A Fast Algorithm for Filtering and Wavelet Decomposition on the Sphere

Diplomarbeit

im Rahmen des Diplomstudiengangs Informatik

vorgelegt von

Martin Böhme

Betreuer

Prof. Dr. Jürgen Prestin

Medizinische Universität zu Lübeck
Technisch-Naturwissenschaftliche Fakultät
Institut für Mathematik

Juni 2002

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Lübeck, den 13. Juni 2002

Contents

1. Introduction

The manipulation of functions defined on the surface of the sphere and the integration of partial differential equations (PDE) on the sphere are important in areas such as weather forecasting and climate modelling [Bo01, Section 18.7].

Spherical harmonics are popular as a basis set for these applications because they have a number of useful properties. In particular, the band-limited functions that are obtained through a so-called "triangular truncation" of the spherical harmonic expansion exhibit the property of "uniform resolution"; broadly speaking, this means that the level of resolution is the same all over the sphere [JA97]. This is an important property for the integration of PDE, for example, since it eliminates the "pole problem" [Bo01, Section 18.10]. This term refers to the fact that the greater density of the grid points near the poles will give rise to instabilities if a time step that is appropriate for the grid spacing at the equator is chosen. Performing a triangular truncation in each time step is one of the countermeasures that can be used to avoid these instabilities.

Thus, one of the subproblems of integrating PDE on the sphere is projecting a function onto the space of band-limited functions. This operation is referred to as the "spherical filter" [JA97]. The computational cost of the spherical filter is quite high if implemented in a straightforward way — $\mathcal{O}(N^3)$ for $\mathcal{O}(N^2)$ grid points. In this work, we will derive a fast algorithm that cuts the complexity to $\mathcal{O}(N^2 \log N)$. We will also show how the fast spherical filter can be used to derive a fast wavelet decomposition algorithm.

Since the spherical filter is quite a costly operation, significant effort has been expended on speeding it up. Jakob-Chien and Alpert [JA97] present an $\mathcal{O}(N^2 \log N)$ algorithm that is based on Fast Multipole Methods (FMM) (see, for example, [DGR96]) to evaluate the sums that occur in the spherical filter. This work was extended by Yarvin and Rokhlin [YaRo98], who derive a generalised FMM and apply it to the spherical filter. Swarztrauber and Spotz [SwSp00] also present optimisations to the spherical filter but take a slightly different approach. They do not improve on the $\mathcal{O}(N^3)$ complexity of the straightforward filter algorithm but reduce the hidden constant while at the same time reducing the memory requirement from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$. They argue that, for the grid resolutions currently in practical use, an efficient $\mathcal{O}(N^3)$ algorithm is at least as fast as the existing $\mathcal{O}(N^2 \log N)$ algorithms, which tend to have a fairly large hidden constant. This point is debatable; in the numerical experiments we performed, our algorithm broke even with the standard $\mathcal{O}(N^3)$ algorithm for $N = 64$. The more efficient Swarztrauber-Spotz algorithm was not tested, but it would seem that $\mathcal{O}(N^2 \log N)$ algorithms are beginning to rival even efficient $\mathcal{O}(N^3)$ schemes.

The algorithm we will present is based on the ideas from [JA97]. However, we wish to avoid using the Fast Multipole Methods employed there because we feel they are rather technical and difficult to implement. Instead, we will use a fast summation algorithm developed recently by Potts and Steidl [PoSt02] that is based on the Nonequispaced Fast Fourier Transform (NFFT) [PST00]. We believe that this algorithm is conceptually simpler and easier to implement. An additional advantage of our approach is that the NFFT builds on the standard (equispaced) Fast Fourier Transform, for which highly

optimised libraries are available.

The asymptotic complexity of the NFFT summation algorithm depends on the distribution of the nodes at which the sum is to be computed. We will show that, for the Legendre nodes used in the spherical filter, the summation algorithm has a complexity of $\mathcal{O}(N \log N)$, giving a total complexity for the spherical filter algorithm of $\mathcal{O}(N^2 \log N)$.

