A Fast Algorithm for Nonequispaced Fourier Transforms on the Rotation Group

Daniel Potts [*] Jürgen Prestin [†] Antje Vollrath[†, ‡]

In this paper we present algorithms to calculate fast Fourier transforms and its adjoint on the rotation group $SO(3)$ for arbitrary sampling sets. It is based on the fast Fourier transform for nonequispaced nodes on the three-dimensional torus. This algorithm evaluates the $SO(3)$ Fourier transform of B-bandlimited functions at M arbitrary input nodes in $\mathcal{O}(M + B^4)$ or even $\mathcal{O}(M + B^3 \log^2 B)$ flops instead of $\mathcal{O}(MB^3)$. Numerical results will be presented establishing the algorithm's numerical stability and time requirements.

2000 AMS Subject Classification: 65T99, 33C55, 42C10, 65T50
Keywords: rotation group, spherical Fourier transform, generalized spherical harmonics, Wigner-D functions, Wigner-d functions, fast discrete transforms, fast Fourier transform at nonequispaced nodes

1 Introduction

During the past years, several different techniques have been proposed for computing Fourier transforms on the rotation group $SO(3)$ motivated by a variety of applications, like protein-protein-docking [4] or texture analysis [3, 28]. Many of these algorithms are based on discrete Fourier transforms on the two-dimensional sphere and a generalization thereof.
In this paper the algorithm to compute a $SO(3)$ Fourier transform is based on evaluating the Wigner-D functions D_l^{mn}, which yield an orthogonal basis of $L^2(SO(3))$. Using these functions we expand B-bandlimited functions $f \in L^2(SO(3))$ into the sum

$$f = \sum_{l=0}^{B} \sum_{m=-l}^{l} \sum_{n=-l}^{l} \hat{f}_l^{mn} D_l^{mn}. \tag{1.1}$$

[*]Department of Mathematics, Chemnitz University of Technology, Chemnitz, Germany
[†]Institut für Mathematik, Universität zu Lübeck
[‡]Lübeck Graduate School for Computing in Medicine and Life Sciences

We will present an algorithm for the efficient and accurate calculation of such B-band-limited functions $f \in L^2(SO(3))$ at arbitrary samples $\boldsymbol{g}_q \in SO(3)$, for $q = 0, \ldots, M-1$ outlining that the above sum is the discrete Fourier transform of a function defined on the rotation group. Moreover we will also present the adjoint transform to the sum above pointing out that it has indeed the potential to efficiently compute inverse $SO(3)$ Fourier transforms, as well.
By splitting up the Wigner-D functions $D_l^{mn}(\boldsymbol{g})$ where $\boldsymbol{g} = \boldsymbol{g}(\alpha, \beta, \gamma) \in SO(3)$ in three components depending on one Euler angle (see Definition 2.1) each as

$$D_l^{m,n}(\boldsymbol{g}) = \mathrm{e}^{-\mathrm{i}m\alpha}\mathrm{e}^{-\mathrm{i}n\gamma} d_l^{m,n}(\cos\beta)$$

we use the nonequispaced fast Fourier transform (NFFT) algorithm (see e.g., [8, 2, 23] and [14] for the implementation of both algorithms). Based on the fast polynomial transform, (cf. e.g., [7, 21]) we develop an algorithm for transforming sums of Wigner-d functions $d_l^{m,n}$ quickly into sums of Chebyshev polynomials. The actual transform will then become a trivariate nonequispaced fast Fourier transform on the torus.
Putting together our efforts the $SO(3)$ Fourier transform will be sped up from $\mathcal{O}(MB^3)$ flops to $\mathcal{O}(M + B^4)$ and under certain circumstances even to $\mathcal{O}(M + B^3 \log^2 B)$ flops, with B being the bandwidth and M the number of input nodes which is independent from the bandwidth. Nevertheless, a typical size of M would be $\mathcal{O}(B^3)$.
Other approaches to compute function values at specially structured, i.e., equispaced grids on $SO(3)$ using $SO(3)$ Fourier transforms can be found in [16] and [26]. Although we used the same separation of variables based technique as the authors of [16] they split their transform differently into a bivariate FFT and a direct recursive evaluation of Wigner-d functions. In contrast [26] describes how to expand the Wigner-d functions directly into a Fourier sum and thus evaluating a quadruple sum by means of an FFT. Both works describe $\mathcal{O}(B^4)$ algorithms while the authors of [16] point out that their algorithm has the potential to be a $\mathcal{O}(B^3 \log^2 B)$, as well. They mention problems in stability and a trade-off between accuracy and time requirements that occur using the fast $\mathcal{O}(B^3 \log^2 B)$ approach as the reason to decide for the slower algorithm. We will address these stability issues in our paper as we encountered them as well in our numerical experiments.

Our paper is structured as follows. We begin with a short introduction to the group $SO(3)$ including Euler angles and sampling on $SO(3)$. Important properties of the Wigner-D functions D_l^{mn} as well as normalization and relation to functions known from harmonic analysis on the two-dimensional sphere are outlined briefly. We then present an approach to the $SO(3)$ Fourier transform for B-bandlimited functions $f \in \mathrm{L}^2(SO(3))$ at M arbitrary nodes $\boldsymbol{g}_q \in SO(3)$. Following this, the adjoint transform which is of independent interest is described.
Introducing the matrix-vector notation $\boldsymbol{f} = \boldsymbol{D}\hat{\boldsymbol{f}}$ for the transform (1.1) we split up the computation of the whole transform into three steps $\boldsymbol{f} = \boldsymbol{D}\hat{\boldsymbol{f}} = \boldsymbol{F}\boldsymbol{A}\boldsymbol{W}\hat{\boldsymbol{f}}$ where $\boldsymbol{f}$ contains the M function samples $f(\boldsymbol{g}_q)$, $\hat{\boldsymbol{f}}$ the $SO(3)$ Fourier coefficients $\hat{f}_l^{mn}$ and $\boldsymbol{D} = \{D_l^{mn}(\boldsymbol{g}_q)\}$ is a matrix containing Wigner-D functions evaluated at the given sampling nodes. The corresponding matrices $\boldsymbol{F}, \boldsymbol{A}$ and $\boldsymbol{W}$ will be discussed in the following subsections. The first step, the multiplication by the matrix $\boldsymbol{W}$ represents

the Wigner-d transform which is discussed in Section 3.1. There the transform itself is explained as well as a fast algorithm in Section 3.2. Here, the derivation of a modified three-term recurrence relation plays an important role. We will also address stability issues in this section.
The following section deals with an intermediate step, the multiplication by the matrix $\boldsymbol{A}$, transforming Chebyshev sums into trigonometric sums. Moreover we consider the whole $SO(3)$ Fourier transform in Section 3.4 using the NFFT to handle the computation represented by the nonequispaced Fourier matrix $\boldsymbol{F}$. Again, also the adjoint transform is considered briefly. Additionally, Section 3.5 applies our algorithm to the equispaced $SO(3)$ Fourier transform setting by using a Clenshaw-Curtis quadrature rule in our adjoint transform. This gives a brief example on how the evaluation of $SO(3)$ Fourier coefficients from function samples can be accomplished. Finally, in Section 4 we present time and error estimates of our implementations for the Wigner transform and the $SO(3)$ Fourier transforms. We will address once more the mentioned stability issues, compute an equispaced SOFT to test the quadrature rule from Section 3.5 and end with a short conclusion.

parameterize	complete name	complexity	reference
FFT	Fast Fourier Transform	$\mathcal{O}(B^d \log B)$	[9]
NFFT	Nonequispaced Fast Fourier Transform	$\mathcal{O}(M + B^d \log B)$	[14]
SOFT	(Equispaced) SO(3) Fourier Transform	$\mathcal{O}(B^4)$, $M = (2B)^3$	[16]
NFSOFT	Nonequispaced Fast SO(3) Fourier Transform	$\mathcal{O}(M + B^3 \log^2 B)$	Sect. 3.4
NDSOFT	Nonequispaced Discrete SO(3) Fourier Transform	$\mathcal{O}(MB^3)$	(3.2)
FWT	Fast Wigner Transform	$\mathcal{O}(B \log^2 B)$	Sect. 3.2
DWT	Discrete Wigner Transform	$\mathcal{O}(B^2)$	Sect. 3.1

Table 1.1: A list of the transforms mentioned in this paper with references and their asymptotic complexities depending on the bandwidth B and the number of input nodes M.

Note the following naming conventions, that are used throughout the paper. The computation of function samples from $SO(3)$ Fourier coefficients, i.e., the synthesis step will be called the $SO(3)$ Fourier transform while the computation of $SO(3)$ Fourier coefficients from function samples is the inverse transform. For some special sampling sets one can use quadrature rules in order to compute the $SO(3)$ Fourier coefficients with the help of the adjoint $SO(3)$ Fourier transform (see Section 3.5). We avoid the name analysis for this transform, since in the nonequispaced setting the adjoint transform is *not* the inverse transform, i.e., the evaluation of $SO(3)$ Fourier coefficients from function samples. Note that two different types of inverse transforms were considered in [10] and

[29]. These inverse transforms based on our $SO(3)$ Fourier transform as well as on the adjoint $SO(3)$ Fourier transform.

2 Preliminaries

2.1 The Rotation Group $SO(3)$

An orthogonal matrix $\mathbf{R} \in \mathbb{R}^{3\times 3}$ with determinant $\det(\mathbf{R}) = 1$ can be identified with a rotation in $\mathbb{R}^3$. It is an orientation-preserving orthogonal transformation and can be interpreted as the circular movement of an object in $\mathbb{R}^3$ by an angle about a fixed axis. The set of all such matrices $\{\mathbf{R} \in \mathbb{R}^{3\times 3} : \det(\mathbf{R}) = 1, \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}_3\}$ constitutes the special orthogonal group $SO(3)$. There are various ways to parameterize this group [31, Sect. 1.4]. We use one variant of the well known Euler angles.

Definition 2.1 *Given three angles $\alpha, \gamma \in [0, 2\pi)$ and $\beta \in [0, \pi]$, the corresponding rotation $\mathbf{R}(\alpha, \beta, \gamma)$ is given by*

$$\mathbf{R}(\alpha,\beta,\gamma) = \mathbf{R}_{\mathrm{ZYZ}}(\alpha,\beta,\gamma) = \mathbf{R}_{\mathrm{Z}}(\alpha)\mathbf{R}_{\mathrm{Y}}(\beta)\mathbf{R}_{\mathrm{Z}}(\gamma)$$

where

$$\mathbf{R}_{\mathrm{Z}}(\varphi) = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{R}_{\mathrm{Y}}(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

denote rotations about the z- and y-axis, respectively.

