Fast evaluation of quadrature formulae on the sphere

Jens Keiner* Daniel Potts†

Recently, a fast approximate algorithm for the evaluation of expansions in terms of standard $\mathrm{L}^2\left(\mathbb{S}^2\right)$-orthonormal spherical harmonics at arbitrary nodes on the sphere $\mathbb{S}^2$ has been proposed in [S. Kunis and D. Potts. Fast spherical Fourier algorithms. J. Comput. Appl. Math., 161:75 – 98, 2003]. The aim of this paper is to develop a new fast algorithm for the adjoint problem which can be used to compute expansion coefficients from sampled data by means of quadrature rules.

We give a formulation in matrix-vector notation and an explicit factorisation of the spherical Fourier matrix based on the former algorithm. Starting from this, we obtain the corresponding factorisation of the adjoint spherical Fourier matrix and are able to describe the associated algorithm for the adjoint transformation which can be employed to evaluate quadrature rules for arbitrary weights and nodes on the sphere. We provide results of numerical tests showing the stability of the obtained algorithm using as examples classical Gauß-Legendre and Clenshaw-Curtis quadrature rules as well as the HEALPix pixelation scheme and an equidistribution.

Key words and phrases : two-sphere, quadrature, nonequispaced fast spherical Fourier transform, NFFT, FFT

2000 AMS Mathematics Subject Classification : 65T99, 33C55, 42C10, 65T50

1 Introduction

Discrete Fourier analysis on a multi-dimensional torus plays an important role in a wide range of applications, among them signal processing in general, image processing, computed tomography, and a lot more. However, in many fields of interest, data naturally arises on a geometry that can be identified with the surface of the two-dimensonal unit-sphere – two-sphere in short – $\mathbb{S}^2 := \left\{\mathbf{x} \in \mathbb{R}^3 : \ \|\mathbf{x}\|_2 = 1\right\}$, embedded into the three-dimensional euclidean space $\mathbb{R}^3$. As a small indication of the impact on current research we mention here the solution

*Institute of Mathematics, University of Lübeck, Wallstraße 40, 23560 Lübeck, Germany, E-Mail: keiner@math.uni-luebeck.de

†Department of Mathematics, Chemnitz University of Technology, Reichenhainer Straße 39, 09107 Chemnitz, Germany, E-Mail: potts@mathematik.tu-chemnitz.de

of inverse problems arising in astrophysics or the solution of systems of differential equations in weather forecast computation as related examples.

In the context of Fourier analysis on the sphere $\mathbb{S}^2$, the spherical analogue of the usual Fourier basis functions $\mathrm{e}^{\mathrm{i}kx}$ in $\mathrm{L}^2([0,2\pi))$, namely the standard orthonormal spherical harmonics Y_k^n in $\mathrm{L}^2(\mathbb{S}^2)$ play a fundamental role. Recently, a fast algorithm for evaluating a function $f \in \mathrm{L}^2(\mathbb{S}^2)$ with finite orthogonal expansion

$$f(\vartheta,\varphi) = \sum_{k=0}^{M} \sum_{n=-k}^{k} a_k^n Y_k^n(\vartheta,\varphi) \quad (M \in \mathbb{N}_0) \tag{1.1}$$

in terms of spherical harmonics Y_k^n on a set of arbitary nodes (ϑ_d, φ_d) with $d = 1,\ldots,D$, $D \in \mathbb{N}$, in spherical coordinates was given in [9]. The key idea is to first perform a change of basis such that the function f in (1.1) takes the form

$$f(\vartheta,\varphi) = \sum_{n=-M}^{M} \sum_{k=-M}^{M} c_k^n \mathrm{e}^{\mathrm{i}k\vartheta} \mathrm{e}^{\mathrm{i}n\varphi} \tag{1.2}$$

of an ordinary two-dimensional Fourier sum with new complex coefficients c_k^n. Then, the evaluation of the function f can be performed using the fast Fourier transform for nonequispaced nodes (NFFT; see for example [14, 20]).

In this paper, we are intrested in the *adjoint problem*, i.e. the fast evaluation of sums

$$\tilde{a}_k^n := \sum_{d=1}^{D} f(\vartheta_d,\varphi_d) \overline{Y_k^n(\vartheta_d,\varphi_d)} \tag{1.3}$$

for given function values $f(\vartheta_d,\varphi_d) \in \mathbb{C}$ and all indices $k = 0,\ldots,M$ and $n = -k,\ldots,k$. Note that this usually does not yet recover the coefficients a_k^n from (1.1) for which we denote the computed coefficients $\tilde{a}_k^n$. The coefficients a_k^n can be obtained from values of the function f on a set of arbitrary nodes (ϑ_d,φ_d) provided that a quadrature rule with weights w_d and sufficient high degree of exactness is available (see also [5, 10]). Then the sum in (1.3) changes to

$$a_k^n = \sum_{d=1}^{D} w_d f(\vartheta_d,\varphi_d) \overline{Y_k^n(\vartheta_d,\varphi_d)}. \tag{1.4}$$

The computation of spherical Fourier coefficients from discrete sampled data has major importance in the whole field of data analysis on the sphere $\mathbb{S}^2$. In many applications however, the distribution of the available data on the sphere is predetermined by the underlying measurement process or as well by data storage and access considerations. This often requires the use of techniques like spherical hyperinterpolation ([16]) or approximate quadrature rules that differ from classical quadrature formulae. The implementation of the algorithm for the adjoint problem (1.3) developed in this paper provides for the first time a means of evaluating quadrature formulae for arbitrary nodes in a fast way and thus allows for the efficient use of new quadrature schemes.

The outline of this paper is as follows: Section 2 introduces basic notation and definitions, and recalls the fast algorithm for evaluating the expansion (1.1) from [9]. We give a matrix-vector formulation of the algorithm where we distinguish the initial change of basis to arrive at (1.2) and the application of the NFFT algorithm. In Section 3, we describe the change of basis by means of a fast polynomial transform in more detail. In Section 4, following the algorithm, the factorisation of the corresponding transform matrix into a product of sparse matrices is derived. Consequently, once obtained a fast algorithm for (1.1), a fast algorithm for the adjoint problem (1.3) comes by taking the adjoint matrix product in Section 5. Finally, Section 6 provides results of numerical tests showing properties of the described algorithm by using a range of different test functions and quadrature formulae.

2 Discrete spherical Fourier transforms

This section reviews basic notation, definitions, and the fast algorithm for (1.1) from [9].

2.1 Fourier analysis on the sphere $\mathbb{S}^2$

In spherical coordinates we identify each point $\mathbf{x} \in \mathbb{S}^2$ with a tuple $(\vartheta, \varphi) \in [0, \pi] \times [0, 2\pi)$ of two angles ϑ and φ. The space $\mathrm{L}^2\left(\mathbb{S}^2\right)$ is the Hilbert space of square integrable functions on the sphere $\mathbb{S}^2$ with the usual inner product given by

$$\langle f, g\rangle_{\mathrm{L}^2(\mathbb{S}^2)} := \int_0^{\pi} \int_0^{2\pi} f(\vartheta, \varphi)\overline{g(\vartheta, \varphi)}\, \mathrm{d}\varphi\, \sin\vartheta\, \mathrm{d}\vartheta.$$

With the standard orthonormal basis of spherical harmonics Y_k^n with indices $k = 0, 1, \ldots$ and $n = -k, \ldots, k$ for the space $\mathrm{L}^2\left(\mathbb{S}^2\right)$, any function f from $\mathrm{L}^2\left(\mathbb{S}^2\right)$ can be developed into a generally infinite orthogonal expansion

$$f(\vartheta, \varphi) = \sum_{k=0}^{\infty} \sum_{n=-k}^{k} a_k^n Y_k^n(\vartheta, \varphi). \tag{2.1}$$

The functions Y_k^n are harmonic homogeneous polynomials of degree k and are defined by

$$Y_k^n : \mathbb{S}^2 \to \mathbb{C}, \quad Y_k^n(\vartheta, \varphi) := \sqrt{\frac{2k+1}{4\pi}} P_k^{|n|}(\cos\vartheta)\mathrm{e}^{\mathrm{i}n\varphi}. \tag{2.2}$$

2.2 Associated Legendre functions and polynomials

The functions $P_k^{|n|}$ are the associated Legendre functions,

$$P_k^n : [-1, 1] \to \mathbb{R}, \quad P_k^n(x) := \left(\frac{(k-n)!}{(k+n)!}\right)^{1/2} \left(1 - x^2\right)^{n/2} \frac{\mathrm{d}^n}{\mathrm{d}x^n} P_k(x) \tag{2.3}$$

for $n = 0, 1, \ldots$ and $k = n, n+1, \ldots$, where the classical Legendre polynomials P_k are given by their Rodrigues formula

$$P_k : [-1, 1] \to \mathbb{R}, \quad P_k(x) := \frac{1}{2^k k!} \frac{\mathrm{d}^k}{\mathrm{d}x^k} \left(x^2 - 1\right)^k.$$

For a concise notation, we let $P_{-1}(x) := 0$. The associated Legendre functions P_k^n have the three-term recurrence relation

$$P_{k+1}^n(x) = v_k^n x P_k^n(x) + w_k^n P_{k-1}^n(x) \quad (k \geq n) \tag{2.4}$$

with initial values

$$P_{n-1}^n(x) := 0, \quad P_n^n(x) = \lambda_n \left(1-x^2\right)^{n/2}, \quad \lambda_n := \frac{\sqrt{(2n)!}}{2^n n!}$$

and the coefficients

$$v_k^n := \frac{2k+1}{((k-n+1)(k+n+1))^{1/2}}, \qquad w_k^n := -\frac{((k-n)(k+n))^{1/2}}{((k-n+1)(k+n+1))^{1/2}}. \tag{2.5}$$

A simple but at the same time powerful idea is to define the associated Legendre functions P_k^n also for $k = 0, \ldots, n$ by means of the modified three-term recurrence relation

$$P_{k+1}^n(x) = \left(\alpha_k^n x + \beta_k^n\right) P_k^n(x) + \gamma_k^n P_{k-1}^n(x) \tag{2.6}$$

for $k \geq 0$ with

$$\begin{aligned} \alpha_0^n &:= \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{if } n \text{ odd}, \\ -1 & \text{if } n \text{ even}, n \neq 0, \end{cases} & \alpha_k^n &:= \begin{cases} (-1)^{k+1} & \text{if } 0 < k < n, \\ v_k^n & \text{if } n \leq k, \end{cases} \\ \beta_k^n &:= \begin{cases} 1 & \text{if } 0 \leq k < n, \\ 0 & \text{if } n \leq k, \end{cases} & \gamma_k^n &:= \begin{cases} 0 & \text{if } 0 \leq k < n, \\ w_k^n & \text{if } n \leq k. \end{cases} \end{aligned} \tag{2.7}$$

Here, we let $P_{-1}^n(x) := 0$, and $P_0^n(x) := \lambda_n$ for even n and $P_0^n(x) := \lambda_n \left(1-x^2\right)^{1/2}$ for odd n. For $k \geq n$, this definition coincides with the recurrence (2.4). As easily verified by the defining equation (2.3), P_k^n is a polynomial of degree k if n is even, while $(1-x^2)^{-1/2} P_k^n$ is a polynomial of degree $k-1$ if n is odd. Based on the recurrence coefficients from (2.7) and introducing a shift parameter $c \in \mathbb{N}_0$, we define the associated Legendre polynomials $P_k^n(\,\cdot\,, c)$ by

$$\begin{aligned} &P_{-1}^n(x,c) := 0, \; P_0^n(x,c) := 1, \\ &P_{k+1}^n(x,c) = \left(\alpha_{k+c}^n x + \beta_{k+c}^n\right) P_k^n(x,c) + \gamma_{k+c}^n P_{k-1}^n(x,c) \quad (k \geq 0). \end{aligned} \tag{2.8}$$

It is not difficult to prove the following lemma by a straightforward induction. ([1]):

Lemma 2.1. *Let $c, k, n \geq 0$ and let the functions P_k^n and $P_k^n(\,\cdot\,, c)$ be given as in* (2.6), (2.7), *and* (2.8). *Then we have*

$$P_{k+c}^n(x) = P_c^n(x,k) P_k^n(x) + \gamma_k^n P_{c-1}^n(x, k+1) P_{k-1}^n(x).$$

2.3 Discrete Fourier transforms on the sphere $\mathbb{S}^2$

We recall that our goal in this section is the evaluation of a finite expansion

$$f(\vartheta, \varphi) = \sum_{k=0}^{M} \sum_{n=-k}^{k} a_k^n Y_k^n(\vartheta, \varphi) = \sum_{(k,n) \in \mathcal{I}_M} a_k^n Y_k^n(\vartheta, \varphi). \tag{2.9}$$

of a function f in terms of spherical harmonics Y_k^n, where $\mathcal{I}_M$ denotes the index set

$$\mathcal{I}_M := \{(k,n) \,:\, k = 0, \ldots, M; \; n = -k, \ldots, k\}.$$

The sum in (2.9) is the spherical Fourier sum of the function f and $M \in \mathbb{N}_0$ is called the bandwidth of f. Likewise, the function f is said to be a bandlimited function on the sphere $\mathbb{S}^2$ with bandwidth M. The complex expansion coefficients $a_k^n \in \mathbb{C}$ are the spherical Fourier coefficients of the function f. The index k denotes the degree and n is the order with respect to the orthonormal basis of spherical harmonics Y_k^n. Alluding to the geographic coordinate system, the angles ϑ are called co-latitudes and the angles φ longitudes. A set $\mathcal{X} := \{(\vartheta_d, \varphi_d)\}_{d=1}^D$, $D \in \mathbb{N}$, of arbitrary nodes on the sphere $\mathbb{S}^2$ is called a sampling set.

