigo and S. G. Johnson. FFTW, C subroutine library. `http://www.fftw.org`.

[8] I. Gradstein and I. Ryshik. *Tables of Series, Products, and Integrals*, volume 2. Verlag Harri Deutsch, Thun, Frankfurt am Main, Germany, 1981.

[9] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325 – 348, 1987.

[10] R. Hielscher, J. Prestin, and A. Vollrath. Fast summation of functions on SO(3). *Math. Geosci.*, 42:773–794, 2010.

[11] J. Keiner. Computing with Expansions in Gegenbauer Polynomials. *SIAM J. Sci. Comput.*, 31:2151 – 2171, 2009.

[12] J. Keiner. *Fast Polynomials Transforms.* Logos Verlag, Berlin, Germany, 2011.

[13] J. Keiner, S. Kunis, and D. Potts. NFFT 3.0, C subroutine library. `http://www.tu-chemnitz.de/~potts/nfft`.

[14] J. Keiner and D. Potts. Fast evaluation of quadrature formulae on the sphere. *Math. Comput.*, 77:397 – 419, 2008.

[15] P. J. Kostelec and D. N. Rockmore. FFTs on the rotation group. *J. Fourier Anal. Appl.*, 14:145 – 179, 2008.

[16] J. A. Kovacs, P. Chacón, Y. Cong, E. Metwally, and W. Wriggers. Fast rotational matching of rigid bodies by fast Fourier transform acceleration of five degrees of freedom. *Acta Crystallogr. Sect. D*, 59(8):1371 – 1376, 2003.

[17] S. Kunis and D. Potts. Fast spherical Fourier algorithms. *J. Comput. Appl. Math.*, 161:75 – 98, 2003.

[18] G. Lohöfer. Inequalities for the associated legendre functions. *Journal of Approximation Theory*, 95(2):178–193, 1998.

[19] D. K. Maslen and D. N. Rockmore. Generalized FFTs - A Survey of Some Recent Results. In L. Finkelstein and W. Kantor, editors, *DIMACS Workshop in Groups and Computation*, volume 28, pages 183 – 238, 1995.

[20] M. J. Mohlenkamp. A fast transform for spherical harmonics. *J. Fourier Anal. Appl.*, 5:159 – 184, 1999.

[21] M. Nakata. The MPACK (MBLAS/MLAPACK); a multiple precision arithmetic version of BLAS and LAPACK. `http://mplapack.sourceforge.net`, 2010.

[22] D. Potts, J. Prestin, and A. Vollrath. A fast algorithm for nonequispaced Fourier transforms on the rotation group. *Numer. Algorithms*, 52:355 – 384, 2009.

[23] D. Potts, G. Steidl, and M. Tasche. Fast Fourier transforms for nonequispaced data: A tutorial. In J. J. Benedetto and P. J. S. G. Ferreira, editors, *Modern Sampling Theory: Mathematics and Applications*, pages 247 – 270, Boston, MA, USA, 2001. Birkhäuser.

[24] D. W. Ritchie and G. J. L. Kemp. Protein docking using spherical polar Fourier correlations. *PROTEINS: Struct. Funct. Genet.*, 39:178 – 194, 2000.

[25] V. Rokhlin and M. Tygert. Fast algorithms for spherical harmonic Expansions. *SIAM J. Sci. Comput.*, 27:1903 – 1928, 2006.

[26] A. Smoktunowicz. Backward stability of Clenshaw's algorithm. *BIT Numerical Mathematics*, 42:600–610, 2002.

[27] B. Stevensson and M. Edén. Interpolation by fast wigner transform for rapid calculations of magnetic resonance spectra from powders. *J Chem Phys*, 134:124104, 2011.

[28] R. Suda and M. Takami. A fast spherical harmonics transform algorithm. *Math. Comput.*, 71:703 – 715, 2002.

[29] G. Szegő. *Orthogonal Polynomials*. Amer. Math. Soc., Providence, RI, USA, 4th edition, 1975.

[30] M. Tygert. Fast algorithms for spherical harmonic expansions II. *J. Comput. Phys.*, 227:4260 – 4279, 2008.

[31] R. Vandebril, M. V. Barel, G. Golub, and N. Mastronardi. A bibliography on semiseparable matrices. *Calcolo*, 42:249 – 270, 2005.

[32] R. Vandebril, M. van Barel, and N. Mastronardi. *Matrix Computations and Semiseparable Matrices Volume 1 – Linear Systems*. The Johns Hopkins University Press, Baltimore, MD, USA, 2008.

[33] D. Varshalovich, A. Moskalev, and V. Khersonskii. *Quantum Theory of Angular Momentum*. World Scientific Publishing, Singapore, 1988.

[34] N. Vilenkin. *Special Functions and the Theory of Group Representations*. Amer. Math. Soc., Providence, RI, USA, 1968.