In the literature there are different conventions for choosing these angles. Most commonly we find the ZXZ- and the ZYZ-convention where the Euler angles denote a sequence of rotations about the z, x, z-axes or the z, y, z-axes respectively. The two representations can be transformed into each other as

$$\mathbf{R}_{\mathrm{ZYZ}}(\alpha,\beta,\gamma) = \mathbf{R}_{\mathrm{ZXZ}}(\alpha + \pi/2, \beta, \gamma - \pi/2).$$

Throughout this paper we will use the ZYZ-convention, identifying an element $\boldsymbol{g} \in SO(3)$ with $\boldsymbol{g}(\alpha,\beta,\gamma) = \mathbf{R}_{\mathrm{ZYZ}}(\alpha,\beta,\gamma)$. For the sake of simplicity, we will also denote functions $f : SO(3) \to \mathbb{C}$ by

$$f(\boldsymbol{g}(\alpha,\beta,\gamma)) = f(\alpha,\beta,\gamma).$$

2.2 A basis for $L^2(SO(3))$

We now consider the Hilbert space $L^2(SO(3))$ with the inner product of two functions $f_1, f_2 \in L^2(SO(3))$ given by

$$\begin{aligned} \langle f_1, f_2 \rangle &= \int_{SO(3)} f_1(\boldsymbol{g})\overline{f_2(\boldsymbol{g})}\,\mathrm{d}\boldsymbol{g} \\ &= \int_0^{2\pi}\int_0^{\pi}\int_0^{2\pi} f_1(\alpha,\beta,\gamma)\overline{f_2(\alpha,\beta,\gamma)}\sin(\beta)\,\mathrm{d}\alpha\mathrm{d}\beta\mathrm{d}\gamma \qquad (2.1) \end{aligned}$$

and the corresponding norm $||f||_{L^2(SO(3))} = \sqrt{\langle f, f\rangle}$.
In this section we introduce the Wigner-D and Wigner-d functions. They play a major role in Fourier analysis on $SO(3)$. The Wigner-D functions $D_l^{m,n}(\boldsymbol{g})$ are the eigenfunctions of the Laplace operator for $SO(3)$. When parameterizing these eigenfunctions in Euler angles we obtain an explicit expression for the Wigner-D functions (see [5, Sect. 9]). They are given for $|m|, |n| \le l \in \mathbb{N}_0$ by

$$D_l^{m,n}(\alpha,\beta,\gamma) = \mathrm{e}^{-\mathrm{i}m\alpha}\mathrm{e}^{-\mathrm{i}n\gamma} d_l^{m,n}(\cos\beta) \tag{2.2}$$

where

$$d_l^{m,n}(x) = \frac{(-1)^{l-m}}{2^l}\sqrt{\frac{(l+m)!}{(l-n)!(l+n)!(l-m)!}}\sqrt{\frac{(1-x)^{n-m}}{(1+x)^{m+n}}}\frac{\mathrm{d}^{l-m}}{\mathrm{d}x^{l-m}}\frac{(1+x)^{n+l}}{(1-x)^{n-l}} \tag{2.3}$$

are called Wigner-d functions. They satisfy the orthogonality condition

$$\int_0^{\pi} d_l^{m,n}(\cos\beta) d_{l'}^{m,n}(\cos\beta)\sin\beta\,\mathrm{d}\beta = \frac{1}{l+\frac{1}{2}}\delta_{ll'}. \tag{2.4}$$

Wigner-d as well as Wigner-D functions generalize functions that are known from harmonic analysis on the sphere $\mathbb{S}^2$, e.g.,

$$d_l^{0,-n}(x) = P_l^n(x) = \frac{1}{2^l l!}\sqrt{\frac{(l-n)!}{(l+n)!}}\sqrt{(1-x^2)^n}\frac{\mathrm{d}^{l+n}}{\mathrm{d}x^{l+n}}(x^2-1)^l \tag{2.5}$$

where P_l^n denotes the associated Legendre functions. Due to (2.5) the Wigner-d functions are also called generalized associated Legendre functions. We now obtain immediately the relation between Wigner-D functions $D_l^{m,n}(\alpha,\beta,\gamma)$ and spherical harmonics Y_l^n as

$$Y_l^n(\boldsymbol{\xi}) = Y_l^n(\beta,\gamma) = \sqrt{\frac{2l-1}{4\pi}}\mathrm{e}^{\mathrm{i}n\gamma} d_l^{0,n}(\cos\beta) = (-1)^{\delta_{n|n|}}\sqrt{\frac{2l-1}{4\pi}} D_l^{0,-n}(\alpha,\beta,\gamma)$$

where $(\beta,\gamma) \in [0,\pi]\times[0,2\pi)$ are the polar coordinates of the point $\boldsymbol{\xi} \in \mathbb{S}^2$. As a result of this relationship Wigner-D functions are sometimes also called generalized spherical harmonics (cf. e.g. [3]). Note that subsequently we will simplify $D_l^{m,n} = D_l^{mn}$ as well as $d_l^{m,n} = d_l^{mn}$ when there is no possibility for confusion.
By means of the Peter-Weyl-Theorem (cf. [32, Sect. 3.3] for details) the harmonic spaces

$$\mathrm{Harm}_l(SO(3)) = \mathrm{span}\,\{D_l^{mn} : m,n = -l,\dots,l\}$$

spanned by the Wigner-D functions satisfy

$$L^2(SO(3)) = \mathrm{clos}_{L^2}\bigoplus_{l=0}^{\infty}\mathrm{Harm}_l(SO(3)).$$

Hence the collection of Wigner-D functions $\{D_l^{mn}(\boldsymbol{g}) : l \in \mathbb{N}_0, m, n = -l, \ldots, l\}$ forms an orthogonal set in $\mathrm{L}^2(SO(3))$. Moreover we define the function spaces

$$\mathbb{D}_B = \bigoplus_{l=0}^{B} \mathrm{Harm}_l(SO(3))$$

for arbitrary $B \in \mathbb{N}$. The dimension of these spaces is given by

$$\dim(\mathbb{D}_B) = \sum_{l=0}^{B} (2l+1)^2 = \frac{1}{3}(B+1)(2B+1)(2B+3). \tag{2.6}$$

We define an ordered set of indices

$$\mathcal{I}_B = \{(l, m, n) : l = 0, \ldots, B; m, n = -l, \ldots, l\} \tag{2.7}$$

corresponding to all sets of admissible indices (l, m, n) within the space $\mathbb{D}_B$. Throughout this paper we keep a particular order of the indices fixed.
The Wigner-D functions D_l^{mn} are not normalized with respect to the inner product (2.1) but for all $\boldsymbol{g} \in SO(3)$ they satisfy

$$\|D_l^{mn}(\boldsymbol{g})\|^2_{L^2(SO(3))} = \frac{4\pi^2}{l + \frac{1}{2}}.$$

Every element $f \in \mathrm{L}^2(SO(3))$ has a unique series expansion in terms of the Wigner-D functions

$$f(\boldsymbol{g}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{n=-l}^{l} \widehat{f}_l^{mn} D_l^{mn}(\boldsymbol{g}), \tag{2.8}$$

where $\boldsymbol{g} \in SO(3)$ and the $SO(3)$ Fourier coefficients $\widehat{f}_l^{mn}$ are given by the integral

$$\widehat{f}_l^{mn} = \frac{l + \frac{1}{2}}{4\pi^2} \langle f, D_l^{mn} \rangle. \tag{2.9}$$

For approximation purposes we are dealing with B-bandlimited functions $f \in \mathbb{D}_B$ which can be written as their own finite Fourier partial sum

$$f(\boldsymbol{g}) = \sum_{(l,m,n) \in \mathcal{I}_B} \widehat{f}_l^{mn} D_l^{mn}(\boldsymbol{g}). \tag{2.10}$$

The fast evaluation of (2.10) for a set of samples $\boldsymbol{g} \in SO(3)$ is the main subject of this paper.

3 Discrete Fourier Transforms on $SO(3)$

From now on we restrict ourselves to B-bandlimited functions $f \in \mathbb{D}_B$. These functions will be evaluated at a certain number of arbitrary sampling nodes $\boldsymbol{g}_q = (\alpha_q, \beta_q, \gamma_q)$ from a not necessarily equispaced sampling set

$$\mathcal{X}_M = \{(\alpha_q, \beta_q, \gamma_q) : q = 0, \ldots, M-1\} \tag{3.1}$$

where $0 \leq \alpha_q, \gamma_q < 2\pi$ and $0 \leq \beta_q \leq \pi$ are Euler angles. The Fourier sum (2.8) of a B-bandlimited function $f \in \mathbb{D}_B$ reads as

$$f(\alpha_q, \beta_q, \gamma_q) = \sum_{(l,m,n)\in\mathcal{I}_B} \widehat{f_l^{mn}} D_l^{mn}(\boldsymbol{g}_q) = \sum_{l=0}^{B} \sum_{m=-l}^{l} \sum_{n=-l}^{l} \widehat{f_l^{mn}} D_l^{mn}(\alpha_q, \beta_q, \gamma_q) \tag{3.2}$$

for $q = 0, \ldots, M-1$. The evaluation of this finite triple sum where the complex-valued $SO(3)$ Fourier coefficients $\widehat{f_l^{mn}} \in \mathbb{C}$ are given will be called the nonequispaced discrete Fourier transform on the rotation group (NDSOFT).
Using this sum we can synthesize non-equispaced samples of a function f of which we know its $SO(3)$ Fourier coefficients $\widehat{f_l^{mn}}$. If on the other hand one has given M samples $f(\alpha_q, \beta_q, \gamma_q)$ of a function f and seeks to compute $SO(3)$ Fourier coefficients $\widehat{f_l^{mn}}$ from them an iterative algorithm is needed. As we only know samples of the function f we can use quadrature rules for this integral. In general we do not know the quadrature rule for an arbitrary set of sampling nodes in $SO(3)$ (see [29]). By computing

$$\tilde{f}_l^{mn} = \sum_{q=0}^{M-1} f(\alpha_q, \beta_q, \gamma_q)\, \overline{D_l^{mn}(\alpha_q, \beta_q, \gamma_q)} \tag{3.3}$$

for all admissible $(l, m, n) \in \mathcal{I}_B$ we obtain coefficients $\tilde{f}_l^{mn}$ out of the given samples. In general $\tilde{f}_l^{mn} \neq \widehat{f_l^{mn}}$ holds. We will call the evaluation of the sum (3.3) the adjoint NDSOFT. To get a inverse transform to (3.2) we would need additional weights to be included in the sum. If we found a quadrature rule associated to the sampling set $\mathcal{X}_M$ we could compute

$$\widehat{f_l^{mn}} = \sum_{q=0}^{M-1} u_q^B f(\alpha_q, \beta_q, \gamma_q)\, \overline{D_l^{mn}(\alpha_q, \beta_q, \gamma_q)} \tag{3.4}$$

where u_q^B would be quadrature weights depending on the bandwidth B for $q = 0, \ldots, M-1$. This inverse transform could analyze the $SO(3)$ Fourier coefficients out of given samples. We will give a an example of a suitable quadrature rule for a special sampling set in Section 3.5. The more general case where we use arbitrary points and determine the corresponding quadrature rules first is beyond the scope of this paper and will be considered in future work.

Remark 3.1 In the following we point out some applications of the $SO(3)$ Fourier transform as well as the adjont transform. First note that these transforms are instrumental when it comes to iterative algorithms that might lead to an inverse Fourier transform without knowing a quadrature rule explicitly or in case we want to use interpolation. Examples for these iterative methods are CGNE or CGNR algorithms that have been proposed for the $SO(3)$ in [29] and [10].
In further applications that are based on the convolution theorem one obtains immediately the SO(3) Fourier coefficients, from the spherical Fourier coefficents on the two-sphere (see e.g. [19, Eq.(8)] and [18]) and one is interested in the evaluation of the function on $SO(3)$.
In both applications mentioned in the introduction, protein-protein-docking [4] and texture analysis [28] one indeed just needs the $SO(3)$ Fourier transform and its adjoint transform.