2.4 Matrix-vector notation

From a linear algebra point of view, evaluating the bandlimited function f with bandwidth M as in (2.9) on a sampling set $\mathcal{X}$ amounts to evaluating matrix-vector product

$$\mathbf{f}_{\mathcal{X}} = \mathbf{Y}_{M,\mathcal{X}}\, \mathbf{a}_M$$

with

$$\begin{aligned} \mathbf{f}_{\mathcal{X}} &:= (f_d)_{d=1}^D \in \mathbb{C}^D, \;\; f_d := f\left(\vartheta_d, \varphi_d\right), \\ \mathbf{Y}_{M,\mathcal{X}} &:= \left(Y_k^n\left(\vartheta_d, \varphi_d\right)\right)_{d=1,\ldots,D;(k,n)\in\mathcal{I}_M} \in \mathbb{C}^{D\times(M+1)^2}, \\ \mathbf{a}_M &:= \left(a_k^n\right)_{(k,n)\in\mathcal{I}_M} \in \mathbb{C}^{(M+1)^2}. \end{aligned}$$

We write $\mathbf{f}_{\mathcal{X}}$, $\mathbf{Y}_{M,\mathcal{X}}$ and $\mathbf{a}_M$ to emphasise the dependence of these quantities on the concrete sampling set $\mathcal{X}$ and the bandwidth M. Furthermore, we introduce the notation

$$\mathbf{a}_M^n := (a_k^n)_{k=|n|}^M \in \mathbb{C}^{M-|n|+1}$$

with $n = -M, \ldots, M$ for subvectors $\mathbf{a}_M^n$ of $\mathbf{a}_M$, each containing the spherical Fourier coefficients a_k^n for a fixed order n.

2.5 A fast Fourier transform algorithm for arbitrary nodes on the sphere $\mathbb{S}^2$

We are now ready to describe the fast evaluation of a bandlimited function f with bandwidth M on an arbitrary sampling set $\mathcal{X}$ on the sphere $\mathbb{S}^2$, or equivalently, the fast evaluation of the matrix-vector product $\mathbf{f}_{\mathcal{X}} = \mathbf{Y}_{M,\mathcal{X}}\, \mathbf{a}_M$. First, we rearrange terms in the spherical Fourier sum and obtain

$$\begin{aligned} f(\vartheta,\varphi) &= \sum_{k=0}^{M}\sum_{n=-k}^{k} a_k^n Y_k^n(\vartheta,\varphi) = \sum_{n=-M}^{M}\sum_{k=|n|}^{M} a_k^n Y_k^n(\vartheta,\varphi) \\ &= \sum_{n=-M}^{M} \underbrace{\left(\sum_{k=|n|}^{M} a_k^n \sqrt{\frac{2k+1}{4\pi}} P_k^{|n|}(\cos\vartheta)\right)}_{:=h_n(\cos\vartheta)} \mathrm{e}^{\mathrm{i}n\varphi}, \end{aligned} \tag{2.10}$$

Then we consider polynomials of degree M

$$g_n(x) := \sum_{k=|n|}^{M} a_k^n P_k^{|n|}(x) \tag{2.11}$$

for even n, and of degree $M-1$

$$g_n(x) := (1-x^2)^{-1/2} \sum_{k=|n|}^{M} a_k^n P_k^{|n|}(x) \tag{2.12}$$

for odd n with the given complex spherical Fourier coefficients a_k^n. It is not difficult to verify that

$$h_n(\cos\vartheta) = \begin{cases} g_n(\cos\vartheta) & \text{if} \quad n \text{ even}, \\ (\sin\vartheta) g_n(\cos\vartheta) & \text{if} \quad n \text{ odd}. \end{cases}$$

We perform a change of basis via a fast polynomial transform (see [12]) and obtain $h_n(\cos\vartheta)$ in Chebyshev representation

$$h_n(\cos\vartheta) = \begin{cases} \displaystyle\sum_{k=0}^{M} b_k^n T_k(\cos\vartheta), & \text{if } n \text{ even}, \\ \displaystyle(\sin\vartheta) \sum_{k=0}^{M-1} b_k^n T_k(\cos\vartheta), & \text{if } n \text{ odd}, \end{cases} \tag{2.13}$$

with the Chebyshev polynomials of the first kind $T_k : [-1,1] \to \mathbb{R}$, $T_k(x) := \cos(k \arccos x)$. Since $T_k(\cos\vartheta) = \cos(k\vartheta)$, we can write

$$h_n(\cos\vartheta) = \begin{cases} \displaystyle\sum_{k=0}^{M} b_k^n \cos(k\vartheta), & \text{if } n \text{ even}, \\ \displaystyle(\sin\vartheta) \sum_{k=0}^{M-1} b_k^n \cos(k\vartheta), & \text{if } n \text{ odd}. \end{cases}$$

In addition, we use $\cos(k\vartheta) = \frac{1}{2}\left(\mathrm{e}^{\mathrm{i}k\vartheta} + \mathrm{e}^{-\mathrm{i}k\vartheta}\right)$ and arrive at

$$h_n(\cos\vartheta) = \begin{cases} \displaystyle\sum_{k=-M}^{M} \tilde{b}_k^n \mathrm{e}^{\mathrm{i}k\vartheta}, & \text{if } n \text{ even}, \\ \displaystyle(\sin\vartheta) \sum_{k=-(M-1)}^{M-1} \tilde{b}_k^n \mathrm{e}^{\mathrm{i}k\vartheta}, & \text{if } n \text{ odd}, \end{cases}$$

where we let

$$\tilde{b}_k^n := \begin{cases} b_0^n & \text{if } k = 0, \\ \frac{1}{2} b_{|k|}^n & \text{if } 0 < |k| \le M. \end{cases} \tag{2.14}$$

In view of that $\sin\vartheta = \frac{1}{2\mathrm{i}}\left(\mathrm{e}^{\mathrm{i}\vartheta} - \mathrm{e}^{-\mathrm{i}\vartheta}\right)$, a last manipulation finally yields the representation

$$h_n(\cos\vartheta) = \sum_{k=-M}^{M} c_k^n \mathrm{e}^{\mathrm{i}k\vartheta}, \tag{2.15}$$

with the coefficients

$$c_k^n := \begin{cases} \tilde{b}_k^n, & \text{if } n \text{ even}, \\ -\frac{1}{2\mathrm{i}} \tilde{b}_{k+1}^n, & \text{if } n \text{ odd and } k = -M, -M+1, \\ \frac{1}{2\mathrm{i}} \tilde{b}_{k-1}^n, & \text{if } n \text{ odd and } k = M, M-1, \\ \frac{1}{2\mathrm{i}} (\tilde{b}_{k-1}^n - \tilde{b}_{k+1}^n), & \text{if } n \text{ odd and } k = -M+2, \ldots, M-2. \end{cases} \tag{2.16}$$

For the function f in (2.10), this now yields the ordinary two-dimensional Fourier sum

$$f(\vartheta, \varphi) = \sum_{n=-M}^{M} \sum_{k=-M}^{M} c_k^n \mathrm{e}^{\mathrm{i}k\vartheta} \mathrm{e}^{\mathrm{i}n\varphi}. \tag{2.17}$$

The double sum in (2.17) can be evaluated by the NFFT algorithm on an arbitrary sampling set $\mathcal{X}$ with D nodes with $\mathcal{O}\left(M^2 \log M + \log^2(1/\varepsilon) D\right)$ floating point operations (flops), where ε is a prescribed accuracy. In matrix-vector notation, the complete procedure reads

$$\mathbf{f}_{\mathcal{X}} = \mathbf{F}_{M,\mathcal{X}} \, \mathbf{C}_M \, \mathbf{B}_M \, \mathbf{a}_M. \tag{2.18}$$

Here, the block-diagonal matrix $\mathbf{B}_M$ with blocks $\mathbf{B}_M^n$, $n = -M, \ldots, M$, on its main diagonal stands for the fast polynomial transform algorithm acting with block $\mathbf{B}_M^n$ on the subvector $\mathbf{a}_M^n$ of $\mathbf{a}_M$ which comprises the Fourier coefficients a_k^n for a fixed order n. The matrix $\mathbf{C}_M$ represents the intermediate steps to convert the Chebyshev coefficients b_k^n in (2.13) into the final Fourier coefficients c_k^n in (2.15), and the matrix $\mathbf{F}_{M,\mathcal{X}}$ realises the evaluation of the ordinary Fourier sum in (2.17). The matrices $\mathbf{C}_M$ and $\mathbf{B}_M$ do not depend on the sampling set $\mathcal{X}$ but only on the bandwidth M.

Let us state the entries of these matrices. We have

$$\mathbf{F}_{M,\mathcal{X}} := \left(\mathrm{e}^{\mathrm{i}(k\vartheta_d + n\varphi_d)}\right)_{d=1,\ldots,D;(k,n)\in\mathcal{I}_M} \in \mathbb{C}^{D\times(2M+1)^2}$$

and according to (2.14) and (2.16)

$$\begin{aligned}
\mathbf{B}_M &:= \operatorname{diag}\left[\mathbf{B}_M^n\right]_{n=-M}^{M} && \in \mathbb{R}^{(2M+1)(M+1)\times(M+1)^2}, \\
\mathbf{B}_M^n & && \in \mathbb{R}^{(M+1)\times(M-|n|+1)}, \\
\mathbf{C}_M &:= \mathbf{C}_{M,2}\,\mathbf{C}_{M,1} && \in \mathbb{R}^{(2M+1)^2\times(2M+1)(M+1)}, \\
\mathbf{C}_{M,1} &:= \mathbf{I}_{2M+1} \otimes \tilde{\mathbf{C}}_{M,1} && \in \mathbb{R}^{(2M+1)^2\times(2M+1)(M+1)}, \\
\tilde{\mathbf{C}}_{M,1} &:= \begin{pmatrix} & & & \frac{1}{2} \\ & & \iddots & \\ & \frac{1}{2} & & \\ \frac{1}{2} & & & \\ 1 & & & \\ \frac{1}{2} & & & \\ & \frac{1}{2} & & \\ & & \ddots & \\ & & & \frac{1}{2} \end{pmatrix} && \in \mathbb{R}^{(2M+1)\times(M+1)}, \\
\mathbf{C}_{M,2} &:= \operatorname{diag}\left(\left(\delta_{0,n \bmod 2}\right)_{n=-M}^{M}\right) \otimes \tilde{\mathbf{C}}_{M,2} && \in \mathbb{R}^{(2M+1)^2\times(2M+1)^2}, \\
\tilde{\mathbf{C}}_{M,2} &:= \operatorname{tridiag}\left[(-\mathrm{i}/2)_{l=0}^{2M-1}, (0)_{l=0}^{2n}, (\mathrm{i}/2)_{l=0}^{2M-1}\right] && \in \mathbb{R}^{(2M+1)\times(2M+1)}.
\end{aligned} \tag{2.19}$$

As usual, $\delta_{j,k}$ denotes the Kronecker delta function for $j, k \in \mathbb{Z}$. The matrices $\mathbf{B}_M$ and $\mathbf{C}_M$ are real matrices acting on complex vectors. The matrix $\mathbf{C}_M$ decomposes into a product of two matrices $\mathbf{C}_{M,1}$ and $\mathbf{C}_{M,2}$, where $\mathbf{C}_{M,1}$ represents step (2.14) and $\mathbf{C}_{M,2}$ stands for (2.16).