The structure of this work is as follows. Section 2 presents some basic definitions, including the spherical harmonics. Section 3 introduces the concept of band-limited functions, defines the spherical filter and wavelet decomposition and shows how these can be computed using a forward and an inverse discrete spherical Fourier transform. This section also discusses how the two transforms can be combined in a way that allows fast summation algorithms to be employed. Section 4 presents the fast NFFT summation algorithm from [PoSt02]. Section 5 summarises the complete spherical filter algorithm and provides a complexity estimate. Section 6 discusses an implementation of the algorithm and presents the results of some numerical experiments performed on it. Finally, Section 7 summarises the results that have been obtained.

2. Definitions

In this section, we present several functions and definitions that will be needed later on.

Definition 2.1 The *normalised Legendre polynomials* are defined as

$$P_k(x) := \frac{1}{2^k k!}\frac{\mathrm{d}^k}{\mathrm{d}x^k}(x^2-1)^k \qquad (x \in [-1,1]; k \in \mathbb{N}_0).$$

Definition 2.2 The *normalised associated Legendre functions* are defined as

$$P_k^n(x) := \sqrt{\left(k+\frac{1}{2}\right)\frac{(k-n)!}{(k+n)!}}(1-x^2)^{\frac{n}{2}}\frac{\mathrm{d}^n}{\mathrm{d}x^n}P_k(x) \qquad (x \in [-1,1]; n,k \in \mathbb{N}_0, k \geq n). \tag{2.1}$$

For $k < n$, we define $P_k^n(x) := 0$. The P_k^n satisfy the three-term recurrence

$$xP_k^n = \alpha_k^n P_{k-1}^n + \alpha_{k+1}^n P_{k+1}^n \tag{2.2}$$

with

$$\alpha_k^n := \left(\frac{(k-n)(k+n)}{(2k-1)(2k+1)}\right)^{\frac{1}{2}}$$

for $k \geq n$ and $\alpha_k^n := 0$ otherwise [SwSp00, equation A.3.1]. The P_k^n are normalised so as to satisfy the orthogonality relation

$$\int_{-1}^{1} P_k^n(x)P_l^n(x)\mathrm{d}x = \delta_{k,l} \qquad (n \in \mathbb{N}_0; k,l = n, n+1, \ldots).$$

Remark 2.3 To evaluate the P_k^n, the recurrence (2.2) is used. As a starting point, we use $P_{n-1}^n(x) = 0$ and

$$P_n^n(x) = \frac{1}{2^n\, n!}\left[\frac{2n+1}{2}\,(2n)!\right]^{1/2}(1-x^2)^{n/2}.$$

For large n, care must be taken in evaluating the constant $c := \frac{1}{2^n\, n!}\left[\frac{2n+1}{2}\,(2n)!\right]^{1/2}$. If the subexpressions 2^n, $n!$ and $(2n)!$ are evaluated directly, the floating-point number range will be exceeded at some point.

Because of this, we use the following strategy: We start by setting $c := \left(\frac{2n+1}{2}\right)^{1/2}$. Then we repeatedly do the following: If $c > 1$, we multiply in one of the factors from the denominator (i.e. one of the factors from $\frac{1}{2^n}$ or $\frac{1}{n!}$). If $c < 1$, we multiply in one of the factors from the numerator (i.e. one of the factors from $[(2n)!]^{1/2}$). We continue in this way until all of the factors from either the numerator of the denominator have been used up, then multiply the remaining factors in. □

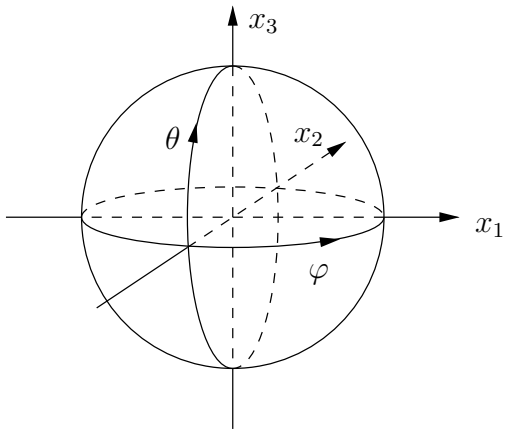


Figure 2.1: Coordinate system on the sphere.