For instance in texture analysis one is interested in computing the so-called kernel density function

$$f(\boldsymbol{q}_d) = \sum_{k=1}^{K} c_k \psi(\boldsymbol{q}_d \boldsymbol{g}_k^{-1})$$

at arbitrary nodes $\boldsymbol{q}_d \in SO(3)$ for given coefficients $c_k \in \mathbb{C}$. By using the Wigner-D functions one can separate the source nodes $\boldsymbol{g}_k \in SO(3)$, $k = 1, \ldots, K$, from the target nodes $\boldsymbol{q}_d$, $d = 1, \ldots, D$, as

$$f(\boldsymbol{q}_d) \approx \sum_{l=0}^{B} \sum_{m,n=-l}^{l} \hat{\psi}(l) \left(\sum_{k=1}^{K} c_k \overline{D_l^{mn}(\boldsymbol{g}_k)} \right) D_l^{mn}(\boldsymbol{q}_d)$$

where $\hat{\psi}(l)$ are the known Fourier-Legendre coefficients of the kernel ψ. What we see in the inner brackets is an adjoint (not inverse) $SO(3)$ Fourier transform for given coefficients c_k and a $SO(3)$ Fourier transform for the outer sums. This problem is considered in [13].
Another application in texture analysis where one has to compute values on the $SO(3)$ from the $SO(3)$ Fourier coefficients was discussed in [12, Lemma 2.4, Theorem 2.7] in order to invert the Radon transform. □

Let us now consider the NDSOFT (3.2) from an algebraic point of view. According to (2.6) we are dealing with $\frac{1}{3}(B+1)(2B+1)(2B+3)$ Fourier coefficients $\widehat{f_l^{mn}}$. These coefficients are mapped onto M sampled function values of a function $f \in \mathbb{D}_B$. The NDSOFT can thus be represented in matrix-vector notation as

$$\boldsymbol{f} = \boldsymbol{D}\widehat{\boldsymbol{f}} \tag{3.5}$$

where the vector $\boldsymbol{f} = (f(\boldsymbol{g}))_{\boldsymbol{g} \in \mathcal{X}_M} \in \mathbb{C}^M$ contains the sampled function values of f, and the vector $\widehat{\boldsymbol{f}} = (\widehat{f_l^{mn}})_{(l,m,n) \in \mathcal{I}_B} \in \mathbb{C}^{\frac{1}{3}(B+1)(2B+1)(2B+3)}$ contains the Fourier coefficients for all triples from the set of indices $\mathcal{I}_B$ (see (2.7)). The matrix

$$\boldsymbol{D} = (D_l^{mn}(\alpha_q, \beta_q, \gamma_q))_{(\alpha_q, \beta_q, \gamma_q) \in \mathcal{X}_M; (l,m,n) \in \mathcal{I}_B} \in \mathbb{C}^{M \times \frac{1}{3}(B+1)(2B+1)(2B+3)}$$

will be called the nonequispaced $SO(3)$ Fourier matrix.
The calculation of the adjoint problem (3.3) is realized by a matrix-vector multiplication

$$\tilde{\boldsymbol{f}} = \boldsymbol{D}^H \boldsymbol{f}.$$

Provided there is a known quadrature rule matching the given sampling set, the inverse problem would yield

$$\widehat{\boldsymbol{f}} = \boldsymbol{D}^H \boldsymbol{U} \boldsymbol{f}$$

where $\widehat{\boldsymbol{f}}, \boldsymbol{f}$ and $\boldsymbol{D}$ are the same as in (3.5) and $\boldsymbol{U} \in \mathbb{C}^{M \times M}$ is a diagonal matrix containing the quadrature weights u_q^B. In cases where we do not have a matching quadrature rule (e.g. there are less function samples than Fourier coefficients) and we want to apply interpolation we still need the adjoint NDSOFT (cf. Remark 3.1 and [10]).
Let us analyze the computational complexity of computing the NDSOFT naively via the matrix-multiplication (3.5). Owing to the size of $\boldsymbol{D}$, the asymptotic complexity for evaluating the NDSOFT of a B-bandlimited function $f \in \mathbb{D}_B$ at M nodes would be $\mathcal{O}(MB^3)$ flops. Analogously the adjoint transform in (3.3), i.e., the multiplication of the sample vector $\boldsymbol{f}$ with $\boldsymbol{D}^H$ requires $\mathcal{O}(MB^3)$ operations as well.
Note that the number of nodes M solely depends on the sampling set not on the bandwidth B although typically M is of order $\mathcal{O}(B^3)$. The lower bound of the algorithm is given by the number of input values of which there are $\mathcal{O}(M + B^3)$ consisting of $\mathcal{O}(B^3)$ Fourier coefficients and $\mathcal{O}(M)$ nodes. We will not achieve this lower bound but approach it with $\mathcal{O}(M + B^3 \log^2 B)$ flops. This will be accomplished by generalizing different algorithms. In particular we will adopt an algorithm which has been presented in [17] for the two-dimensional sphere and combine it with the separation of variables technique from [16]. In the following sections we will describe how to split up the NDSOFT (3.5) such that we can use the NFFT [14, 23] on one hand and a fast polynomial transform [7, 21] on the other hand for its evaluation. We start by focusing on the latter.

3.1 Discrete Wigner Transform

In this section we are dealing with the Wigner-d functions d_l^{mn} as they are the main source for the high computational complexity with $\mathcal{O}(B)$ flops needed for their evaluation. One important feature we will use is the orthogonality of the Wigner-d functions d_l^{mn}. Since they are orthogonal functions in the sense of (2.4) they can be described not only by their Rodrigues formula given in (2.3) but also by a three-term recurrence relation that reads for $|m|, |n| \leq l$ as

$$d_{l+1}^{mn}(x) = (u_l^{mn} x + v_l^{mn}) d_l^{mn}(x) + w_l^{mn} d_{l-1}^{mn}(x), \qquad x = \cos\theta, \tag{3.6}$$

with the recurrence coefficients given by

$$\begin{aligned} u_l^{mn} &= \frac{(l+1)(2l+1)}{\sqrt{((l+1)^2-m^2)((l+1)^2-n^2)}}, \\ v_l^{mn} &= \frac{-mn(2l+1)}{l\sqrt{((l+1)^2-m^2)((l+1)^2-n^2)}}, \\ w_l^{mn} &= \frac{-(l+1)\sqrt{(l^2-m^2)(l^2-n^2)}}{l\sqrt{((l+1)^2-m^2)((l+1)^2-n^2)}} \end{aligned}$$

where we set $d_l^{mn}(x) = 0$ for all $l < \max(|m|,|n|)$ (cf. [31, Sect. 4.3]).
For the evaluation of this three-term recurrence relation we can now employ the Clenshaw algorithm e.g., [25, pp. 184]. Although it decreases the memory requirements as compared with the naive matrix multiplication from (3.5) it is still too slow for most applications as it does not reduce the computational complexity of $\mathcal{O}(MB^3)$ flops in total. This recursive evaluation however will be used in the discrete Wigner transform (DWT) which will be described now.
The aim of the DWT is to transform a linear combination of Wigner-d functions d_l^{mn} from (2.3) into a linear combination of Chebyshev polynomials of the first kind T_l.
We will take advantage of the fact that Wigner-d functions are similar to the associated Legendre functions that are the subject of [17]. In fact they are a generalization of those functions as mentioned before in (2.5).
From (2.3) it can be seen that for $m+n$ even the Wigner-d functions $d_l^{mn}(x)$ are polynomials of degree at most l whereas for odd $m+n$ they have to be transformed into polynomials of degree $l-1$ by multiplying them with $(1-x^2)^{-1/2}$. Performing a change of the polynomial basis we get the following Chebyshev coefficients t_l^{mn} from the input coefficients $\hat{f}_l^{mn}$ by computing

$$\sum_{l=\max(|m|,|n|)}^{B} \hat{f}_l^{mn} d_l^{mn}(\cos\theta) = \begin{cases} \displaystyle\sum_{l=0}^{B} t_l^{mn} T_l(\cos\theta) & \text{for } m+n \text{ even}, \\ \displaystyle\sin\theta \sum_{l=0}^{B-1} t_l^{mn} T_l(\cos\theta) & \text{for } m+n \text{ odd}, \end{cases} \tag{3.7}$$

at nodes $\theta \in \mathcal{X}_C$ on the equispaced grid $\mathcal{X}_C = \left\{ \frac{(2k+1)\pi}{2(B+1)} : k = 0,\ldots,B \right\}$ where m,n are fixed and $T_l(\cos\theta) = \cos l\theta$ are Chebyshev polynomials of degree l.

Definition 3.2 *For fixed orders m and n the change of basis given in equation (3.7) from the vector of SO(3) Fourier coefficients $\hat{\boldsymbol{f}}^{mn} = \left(\hat{f}^{mn}_{\max(|m|,|n|)},\ldots,\hat{f}^{mn}_B\right)^T$ to the vector of Chebyshev coefficients $\boldsymbol{t}^{mn} = (t_0^{mn},\ldots,t_B^{mn})^T$ is described by the matrix $\boldsymbol{W}^{mn} \in \mathbb{C}^{(B+1)\times(B-\max(|m|,|n|))}$ through*

$$\boldsymbol{t}^{mn} = \boldsymbol{W}^{mn}\hat{\boldsymbol{f}}^{mn}.$$

Lemma 3.3 *The matrix $\boldsymbol{W}^{mn}$ given in Definition 3.2 can be separated into*

$$\boldsymbol{W}^{mn} = \begin{cases} \boldsymbol{T}\tilde{\boldsymbol{D}}^{mn} & \text{for } m+n \text{ even,} \\ \boldsymbol{S}^{-1}\boldsymbol{T}\tilde{\boldsymbol{D}}^{mn} & \text{for } m+n \text{ odd} \end{cases} \tag{3.8}$$

where

$$\begin{aligned} \boldsymbol{T} = & \left(\frac{2-\delta_{0k}}{B+1} \cos \frac{(2l+1)k\,\pi}{2(B+1)} \right)_{k,l=0,\dots,B}, \\ \tilde{\boldsymbol{D}}^{mn} = & \left(d_l^{mn}(\cos \frac{(2k+1)\pi}{2(B+1)}) \right)_{(k=0,\dots,B);(l=\max(|m|,|n|),\dots,B)} \end{aligned}$$

and

$$\boldsymbol{S} = \operatorname{diag}\left(\sin \frac{(q+1)\pi}{B+2} \right)_{q=0,\dots,B}.$$

Proof. We consider the $B+1$ nodes of an equispaced grid $\mathcal{X}_C$. For these nodes the matrix-vector notation of equation (3.7) reads as

$$\tilde{\boldsymbol{D}}^{mn}\widehat{\boldsymbol{f}}^{mn} = \begin{cases} \boldsymbol{T}^{-1}\boldsymbol{t}^{mn} & \text{for } m+n \text{ even,} \\ \boldsymbol{T}^{-1}\boldsymbol{S}\boldsymbol{t}^{mn} & \text{for } m+n \text{ odd} \end{cases}$$

where $\widehat{\boldsymbol{f}}^{mn}, \boldsymbol{t}^{mn}, \boldsymbol{S}$, and $\tilde{\boldsymbol{D}}^{mn}$ are defined as in Lemma 3.3. A simple calculation shows that the inverse of $\boldsymbol{T}$ is given by

$$\boldsymbol{T}^{-1} = \left(\cos \frac{(2k+1)l\,\pi}{2(B+1)} \right)_{k,l=0,\dots,B}.$$