The adjoint counterpart of the method just described is obtained from the factorisation of the spherical Fourier matrix $\mathbf{Y}_{M,\mathcal{X}}$ in the matrix-vector product (2.18) and reads

$$\tilde{\mathbf{a}}_M = \mathbf{B}_M^{\mathrm{T}}\, \mathbf{C}_M^{\mathrm{T}}\, \mathbf{F}_{M,\mathcal{X}}^{\mathrm{H}}\, \mathbf{f}_{\mathcal{X}}. \tag{2.20}$$

This represents the adjoint Fourir sum in (1.3) in matrix-vector notation. Again, we use tildes to emphasize that the adjoint transform in general does not yield the spherical Fourier coefficients a_k^n. The multiplication with the matrix $\mathbf{F}_{\mathcal{X}}^{\mathrm{H}}$, which is identical to the evaluation of sums

$$\tilde{c}_k^n := \sum_{d=1}^{D} f\left(\vartheta_d, \varphi_d\right) \mathrm{e}^{-\mathrm{i}k\vartheta_d} \mathrm{e}^{-\mathrm{i}n\varphi_d}$$

for indices $(k,n) \in \mathcal{I}_M$, can also be carried out by the NFFT as described in [14]. The transposed algorithm for the intermediate steps, hence the multiplication with the matrix $\mathbf{C}_M^{\mathrm{T}}$, can be derived directly from the explicit representation in (2.19). The details of the transposed polynomial transform algorithm, or, equivalently, the multiplication with the matrix $\mathbf{B}_M^{\mathrm{T}}$, are derived in Section 5 after the matrix $\mathbf{B}_M$ has been further decomposed in Section 4. The final adjoint algorithm is called adjoint non-uniform fast spherical Fourier transform (adjoint NFSFT).

3 Fast Legendre function transform

In this section, we review a fast algorithm for the transformation of sums

$$g^n(x) = \begin{cases} \sum_{k=|n|}^{M} a_k^n P_k^n(x), & \text{if } n \text{ even}, \\ (1-x^2)^{-1/2} \sum_{k=n}^{M} a_k^n P_k^n(x), & \text{if } n \text{ odd} \end{cases}$$

of associated Legendre functions P_k^n into their respective Chebyshev representation

$$g^n(x) = \sum_{k=0}^{M} b_k^n T_k^n(x)$$

in terms of Chebyshev polynomials of the first kind T_k. This algorithm is called a fast Legendre function transform (FLFT).

3.1 Existing algorithms

The first originating paper [4] is due to Driscoll and Healy describing an exact algorithm for the transposed problem, i.e. projecting a sum of Chebyshev polynomials T_k onto associated Legendre functions P_k^n. In [13], an exact algorithm is derived, introducing for the first time a stabilisation technique to compensate for errors due to finite precision arithmetic. Both, the Driscoll-Healy algorithm for the transposed problem and the latter algorithm in its initial version are subject to numerical instabilities. In [7], Healy, Kostelec, Moore, and Rockmore show variations of the Driscoll-Healy algorithm also including stabilization techniques.

Mohlenkamp ([11]) introduces the first related approximate algorithm. Further approximate algorithms are developed by Suda and Takami ([17]) using a stabilisation method relying on fast multipole methods and optimised interpolation nodes. Rokhlin and Tygert ([15]) exploit a relation to semi-separable matrices. We follow the lines in [13] and [9].

3.2 An exact algorithm

Let now $M \geq 4$, $t := \lceil \log_2 M \rceil$, and $N := 2^t$. Furthermore, set $a_k^n := 0$ for $0 \leq k < |n|$ and $M < k \leq N$. In the first step, we define $\tilde{n} := \min(|n|, N-2)$ and $\tilde{M} := \min(M, N-1)$. We can write

$$g^n = \sum_{k=\tilde{n}}^{\tilde{M}} a_{k,0}^n P_k^n \tag{3.1}$$

with the polynomials

$$\begin{aligned} a_{k,0}^n(x) &:= a_k^n \qquad\qquad (k = 0, \ldots, N-3), \\ a_{N-2,0}^n(x) &:= a_{N-2}^n + \gamma_{N-1}^n a_N^n, \\ a_{N-1,0}^n(x) &:= a_{N-1}^n + \left(\alpha_{N-1}^n x + \beta_{N-1}^n\right) a_N^n. \end{aligned} \tag{3.2}$$

of degree at most one. By grouping terms we arrive at

$$g^n = \sum_{l=\left\lfloor \frac{\tilde{n}}{4} \right\rfloor}^{\left\lceil \frac{\tilde{M}+1}{4} \right\rceil - 1} \left(\sum_{k=0}^{3} a_{4l+k,0}^n P_{4l+k}^n \right), \tag{3.3}$$

which is a partition of the sum (3.1) into blocks of four consecutive summands. We note that we immediately know the Chebyshev representation of each of the polynomials $a_{k,0}^n$ involved which is due to the fact that these polynomials have degree at most one, and that $T_0(x) = 1$ and $T_1(x) = x$. The generalised recurrence in Lemma 2.1 implies

$$\begin{pmatrix} P_{k+c}^n \\ P_{k+c+1}^n \end{pmatrix} = \mathbf{U}_c^n(\cdot, k)^{\mathrm{T}} \begin{pmatrix} P_{k-1}^n \\ P_k^n \end{pmatrix}, \tag{3.4}$$

where

$$\mathbf{U}_c^n(\cdot, k)^{\mathrm{T}} := \begin{pmatrix} \gamma_k^n P_{c-1}^n(\cdot, k+1) & \gamma_k^n P_c^n(\cdot, k+1) \\ P_c^n(\cdot, k) & P_{c+1}^n(\cdot, k) \end{pmatrix}. \tag{3.5}$$

From the matrix-vector form (3.4) and with $c = 1$ and $k = l + 1$ it follows that

$$\left(P_{4l+2}^n, P_{4l+3}^n\right) \begin{pmatrix} a_{4l+2,0}^n \\ a_{4l+3,0}^n \end{pmatrix} = \left(P_{4l}^n, P_{4l+1}^n\right) \mathbf{U}_1^n\left(\cdot, 4l+1\right) \begin{pmatrix} a_{4l+2,0}^n \\ a_{4l+3,0}^n \end{pmatrix}.$$

This lets us rewrite the sum (3.3) as

$$g^n = \sum_{l=\left\lfloor \frac{\tilde{n}}{4} \right\rfloor}^{\left\lceil \frac{\tilde{M}+1}{4} \right\rceil - 1} \left(a_{4l,1}^n P_{4l}^n + a_{4l+1,1}^n P_{4l+1}^n \right)$$

with new polynomials of degree at most three,

$$\begin{pmatrix} a_{4l,1}^n \\ a_{4l+1,1}^n \end{pmatrix} := \begin{pmatrix} a_{4l,0}^n \\ a_{4l+1,0}^n \end{pmatrix} + \mathbf{U}_1^n\left(\cdot, 4l+1\right) \begin{pmatrix} a_{4l+2,0}^n \\ a_{4l+3,0}^n \end{pmatrix}. \tag{3.6}$$

We can use a fast cosine transform to compute the polynomial products in (3.6) and obtain the polynomials $a_{4l,1}^n$ and $a_{4l+1,1}^n$ in Chebyshev representation again. Applying this idea repeatedly

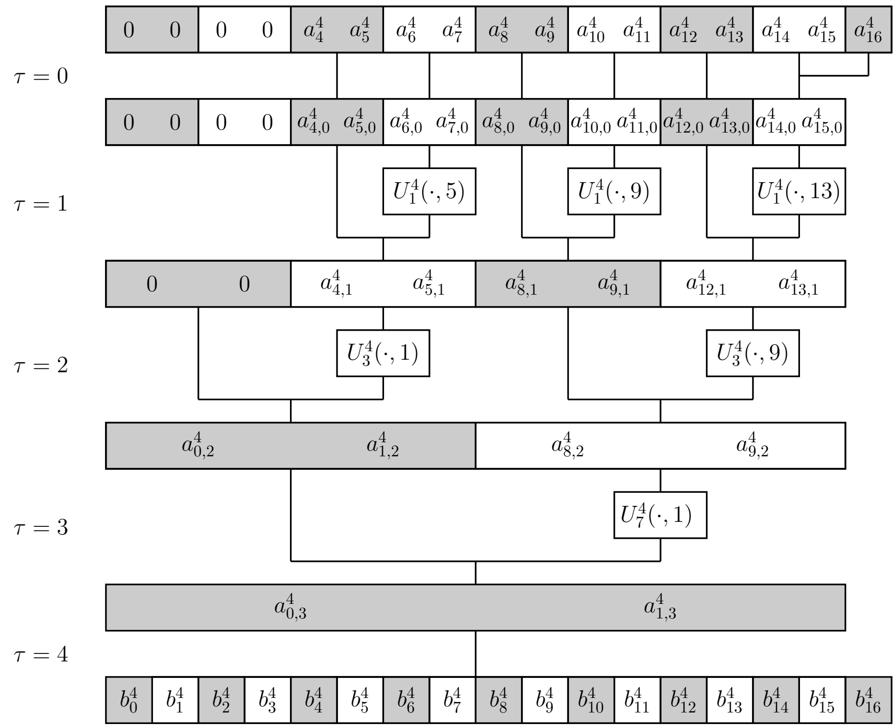

Figure 3.1: Schematic representation of the FLFT cascade summation for $M = 16$ ($t = 4$) and $n = 4$. The steps $\tau = 0$ and $\tau = 4$ are the first and the last step, respectively.

leads to a cascade summation scheme as illustrated in Figure 3.1. For $\tau = 1, \dots, t-1$, we compute

$$\begin{pmatrix} a^n_{2^{\tau+1}l,\tau} \\ a^n_{2^{\tau+1}l+1,\tau} \end{pmatrix} = \begin{pmatrix} a^n_{2^{\tau+1}l,\tau-1} \\ a^n_{2^{\tau+1}l+1,\tau-1} \end{pmatrix} + \mathbf{U}^n_{2^\tau-1}\left(\cdot, 2^{\tau+1}l+1\right) \begin{pmatrix} a^n_{2^{\tau+1}l+2^\tau,\tau-1} \\ a^n_{2^{\tau+1}l+2^\tau+1,\tau-1} \end{pmatrix} \tag{3.7}$$

for $l = \lfloor \tilde{n}/2^{\tau+1} \rfloor, \dots, \lceil (\tilde{M}+1)/2^{\tau+1} \rceil - 1$. It is not difficult to see that $a^n_{2^{\tau+1}l,\tau} = a^n_{2^{\tau+1}l+1,\tau} = 0$ for larger or smaller l. After step $\tau = t-1$, we arrive at $g^n = a^n_{0,t-1}P^n_0 + a^n_{1,t-1}P^n_1$, where we have the Chebyshev coefficients $b_{0,k}$ and $b_{1,k}$ for $k = 0, \dots, N-1$ of the polynomials $a^n_{0,t-1}$ and $a^n_{1,t-1}$ with degree at most $N-1$, respectively. Since

$$\lambda_n = \begin{cases} P^n_0(x), & \text{if } n \text{ even}, \\ (1-x^2)^{-1/2} P^n_0(x), & \text{if } n \text{ odd}, \end{cases} \qquad P^n_1(x) = \left(\alpha^n_0 x + \beta^n_0\right) P^n_0(x),$$

we have

$$g^n(x) = \lambda_n \left(a^n_{0,t-1}(x) + \left(\alpha^n_0 x + \beta^n_0\right) a^n_{1,t-1}(x) \right). \tag{3.8}$$