Throughout this work, we will be dealing with functions that are defined on the unit sphere $S \coloneqq \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\|_2 = 1\}$. In particular, we will be focusing on the space $L^2(S)$ of square-integrable functions on the sphere. We will describe points on the unit sphere by their latitude $\theta \in [0, \pi]$ and longitude $\varphi \in [0, 2\pi)$ (see Figure 2.1), which are more convenient for our purposes than the Cartesian coordinates x_1, x_2, x_3.

Definition 2.4 The *spherical harmonics* are functions on the unit sphere defined as

$$Y_k^n(\theta, \varphi) \coloneqq P_k^{|n|}(\cos\theta)\, \mathrm{e}^{\mathrm{i}n\varphi},$$

where $\theta \in [0, \pi]$; $\varphi \in [0, 2\pi)$; $n \in \mathbb{Z}, k \in \mathbb{N}_0, k \geq |n|$.

These form an orthonormal basis of $L^2(S)$ with respect to the scalar product

$$\langle f, g \rangle \coloneqq \frac{1}{2\pi} \int_0^{\pi} \int_0^{2\pi} f(\theta, \varphi)\, \overline{g(\theta, \varphi)} \sin\theta \, \mathrm{d}\varphi \, \mathrm{d}\theta \qquad (f, g \in L^2(S)).$$

Definition 2.5 The *discrete Fourier transform (DFT)* of a discrete, periodic signal $s : \mathbb{Z} \to \mathbb{C}$ with period N (i.e. $s(n) = s(n + kN)$ for all $k \in \mathbb{Z}$) is defined as

$$a_k \coloneqq \frac{1}{N} \sum_{n=0}^{N-1} s(n)\, \mathrm{e}^{-\mathrm{i}2\pi \frac{kn}{N}},$$

where $k = 0, \ldots, N-1$.

The *inverse discrete Fourier transform (IDFT)* of $(a_k)_{k=0}^{N-1}$, $a_k \in \mathbb{C}$ is defined as

$$s(n) \coloneqq \sum_{k=0}^{N-1} a_k\, \mathrm{e}^{\mathrm{i}2\pi \frac{kn}{N}},$$

where $n \in \mathbb{Z}$.

3. The Spherical Filter and Wavelet Decomposition on the Sphere

In this section, we will introduce the concept of band-limited functions and use these to define the spherical filter and the wavelet decomposition on the sphere. We will introduce the forward and inverse discrete spherical Fourier transforms for transforming from the longitude / latitude grid to Fourier space and back. And finally, we will combine these two transforms to yield an algorithm for the spherical filter.

3.1. Definition of the Spherical Filter and the Wavelet Decomposition

Definition 3.1 A function $f \in L^2(S)$ is said be *band-limited* with bandwidth N if it can be expressed as

$$f(\theta, \varphi) = \sum_{k=0}^{N} \sum_{n=-k}^{k} a_k^n \, Y_k^n(\theta, \varphi) \qquad (a_k^n \in \mathbb{C}).$$

Note that, for any N, the set of band-limited functions with bandwidth N forms a subspace of $L^2(S)$.

Remark 3.2 Band-limited functions possess the useful property that any rotated version of a band-limited function is also a band-limited function with the same bandwidth (see [JA97, Section 2.1]). Thus, band-limited functions are referred to as having *uniform resolution*, meaning that features are resolved equally well at all points on the sphere.