Since $\boldsymbol{S}$ is also a nonsingular matrix we obtain the unique solution of

$$\boldsymbol{t}^{mn} = \boldsymbol{T}\tilde{\boldsymbol{D}}^{mn}\widehat{\boldsymbol{f}}^{mn} = \boldsymbol{W}^{mn}\widehat{\boldsymbol{f}}^{mn}$$

for even orders $m+n$, and

$$\boldsymbol{t}^{mn} = \boldsymbol{S}^{-1}\boldsymbol{T}\tilde{\boldsymbol{D}}^{mn}\widehat{\boldsymbol{f}}^{mn} = \boldsymbol{W}^{mn}\widehat{\boldsymbol{f}}^{mn}$$

for odd orders $m+n$. ■

Note that the matrix-vector multiplication $\tilde{\boldsymbol{D}}^{mn}\widehat{\boldsymbol{f}}^{mn}$ will be computed recursively in $\mathcal{O}(B^2)$ steps with the Clenshaw algorithm using (3.6) for fixed m and n. Neither the multiplication by $\boldsymbol{T}^{-1}$ which is realized with the discrete cosine transform (DCT) from [1] in $\mathcal{O}(B \log B)$ flops nor the multiplication by the diagonal matrix $\boldsymbol{S}$ increases the asymptotic complexity. Thus the total complexity of the DWT, i.e., a multiplication by $\boldsymbol{t}^{mn} = \boldsymbol{W}^{mn}\widehat{\boldsymbol{f}}^{mn}$ is $\mathcal{O}(B^2)$. As there are $(2B+1)^2$ vectors $\boldsymbol{t}^{mn}$ to be computed the total complexity of the whole transformation step sums to $\mathcal{O}(B^4)$ flops.

3.2 Fast Wigner Transform

In this section we outline how to speed up the DWT by adopting the fast polynomial transform together with the stabilization presented in [22]. Other stabilization methods for the fast spherical harmonic transform were presented in [30, 11]. Our resulting algorithm will be referred to as the fast Wigner transform (FWT).

Our aim is the efficient computation of matrix-vector multiplications using the matrices $\boldsymbol{W}^{mn}$ for all $m, n = -B, \ldots, B$. Lemma 3.3 shows that we can split $\boldsymbol{W}^{mn}$ into a product of the matrices $\boldsymbol{T}, \boldsymbol{S}^{-1}$ and $\tilde{\boldsymbol{D}}^{mn}$. As $\boldsymbol{S}$ is a diagonal matrix a multiplication by $\boldsymbol{S}$ can be computed in $\mathcal{O}(B)$ flops. The effect of matrix $\boldsymbol{T}$ can be computed by means of the DCT in $\mathcal{O}(B \log B)$ flops. What remains to be done is to employ a fast algorithm for the transform represented by the matrix $\tilde{\boldsymbol{D}}^{mn}$.

A simple but powerful idea is to modify the three-term recurrence relation from (3.6) by defining the Wigner-d functions for $|m|, |n| > l$ as well. Note that this is the important step, which allows us to realize the fast Wigner-d transform in a stable way. This technique has been described in [22] as a key step in the development of a stable fast spherical Fourier transform. By that trick we obtain for $m, n = -B, \ldots, B$ and $l = 0, \ldots, B$ for arbitrary $B \in \mathbb{N}_0$ an extended three-term recurrence formula

$$d_{l+1}^{mn}(x) = (\alpha_l^{mn} x + \beta_l^{mn}) d_l^{mn}(x) + \gamma_l^{mn} d_{l-1}^{mn}(x), \qquad x = \cos\theta, \tag{3.9}$$

where for $\mu = \min(|m|, |n|)$ and $\nu = \max(|m|, |n|)$ the recurrence coefficients read as

$$\alpha_0^{mn} = \begin{cases} 1 & \text{for } m = n, \\ -1 & \text{for } m+n \text{ even, } m \neq n, \\ 0 & \text{otherwise,} \end{cases} \qquad \alpha_l^{mn} = \begin{cases} (-1)^{m+n+1} & \text{for } 0 < l \le \nu - \mu, \\ \frac{mn}{|mn|} & \text{for } \nu - \mu < l < \nu, \\ u_l^{mn} & \text{for } \nu \le l, \end{cases}$$

$$\beta_l^{mn} = \begin{cases} 1 & \text{for } 0 \le l < \nu, \\ 0 & \text{for } m = n = 0, \\ v_l^{mn} & \text{otherwise} \end{cases} \qquad \text{and} \qquad \gamma_l^{mn} = \begin{cases} 0 & \text{for } l \le \nu, \\ w_l^{mn} & \text{otherwise,} \end{cases}$$

using u_l^{mn}, v_l^{mn} and w_l^{mn}, the recurrence coefficients from (3.6). To start the recurrence for all $m, n = -B, \ldots, B$ we set $d_{-1}^{mn} = 0$ and with $\lambda_{mn} = \frac{\sqrt{(2\mu)!}}{2^\mu \mu!}$ we obtain

$$d_0^{mn}(x) = \begin{cases} \lambda_{mn} & \text{for } m+n \text{ even,} \\ \lambda_{mn}\sqrt{1-x^2} & \text{for } m+n \text{ odd.} \end{cases}$$

The FWT will exploit the concepts of the fast polynomial transforms as it performs a change of basis replacing the Wigner-d functions with Chebyshev polynomials of the first kind. Following [17] we perform the fast polynomial multiplication based on discrete cosine transforms (DCT).

For this we derive associated Wigner-d functions $d_l^{mn}(\cdot, c)$ with a shift parameter $c \in \mathbb{N}_0$

by

$$\begin{aligned}
d_{-1}^{mn}(x,c) &= 0, \\
d_{0}^{mn}(x,c) &= 1, \\
d_{l+1}^{mn}(x,c) &= (\alpha_{l+c}^{mn} x + \beta_{l+c}^{mn}) d_{l}^{mn}(x,c) + \gamma_{l+c}^{mn} d_{l-1}^{mn}(x,c). \qquad (3.10)
\end{aligned}$$

Instead of only one step as in the modified recurrence (3.9) we will now shift the degree l of d_l^{mn} by $c \in \mathbb{N}_0$ steps using the result of the following Lemma.

Lemma 3.4 *Let $B \in \mathbb{N}_0$, $m,n = -B,\ldots,B$ and $c,l = 0,\ldots,B$ and let the functions d_l^{mn} and $d_l^{mn}(\cdots,c)$ be given as in* (3.9) *and* (3.10)*, respectively. We then obtain*

$$d_{l+c}^{mn}(x) = d_c^{mn}(x,l)d_l^{mn}(x) + \gamma_l^{mn} d_{c-1}^{mn}(x,l+1)d_{l-1}^{mn}(x).$$

Proof. The proof is done by induction over c. ■

On these functions we perform the fast polynomial transform that has been described in [17] and [7] for the sphere $\mathbb{S}^2$. Applying their algorithms we can speed up our DWT from $\mathcal{O}(B^2)$ to only $\mathcal{O}(B \log^2 B)$ flops per set of orders m,n. For all $\mathcal{O}(B^2)$ different sets of orders m,n we obtain a total complexity of $\mathcal{O}(B^3 \log^2 B)$ flops for all FWTs instead of the previous $\mathcal{O}(B^4)$ flops for all DWTs.
When using exact arithmetic the algorithm is exact. But when computing in finite precision small errors introduced by the DCT algorithm cause numerical instabilities (cf. [24]) as they are multiplied with large function values of the associated Wigner-d functions $d_c^{mn}(x,l)$ for $|x| \approx 1$ at certain admissible triples $(c,m,n) \in \mathcal{I}_M$.
The algorithms mentioned above use a cascade summation to compute the polynomials. An effective approach to improve the stability of this summation has been developed in [22]. The authors replace a standard multiplication step in the cascade built by the algorithm with a stabilization step whenever the functions to be transformed exceed a certain threshold κ. This corresponds to removing the critical polynomial from the cascade, dealing with it separately and inserting it again at the very end. In a scenario were every single polynomial is removed from the cascade we would get back exactly the slow DWT algorithm.
The algorithmic details on this method as well as its implementation for associated Legendre functions can be found in [17] and [14], respectively. This method can be directly applied to the Wigner-d functions as they show the same behavior.
Although we now have a method to improve the stability of our computation its application will increase the runtime of the algorithm. So far we do not know an upper bound for the number of stabilization steps with respect to the bandwidth B for a given threshold κ and thus the asymptotical complexity of the stabilized FWT. However numerical experiments have been conducted and their results are shown in Section 4. They support the conjecture that although the stabilized FWT is slower than the unstabilized version with $\mathcal{O}(B \log^2 B)$ flops it is still asymptotically faster than the DWT with $O(B^2)$ flops.

Remark 3.5 As mentioned in the introduction, $SO(3)$ Fourier transforms arise as a generalization of spherical Fourier transforms. That means apart from the polynomial cascade we could have used other algorithms for transforming the polynomial part of our $SO(3)$ basis functions after applying the DCT to them. Indeed we find other works that deal with other techniques to compute spherical harmonic expansions, like [20, 30, 11, 27]. We suppose that these approaches can also be extended for the NFSOFT. $\square$

3.3 An intermediate step

The FWT generates Chebyshev coefficients t_l^{mn} from the SO(3) Fourier coefficients $\widehat{f}_l^{mn}$ as seen in (3.7). Yet another change of basis needs to be done. Our aim is to obtain suitable coefficients for a trivariate Fourier transform. Thus we need to find coefficients h_l^{mn} which satisfy

$$\sum_{l=0}^{B} t_l^{mn} T_l(\cos\beta) = \sum_{l=-B}^{B} h_l^{mn} \mathrm{e}^{-\mathrm{i}l\beta} \tag{3.11}$$

for $m+n$ even and

$$\sin\beta \sum_{l=0}^{B} t_l^{mn} T_l(\cos\beta) = \sum_{l=-B}^{B} h_l^{mn} \mathrm{e}^{-\mathrm{i}l\beta} \tag{3.12}$$

for $m+n$ odd. For fixed orders m and n the change of basis from the Chebyshev coefficients $\boldsymbol{t}^{mn} = (t_0^{mn}, \ldots, t_B^{mn})^T$ to the Fourier coefficients $\boldsymbol{h^{mn}} = (h_{-B}^{mn}, \ldots, h_B^{mn})^T$ given in equations (3.11) and (3.12) is described by the matrix $\boldsymbol{A}^{mn} \in \mathbb{C}^{(2B+1)\times(B+1)}$ through

$$\boldsymbol{h}^{mn} = \boldsymbol{A}^{mn} \boldsymbol{t}^{mn}.$$

To obtain an explicit expression for the matrix $\boldsymbol{A}^{mn}$ we first consider the slightly easier case in which $m+n$ is even. By $\cos l\beta = \frac{1}{2}(\mathrm{e}^{\mathrm{i}l\beta} + \mathrm{e}^{-\mathrm{i}l\beta})$ we obtain

$$\sum_{l=0}^{B} t_l^{mn} T_l(\cos\beta) = \sum_{l=0}^{B} t_l^{mn} \cos l\beta = \frac{1}{2}\left(t_0^{mn} + \sum_{l=-B}^{B} t_{|l|}^{mn} \mathrm{e}^{\mathrm{i}l\beta} \right).$$