Now, using $x\,T_0(x) = x = T_1(x)$ and $x\,T_k(x) = \frac{1}{2}\left(T_{k+1}(x) + T_{k-1}(x)\right)$ for $k \geq 1$, we finally obtain in the last step the sought Chebyshev coefficients b^n_k for $k = 0, \dots, M$ if n even or $k = 0, \dots, M-1$ if n odd of the polynomial $g^n(x)$ of degree at most M or $M-1$, respectively, by computing

$$
\begin{aligned}
b_0^n &= \lambda_n \left(b_{0,0} + \beta_0^n b_{1,0} + \frac{\alpha_0^n}{2} b_{1,1} \right), \\
b_1^n &= \lambda_n \left(b_{0,1} + \beta_0^n b_{1,1} + \alpha_0^n \left(b_{1,0} + \frac{b_{1,2}}{2} \right) \right), \\
b_k^n &= \lambda_n \left(b_{0,k} + \beta_0^n b_{1,k} + \frac{\alpha_0^n}{2} \left(b_{1,k-1} + b_{1,k+1} \right) \right) \quad (k = 2, \ldots, N-2), \\
b_{N-1}^n &= \lambda_n \left(b_{0,N-1} + \beta_0^n b_{1,N-1} + \frac{\alpha_0^n}{2} b_{1,N-2} \right), \\
b_N^n &= \lambda_n \frac{\alpha_0^n}{2} b_{1,N-1}.
\end{aligned}
\tag{3.9}
$$

In total, the FLFT algorithm consists of $t + 1 = \mathcal{O}(\log M)$ steps. The first step ($\tau = 0$) and the last step ($\tau = t$) clearly have a complexity of $\mathcal{O}(M)$ flops. The rest is the cascade summation where each step has a complexity of $\mathcal{O}(M \log M)$ flops due to the DCT applications used for the multiplication with the matrices $\mathbf{U}_{2^\tau-1}^n \left(\cdot, 2^{\tau+1} l + 1 \right)$. In total, this accumulates to $\mathcal{O}\left(M \log^2 M \right)$ flops for the whole algorithm.

3.3 Stabilisation

The described algorithm is only exact in exact arithmetic. Unfortunately, it becomes unstable in finite precision arithmetic for $|n| > 16$ ([13]). The computation is subject to numerical instabilities owing to small errors introduced by the DCT algorithm that get multiplied by large function values of the associated Legendre polynomials $P_k^n(x, c)$ for certain admissible triples (k, n, c) and $|x| \approx 1$ ([9]). A simple but effective idea (see [13]) is to replace the ordinary multiplication steps by so-called 'stabilisation' steps, whenever the values $P_k^n(x, c)$ exceed a certain threshold $\kappa > 0$. The multiplication with the matrix $\mathbf{U}_{2^{\tau+1}}^n \left(\cdot, 2^{\tau+1} l + 1 \right)$ is replaced by a multiplication with the matrix $\mathbf{U}_{2^\tau(2l+1)-1}^n(\cdot, 1)$ fulfilling

$$
\begin{pmatrix} P_{2^{\tau+1}l+2^\tau}^n \\ P_{2^{\tau+1}l+2^\tau+1}^n \end{pmatrix} = \mathbf{U}_{2^\tau(2l+1)-1}^n \left(\cdot, 1 \right)^{\mathrm{T}} \begin{pmatrix} P_0^n \\ P_1^n \end{pmatrix}.
$$

This is nothing else than taking the polynomials $a_{2^{\tau+1}l+2^\tau,\tau-1}^n$ and $a_{2^{\tau+1}l+2^\tau+1,\tau-1}^n$ out of the cascade and updating

$$
\begin{pmatrix} a_{0,t-1}^n \\ a_{1,t-1}^n \end{pmatrix} := \begin{pmatrix} a_{0,t-1}^n \\ a_{1,t-1}^n \end{pmatrix} + \mathbf{U}_{2^\tau(2l+1)-1}^n \left(\cdot, 1 \right) \begin{pmatrix} a_{2^{\tau+1}l+2^\tau,\tau-1}^n \\ a_{2^{\tau+1}l+2^\tau+1,\tau-1}^n \end{pmatrix} \tag{3.10}
$$

after the cascade summation is completed. These 'stabilisation' steps are costly compared to the cascade summation without stabilisation. We sacrifice runtime efficiency for accuracy. Unfortunately, an upper bound for the number

$$
s_n := \# \left\{ (k, n, c) : \ (k, n, c) \text{ is admissible and } P_k^n(\cdot, c) \text{ exceeds the threshold } \kappa \right\}
$$

of stabilisation steps with respect to M for a given threshold κ is not known. If $s_n = \mathcal{O}(\log M)$, we would still have an $\mathcal{O}\left(M \log^2 M \right)$ algorithm. Figure 3.2 shows time measurements supporting that this might be a reasonable conjecture. Figure 3.3 shows accuracy measurements

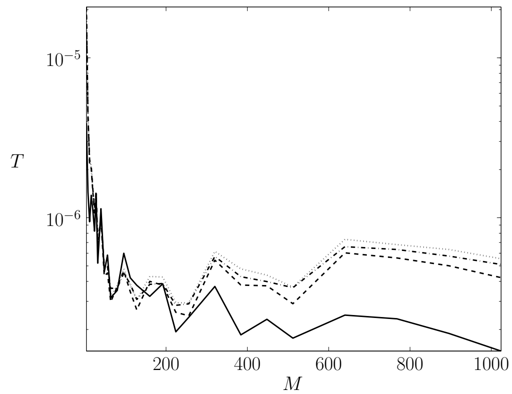

Figure 3.2: The time $T := \frac{\tilde{T}}{M \log^2 M}$ in seconds, with $\tilde{T}$ being the time for a single fast Legendre function transform, as a function of the bandwidth M up to $M = 1024$ and orders $n = 0$ (solid), $n = \frac{M}{4}$ (dashed), $n = \frac{M}{2}$ (dashed-dotted), and $n = \frac{3M}{4}$ (dotted). The time measurements have been averaged over 10.000 single transforms for each bandwidth M support an estimate of the arithmetic complexity of $\mathcal{O}(M \log^2 M)$.

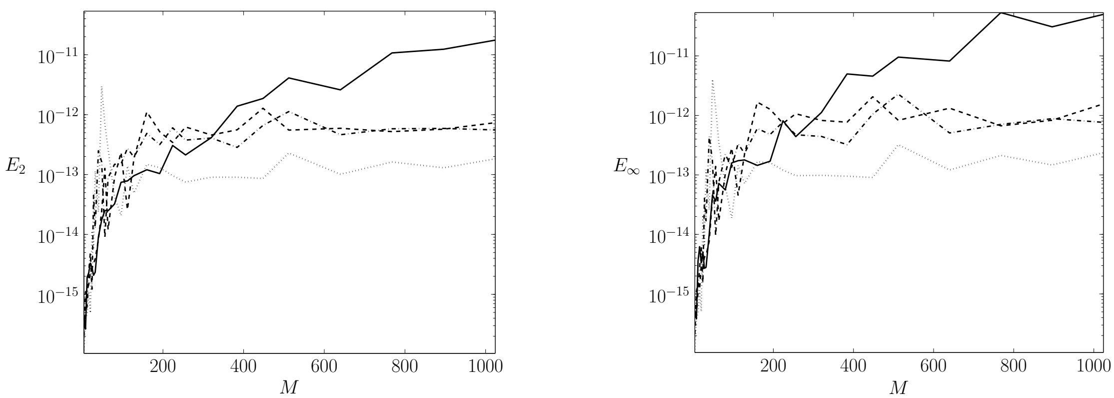

Figure 3.3: The errors E_2 (left) and E_∞ (right) from (3.11) for single fast Legendre function transforms as a function of the bandwidth M up to $M = 1024$ and orders $n = 0$ (solid), $n = \frac{M}{4}$ (dashed), $n = \frac{M}{2}$ (dashed-dotted), and $n = \frac{3M}{4}$ (dotted), respectively. We used uniformly distributed pseudo-random coefficients a_k^n from $\left[-\frac{1}{2}, \frac{1}{2}\right] \times \mathrm{i} \left[-\frac{1}{2}, \frac{1}{2}\right]$. Reference values where calculated with Mathematica 5.0 using arbitrary precision arithmetic up to accuracy $\varepsilon = 10^{-32}$.

with respect to the relative error measures

$$E_\infty := \frac{\|\mathbf{g} - \tilde{\mathbf{g}}\|_\infty}{\|\mathbf{g}\|_\infty}, \qquad E_2 := \frac{\|\mathbf{g} - \tilde{\mathbf{g}}\|_2}{\|\mathbf{g}\|_2}, \tag{3.11}$$

where the vectors $\mathbf{g}$ and $\tilde{\mathbf{g}}$ represent a sum

$$g^n(\cos\vartheta) = \sum_{k=|n|}^{M} a_k^n P_k^{|n|}(\cos\vartheta)$$

evaluated at the $M+1$ Chebyshev nodes $\vartheta_l = \pi(2l+1)/(2(M+1))$, $l = 0, \ldots, M$ using Mathematica 5.0 with arbitrary precision arithmetic for $\mathbf{g}$, and the stabilized FLFT algorithm followed by a DCT-III for $\tilde{\mathbf{g}}$, respectively.

4 A factorization of the matrix B_M^n

In this section, we represent the FLFT algorithm as a linear mapping acting on a vector of coefficients $\mathbf{a}_M^n = (a_k^n)_{k=|n|}^M \in \mathbb{C}^{M-|n|+1}$ for fixed order n with $|n| \leq M$. For the sake of simplicity, we restrict ourselves to the case when M is a power of two, hence $M = N$. For values of M no being a power of two, we set $a_k^n := 0$ for $M < k \leq N$. Furthermore, we let $a_k^n := 0$ for $0 \leq k < |n|$ as before and use the extended vector

$$\mathbf{a}_M^n := (a_k^n)_{k=0}^N \in \mathbb{C}^{N+1}$$

We also omit the stabilisation procedure from Section 3.3 here and do not exploit the fact that $a_k^n = 0$ for $0 \leq k < n$ or $M < k \leq N$ in the following.

As seen before in Section 2.4, the FLFT algorithm can be represented as a matrix $\mathbf{B}_M^n \in \mathbb{R}^{(N+1)\times(N+1)}$ that multiplied with the vector $\mathbf{a}_M^n$ results in a vector $\mathbf{b}_M^n := (b_0^n, b_1^n, \ldots, b_N^n)^{\mathrm{T}} \in \mathbb{C}^N$ containing the Chebyshev coefficients b_k^n of the polynomial $g^n(x)$ from (2.11) or (2.12), admitting the representation

$$g^n(x) = \sum_{k=0}^{N} b_k^n T_k(x)$$

In matrix-vector notation, this reads

$$\mathbf{b}_M^n = \mathbf{B}_M^n \ \mathbf{a}_M^n.$$

The FLFT algorithm implies a factorisation of the matrix $\mathbf{B}_M^n$ into a product of sparse matrices. We will use this fact later on in Section 5 to obtain an algorithm for the transposed problem, i.e. computing a matrix-vector product of the form $\tilde{\mathbf{a}}_{M,n} = {\mathbf{B}_M^n}^{\mathrm{T}}\, \tilde{\mathbf{b}}_{M,n}$.