This is especially useful when the sphere is discretised with a latitude / longitude grid — such a grid cannot in general be expected to afford uniform resolution since the grid points are more dense at the poles than at the equator. This can cause problems when integrating PDE, for example, which is why a function defined on a spherical grid is often projected onto the space of band-limited functions of a certain bandwidth to guarantee uniform resolution. □

Definition 3.3 The orthogonal projection of a function $f \in L^2(S)$ onto the subspace of band-limited functions with bandwidth N is called the *spherical filter with bandwidth N*. The result of this operation is denoted by f^N, i.e. if $f = \sum_{n \in \mathbb{Z}, k \in \mathbb{N}_0, k \geq |n|} a_k^n \, Y_k^n$, then

$$f^N(\theta, \varphi) = \sum_{k=0}^{N} \sum_{n=-k}^{k} a_k^n \, Y_k^n(\theta, \varphi).$$

This operation is also referred to as a "triangular truncation", since the Fourier coefficients in f^N can be arranged in the shape of a triangle (see Figure 3.1).

Definition 3.4 Given a band-limited function f with bandwidth $N - 1$, we define its *wavelet decomposition* as the decomposition $f = f^{N-1} = f^{N/2-1} + g$, where

$$
\begin{array}{ccccc}
 & & & & a_3^3 \\
 & & & a_2^2 & a_3^2 \\
 & & a_1^1 & a_2^1 & a_3^1 \\
 & a_0^0 & a_1^0 & a_2^0 & a_3^0 \\
 & & a_1^{-1} & a_2^{-1} & a_3^{-1} \\
 & & & a_2^{-2} & a_3^{-2} \\
 & & & & a_3^{-3}
\end{array}
$$

Figure 3.1: The coefficients that remain after triangular truncation can be arranged in the shape of a triangle, hence the name. $N = 3$ in this case.

$$
f^{N/2-1}(\theta, \varphi) = \sum_{k=0}^{N/2-1} \sum_{n=-k}^{k} a_k^n \, Y_k^n(\theta, \varphi)
$$

(as defined above) is the *low-frequency component* and

$$
g(\theta, \varphi) = \sum_{k=N/2}^{N-1} \sum_{n=-k}^{k} a_k^n \, Y_k^n(\theta, \varphi)
$$

is the *wavelet component*.

In practice, to perform the wavelet decomposition, we will compute $f^{N/2-1}$ and then obtain g by $g = f^{N-1} - f^{N/2-1}$.

Details on spherical wavelets, including an explanation of why the above operation can be referred to as a wavelet decomposition, are given in [PST96] and [FGS98]. The underlying wavelets are called *Shannon wavelets*.

The definition of the wavelet decomposition assumes that the Fourier coefficients of f are known. In numerical applications, however, we often know only the function values at certain grid points on the sphere. In this case, to carry out the wavelet decomposition, we first have to determine the Fourier coefficients. Since the Y_k^n are orthonormal, as noted above, this can be accomplished by computing

$$
\begin{aligned}
a_k^n &= \langle f, Y_k^n \rangle \\
&= \frac{1}{2\pi} \int_0^{\pi} \int_0^{2\pi} f(\theta, \varphi) \; P_k^{|n|}(\cos\theta) \mathrm{e}^{-\mathrm{i}n\varphi} \sin\theta \; \mathrm{d}\varphi \; \mathrm{d}\theta \\
&= \int_0^{\pi} P_k^{|n|}(\cos\theta) \sin\theta \; \frac{1}{2\pi} \int_0^{2\pi} f(\theta, \varphi) \mathrm{e}^{-\mathrm{i}n\varphi} \; \mathrm{d}\varphi \; \mathrm{d}\theta.
\end{aligned}
$$

For convenience, we denote the inner integral, together with the normalisation constant $\frac{1}{2\pi}$, by $\mathfrak{f}_n(\theta)$ and thus obtain

$$\mathfrak{f}_n(\theta) := \frac{1}{2\pi}\int_0^{2\pi} f(\theta,\varphi)\mathrm{e}^{-\mathrm{i}n\varphi}\,\mathrm{d}\varphi \tag{3.1}$$

and

$$a_k^n = \int_0^{\pi} P_k^{|n|}(\cos\theta)\mathfrak{f}_n(\theta)\sin\theta\,\mathrm{d}\theta. \tag{3.2}$$

3.2. Discretising the Integrals

We will now proceed to show that, for band-limited functions, the integrals in (3.1) and (3.2) can be computed by discrete sums, making it possible to evaluate them numerically. To do this, we will first prove an aliasing theorem for Fourier series.