Setting

$$h_l^{mn} = \begin{cases} 0 & \text{for } |l| = M-1, M, \\ t_0^{mn} & \text{for } l = 0, \\ \frac{1}{2} t_{|l|}^{mn} & \text{otherwise,} \end{cases} \tag{3.13}$$

we achieve the desired change of basis. For arbitrary but fixed orders m and n this transformation can be written in matrix-vector-notation as

$$\boldsymbol{H}^{mn} = \begin{pmatrix} & & & \frac{1}{2} \\ & & \iddots & \\ & \frac{1}{2} & & \\ 1 & & & \\ & \frac{1}{2} & & \\ & & \ddots & \\ & & & \frac{1}{2} \end{pmatrix} \in \mathbb{C}^{(2B+1)\times(B+1)}. \tag{3.14}$$

Hence $\boldsymbol{h}^{mn} = \boldsymbol{H}^{mn}\boldsymbol{t}^{mn}$ for $m+n$ even. In the case of $m+n$ odd, we obtain

$$\sin\beta \sum_{l=0}^{B-1} t_l^{mn} T_l(\cos\beta) = \sin\beta \sum_{l=0}^{B-1} t_l^{mn} \cos l\beta = \frac{\sin\beta}{2}\left(t_0^{mn} + \sum_{l=-B}^{B} t_{|l|}^{mn} \mathrm{e}^{\mathrm{i}l\beta}\right).$$

It remains to convert $\sin\beta = \frac{\mathrm{i}}{2}(\mathrm{e}^{-\mathrm{i}\beta} - \mathrm{e}^{\mathrm{i}\beta})$ and to replace the t_l^{mn} according to (3.13) which yields

$$\sin\beta \sum_{l=0}^{B-1} t_l^{mn} T_l(\cos\beta) \quad = \quad \sum_{l=-B}^{B} \frac{\mathrm{i}}{2}(h_{|l+1|}^{mn} - h_{|l-1|}^{mn})\mathrm{e}^{\mathrm{i}l\beta}.$$

In other words, we need to multiply the matrix $\boldsymbol{H}^{mn}$ with

$$\boldsymbol{O}^{mn} = \frac{\mathrm{i}}{2}\begin{pmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & -1 & 0 \end{pmatrix} \in \mathbb{C}^{(2B+1)\times(B+1)}.$$

which yields $\boldsymbol{h}^{mn} = \boldsymbol{O}^{mn}\boldsymbol{H}^{mn}\boldsymbol{t}^{mn}$ for $m+n$ odd. All in all the matrix $\boldsymbol{A}^{mn}$ is composed as follows

$$\boldsymbol{A}^{mn} = \begin{cases} \boldsymbol{H}^{mn} & \text{for } m+n \text{ even,} \\ \boldsymbol{O}^{mn}\boldsymbol{H}^{mn} & \text{for } m+n \text{ odd.} \end{cases} \tag{3.15}$$

The multiplication of $\boldsymbol{A}^{mn}$ can be done in $\mathcal{O}(B)$ steps as it is sparse. So for all possible m, n this yields $\mathcal{O}(B^3)$ flops in total for the intermediate step.

3.4 The Nonequispaced Fast SO(3) Fourier Transform (NFSOFT)

Sections 3.1 and 3.3 provides us with the tools necessary to manage the fast computation of $SO(3)$ Fourier transforms of bandlimited functions. Looking again at the NDSOFT in (3.2) the naive approach of computation would yield $\mathcal{O}(MB^3)$ flops.
The basic idea for obtaining a fast algorithm is a simple reorganization of the sums from NDSOFT in (3.2) and (3.3). Two steps are necessary for this factorization: a separation of variables and a re-indexing of the sums. In Algorithm 1 we will see in detail that the application of Theorem 3.6 reduces the asymptotic complexity from $\mathcal{O}(MB^3)$ to $\mathcal{O}(M + B^3\log^2 B)$ when omitting stabilization. In the following theorem we describe the matrix factorization of the NDSOFT given in (3.5).

Theorem 3.6 *The matrix $\boldsymbol{D}$ given in (3.5) by $\boldsymbol{f} = \boldsymbol{D}\hat{\boldsymbol{f}}$ representing the NDSOFT can be split into the matrix product*

$$\boldsymbol{D} = \boldsymbol{F}\boldsymbol{A}\boldsymbol{W}$$

where

$$\boldsymbol{W} = \operatorname{diag}\left(\boldsymbol{W}^{mn}\right)_{m,n=-B,\dots,B} \in \mathbb{C}^{(2B+1)^2(B+1)\times(2B+1)^2(B+1)}$$

is the diagonal block matrix consisting of the matrices $\boldsymbol{W}^{mn}$ *from Definition 3.2,*

$$\boldsymbol{A} = \operatorname{diag}\left(\boldsymbol{A}^{mn}\right)_{m,n=-B,\ldots,B} \in \mathbb{C}^{(2B+1)^3\times(2B+1)^2(B+1)}$$

the diagonal block matrix composed of blocks from (3.15) *and finally a trivariate nonequispaced Fourier matrix* $\boldsymbol{F} \in \mathbb{C}^{M\times(2B+1)^3}$ *with*

$$\boldsymbol{F} = \left(\mathrm{e}^{-\mathrm{i}\left((m,l,n)(\alpha_q,\beta_q,\gamma_q)^T\right)}\right)_{q=0,\ldots,M-1;(l,m,n)\in\mathcal{I}_B}. \tag{3.16}$$

Proof. For the separation of variables we simply use formula (2.2), which splits up the Wigner-D functions according to the Euler angles of $f(\alpha_q,\beta_q,\gamma_q) \in \mathbb{D}_B$. We may rewrite (3.2) for $q = 0,\ldots,M-1$ as

$$\begin{aligned} f(\alpha_q,\beta_q,\gamma_q) &= \sum_{l=0}^{B}\sum_{m=-l}^{l}\sum_{n=-l}^{l} \hat{f}_l^{mn} D_l^{mn}(\alpha_q,\beta_q,\gamma_q) \\ &= \sum_{l=0}^{B}\sum_{m=-l}^{l}\sum_{n=-l}^{l} \hat{f}_l^{mn} \mathrm{e}^{-\mathrm{i}m\alpha_q} d_l^{mn}(\cos\beta_q)\mathrm{e}^{-\mathrm{i}n\gamma_q}. \end{aligned} \tag{3.17}$$

The next step is to rearrange these sums into

$$f(\alpha_q,\beta_q,\gamma_q) = \sum_{m=-B}^{B} \mathrm{e}^{-\mathrm{i}m\alpha_q} \sum_{n=-B}^{B} \mathrm{e}^{-\mathrm{i}n\gamma_q} \sum_{l=\max(|m|,|n|)}^{B} \hat{f}_l^{mn} d_l^{mn}(\cos\beta_q).$$

Performing the change of basis from equation (3.7) we are able to replace the last sum by a linear combination of Chebyshev polynomials

$$f(\alpha_q,\beta_q,\gamma_q) = \sum_{m=-B}^{B} \mathrm{e}^{-\mathrm{i}m\alpha_q} \sum_{n=-B}^{B} \mathrm{e}^{-\mathrm{i}n\gamma_q} \sum_{l=0}^{B} t_l^{mn} T_n(\cos\beta_q)(\sin\beta_q)^{\mathrm{mod}(m+n,2)}.$$

Then (3.11) and (3.12) provide the change from Chebyshev coefficients to standard Fourier coefficients, i.e.,

$$\begin{aligned} f(\alpha_q,\beta_q,\gamma_q) &= \sum_{m=-B}^{B} \mathrm{e}^{-\mathrm{i}m\alpha_q} \sum_{n=-B}^{B} \mathrm{e}^{-\mathrm{i}n\gamma_q} \sum_{l=-B}^{B} \mathrm{e}^{-\mathrm{i}l\beta_q} h_l^{mn} \\ &= \sum_{m=-B}^{B}\sum_{n=-B}^{B}\sum_{l=-B}^{B} h_l^{mn} \mathrm{e}^{-\mathrm{i}(m\alpha_q+n\gamma_q+l\beta_q)} \end{aligned}$$

for $q = 0,\ldots,M-1$. Thus we obtain a trivariate Fourier transform which can be represented by the matrix $\boldsymbol{F}$ in (3.16). ■

Corollary 3.7 The adjoint NFSOFT, i.e., the matrix-vector multiplication by $\boldsymbol{D}^H$ as in (3.4) reads in matrix-vector notation as

$$\hat{\boldsymbol{f}} = \boldsymbol{D}^H \boldsymbol{Q} \boldsymbol{f}$$

where $\boldsymbol{Q}$ is a matrix containing the quadrature weights u_q^B. Corresponding to the matrix $\boldsymbol{D}$ we split up its adjoint in a similar way into

$$\widehat{\boldsymbol{f}} = \boldsymbol{W}^H \boldsymbol{A}^H \boldsymbol{F}^H \boldsymbol{Q} \boldsymbol{f}.$$

□

A possible choice for the matrix $\boldsymbol{Q}$ will be discussed in Section 3.5. Our implementations also provide an algorithm to compute this adjoint problem quickly, i.e., with the same asymptotic complexity as the NFSOFT. Note that the explicit factorization of $\boldsymbol{Q}$ is accomplished in a manner similar to that in [15].

Algorithm 1 Nonequispaced Fast $SO(3)$ Fourier Transform (NFSOFT)

Input:	$B \in \mathbb{N}$	the bandwidth
	$\mathcal{X}_M$	a sampling set according to (3.1)
	$\widehat{\boldsymbol{f}} = (\widehat{f}_l^{mn})$	the $\frac{1}{3}(B+1)(2B+1)(2B+3)$ SO(3) Fourier coefficients of $f \in \mathbb{D}_B$

1. Compute $\boldsymbol{t} = \boldsymbol{W}\widehat{\boldsymbol{f}}$, the vector of $(B+1)(2B+1)^2$ Chebyshev coefficients t_l^{mn} in $\mathcal{O}(B^3 \log^2 B)$ flops as described in Section 3.2.
2. Compute $\boldsymbol{h} = \boldsymbol{A}\boldsymbol{t}$, the vector of $(2B+1)^3$ Fourier coefficients h_l^{mn} in $\mathcal{O}(B^3)$ flops as in Section 3.3.
3. Compute $\boldsymbol{f} = \boldsymbol{F}\boldsymbol{h}$ by means of a trivariate NFFT in $\mathcal{O}(M + B^3 \log B)$ flops.

Output: $\boldsymbol{f} = (f(\alpha_q, \beta_q, \gamma_q))_{q=0,\ldots,M-1}$ the function samples of $f \in \mathbb{D}_B$ as given in (3.17)

Complexity: $\mathcal{O}(M + B^3 \log^2 B)$ flops

Clearly in Algorithm 1 the application of the NFFT in Step 3 can be replaced by a FFT in case we have a sampling set of equispaced input nodes. In that case we will refer to the algorithm as the fast $SO(3)$ Fourier transform. Details on this special case can be found in Section 3.5.