In general, the FLFT algorithm consists of $t+1$ steps, such that the matrix $\mathbf{B}_M^n$ can be decomposed into the matrix product

$$\mathbf{B}_M^n = \mathbf{B}_{M,t}^n \ \cdot \ \mathbf{B}_{M,t-1}^n \ \cdot \ \ldots \ \cdot \ \mathbf{B}_{M,1}^n \ \cdot \ \mathbf{B}_{M,0}^n, \tag{4.1}$$

with the matrices

$$\mathbf{B}_{M,\tau}^n \in \begin{cases} \mathbb{R}^{2N\times(N+1)} & \text{if} \quad \tau = 0, \\ \mathbb{R}^{2N\times 2N} & \text{if} \quad 1 \leq \tau < t, \\ \mathbb{R}^{(N+1)\times 2N} & \text{if} \quad \tau = t. \end{cases}$$

4.1 The First Step

The first step converts the spherical Fourier coefficents a_k^n, $k = 0, \ldots, N$, into polynomials $a_{k,0}^n$, $k = 0, \ldots, N-1$ of degree at most one in *Chebyshev representation*, i.e. vectors $\mathbf{a}_{k,0}^n \in \mathbb{C}^2$ containing the coefficients of their expansions in terms of the Chebyshev polynomials T_k for $k = 0, 1$. In matrix-vector notation, this reads

$$\mathbf{a}_{M,0}^n = \mathbf{B}_{M,0}^n \, \mathbf{a}_M^n$$

where the result vector

$$\mathbf{a}^n_{M,0} := \begin{bmatrix} \tilde{\mathbf{a}}^n_{0,0} \\ \vdots \\ \tilde{\mathbf{a}}^n_{N-1,0} \end{bmatrix} \in \mathbb{C}^{2N}$$

is a vector of length $2N$. For $k = 0, \dots, N-3$ we have

$$\tilde{\mathbf{a}}^n_{k,0} = \mathbf{e}\, a^n_k, \quad \text{with } \mathbf{e} := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

since these new polynomials remain constants. The last polynomial $a^n_{N,0}$ is mapped onto the preceeding ones, $a^n_{N-1,0}$ and $a^n_{N-2,0}$, by means of the three-term recurrence relation (2.6), i.e.

$$a^n_{N,0} = \left(\alpha^n_{N-1} x + \beta^n_{N-1}\right) a^n_{N-1,0} + \gamma^n_{N-1} a^n_{N-2,0}.$$

Consequently, the matrix $\mathbf{B}^n_{M,0}$ can be written as

$$\mathbf{B}^n_{M,0} = \left[\mathbf{I}_N \otimes \mathbf{e},\ \tilde{\mathbf{e}}\right], \tag{4.2}$$

with $\tilde{\mathbf{e}} := \left(0, 0, \dots, 0, \gamma^n_{N-1}, 0, \beta^n_{N-1}, \alpha^n_{N-1}\right)^{\mathrm{T}} \in \mathbb{R}^{2N}$.

4.2 Steps $\tau = 1, \dots, t-1$ (Cascade Summation)

These steps represent the cascade summation applied to a sum of associated Legendre functions P^n_k as in Figure 3.1. Notice that after the first step, we have N polynomials $a^n_{k,0}$ for $k = 0, \dots, N-1$, where N is a power of two. In each step now following, half the functions P^n_k are eliminated by mapping the polynomials $a^n_{k,0}$ in front of them onto the remaining ones, employing the three-term recurrence. The input of step τ, the vector $\mathbf{a}^n_{M,\tau-1}$, containing the Chebyshev coefficients of the polynomials $a^n_{2^{\tau+1}l+j,\tau-1}$ as vectors $\tilde{\mathbf{a}}^n_{2^{\tau+1}l+j,\tau-1}$ and $j = 0, 1, 2^\tau, 2^\tau+1$, is grouped by the idnex j into consecutive blocks $\hat{\mathbf{a}}^n_{l,\tau-1}$ of four polynomials for $l = 0, \dots, N/2^{\tau+1} - 1$, hence

$$\mathbf{a}^n_{M,\tau-1} := \begin{bmatrix} \hat{\mathbf{a}}^n_{0,\tau-1} \\ \vdots \\ \hat{\mathbf{a}}^n_{\frac{N}{2^{\tau+1}}-1,\tau-1} \end{bmatrix} \in \mathbb{C}^{2N}, \qquad \hat{\mathbf{a}}^n_{l,\tau-1} := \begin{bmatrix} \tilde{\mathbf{a}}^n_{2^{\tau+1}l,\tau-1} \\ \tilde{\mathbf{a}}^n_{2^{\tau+1}l+1,\tau-1} \\ \tilde{\mathbf{a}}^n_{2^{\tau+1}l+2^\tau,\tau-1} \\ \tilde{\mathbf{a}}^n_{2^{\tau+1}l+2^\tau+1,\tau-1} \end{bmatrix} \in \mathbb{C}^{2^{\tau+2}}$$

In every block $\hat{\mathbf{a}}^n_{l,\tau-1}$, the first and the second polynomial, $a^n_{2^{\tau+1}l,\tau-1}$ and $a^n_{2^{\tau+1}l+1,\tau-1}$ remain unchanged. The third and the fourth polynomial, $a^n_{2^{\tau+1}l+2^\tau,\tau-1}$ and $a^n_{2^{\tau+1}l+2^\tau+1,\tau-1}$, get multiplied by the matrix $\mathbf{U}^n_{2^\tau-1}\left(\cdot, 2^{\tau+1}l+1\right)$ and the result is added to $a^n_{2^{\tau+1}l,\tau-1}$ and $a^n_{2^{\tau+1}l+1,\tau-1}$ yielding $a^n_{2^{\tau+1}l,\tau}$ and $a^n_{2^{\tau+1}l+1,\tau}$.

The output vector $\mathbf{a}^n_{M,\tau}$ of this step contains only half as many polynomials as before, but due to the multiplication with the matrix $\mathbf{U}^n_{2^\tau-1}\left(\cdot, 2^{\tau+1}l+1\right)$, the degree of each polynomial nearly doubles each time so that twice the space is needed to store the Chebyshev coefficients, hence the vectors $\tilde{\mathbf{a}}^n_{2^{\tau+1}l,\tau}$ and $\tilde{\mathbf{a}}^n_{2^{\tau+1}l+1,\tau}$. In total, the result vector $\mathbf{a}^n_{M,\tau}$ still has length $2N$.

Reviewing this polynomial multiplication and addition scheme, we need to keep the first two polynomials, but with their vectors zero-padded to twice the length. Furthermore, we have to add the vectors of Chebyshev coefficients due to the multiplication of the third and fourth polynomial with the matrix $\mathbf{U}^n_{2^\tau-1}\left(\cdot, 2^{\tau+1}l+1\right)$. By writing these steps now as

$$\begin{bmatrix} \tilde{\mathbf{a}}^n_{2^{\tau+1}l,\tau} \\ \tilde{\mathbf{a}}^n_{2^{\tau+1}l+1,\tau} \end{bmatrix} = \mathbf{V}^n_{l,\tau} \begin{bmatrix} \tilde{\mathbf{a}}^n_{2^{\tau+1}l,\tau-1} \\ \tilde{\mathbf{a}}^n_{2^{\tau+1}l+1,\tau-1} \\ \tilde{\mathbf{a}}^n_{2^{\tau+1}l+2^\tau,\tau-1} \\ \tilde{\mathbf{a}}^n_{2^{\tau+1}l+2^\tau+1,\tau-1} \end{bmatrix} = \mathbf{V}^n_{l,\tau}\, \hat{\mathbf{a}}^n_{l,\tau-1}$$

we introduce the matrices

$$\begin{aligned} \mathbf{V}^n_{l,\tau} &:= [\mathbf{Z}_\tau, \mathbf{U}_{l,\tau}] && \in \mathbb{R}^{2^{\tau+2}\times 2^{\tau+2}}, \\ \mathbf{Z}_\tau &:= \begin{bmatrix} \mathbf{I}_{2^\tau} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{2^\tau} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} && \in \mathbb{R}^{2^{\tau+2}\times 2^{\tau+1}}, \\ \mathbf{U}^n_{l,\tau} & && \in \mathbb{R}^{2^{\tau+2}\times 2^{\tau+1}} \end{aligned}$$

where the matrix $\mathbf{U}^n_{l,\tau}$ representing the multiplication of the third and fourth polynomial with the matrix $\mathbf{U}^n_{2^\tau-1}\left(\cdot, 2^{\tau+1}l+1\right)$ can be further factorised as:

$$\mathbf{U}^n_{l,\tau} := \mathbf{D}_{\mathrm{II},\tau}\, \mathbf{S}_\tau\, \mathbf{P}^n_{l,\tau}\, \mathbf{D}_{\mathrm{III},\tau}, \tag{4.3}$$

defining the matrices

$$\begin{aligned} \mathbf{D}_{\mathrm{II},\tau} &:= \mathbf{I}_2 \otimes (\mathbf{D}_{2^{\tau+1}}\mathbf{C}_{2^{\tau+1}}) && \in \mathbb{R}^{2^{\tau+2}\times 2^{\tau+2}}, \\ \mathbf{S}_\tau &:= \mathbf{I}_2 \otimes \left[\ \mathbf{I}_{2^{\tau+1}}, \mathbf{I}_{2^{\tau+1}}\ \right] && \in \mathbb{R}^{2^{\tau+2}\times 2^{\tau+3}}, \\ \mathbf{P}^n_{l,\tau} &:= \operatorname{diag}\big(\big[\gamma^n_{2^{\tau+1}l+1}\mathbf{P}^n_{2^\tau-2}(2^{\tau+1}l+1+1), \gamma^n_{2^{\tau+1}l+1}\mathbf{P}^n_{2^\tau-1}(2^{\tau+1}l+1+1), \\ &\qquad \mathbf{P}^n_{2^\tau-1}(2^{\tau+1}l+1), \mathbf{P}^n_{2^\tau}(2^{\tau+1}l+1)\big]\big) && \in \mathbb{R}^{2^{\tau+3}\times 2^{\tau+3}}, \\ \mathbf{D}_{\mathrm{III},\tau} &:= \mathbf{I}_2 \otimes \left(\left(\mathbf{I}_2 \otimes \mathbf{C}^{\mathrm{T}}_{2^{\tau+1}}\right)\mathbf{Z}_\tau\right) && \in \mathbb{R}^{2^{\tau+3}\times 2^{\tau+1}}, \end{aligned}$$

and using the DCT related matrices $\mathbf{D}_{\mathbf{2}^{\tau+1}}$ and $\mathbf{C}_{\mathbf{2}^{\tau+1}}$ given by

$$\mathbf{C}_N := \left(\cos\frac{j(2k+1)\pi}{2N}\right)_{j,k=0}^{N-1}, \quad \mathbf{D}_N := \operatorname{diag}\left((\varepsilon_j)_{j=0}^{N-1}\right) \quad (N \in \mathbb{N}),$$

with $\varepsilon_0 := \frac{1}{2}$ and $\varepsilon_j := 1$ for $j = 1, \ldots, N-1$.

The matrix $\mathbf{D}_{\mathrm{III},\tau}$ realizes the zero-padding (see the definition of $\mathbf{Z}_\tau$ from above) of the two polynomials $a^n_{2^{\tau+1}l+2^\tau,\tau-1}$ and $a^n_{2^{\tau+1}l+2^\tau+1,\tau-1}$, second, the evaluation at the Chebyshev nodes $\cos\left(\frac{(2j+1)\pi}{2^{\tau+2}}\right)$ for $j = 0, \ldots, 2^{\tau+1}-1$ by the DCT-III related matrix $\mathbf{C}^{\mathrm{T}}_{2^{\tau+1}}$, and finally, a duplication of the resulting vector to allow for multiplication with two different associated Legendre polynomials. Combined, the zero-padding and evaluation are an interpolation of function values of the polynomials at additional nodes.

The diagonal matrix $\mathbf{P}^n_{l,\tau}$ contains the values of the associated Legendre polynomials in the matrix $\mathbf{U}^n_{2^\tau-1}\left(\cdot, 2^{\tau+1}l+1\right)$ Chebyshev nodes $\cos\left((2j+1)\pi/2^{\tau+2}\right)$ for $j = 0, \ldots, 2^\tau - 1$ on its main diagonal. It represents a pointwise multiplication with the interpolated polynomial function values of $a^n_{2^{\tau+1}l+2^\tau,\tau-1}$ and $a^n_{2^{\tau+1}l+2^\tau+1,\tau-1}$.