Theorem 3.5 *Let f be a one-periodic function with pointwise convergent Fourier series, i.e.*

$$f(x) = \sum_{k\in\mathbb{Z}} c_k(f)\,\mathrm{e}^{\mathrm{i}2\pi kx} \tag{3.3}$$

with Fourier coefficients

$$c_k(f) := \int_{-\frac{1}{2}}^{\frac{1}{2}} f(x)\,\mathrm{e}^{-\mathrm{i}2\pi kx}\mathrm{d}x. \tag{3.4}$$

If the $c_k(f)$ are approximated as

$$\hat{c}_k(f) := \frac{1}{n}\sum_{j=-n/2}^{n/2-1} f\left(\frac{j}{n}\right)\,\mathrm{e}^{-\mathrm{i}2\pi jk/n} \tag{3.5}$$

using the rectangle quadrature rule, then the following aliasing relation holds:

$$\hat{c}_k(f) = c_k(f) + \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} c_{k+rn}(f).$$

Proof. Substituting the Fourier expansion of f from (3.3) into the definition of the $\hat{c}_k(f)$ (given in (3.5)) yields

$$\begin{aligned}\hat{c}_k(f) &= \frac{1}{n}\sum_{j=-n/2}^{n/2-1}\sum_{l\in\mathbb{Z}} c_l(f)\,\mathrm{e}^{\mathrm{i}2\pi lj/n}\,\mathrm{e}^{-\mathrm{i}2\pi jk/n}\\ &= \sum_{l\in\mathbb{Z}} c_l(f)\,\frac{1}{n}\sum_{j=-n/2}^{n/2-1}\mathrm{e}^{\mathrm{i}2\pi j(l-k)/n}\\ &= \sum_{l\in\mathbb{Z}} c_l(f)\,\frac{1}{n}\sum_{j=0}^{n-1}\mathrm{e}^{\mathrm{i}2\pi j(l-k)/n}.\end{aligned} \tag{3.6}$$

We claim that

$$\frac{1}{n}\sum_{j=0}^{n-1} \mathrm{e}^{\mathrm{i}2\pi j(l-k)/n} = \begin{cases} 1 & \text{if } \frac{l-k}{n} \in \mathbb{Z} \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

In the case where $\frac{l-k}{n} \in \mathbb{Z}$, this holds because all of the terms in the sum are 1. In the case where $\frac{l-k}{n} \notin \mathbb{Z}$, we apply the geometrical sum $\sum_{k=0}^{n-1} q^k = \frac{q^n-1}{q-1}$. This yields

$$\sum_{j=0}^{n-1} \mathrm{e}^{\mathrm{i}2\pi j(l-k)/n} = \frac{\mathrm{e}^{\mathrm{i}2\pi(l-k)} - 1}{\mathrm{e}^{\mathrm{i}2\pi(l-k)/n} - 1} = \frac{0}{\mathrm{e}^{\mathrm{i}2\pi(l-k)/n} - 1} = 0$$

because $\frac{l-k}{n} \notin \mathbb{Z}$ and thus $\mathrm{e}^{\mathrm{i}2\pi(l-k)/n} \neq 1$.

Applying (3.7) to (3.6) yields

$$\hat{c}_k(f) = \sum_{\substack{l\in\mathbb{Z} \\ (l-k)/n\in\mathbb{Z}}} c_l(f) = \sum_{r\in\mathbb{Z}} c_{k+rn}(f) = c_k(f) + \sum_{\substack{r\in\mathbb{Z} \\ r\neq 0}} c_{k+rn}(f).$$

■

Corollary 3.6 *If f is a one-periodic function of which only the lowest n Fourier coefficients are non-zero, i.e.*

$$f(x) = \sum_{k=-n/2}^{n/2-1} c_k(f)\, \mathrm{e}^{\mathrm{i}2\pi kx},$$

then the approximation $\hat{c}_k(f)$ for the Fourier coefficients (as defined in Theorem 3.5) is exact for $k = -n/2, \ldots, n/2-1$.