Remark 3.8 Instead of transforming the SO(3) Fourier coefficients into Chebyshev coefficients first, they could also be directly transformed into standard Fourier coefficients. This is done by expanding the Wigner-d functions directly into a Fourier series

$$d_l^{mn}(\cos\theta) = \sum_{s=-l}^{l} \widehat{d}_{ls}^{mn} \mathrm{e}^{\mathrm{i}s\theta}, \qquad |m|, |n| \le l,$$

where the Fourier coefficients $\widehat{d}_{ls}^{mn}$ are given by

$$\widehat{d}_{ls}^{mn} = (-1)^s \mathrm{e}^{\mathrm{i}\pi\frac{m+n}{2}} d_l^{ms}\left(\frac{\pi}{2}\right) d_l^{sn}\left(\frac{\pi}{2}\right),$$

see [26] for details. What we gained by this rearrangement is that we omit the polynomial transform and can now compute the DWT by means of an NFFT as well. But in contrast to the previously suggested method we do not improve the computational complexity of the discrete Wigner transform as we have to deal with an additional sum depending on the bandwidth B. The asymptotic complexity would then increase to $\mathcal{O}(M + B^4)$. $\square$

3.5 An exemplary quadrature rule

Algorithm 1 enables us to evaluate function values $f(\alpha, \beta, \gamma)$ from given $SO(3)$ Fourier coefficients $\widehat{f}_l^{mn}$ for any arbitrary sampling set $\mathcal{X}_M$. For special choices of $\mathcal{X}_M$ we are able to reconstruct the $SO(3)$ Fourier coefficients $\widehat{f}_l^{mn}$ from function samples $f(\alpha, \beta, \gamma)$ using a quadrature rule. The investigation of the adjoint NFSOFT and further quadrature rules will be subject of future work but we will present one example here. The reason to do so is we want to validate the results of the NFSOFT in Section 4. One idea is to transform $SO(3)$ Fourier coefficients into function samples by the NFSOFT from Theorem 3.6. Then, by means of an adjoint NFSOFT, see Corollary 3.7, we reconstruct the $SO(3)$ Fourier coefficients used as input.
We assume the equispaced grid

$$\mathcal{X}_B^{CC} = \{(\alpha_{a_1}, \beta_b, \gamma_{a_2}) : a_1, a_2 = 0, \ldots, 2B + 1; b = 0, \ldots, 2B\} \tag{3.18}$$

with uniformly distributed sample points

$$\alpha_{a_1} = \frac{\pi a_1}{B+1}, \; \gamma_{a_2} = \frac{\pi a_2}{B+1} \text{ and } \beta_b = \frac{\pi b}{2B}.$$

Note that technically speaking we are now dealing with an $SO(3)$ Fourier transform at equispaced nodes which can be computed by using the standard FFT instead of the NFFT in the last step of Algorithm 1. Other algorithms for equispaced discrete $SO(3)$ Fourier transforms can also be found e.g. in [16] (see Remark 3.9).
By using the grid $\mathcal{X}_B^{CC}$ we can compare accuracy and time requirements of our results with the exact computations, as we know the associated quadrature rule. In our example we use a Clenshaw-Curtis quadrature rule for the Wigner-d functions. Let us now collect the formulae necessary for the reconstruction.
We would like to evaluate a function $f \in \mathbb{D}_B$ for given Fourier coefficients $\widehat{f}_l^{mn}$. Starting with the Fourier coefficients given by the integral

$$\begin{aligned}
\widehat{f}_l^{mn} &= \frac{2l+1}{8\pi^2} \int_0^{2\pi} \int_0^{\pi} \int_0^{2\pi} f(\alpha, \beta, \gamma) \overline{D_l^{mn}(\alpha, \beta, \gamma)} \sin\beta \, \mathrm{d}\alpha \, \mathrm{d}\beta \, \mathrm{d}\gamma, \\
&= \frac{2l+1}{8\pi^2} \int_0^{2\pi} \mathrm{e}^{\mathrm{i}m\alpha} \int_0^{2\pi} \mathrm{e}^{\mathrm{i}n\gamma} \int_0^{\pi} d_l^{mn}(\beta) f(\alpha, \beta, \gamma) \sin\beta \, \mathrm{d}\beta \, \mathrm{d}\gamma \, \mathrm{d}\alpha
\end{aligned}$$

we can employ the following quadrature rule to describe the integral with respect to the two angles α and γ about the z-axis with

$$\begin{aligned} f_b(\beta) &= \int_0^{2\pi}\int_0^{2\pi} f(\alpha,\beta,\gamma)\mathrm{e}^{\mathrm{i}m\alpha}\mathrm{e}^{\mathrm{i}n\gamma}\,\mathrm{d}\gamma\,\mathrm{d}\alpha \\ &= \frac{1}{4(B+1)^2}\sum_{a_1=0}^{2B+1}\sum_{a_2=0}^{2B+1} f(\alpha_{a_1},\beta,\gamma_{a_2})\mathrm{e}^{\mathrm{i}(m\alpha_{a_1}+n\gamma_{a_2})} \end{aligned} \tag{3.19}$$

at nodes $\alpha_{a_1} = \frac{a_1\pi}{B+1}$ and $\gamma_{a_2} = \frac{a_2\pi}{B+1}$ for $a_1, a_2 = 0, \ldots, 2B+1$. For the quadrature of the Wigner-d function we use a Clenshaw-Curtis rule with nodes $\beta_b = \frac{l\pi}{2B}$ for $b = 0, \ldots, 2B$ and weights u_b^B such that the integral transforms to

$$\int_0^{\pi} d_l^{mn}(\beta) f_b(\beta)\sin\beta\mathrm{d}\beta = \frac{1}{2l+1}\sum_{b=0}^{2B} u_b^B d_l^{mn}(\beta_b) f_b(\beta_b). \tag{3.20}$$

The weights are given according to [6, pp. 86] by

$$u_b^B = u_{b+B}^B = \frac{\epsilon_b^{2B}}{B}\sum_{j=0}^{B}\epsilon_j^B \frac{-2}{4j^2-1}\cos\left(\frac{\pi b j}{B}\right) \tag{3.21}$$

with $\epsilon_l^B = 1 - \dfrac{\delta_{l0}+\delta_{lB}}{2}$ for $b = 0, \ldots, B$. We are now able to weight the adjoint transform with combined weights such that the Fourier transform is exactly computed for a set of nodes on the grid $\mathcal{X}_B^{CC}$. Putting together (3.19), (3.20) and (3.21) yields

$$\widehat{f_l^{mn}} = \frac{1}{4(B+1)^2(2l+1)}\sum_{a_1=0}^{2B+1}\sum_{a_2=0}^{2B+1}\sum_{b=0}^{2B} u_b^B f(\alpha_{a_1},\beta_b,\gamma_{a_2})\overline{D_l^{mn}(\alpha_{a_1},\beta_b,\gamma_{a_2})}. \tag{3.22}$$

Seen from an algebraic point of view, we have the vector of $SO(3)$ Fourier coefficients

$$\widehat{\boldsymbol{f}} = (\widehat{f_l^{mn}})_{(l,m,n)\in\mathcal{I}_B} \in \mathbb{C}^{(\frac{4B^3-B}{3})},$$

the data vector

$$\boldsymbol{f} = (f(\alpha_{a_1},\beta_b,\gamma_{a_2}))_{(\alpha_{a_1},\beta_b,\gamma_{a_2})\in\mathcal{X}_B^{CC}} \in \mathbb{C}^{(2B+2)^2(2B+1)}$$

containing the sampled values of the original function as well as the matrix $\mathbf{D} \in C^{(2B+2)^2(2B+1)\times(\frac{4B^3-B}{3})}$ consisting of the Wigner-D functions for all indices $(l,m,n) \in \mathcal{I}_B$ at all sampling nodes $\boldsymbol{g}(\alpha_{a_1},\beta_b,\gamma_{a_2})$ with $(\alpha_{a_1},\beta_b,\gamma_{a_2}) \in \mathcal{X}_B^{CC}$.
The quadrature weights go into two different diagonal matrices. The first matrix

$$\boldsymbol{U}_B = \operatorname{diag}\left(u_{a_1,b,a_2}^B\right)_{(a_1,b,a_2)\in\mathcal{X}_B^{CC}} \in \mathbb{R}^{((2B+2)^2(2B+1))\times((2B+2)^2(2B+1))}$$

depends on the sampling nodes $\boldsymbol{g}(\alpha_{a_1}, \beta_b, \gamma_{a_2})$ and contains the weights $u^B_{a_1,b,a_2} = u^B_b$ ordered according to the indices a_1, b, a_2. The remaining parts of the weights are going into another diagonal matrix

$$\boldsymbol{V}_B = \operatorname{diag}\left(v^B_{m,l,n}\right)_{(l,m,n)\in\mathcal{I}_B} \in \mathbb{R}^{(\frac{4B^3-B}{3})\times(\frac{4B^3-B}{3})}$$

where

$$v^B_{m,l,n} = v^B_l = \frac{1}{4(B+1)^2(2l+1)}.$$

Summarizing, we obtain

$$\begin{aligned} \mathbf{D}\widehat{\mathbf{f}} = \boldsymbol{f} & \quad \text{for the discrete SO(3) Fourier transform,} \\ \boldsymbol{V}_B\mathbf{D}^{\mathrm{H}}\boldsymbol{U}_B\boldsymbol{f} = \widehat{\boldsymbol{f}} & \quad \text{for the adjoint transform,} \end{aligned}$$

see also Corollary 3.7. In Section 4 the accuracy of the NFSOFT is tested by computing a combination of NFSOFT and its adjoint.

Remark 3.9 In our experiments we chose to apply a Clenshaw-Curtis type quadrature rule for arguments β_b of the Wigner-d functions $d^{mn}_l(\beta_b)$ on the closed interval $\beta_b \in [0, \pi]$. In contrast the authors of [16] use a slightly different rule as the authors consider $\beta_b = \frac{(2b+1))\pi}{4B}$ for $b = 0, \ldots, 2B-1$, i.e., on the open interval $\beta_b \in (0, \pi)$ for integrating the Wigner-d functions with weights

$$\tilde{u}^B_l = \frac{2}{B}\sin\left(\frac{\pi(2l+1)}{4B}\right)\sum_{j=0}^{B-1}\frac{\sin((2j+1)(2l+1)\frac{\pi}{4B})}{2l+1}.$$

□

4 Numerical Results

We now present some numerical examples to demonstrate performance and accuracy of the NFSOFT algorithm and certain parts of it. All algorithms mentioned were implemented in C and tested on a 3.00 GHz Intel Xeon$^{\mathrm{TM}}$ processor with 12GB main memory, SuSe-Linux (64 bit), using double precision arithmetic. In addition to that, we used the FFTW 3.2α [9] and NFFT 3.0.3 [14] libraries.
If not mentioned otherwise, we used pseudo-random $SO(3)$ Fourier coefficients $\widehat{f}^{mn}_l$ from the complex square $\left[-\frac{1}{2}, \frac{1}{2}\right] \times \left[-\frac{\mathrm{i}}{2}, \frac{\mathrm{i}}{2}\right]$ and pseudo-random nodes $(\alpha, \beta, \gamma) \in [0, 2\pi) \times [0, \pi] \times [0, 2\pi)$. Whenever involved we invoked the NFFT algorithm [14] with oversampling factor $\rho = 2$ and precomputed Kaiser-Bessel functions.