The matrix $\mathbf{S}^{(\tau)}$ forms the sums for the two rows of the result, and finally, the matrix $\mathbf{D}_{\mathrm{II},\tau}$ transforms the newly obtained polynomials $a^n_{2^{\tau+1}l.\tau}$ and $a^n_{2^{\tau+1}l+1,\tau}$ back into Chebyshev representation by applying the DCT-II related matrix product $\mathbf{D}_{2^{\tau+1}}\,\mathbf{C}_{2^{\tau+1}}$ to each vector of polynomial function values.

From the factorisation in (4.3), the compact representation

$$\mathbf{U}^n_{l,\tau} = \begin{pmatrix} \mathbf{D}_{2^{\tau+1}}\mathbf{C}_{2^{\tau+1}} & \gamma^n_c & (& \mathbf{P}^n_{2^\tau-2}(c+1)\mathbf{C}^{\mathrm{T}}_{2^{\tau+1}}\mathbf{Z}_{1,\tau} & + & \mathbf{P}^n_{2^\tau-1}(c+1)\mathbf{C}^{\mathrm{T}}_{2^{\tau+1}}\mathbf{Z}_{2,\tau} \;) \\ \mathbf{D}_{2^{\tau+1}}\mathbf{C}_{2^{\tau+1}} & & (& \mathbf{P}^n_{2^\tau-1}(c)\mathbf{C}^{\mathrm{T}}_{2^{\tau+1}}\mathbf{Z}_{1,\tau} & + & \mathbf{P}^n_{2^\tau}(c)\mathbf{C}^{\mathrm{T}}_{2^{\tau+1}}\mathbf{Z}_{2,\tau} \;) \end{pmatrix}$$

with $c = 2^{\tau+1}l + 1$ and

$$\mathbf{Z}_{1,\tau} := \begin{pmatrix} \mathbf{I}_{2^\tau} & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{2^{\tau+1}\times 2^{\tau+1}},\ \mathbf{Z}_{2,\tau} := \begin{pmatrix} 0 & \mathbf{I}_{2^\tau} \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{2^{\tau+1}\times 2^{\tau+1}}$$

is obtained.

The whole step $\mathbf{T}^{(\tau)}_n$ can then be represented as

$$\begin{aligned} \mathbf{a}^n_{M,\tau} &= \mathbf{B}^n_{M,\tau}\,\mathbf{a}^n_{M,\tau-1}, \\ \mathbf{B}^n_{M,\tau} &:= \begin{bmatrix} \mathbf{V}^n_{0,\tau} & & & \\ & \mathbf{V}^n_{1,\tau} & & \\ & & \ddots & \\ & & & \mathbf{V}^n_{2^{t-\tau-1}-1,\tau} \end{bmatrix}. \end{aligned} \tag{4.4}$$

4.3 The Last Step

The last step consists in obtaining the polynomial $g^n = a^n_{0,t-1}P^n_0 + a^n_{1,t-1}P^n_1$ in Chebyshev representation, i.e. the vector $\mathbf{b}^n_M = (b^n_k)^N_{k=0} \in \mathbb{C}^{N+1}$. Using (3.9) we write

$$\mathbf{g}^n_M = \lambda_n\left(\tilde{\mathbf{I}}_N\mathbf{a}^n_{0,t-1} + \left(\alpha^n_0\mathbf{W}_N + \beta^n_0\tilde{\mathbf{I}}_N\right)\mathbf{a}^n_{1,t-1}\right).$$

Here, we have defined

$$\tilde{\mathbf{I}}_k := \begin{bmatrix} \mathbf{I}_N \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(N+1)\times N}, \qquad \mathbf{W}_N := \begin{pmatrix} 0 & \frac{1}{2} & & & \\ 1 & 0 & \frac{1}{2} & & \\ & \frac{1}{2} & 0 & \ddots & \\ & & \ddots & \ddots & \frac{1}{2} \\ & & & \frac{1}{2} & 0 \\ & & & & \frac{1}{2} \end{pmatrix} \in \mathbb{R}^{(N+1)\times N}.$$

The last step is computing

$$\mathbf{b}^n_M = \mathbf{B}^n_{M,t}\,\mathbf{a}^n_{M,t-1}$$

where, depending on n, we distinguish three cases:

i) If n is odd we have $\alpha^n_0 = 0$, $\beta^n_0 = 1$, and

$$\mathbf{B}^n_{M,t} = \lambda_n\left[\tilde{\mathbf{I}}_N, \tilde{\mathbf{I}}_N\right] \in \mathbb{R}^{(N+1)\times 2N}. \tag{4.5}$$

ii) For $n = 0$ holds $\alpha_0^n = 1$, $\beta_0^n = 0$, and we find

$$\mathbf{B}_{M,t}^n = \lambda_n \left[\tilde{\mathbf{I}}_{N+1}, \mathbf{W}_N\right] \in \mathbb{R}^{(N+1)\times 2N}. \tag{4.6}$$

iii) Finally, if n is even and $n \neq 0$, one verifies $\alpha_0^n = -1$ and $\beta_0^n = 1$ resulting in

$$\mathbf{B}_{M,t}^n = \lambda_n \left[\tilde{\mathbf{I}}_N, \tilde{\mathbf{I}}_N - \mathbf{W}_N\right] \in \mathbb{R}^{(N+1)\times 2N}. \tag{4.7}$$

5 Transposed Fast Legendre Function Transform

After having analysed the FLFT algorithm in more detail by factorising the associated matrix, the *transposed fast Legendre Function transform (transposed FLFT)* reads

$$\tilde{\mathbf{a}}_M^n = {\mathbf{B}_M^n}^{\mathrm{T}}\, \tilde{\mathbf{b}}_M^n \tag{5.1}$$

with the vectors $\tilde{\mathbf{b}}_M^n := \left(\tilde{b}_k^n\right)_{k=0}^N \in \mathbb{C}^{N+1}$ and $\tilde{\mathbf{a}}_M^n := (\tilde{a}_k^n)_{k=0}^N \in \mathbb{C}^{N+1}$. Notice that we are only interested in the coefficients $\tilde{a}_k^n$ for $|n| \le k \le M$ afterwards.

Following the lines of the preceeding sections, in particular the formulas in (4.1), (4.2), (4.4), and (4.5) – (4.7), we immediately obtain a factorisation of the transposed matrix ${\mathbf{B}_M^n}^{\mathrm{T}}$ as

$${\mathbf{B}_M^n}^{\mathrm{T}} = {\mathbf{B}_{M,0}^n}^{\mathrm{T}} \cdot {\mathbf{B}_{M,1}^n}^{\mathrm{T}} \cdot \ldots \cdot {\mathbf{B}_{M,t-1}^n}^{\mathrm{T}} \cdot {\mathbf{B}_{M,t}^n}^{\mathrm{T}}. \tag{5.2}$$

For $\tau = 0$ and $\tau = t$, we have immediatly

$${\mathbf{B}_{M,0}^n}^{\mathrm{T}} = \begin{pmatrix} \mathbf{I}_N \otimes \mathbf{e}_1^{\mathrm{T}} \\ \tilde{\mathbf{e}}^{\mathrm{T}} \end{pmatrix}, \qquad {\mathbf{B}_{M,t}^n}^{\mathrm{T}} = \lambda_n \begin{cases} \begin{bmatrix} \tilde{\mathbf{I}}_N \\ \tilde{\mathbf{I}}_N \end{bmatrix} & \text{if} \quad \text{if } n \text{ odd}, \\ \begin{bmatrix} \tilde{\mathbf{I}}_N \\ \mathbf{W}_N^{\mathrm{T}} \end{bmatrix} & \text{if} \quad \text{if } n = 0, \\ \begin{bmatrix} \tilde{\mathbf{I}}_N \\ \tilde{\mathbf{I}}_N - \mathbf{W}_N^{\mathrm{T}} \end{bmatrix} & \text{if} \quad \text{if } n \text{ even}, n \neq 0. \end{cases} \tag{5.3}$$

For the rest of the steps, i.e. the 'transposed' cascade summation for $\tau = t-1, \ldots, 1$, we have

$$\begin{aligned}
{\mathbf{B}_{M,\tau}^n}^{\mathrm{T}} &= \begin{bmatrix} {\mathbf{V}_{0,\tau}^n}^{\mathrm{T}} & & & \\ & {\mathbf{V}_{1,\tau}^n}^{\mathrm{T}} & & \\ & & \ddots & \\ & & & {\mathbf{V}_{2^{t-\tau-1}-1,\tau}^n}^{\mathrm{T}} \end{bmatrix}, \\
{\mathbf{V}_{l,\tau}^n}^{\mathrm{T}} &= \begin{bmatrix} {\mathbf{Z}_\tau}^{\mathrm{T}}, \\ {\mathbf{U}_{l,\tau}^n}^{\mathrm{T}} \end{bmatrix}, \\
{\mathbf{U}_{l,\tau}^n}^{\mathrm{T}} &= \left[\gamma_{2^{\tau+1}l+1}^n \left(\mathbf{Z}_1^{\mathrm{T}} \mathbf{C}_{2^{\tau+1}} \mathbf{P}_{2^\tau-2}^n(2^{\tau+1}l+1+1) + \mathbf{Z}_2^T \mathbf{C}_{2^{\tau+1}} \mathbf{P}_{2^\tau-1}^n(2^{\tau+1}l+1+1)\right) \mathbf{C}_{2^{\tau+1}}^{\mathrm{T}}, \right. \\
&\qquad \left. \left(\mathbf{Z}_1^{\mathrm{T}} \mathbf{C}_{2^{\tau+1}} \mathbf{P}_{2^\tau-1}^n(2^{\tau+1}l+1) + \mathbf{Z}_2^{\mathrm{T}} \mathbf{C}_{2^{\tau+1}} \mathbf{P}_{2^\tau}^n(2^{\tau+1}l+1)\right) \mathbf{C}_{2^{\tau+1}}^{\mathrm{T}}\right].
\end{aligned} \tag{5.4}$$

Following this decomposition, a fast algorithm for computing the transposed FLFT, hence computing the matrix-vector product $\tilde{\mathbf{a}}_M = {\mathbf{B}_M^n}^{\mathrm{T}} \tilde{\mathbf{b}}_M$ can be obtained immediately.

At each level $\tau = t-1, \ldots, 1$ in the now backwards traversed cascade, we apply two transformations corresponding to the transposed matrix ${\mathbf{V}_{l,\tau}^n}^{\mathrm{T}}$ to each pair of polynomials $\tilde{a}^n_{2^{\tau+1}l,\tau}$, $\tilde{a}^n_{2^{\tau+1}l+1,\tau}$ for $l = 0, \ldots, \frac{N}{2^{\tau+1}} - 1$:

First, we cut off the higher half of Chebyshev coefficients of every polynomial. This is a multiplication with the matrix ${\mathbf{Z}_\tau}^{\mathrm{T}}$ and can be interpreted as a projection onto a polynomial subspace spanned by Chebyshev polynomials T_k up to degree 2^τ. This yields the polynomials $\tilde{a}^n_{2^{\tau+1}l,\tau-1}$ and $\tilde{a}^n_{2^{\tau+1}l+1,tau-1}$.

The second transformation corresponds to the multiplication with the matrix ${\mathbf{U}_{l,\tau}^n}^{\mathrm{T}}$ and consists in polynomial multiplications with the associated Legendre polynomials contained in $\mathbf{U}^n_{2^\tau-1}\left(\cdot, 2^{\tau+1}l+1\right)$ followed by projections onto different polynomial subspaces, concretely the set of polynomials up to the degree 2^τ (multiplication with ${\mathbf{Z}_{1,\tau}}^{\mathrm{T}}$) and the set of polynomials with strictly higher degree up to $2^{\tau+1}$ (multiplication with ${\mathbf{Z}_{2,\tau}}^{\mathrm{T}}$).