This result can now be used to compute the $\mathfrak{f}_n(\theta)$ using a discrete sum.

Theorem 3.7 *If $f \in L^2(S)$ is a band-limited function of bandwidth $N-1$, then for $n = -(N-1), \ldots, N-1$ the $\mathfrak{f}_n(\theta)$ from (3.1) can be computed as*

$$\mathfrak{f}_n(\theta) = \frac{1}{2N} \sum_{t=0}^{2N-1} f(\theta, \varphi_t)\, \mathrm{e}^{-\mathrm{i}n\varphi_t}, \tag{3.8}$$

where

$$\varphi_t \coloneqq \frac{t\pi}{N} \quad (t = 0, \ldots, 2N-1).$$

Proof. First, we will show that $f(\theta, \varphi)$, as a function of φ, is a Fourier series with a finite number of non-zero coefficients:

$$\begin{aligned} f(\theta,\varphi) &= \sum_{k=0}^{N-1} \sum_{n=-k}^{k} a_k^n\, P_k^{|n|}(\cos\theta)\mathrm{e}^{\mathrm{i}n\varphi} \\ &= \sum_{n=-(N-1)}^{N-1} \mathrm{e}^{\mathrm{i}n\varphi} \sum_{k=|n|}^{N-1} a_k^n\, P_k^{|n|}(\cos\theta). \end{aligned}$$

We now apply Corollary 3.6, setting $n = 2N$ and performing the substitution $\varphi = 2\pi x$. The integral (3.1) then corresponds to the Fourier coefficients (3.4), and the sum (3.8) corresponds to the approximated Fourier coefficients (3.5), and so, by the corollary, the two are equal for $n = -(N-1), \ldots, N-1$. ∎

Theorem 3.8 *If $f \in L^2(S)$ is a band-limited function of bandwidth $N-1$, then the integral (3.2) for the a_k^n can be computed as*

$$a_k^n = \sum_{s=0}^{N-1} w_s P_k^{|n|}(x_s) \mathfrak{f}_n(\arccos x_s), \tag{3.9}$$

where the w_s and x_s $(s = 0, \ldots, N-1)$ are the weights and nodes for Gauss-Legendre quadrature, respectively (see Appendix A).

Proof. We perform the substitution $x = \cos\theta$ on the integral (3.2), yielding

$$a_k^n = \int_{-1}^{1} P_k^{|n|}(x) \mathfrak{f}_n(\arccos x)\, \mathrm{d}x. \tag{3.10}$$

Applying the Gauss-Legendre quadrature rule with N nodes yields the sum given above.

What remains to be shown is that this quadrature is exact for functions of bandwidth $N-1$ or less. To do this, it is sufficient to show that the quadrature is exact for the spherical harmonics Y_k^n $(k = 0, \ldots, N-1; n = -k, \ldots, k)$, since the functions of bandwidth $N-1$ are linear combinations of these spherical harmonics.

For $f = Y_l^m$ where $m \neq n$, we have $\mathfrak{f}_n \equiv 0$ because $\int_0^{2\pi} \mathrm{e}^{\mathrm{i}m\varphi}\, \mathrm{e}^{-\mathrm{i}n\varphi} \mathrm{d}\varphi = 0$ for $m \neq n$. Therefore, we need only examine the case where $f = Y_l^n$, $|n| \leq l \leq N-1$. In this case we have $\mathfrak{f}_n(\theta) = P_l^{|n|}(\cos\theta)$. Using the definition of the P_k^n (2.1), the integrand in (3.10) thus becomes

$$\begin{aligned} P_k^{|n|}(x) P_l^{|n|}(x) &= c \cdot \left((1-x^2)^{\frac{|n|}{2}} \frac{\mathrm{d}^{|n|}}{\mathrm{d}x^{|n|}} P_k(x)\right) \left((1-x^2)^{\frac{|n|}{2}} \frac{\mathrm{d}^{|n|}}{\mathrm{d}x^{|n|}} P_l(x)\right) \\ &= c \cdot (1-x^2)^{|n|} \left(\frac{\mathrm{d}^{|n|}}{\mathrm{d}x^{|n|}} P_k(x)\right) \left(\frac{\mathrm{d}^{|n|}}{\mathrm{d}x^{|n|}} P_l(x)\right). \end{aligned}$$