4.1 The Wigner Transform

First, we examine the Wigner transform itself, i.e., the transform represented by the matrix $\boldsymbol{W}$ from Definition 3.2. We consider three variations of the transform:

1. DWT: the transform corresponding to the matrix $\boldsymbol{W}^{mn}$ as described in Section 3.1, i.e., the transform that converts $SO(3)$ Fourier coefficients into Chebyshev coefficients by the three-term recurrence relation (3.6) using the Clenshaw algorithm.

2. FWT: the same transform as above but using the concepts of the fast polynomial transform to evaluate the modified three-term recurrence relation (3.9) described in Section 3.2. Moreover the stabilization scheme mentioned is applied.

3. FWT (w/o): the same transform as the FWT but without stabilization.

Time requirements: The first test examines the time requirements of the Wigner transform. We computed the vector of Chebyshev coefficients $\boldsymbol{t}^{mn} = (t_0^{mn}, \ldots, t_B^{mn})^T$ from the SO(3) Fourier coefficients $\widehat{\boldsymbol{f}}^{mn} = \left(\widehat{f}^{mn}_{\max(|m|,|n|)}, \ldots, \widehat{f}^{mn}_B\right)^T$ and the matrix $\boldsymbol{W}^{mn} \in \mathbb{C}^{(B+1)\times(B-\max(|m|,|n|))}$ by evaluating $\boldsymbol{t}^{mn} = \boldsymbol{W}^{mn}\widehat{\boldsymbol{f}}^{mn}$ using the FWT algorithm with and without stabilization, and the DWT.
For a fixed bandwidth B, we computed the transform for all possible pairs of orders $|m|, |n| \leq B$. Figure 4.1 shows the average time for one transforms. It shows that without the stabilization the FWT becomes really fast (solid line) compared to the DWT (dotted line). This confirms the different asymptotic complexities of the DWT ($\mathcal{O}(B^2)$) and FWT (w/o) ($\mathcal{O}(B \log^2 B)$) algorithms. That is the good news. Unfortunately, the stabilized FWT (dashed line) does not show the same asymptotical behavior as its unstabilized version. Our numerical tests show that it is slower. As we were not able to determine in advance the number of stabilization steps needed in the FWT, we can not prove the asymptotic complexity of the stabilized FWT, and whether it is closer to the DWT or the FWT (w/o). However, when looking at the absolute times we see that the stabilized FWT is still a large improvement over the DWT as it takes only half the time to compute a transform in average.

Errors: The previous test showed that the time required by the FWT is an improvement over the DWT algorithm, especially when using the FWT (w/o). This next test should shed light on the errors produced during the Wigner transform. Figure 4.2 shows the error

$$E_\infty = \frac{||\boldsymbol{t} - \boldsymbol{t}_{FWT}||_\infty}{||\boldsymbol{t}||_1}$$

between the Chebyshev coefficients $\boldsymbol{t}_{FWT} = (t_l^{mn})_{(l,m,n)\in\mathcal{I}_B}$ computed by the FWT or FWT (w/o) and the coefficients in $\boldsymbol{t}$ computed by the direct algorithm DWT for different bandwidths B. The norms $||\cdot||_1$ and $||\cdot||_\infty$ are the l^p-norms of vectors.
For the tested bandwidths up to $B = 1024$, the error of the stabilized FWT does not exceed 10^{-12}. In contrast the error of the FWT (w/o) grows dramatically losing single precision at about bandwidth $B = 80$.

Stability Issues: In the first experiment we examined the time requirements of the DWT, FWT and FWT (w/o). It showed that the stabilization is costly in terms of

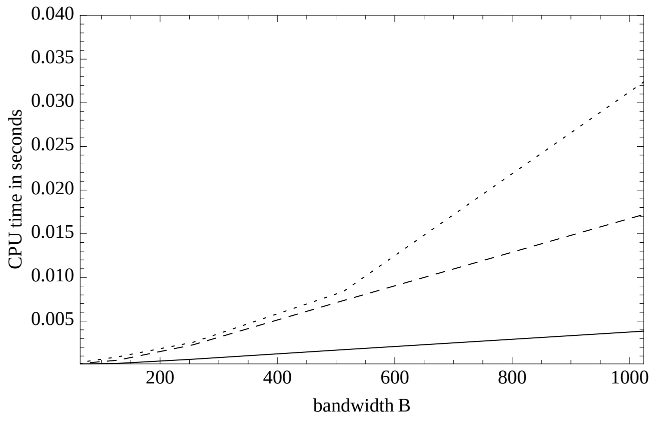


Figure 4.1: For this figure the Wigner transform has been computed for all d_l^{mn} where $(l, m, n) \in \mathcal{I}_B$ occurring in a $SO(3)$ Fourier transform of bandwidth B. The average times for performing one Wigner transform using the FWT (w/o) algorithm (solid), the stabilized FWT (dashed) and the DWT (dotted) are shown as a function of the bandwidth B for all $B = 16, \ldots, 1024$.

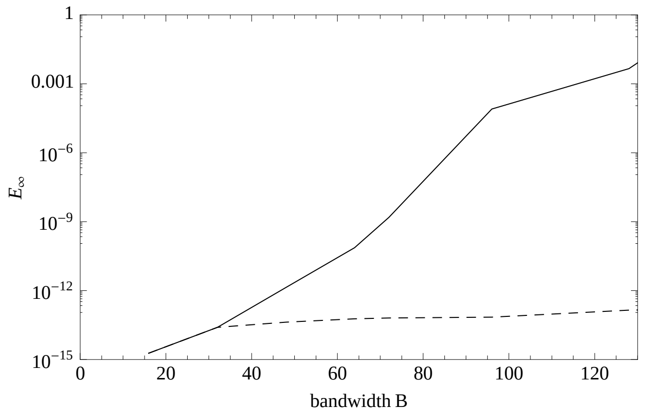


Figure 4.2: For this graph the Wigner transform has been computed for all d_l^{mn} where $(l, m, n) \in \mathcal{I}_B$ occurring in a $SO(3)$ Fourier transform of bandwidth B. The average error occurring in one Wigner transform using the FWT (w/o) algorithm (solid), the stabilized FWT (dashed) are shown as a function of the bandwidth B for all $B = 16, \ldots, 128$.

runtime. On the other hand the second experiment showed that we can not discard stabilizing if we compute with large bandwidths. So in this next test we would like to analyze how often the stabilization scheme is applied within the transform.
Once again, the fast Wigner transform has been computed for all sets of degree l and orders m, n occurring in a $SO(3)$ Fourier transform of bandwidth B, i.e., for a fixed

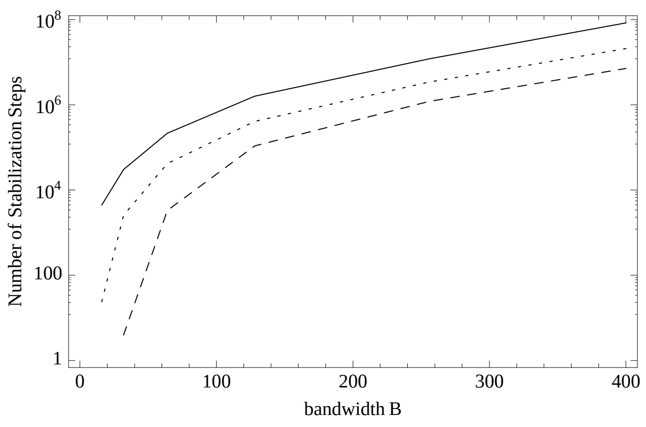


Figure 4.3: The graph shows the number of stabilization steps for all Wigner transforms of orders $|m|, |n| \leq B$ as a function of the bandwidth B for two differently chosen κ along with the upper bound for the number of stabilization steps (solid), i.e., the total number of transform steps in which the algorithm decides whether or not to stabilize. The parameter controlling the accuracy κ is set to $\kappa = 3$ (dotted) and $\kappa = 8$ (dashed).

bandwidth B we computed the transform for all possible pairs of orders $m, n \leq |B|$. We then counted how often the stabilization routine is called. The graph shows the averaged values of the number of stabilization steps. This number does not only depend on the bandwidth B of the transform but also on an internal parameter κ. Whenever the absolute value of an associated Wigner-d function $|d_c^{m,n}(1, l)|$ exceeds κ we use the stabilization scheme. We chose $\kappa = 10^3$ (dotted line) and $\kappa = 10^8$ (dashed line). The solid line represents the total number of steps in the cascade of the polynomial transform and thus the upper bound for the number of stabilization steps.
The graph now shows that by choosing a lower threshold κ, the number of stabilization steps increases, but only by a constant factor. If we compare the two functions with the total number of steps we see they show constant growth as the bandwidth increases. Indeed the relative portion of stabilization steps tends to 8% for $\kappa = 10^8$ and 25% for $\kappa = 10^3$.

4.2 The $SO(3)$ Fourier Transforms

Now we consider the Fourier transform on the whole $SO(3)$. We test our NFSOFT algorithm, (see Algorithm 1), and compare it to the NDSOFT, i.e., the naive evaluation of (3.2). We chose the following three variations of our algorithm:

1. NFSOFT (FWT): the transform described in Algorithm 1 using FWT (see Section 3.2) with stabilization and the NFFT,

2. NFSOFT (DWT): the transform as above but with DWT (see Section 3.1) instead

of the FWT,

3. NDSOFT: the transform as above using the DWT and nonequispaced discrete Fourier transform (NDFT) and thus directly evaluating equation (3.2).

Time requirements: The first test examines the time requirements of the various nonequispaced variations of the $SO(3)$ Fourier transform mentioned above. From the vector of $SO(3)$ Fourier coefficients $\widehat{\boldsymbol{f}} = (\widehat{f}_l^{mn})_{(l,m,n)\in\mathcal{I}_B}$ we compute the vector of function samples $\boldsymbol{f} = (f(\boldsymbol{g}_q))_{\boldsymbol{g}_q\in\mathcal{X}_M}$ of a function $f \in \mathbb{D}_B$ for a sampling set $\mathcal{X}_M$ consisting of pseudo-randomly generated rotations $\boldsymbol{g}_q \in SO(3)$. This was done by evaluating the matrix-vector-product $\boldsymbol{f} = \boldsymbol{D}\widehat{\boldsymbol{f}}$ with the nonequispaced $SO(3)$ Fourier matrix $\boldsymbol{D} = \left(D_l^{mn}(\boldsymbol{g}_q)\right)_{\boldsymbol{g}_q\in\mathcal{X}_M;(l,m,n)\in\mathcal{I}_B}$.
In Figure 4.4 we show the time requirements for the NDSOFT, NFSOFT (FWT) and the NFSOFT (DWT), respectively. The number of nodes is set to $M = B^3$ while we test the algorithms for different bandwidths. We see that the two NFSOFT algorithms outperform the NDSOFT for all bandwidths. Comparing the two NFSOFT versions we see that for bandwidths $B < 64$ they do not show any significant differences. That means the NFFT algorithm dominates the runtime behavior completely. But at bandwidths larger than 64 the NFSOFT (FWT) becomes faster than the NFSOFT(DWT), i.e., the Wigner transform becomes more important. Note that we used the stabilization scheme within the FWT. That verifies that the stabilization does not have major influence on the runtime after all and using the FWT with stabilization is a gain over the slow DWT.