6 Examples

We present numerical examples in order to demonstrate the stability, accuracy, and efficiency of our approach. All algorithms were implemented in C and tested on an AMD Athlon$^{\mathrm{TM}}$XP 2700+ with 2GB main memory, SuSe-Linux (kernel 2.6.5-7.151-default, gcc 3.3.5) using double precision arithmetic. Moreover, we have used the libraries FFTW 3.0.1 [6] and the NFFT 3.0 library [8], now including the fast NFSFT algorithms. Throughout our experiments we have applied the NFFT routines with precomputed Kaiser–Bessel functions and an oversampling factor of two. For the NFSFT routines we used the threshold $\kappa = 1000$ for the stabilisation.

In our tests, we used a collection of test functions $f_1, f_2, \ldots, f_4$ from [21]:

$$
\begin{aligned}
f_1(\mathbf{x}) &:= x_1 x_2 x_3, \\
f_2(\mathbf{x}) &:= 0.1 \left\|x\right\|_1, \\
f_3(\mathbf{x}) &:= 1/\left\|x\right\|_1, \\
f_4(\mathbf{x}) &:= 0.1 \sin^2\left(1 + \left\|x\right\|_1\right).
\end{aligned}
$$

Function f_1 is a cubic polynomial, i.e. the spherical Fourier coefficients a_k^n vanish for k greater than three, while the other functions, not being polynomials, have more or less rapidly decreasing spherical Fourier coefficients a_k^n as k grows. In addition, we used the test function f_5 with

$$
f_5(\vartheta, \varphi) := \begin{cases} 1, & \text{if } \vartheta \in [0, \pi/2], \\ \left(1 + 3\cos^2\vartheta\right)^{-1/2}, & \text{if } \vartheta \in (\pi/2, \pi] \end{cases}
$$

which we took from [2]. It consists of a half-sphere joined with a half-ellipsoid. It is smooth everywhere except at the equator, where the two parts are joined.

We tested our algorithms on a variety of different quadrature formulae each identified with a tuple $(\mathcal{X}, W)$ consisting of a set of nodes $\mathcal{X} = \left\{(\vartheta_d, \varphi_d) \in \mathbb{S}^2 \;:\; d = 1, \ldots, D\right\}$ and positive weights $W := \left\{w\left(\vartheta_d, \varphi_d\right) > 0 \;:\; d = 1, \ldots, D\right\}$ to compute spherical Fourier coefficients a_k^n from the formula

$$
a_k^n = \langle f, Y_k^n \rangle_{\mathrm{L}^2(\mathbb{S}^2)} = \int_0^{2\pi} \int_0^\pi f(\vartheta, \varphi) \overline{Y_k^n(\vartheta, \varphi)} \sin\vartheta \,\mathrm{d}\vartheta \,\mathrm{d}\varphi
$$

by discretising to the sums

$$\sum_{d=1}^{D} w_d f(\vartheta_d, \varphi_d) \overline{Y_k^n(\vartheta_d, \varphi_d)}. \tag{6.1}$$

We considered the following quadrature formulae $(\mathcal{X}, W)$:

i) The *Gauss-Legendre quadrature grid* $\mathcal{X}_S^{\mathrm{G}}$ of size $S \in \mathbb{N}_0$ is the Cartesian product

$$\mathcal{X}_S^{\mathrm{G}} := \left\{ \vartheta_j^{\mathrm{G}} \;:\; j = 0, \ldots, S \right\} \times \left\{ \varphi_k^{\mathrm{G}} \;:\; k = 0, \ldots, 2S+1 \right\}$$

with longitudinal nodes $\varphi_k^{\mathrm{G}} := \frac{k\pi}{S+1}$. For the co-latitudinal direction we use the Gauss-Legendre quadrature with nodes ϑ_j^{G} and weights w_j^{G} which can be obtained as the solution of an eigenvalue problem (see [3, pp. 95]). The weights

$$W_S^G := \left\{ w_d^{\mathrm{G}} = w_{j,k}^{\mathrm{G}} \;:\; j = 0, \ldots, S;\; k = 0, \ldots, 2S+1 \right\}$$

for the entire quadrature formula are then given by $w_d = w_{j,k}^{\mathrm{G}} := \frac{2\pi}{2S+2} w_j^{\mathrm{G}}$. The number of nodes is $\left|\mathcal{X}_S^{\mathrm{G}}\right| = 2S^2 + 4S + 2$. The quadrature formula $\left(\mathcal{X}_S^{\mathrm{G}}, W_S^{\mathrm{G}}\right)$ is exact for polynomials up to degree $M \leq 2S+1$ so that (6.1) gives exact Fourier coefficients for f a polynomial of degree at most S.

ii) The *Clenshaw-Curtis quadrature grid* $\mathcal{X}_S^{\mathrm{C}}$ of size $S \in \mathbb{N}_0$ is the Cartesian product

$$\mathcal{X}_S^{\mathrm{C}} := \left\{ \vartheta_j^{\mathrm{C}} \;:\; j = 0, \ldots, 2S \right\} \times \left\{ \varphi_k^{\mathrm{C}} \;:\; k = 0, \ldots, 2S+1 \right\}$$

with longitudinal nodes $\varphi_k^{\mathrm{C}} := \frac{k\pi}{S+1}$ and co-latitudinal nodes $\vartheta_j^{\mathrm{C}} := \frac{j\pi}{2S}$. The weights

$$W_S^C := \left\{ w_{j,k}^{\mathrm{C}} \;:\; j = 0, \ldots, 2S;\; k = 0, \ldots, 2S+1 \right\}$$

for the quadrature formula are determined by

$$w_{j,k}^{\mathrm{C}} := w_{2S-j,k}^{\mathrm{C}} := \frac{4\pi\varepsilon_j^{2S}}{S(2S+2)} \sum_{l=0}^{S} \varepsilon_l^S \frac{1}{1-4l^2} \cos\frac{jl\pi}{S}$$

for $j = 0, \ldots, S$ and $k = 0, \ldots, 2S+1$. They can be computed efficiently by a DCT or a FFT (see e.g. [19]). Here, we have defined

$$\varepsilon_j^J := \begin{cases} \frac{1}{2} & \text{if } j = 0 \text{ or } j = J, \\ 1 & \text{if } 0 < j < J, \end{cases}$$

for $J \in \mathbb{N}_0$. We have $\left|\mathcal{X}_S^{\mathrm{C}}\right| = 4S^2 + 6S + 2$. The quadrature formula $\left(\mathcal{X}_S^{\mathrm{C}}, W_S^{\mathrm{C}}\right)$ is exact for polynomials up to degree $M \leq 2S$ so that (6.1) gives exact Fourier coefficients for f a polynomial of degree at most S. The Clenshaw-Curtis quadrature rule is computationally attractive since its nodes and weights are easily computed and allow for using fast FFT-based algorithms for evaluation, see also [18].

iii) The *HEALPix grid* $\mathcal{X}_S^{\mathrm{C}}$ of size $S = 2^t$, $t \in \mathbb{N}_0$, is a hierarchical area partitioning scheme on the sphere and has importance as data storage standard in several applications like cosmic microwave background estimation. It comprises $\left|\mathcal{X}_S^{\mathrm{H}}\right| = 12S^2$ nodes, where S must be a power of two, and is given explicitly by

$$
\begin{aligned}
\mathcal{N}_S &:= \left\{ \left(\arccos\left(1 - \frac{k^2}{3S^2}\right), \frac{\pi\left(n+\frac{1}{2}\right)}{2k} \right) \;:\; k = 1,\dots,S-1;\; n = 0,\dots,4k-1 \right\},\\
\mathcal{E}_S &:= \left\{ \left(\arccos\left(\frac{2(2S-k)}{3S}\right), \frac{\pi\left(n+\frac{\delta_{0,k \bmod 2}}{2}\right)}{2S} \right) \;:\; k = S,\dots,3S;\; n = 0,\dots,4S-1 \right\},\\
\mathcal{S}_S &:= \left\{ \left(\arccos\left(-\left(1 - \frac{k^2}{3S^2}\right)\right), \frac{\pi\left(n+\frac{1}{2}\right)}{2k} \right) \;:\; k = 1,\dots,S-1;\; n = 0,\dots,4k-1 \right\},\\
\mathcal{X}_S^{\mathrm{H}} &:= \mathcal{N}_S \cup \mathcal{E}_S \cup \mathcal{S}_S.
\end{aligned}
$$

A drawback is that HEALPix grids lack an exact integration scheme with easily computable weights and a degree of exactness. Nevertheless, it can be used to compute spherical Fourier coefficients a_k^n up to certain accuracy. For simplicity, we use for the weights $W_S^{\mathrm{H}} := \left\{w_d^{\mathrm{H}} \;:\; d = 1,\dots,12S^2\right\}$ the uniform estimate $w_d^{\mathrm{H}} := 4\pi/\left|\mathcal{X}_S^{\mathrm{H}}\right|$.

iv) The last grid used is a so-called *equidistribution grid* $\mathcal{X}_S^{\mathrm{E}}$ of size $S \in \mathbb{N}$ for which in the limit $S \to \infty$ the exact quadrature weights approach the uniform distribution $w_{j,k}^{\mathrm{E}} = \frac{4\pi}{\left|\mathcal{X}_S^{\mathrm{E}}\right|}$ (see [5, Chapter 7]). We took the ensemble from Example 7.1.9 in [5, pp. 171] with nodes given by

$$
\begin{aligned}
\mathcal{X}_S^{\mathrm{E}} :=& \{x_{0,0} = (0,0), x_{S,0} = (\pi,0)\} \cup \\
& \bigcup_{j=1}^{S-1} \left\{ x_{j,k} = \left(\frac{j\pi}{S}, \left(k - \frac{1}{2}\right)\left(\frac{2\pi}{S_j}\right) \right) \;:\; k = 1,\dots,S_j \right\},\\
S_j :=& \begin{cases} 1, & \text{if } j = 0 \text{ or } j = S,\\ \left\lfloor 2\pi/\arccos\left(\left(\cos\frac{\pi}{S} - \cos^2\frac{j\pi}{S}\right)/\sin^2\frac{j\pi}{S}\right)\right\rfloor, & \text{if } 0 < j < S. \end{cases}
\end{aligned}
$$

An upper bound for the number of nodes contained is $\left|\mathcal{X}_S^{\mathrm{E}}\right| \le 2 + \frac{4}{\pi}S^2$. Instead of using uniform weights we employed the Clenshaw-Curtis quadrature rule again and obtained for the weights $W_S^{\mathrm{E}} := \left\{ w_{j,k}^{\mathrm{E}} \;:\; j = 0,\dots,S;\; k = 1,\dots,S_j \right\}$

$$
\begin{aligned}
w_{0,0}^{\mathrm{E}} := w_{S,0}^{\mathrm{E}} &:= \frac{2\pi}{S_j S} \sum_{j=0}^{\lfloor S/2 \rfloor} \varepsilon_j^{S/2} \frac{1}{1-4j^2},\\
w_{j,k}^{\mathrm{E}} := w_{S-j,k}^{\mathrm{E}} &:= \frac{4\pi}{S_j S} \sum_{j=0}^{\lfloor S/2 \rfloor} \varepsilon_j^{S/2} \frac{1}{1-4j^2} \cos\frac{jk\pi}{S/2}.
\end{aligned}
$$

Example 6.1. We first examine the accuracy of the adjoint algorithm. Since we cannot compute the exact output of the adjoint algorithms as reference without extraordinary amount of time, e.g. using Mathematica with arbitrary precision arithmetic and direct evaluation of the appearing sums, we investigate the combined accuracy of the adjoint and non-adjoint