(The normalisation constants are not relevant in this context and have been combined into the constant c for clarity.) We thus have a product of polynomials of degree $2|n|$, $k-|n|$ and $l-|n|$, respectively; hence, the integrand is a polynomial of degree $k+l \leq 2N-2$ (because $k, l \leq N-1$). Gaussian quadrature with N nodes is exact for polynomials up to a degree of $2N-1$, and so the quadrature is exact for functions with bandwidth $N-1$. ∎

3.3. Algorithms

The computation of the a_k^n (also known as the discrete spherical Fourier transform) has been split up into two phases: The computation of the $\mathfrak{f}_n(\theta)$ by a Fourier series and the subsequent computation of the a_k^n by Gauss-Legendre quadrature.

Before we summarise the discrete spherical Fourier transform algorithm obtained in this way, we will investigate how the computation of $f(\theta, \varphi)$ from given Fourier coefficients a_k^n (also known as the inverse discrete spherical Fourier transform) can similarly be split up into two phases. We recall that a function on the sphere with bandwidth $N-1$ has the form

$$f(\theta,\varphi) = \sum_{k=0}^{N-1} \sum_{n=-k}^{k} a_k^n \, Y_k^n(\theta,\varphi) \qquad (a_k^n \in \mathbb{C}).$$

We will introduce an additional parameter N_{lim} for the upper limit of the first sum. This will allow us to obtain a formula for the full reconstruction of f from the a_k^n as well as for $f^{N_{\text{lim}}}$, the result of applying the spherical filter with bandwidth N_{lim}. We will thus compute

$$f^{N_{\text{lim}}}(\theta,\varphi) = \sum_{k=0}^{N_{\text{lim}}} \sum_{n=-k}^{k} a_k^n \, Y_k^n(\theta,\varphi).$$

This double sum can now be split up in the following way:

$$\begin{aligned}
f^{N_{\text{lim}}}(\theta,\varphi) &= \sum_{k=0}^{N_{\text{lim}}} \sum_{n=-k}^{k} a_k^n \, Y_k^n(\theta,\varphi) \\
&= \sum_{n=-N_{\text{lim}}}^{N_{\text{lim}}} \sum_{k=|n|}^{N_{\text{lim}}} a_k^n \, Y_k^n(\theta,\varphi) \\
&= \sum_{n=-N_{\text{lim}}}^{N_{\text{lim}}} \sum_{k=|n|}^{N_{\text{lim}}} a_k^n \, P_k^{|n|}(\cos\theta) \, \mathrm{e}^{\mathrm{i}n\varphi} \\
&= \sum_{n=-N_{\text{lim}}}^{N_{\text{lim}}} \mathrm{e}^{\mathrm{i}n\varphi} \sum_{k=|n|}^{N_{\text{lim}}} a_k^n \, P_k^{|n|}(\cos\theta) \\
&= \sum_{n=-N_{\text{lim}}}^{N_{\text{lim}}} \mathrm{e}^{\mathrm{i}n\varphi} \mathfrak{g}_n(\theta)
\end{aligned}$$

where

$$\mathfrak{g}_n(\theta) := \sum_{k=|n|}^{N_{\text{lim}}} a_k^n \, P_k^{|n|}(\cos\theta). \tag{3.11}$$

As in the forward transform, we have two steps: The computation of the $\mathfrak{g}_n(\theta)$ and the summation of the Fourier series.