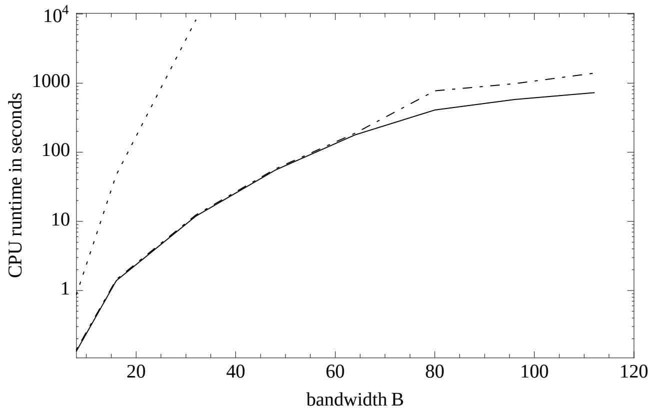


Figure 4.4: The graph shows the runtime of a $SO(3)$ Fourier transform as a function of the bandwidth $B = 16, \ldots, 112$ for the three nonequispaced variations NDSOFT (dotted), NFSOFT(FWT) (solid) and NFSOFT(DWT) (dot-dashed).

The graph of Figure 4.5 shows the runtime as a function of the number of input nodes for $B = 32$ (solid) and $B = 64$ (dashed). We see that up to $M = B^3$ nodes the runtime

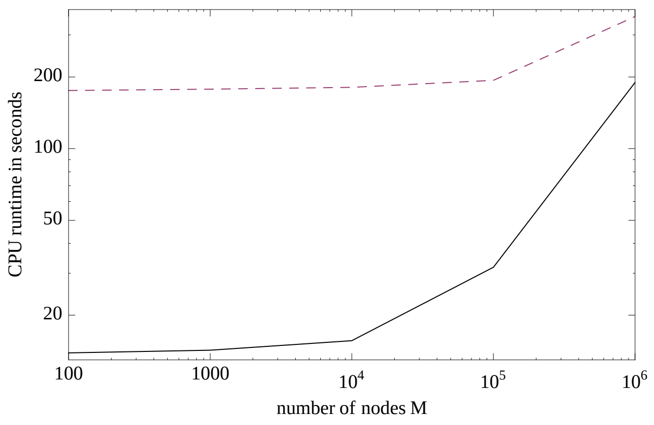


Figure 4.5: In the graph we see the runtime of the NFSOFT(FWT) as a function of the number of input nodes $M = 10^n$ for $n = 2, 3, \ldots, 6$ and different bandwidths $B = 32$ (solid) and $B = 64$ (dashed).

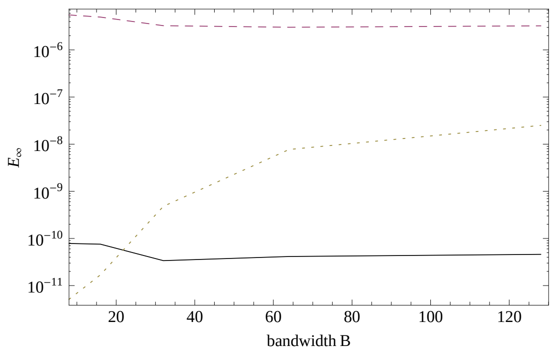
is almost constant, i.e., the bandwidth controls the runtime of the NFSOFT. For larger number of nodes they become the dominant factor over the bandwidth. We then see linear growth of runtime which verifies that the nodes only add linear to the asymptotic complexity. If we computed this figures also for the NFSOFT (DWT) we would not spot a difference in time between the NFSOFT (FWT) here. That is due to the fact that both Wigner transforms, DWT and FWT, are independent of the input nodes (see Section 3.1).

4.3 Equispaced $SO(3)$ Fourier Transforms

We give an example for the results in Section 3.5, i.e., we compute an $SO(3)$ Fourier transform on a given equispaced sampling set, use the Clenshaw-Curtis quadrature from Section 3.5, and test the adjoint algorithm.
We consider the error

$$E_\infty = \frac{||\widehat{\boldsymbol{f}} - \widehat{\tilde{\boldsymbol{f}}}||_\infty}{||\widehat{\boldsymbol{f}}||_1}$$

where the vector $\widehat{\boldsymbol{f}} = (\widehat{f}_l^{mn})_{(l,m,n)\in\mathcal{I}_B}$ contains the original SO(3) Fourier coefficients, and $\widehat{\tilde{\boldsymbol{f}}} = (\widehat{\tilde{f}}_l^{mn})_{(l,m,n)\in\mathcal{I}_B}$ contains the coefficients computed via the fast $SO(3)$ Fourier transform and its adjoint using the Clenshaw-Curtis quadrature rule from (3.22). Figure 4.6 shows this error as a function of the bandwidth. We choose the FWT threshold $\kappa = 10^8$ (dashed line) and $\kappa = 10^3$ (solid line), respectively. The two parameters control the accuracy of the fast Wigner transform. The number of nodes is given by the Clenshaw-Curtis quadrature rule with $M = (2B + 2)^2(2B + 1)$. We compare our results to the ones from the SOFT in [16] (dotted). Note that for this special grid, we



band- width B	equispaced FWT	DWT from [16]
64	5.9e-06	1.9e-05
128	6.2e-05	5.3e-05
256	6.3e-04	4.2e-04
512	1.9e-03	1.3e-03
1024	4.8e-03	5.3e-03

Figure 4.6: The error E_∞ of the fast $SO(3)$ Fourier transform and its adjoint as a function of the bandwidth $B = 16, \ldots, 128$ at equispaced nodes using the quadrature rule as described in Section 3.5 (solid and dashed lines) as well as the error of the SOFT algorithm from [16](dotted line). The table on the right shows the CPU time requirements in seconds of both, the FWT and the Wigner transform from [16], using Wigner-d functions $d_l^{0,0}$ for $l = 0, \ldots, B$.

can replace the NFFT by the FFT.
The graph shows that indeed once we decided for the accuracy controlling parameter κ the error stays at a constant level for our algorithm, i.e., it does not depend on the bandwidths displayed here. That is due to the stabilization scheme we applied here once more. We also see that the error of the SOFT algorithm is slightly bigger than our best case.
After comparing the errors of our equispaced NFSOFT algorithm to the one from [16] we like to mention a comparison in computational time. In the equispaced case from Section 3.5 the difference between both algorithms lies in the evaluation of the sums (3.7). Focussing on this step of the algorithm the table in Figure 4.6 lists the CPU time needed for a Wigner transform either computed by our FWT or by the algorithm from [16] for some bandwidths B. We see that our FWT becomes faster for bandwidths $B \geq 1024$ due to its more favourable complexity but is slower for smaller B as it has larger computational overhead.

4.4 Conclusion

We presented a method for the efficient and accurate calculation of the $SO(3)$ Fourier transform

$$f(\boldsymbol{g}_q) = \sum_{l=0}^{B} \sum_{m=-l}^{l} \sum_{n=-l}^{l} \widehat{f}_l^{mn} D_l^{mn}(\boldsymbol{g}_q), \qquad q = 0, \ldots, M-1,$$

of B-bandlimited functions $f \in \mathbb{D}_B$ at M arbitrary nodes $\boldsymbol{g}_q \in SO(3)$, as well as the adjoint transform.

We showed how to split up the computation of the whole transform

$$\boldsymbol{f} = \boldsymbol{D}\widehat{\boldsymbol{f}} = \boldsymbol{F}\boldsymbol{A}\boldsymbol{W}\widehat{\boldsymbol{f}}$$

into three steps. The new algorithm NFSOFT computes the SO(3) Fourier transforms by means of a trivariate NFFT, represented by the matrix $\boldsymbol{F}$ and a fast Wigner transform, represented by the matrix $\boldsymbol{W}$. Instead of naively $\mathcal{O}(MB^3)$ operations we can compute the NFSOFT in $\mathcal{O}(M + B^3 \log^2 B)$ flops or $\mathcal{O}(M + B^4)$ flops depending on whether we choose the FWT or DWT algorithm. The numerical results show that the NFSOFT outperforms its counterpart, the NDSOFT.

Acknowledgments

D. Potts gratefully acknowledges financial support by Deutsche Forschungsgemeinschaft, grant PO 711/9-1 and A. Vollrath gratefully acknowledges financial support by the Lübeck Graduate School for Computing in Medicine and Life Sciences. Moreover, we like to thank the referees for their valuable suggestions.

References

[1] G. Baszenski and M. Tasche. Fast polynomial multiplication and convolution related to the discrete cosine transform. *Linear Algebra Appl.*, 252:1 – 25, 1997.

[2] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2:363 – 381, 1995.

[3] H. J. Bunge. *Texture Analysis in Material Science.* Butterworths, 1982.

[4] J. E. Castrillon-Candas, V. Siddavanahalli, and C. Bajaj. Nonequispaced Fourier transforms for protein-protein docking. *ICES Report 05-44, Univ. Texas*, 2005.

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

[6] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration (Second Edition).* Academic Press Inc., 1984.

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

[8] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Stat. Comput.*, 14:1368 – 1393, 1993.

[9] M. Frigo and S. G. Johnson. FFTW, C subroutine library. `http://www.fftw.org`, 2005.

[10] M. Gräf and S. Kunis. Stability results for scattered data interpolation on the rotation group. *Electron. Trans. Numer. Anal.*, 31:30 – 39, 2008.

[11] D. Healy, P. Kostelec, S. Moore, and D. Rockmore. FFTs for the 2-sphere - improvements and variations. *J. Fourier Anal. Appl.*, 9:341 – 385, 2003.

[12] R. Hielscher, D. Potts, J. Prestin, H. Schaeben, and M. Schmalz. The Radon transform on SO(3): A Fourier slice theorem and numerical inversion. *Inverse Problems*, 24:025011, 2008.

[13] R. Hielscher, J. Prestin, and A. Vollrath. Fast summation of functions on $SO(3)$. submitted.

[14] J. Keiner, S. Kunis, and D. Potts. NFFT 3.0, C subroutine library. `http://www.tu-chemnitz.de/~potts/nfft`, 2006.

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

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

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

[18] A. Makadia, C. Geyer, S. Sastry, and K. Daniilidis. Radon-based structure from motion without correspondences. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 796–803, Washington, DC, USA, 2005. IEEE Computer Society.

[19] J. D. McEwen, M. P. Hobson, and A. N. Lasenby. A directional continuous wavelet transform on the sphere. *ArXiv*, 2006.

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

[21] D. Potts, G. Steidl, and M. Tasche. Fast algorithms for discrete polynomial transforms. *Math. Comput.*, 67:1577 – 1590, 1998.

[22] D. Potts, G. Steidl, and M. Tasche. Fast and stable algorithms for discrete spherical Fourier transforms. *Linear Algebra Appl.*, 275/276:433 – 450, 1998.

[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. Birkhäuser, Boston, 2001.

[24] D. Potts, G. Steidl, and M. Tasche. Numerical stability of fast trigonometric transforms - a worst case study. *J. Concrete Appl. Math.*, 1:1 – 36, 2003.

[25] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1992.

[26] T. Risbo. Fourier transform summation of Legendre series and D-Functions. *J. Geod.*, 70:383 – 396, 1996.

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

[28] H. Schaeben and K. G. v.d. Boogaart. Spherical harmonics in texture analysis. *Tectonophysics*, 370:253 – 268, 2003.

[29] D. Schmid. Marcinkiewicz-Zygmund inequalities and polynomial approximation from scattered data on $SO(3)$. *Numer. Funct. Anal. Optim.*, 29:855 – 882, 2008.

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

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

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