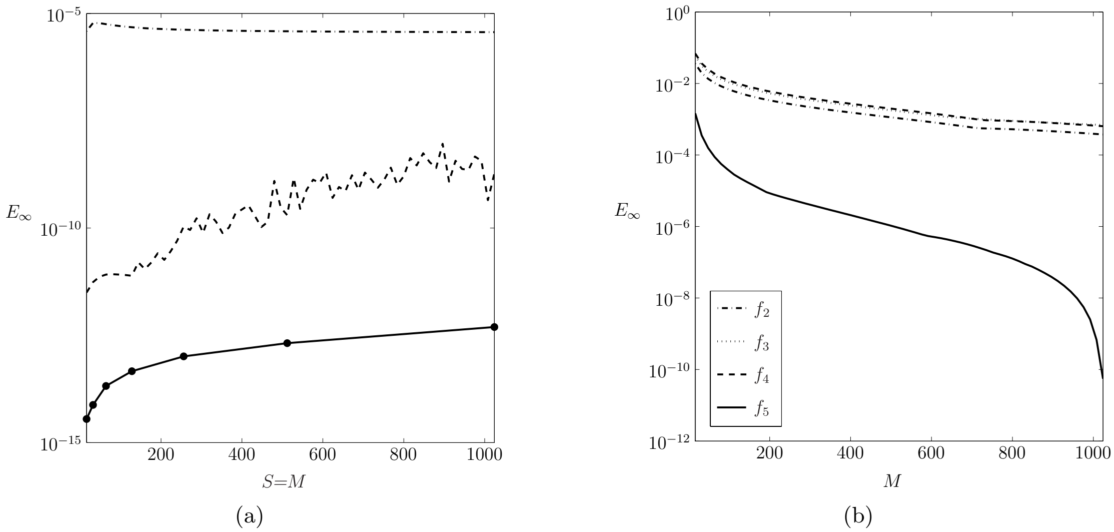

Figure 6.1: (a) The error E_∞ for the fast NFSFT algorithms with NFFT cut-off parameter $m = 3$ (dash-dotted), $m = 6$ (dashed) and $S = M = 16, 32, 48, \ldots, 1024$, and the direct NDSFT algorithms for $S = M = 16, 32, 64, 128, 256, 512, 1024$ (solid).
(b) The error E_∞ for the test functions f_2, f_3, f_4, f_5 with $S = 1024$ and $M = 16, 32, 48, \ldots, 1024$.
In both examples, we used the quadrature rule $\left(\mathcal{X}_S^{\mathrm{G}}, W_S^{\mathrm{G}}\right)$.

algorithms. We therefore evaluate the test function f_1, which is a polynomial of degree three on Gauss-Legendre grids $\mathcal{X}_S^{\mathrm{G}}$ of increasing size S, compute spherical Fourier coefficients a_k^n up to the appropriate degree of exactness $M = S$, and evaluate the obtained trigonometric polynomial, which should be identical with the function f_1, on the same Gauss-Legendre grid $\mathcal{X}_S^{\mathrm{G}}$ using the non-adjoint transform algorithm. In matrix-vector notation, this reads

$$\tilde{\mathbf{f}} = \mathbf{Y}_{M,\mathcal{X}_M^{\mathrm{G}}} \left(\mathbf{Y}_{M,\mathcal{X}_M^{\mathrm{G}}}\right)^{\mathrm{H}} \mathbf{W}_M \, \mathbf{f}.$$

The relative infinity error

$$E_\infty := \frac{\|\mathbf{f} - \tilde{\mathbf{f}}\|_\infty}{\|\mathbf{f}\|_\infty}.$$

with the vectors $\mathbf{f}$ and $\tilde{\mathbf{f}}$ of exact and computed function values, respectively, gives an indication of the backward stability of the adjoint algorithm. Figure 6.1 shows the results of the fast NFSFT algorithms for two different NFFT cut-off parameters $m = 3, 6$ and $S = M = 16, 32, 48, \ldots, 1024$ plus the direct NDSFT algorithms and $S = M = 16, 32, 64, 128, 256, 512, 1024$. While the NFFT cut-off parameter for $m = 3$ limits the achievable accuracy uniformely for all transform sizes, the accuracy of the computations for $m = 6$ decrease with increasing transform size owing to the properties of the polynomials used in the stabilized polynomial transform algorithm. Also the direct NDSFT algorithms show a slight decay of accuracy.

Example 6.2. We use Gauss-Legendre quadrature rules to compute and evaluate polynomial approximants to the test functions f_2, f_3, f_4 and f_5 with the fast NFSFT algorithms and $m = 6$. More exactly, we evaluate the test functions on the fixed Gauss-Legendre grid $\mathcal{X}^{\mathrm{G}}_{1024}$ of size $S = 1024$ and compute spherical Fourier coefficients a^n_k up to increasing degrees $M = 16, 32, 48, \ldots, 1024$. We then evaluate the approximants on the grid $\mathcal{X}^{\mathrm{G}}_{1024}$ again using the relative error measure E_∞. In matrix-vector notation, this reads

$$\tilde{\mathbf{f}}_j = \mathbf{Y}_{M,\mathcal{X}^{\mathrm{G}}_{1024}} \left(\mathbf{Y}_{M,\mathcal{X}^{\mathrm{G}}_{1024}}\right)^{\mathrm{H}} \mathbf{W}_{1024}\, \mathbf{f}_j$$

for the test functions f_j and $j = 3, 4, 5, 6$.

Example 6.3. In Examples 6.1 and 6.2, the sampling sets are spherical grids, but for example Maskhar et al. ([10]) show that suitable quadrature rules also allow reconstruction from scattered sampling sets. Furthermore, one is often also intrested in quadrature rules which only approximate the spherical Fourier coefficients. We therefore considered the Gauss-Legendre grids $\mathcal{X}^{\mathrm{G}}_S$, the Clenshaw-Curtis grids $\mathcal{X}^{\mathrm{C}}_S$, the HEALPix grids $\mathcal{X}^{\mathrm{H}}_S$ and the equidistributions $\mathcal{X}^{\mathrm{E}}_S$ for various sizes S. Up to the fixed degree $M = 128$, we choose spherical Fourier coefficients a^n_k randomly from $\left[-\frac{1}{2}, \frac{1}{2}\right]$. We now evaluate this random polynomial of degree $M = 128$ on the different node sets for increasing sizes $S = 16, 32, 48, \ldots, 1024$ for $\mathcal{X}^{\mathrm{G}}_S$, $\mathcal{X}^{\mathrm{C}}_S$, $\mathcal{X}^{\mathrm{E}}_S$ and $S = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512$ for $\mathcal{X}^{\mathrm{H}}_S$. Using this function values, we try to recover the spherical Fourier coefficients a^n_k up to the degree $M = 128$ and compare the resulting polynomial with the original function on a set of 1000 uniformly distributed random nodes, using the relative error measure E_∞. In matrix-vector notation this reads

$$\tilde{\mathbf{f}}_{\mathrm{x}} = \mathbf{Y}_{128,\mathcal{X}^{\mathrm{G}}_S} \left(\mathbf{Y}_{128,\mathcal{X}^{\mathrm{G}}_S}\right)^{\mathrm{H}} \mathbf{W}_S\, \mathbf{f}_{\mathrm{x}}.$$

Figure 6.2 (a) compares the error E_∞ for the different quadrature rules and the absolute number of nodes used.

Example 6.4. We finally compared the computation time of the adjoint NDSFT and adjoint NFSFT algorithms. Figure 6.2 (b) shows the CPU time required for one transformation as a function of the bandwidth M. For given M we chose the number of nodes as M^2. For larger M, the NFSFT algorithm outperforms the NDSFT algorithm.

References

[1] S. Belmehdi. On the associated orthogonal polynomials. *J. Comput. Appl. Math.*, 32:311 – 319, 1991.

[2] M. Böhme and D. Potts. A fast algorithm for filtering and wavelet decomposition on the sphere. *Electron. Trans. Numer. Anal.*, 16:70 – 92, 2003.

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

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

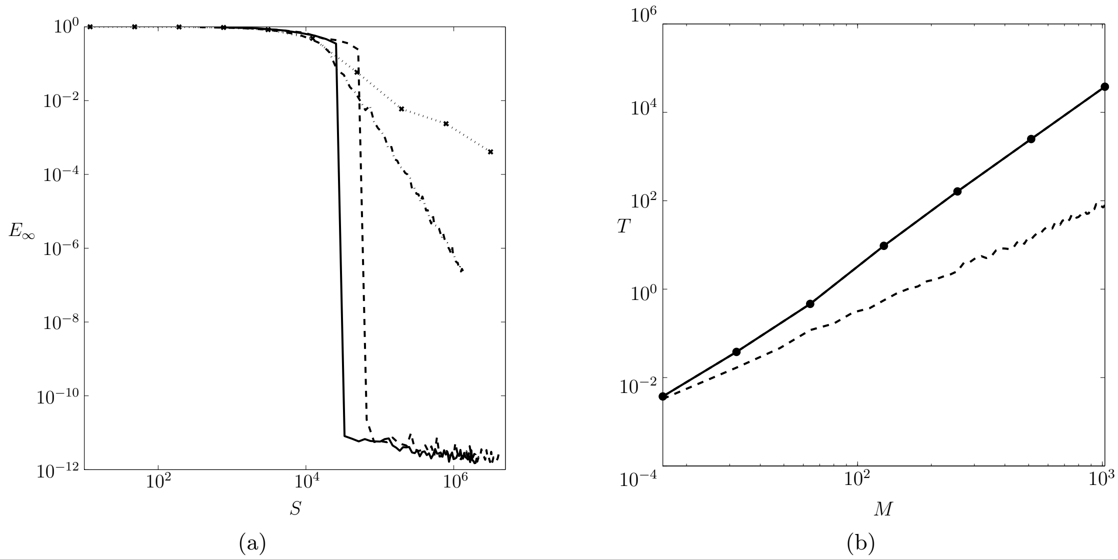

Figure 6.2: (a) The error E_∞ for the fast NFSFT algorithms with NFFT cut-off parameter $m = 6$, a random polynomial with fixed degree $M = 128$ and sizes $S = 16, 32, 48, \ldots, 1024$ for the quadrature rules $\left(\mathcal{X}_S^{\mathrm{G}}, W_S^{\mathrm{G}}\right)$ (solid), $\left(\mathcal{X}_S^{\mathrm{C}}, W_S^{\mathrm{C}}\right)$ (dashed), $\left(\mathcal{X}_S^{\mathrm{E}}, W_S^{\mathrm{E}}\right)$ (dash-dotted) and $S = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512$ for the quadrature rules $\left(\mathcal{X}_S^{\mathrm{H}}, W_S^{\mathrm{H}}\right)$ (dotted).
(b) Time measurements for the fast adjoint NFSFT algorithm (dashed) with NFFT cut-off parameter $m = 6$, $M = 16, 32, 48, \ldots, 1024$, $S = M^2$ and the adjoint direct NDSFT algorithm (solid) with $M = 16, 32, 64, 128, 256, 512, 1024$, $S = M^2$. The time is the duration of a single adjoint transform in seconds executed on a set of S uniformly distributed random nodes.

[5] W. Freeden, T. Gervens, and M. Schreiner. *Constructive Approximation on the Sphere*. Oxford University Press, Oxford, 1998.

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

[7] 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.

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

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

[10] H. N. Mhaskar, F. J. Narcowich, and J. D. Ward. Spherical Marcinkiewicz-Zygmund inequalities and positive quadrature. *Math. Comput.*, 70:1113 – 1130, 2001. Corrigendum on the positivity of the quadrature weights in 71:453 – 454, 2002.

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

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

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

[14] 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.

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

[16] I. H. Sloan and R. S. Womersley. Constructive polynomial approximation on the sphere. *J. Approx. Theory*, 103:91 – 118, 2000.

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

[18] L. N. Trefethen. Is Gauss quadrature better than Clenshaw-Curtis? *SIAM Review*, 2007. accepted.

[19] J. Waldvogel. Fast construction of Fejer and Clenshaw-Curtis quadrature rules. *BIT*, 46:195 – 202, 2006.

[20] A. F. Ware. Fast approximate Fourier transforms for irregularly spaced data. SIAM *Rev.*, 40:838 – 856, 1998.

[21] R. S. Womersley and I. H. Sloan. How good can polynomial interpolation on the sphere be? *Adv. Comput. Math.*, 14:195 – 226, 2001.