Let us summarise the two transforms. In doing so, we will assume from the beginning that the values of f are given on the grid $(\theta_s, \varphi_t)_{s=0,\ t=0}^{N-1\ \ 2N-1}$ where $\theta_s := \arccos x_s$, x_s being the Gauss-Legendre nodes, and $\varphi_t := \frac{t\pi}{N}$. (See Appendix A on how to compute the nodes and weights for Gauss-Legendre quadrature.)

Algorithm 3.1 (Discrete Spherical Fourier Transform)

Input: $f(\theta_s, \varphi_t) \in \mathbb{C}$ $(s = 0, \dots, N-1,\ t = 0, \dots, 2N-1)$
(*f is band-limited with bandwidth $N-1$*)

Output: $a_k^n \in \mathbb{C}$ $(k = 0, \dots, N-1,\ n = -k, \dots, k)$
(*Fourier coefficients for f*)

Constants: $N \in \mathbb{N}$

$x_s, w_s \in \mathbb{R}$ $(s = 0, \dots, N-1)$
(*nodes and weights for Gauss-Legendre quadrature*)

$\theta_s := \arccos x_s$ $(s = 0, \dots, N-1)$, $\varphi_t := \frac{t\pi}{N}$ $(t = 0, \dots, 2N-1)$
(*grid points*)

1. *For $s = 0, \dots, N-1$ compute*

$$\mathfrak{f}_n(\theta_s) := \frac{1}{2N} \sum_{t=0}^{2N-1} f(\theta_s, \varphi_t)\, \mathrm{e}^{-\mathrm{i}n\varphi_t} \quad (n = -(N-1), \dots, N-1)$$

using an FFT. Complexity: $\mathcal{O}(N^2 \log N)$

2. *For $k = 0, \dots, N-1$ and $n = -k, \dots, k$ compute*

$$a_k^n := \sum_{s=0}^{N-1} w_s P_k^{|n|}(x_s) \mathfrak{f}_n(\theta_s)$$

Complexity: $\mathcal{O}(N^3)$

Algorithm 3.2 (Inverse Discrete Spherical Fourier Transform)

Input: $a_k^n \in \mathbb{C}$ $(k = 0, \dots, N_{\mathrm{lim}},\ n = -k, \dots, k)$
(*Fourier coefficients*)

Output: $f^{N_{\mathrm{lim}}}(\theta_s, \varphi_t) \in \mathbb{C}$ $(s = 0, \dots, N-1,\ t = 0, \dots, 2N-1)$

Constants: $N, N_{\mathrm{lim}} \in \mathbb{N}$

$\theta_s := \arccos x_s$ $(s = 0, \dots, N-1)$, $\varphi_t := \frac{t\pi}{N}$ $(t = 0, \dots, 2N-1)$
(*grid points — the x_s are the Gauss-Legendre nodes*)

1. *For* $n = -N_{\text{lim}}, \ldots, N_{\text{lim}}$ *and* $s = 0, \ldots, N-1$ *compute*

$$\mathfrak{g}_n(\theta_s) := \sum_{k=|n|}^{N_{\text{lim}}} a_k^n P_k^{|n|}(x_s)$$

Complexity: $\mathcal{O}({N_{\text{lim}}}^2\, N)$

2. *For* $s = 0, \ldots, N-1$ *compute*

$$f^{N_{\text{lim}}}(\theta_s, \varphi_t) := \sum_{n=-N_{\text{lim}}}^{N_{\text{lim}}} \mathrm{e}^{\mathrm{i}n\varphi_t} \mathfrak{g}_n(\theta_s) \quad (t = 0, \ldots, 2N-1)$$

using an FFT. This will require those components of the input vector with indexes greater than N_{lim} *to be padded with zeros. Complexity:* $\mathcal{O}(N^2 \log N)$

It is, of course, possible to apply the discrete spherical Fourier transform to functions f that are not band-limited. In this case, an inverse transform will not reconstruct the original data. Instead, one obtains the result of applying a spherical filter with bandwidth N_{lim}, which filters out high-frequency components and yields a representation with "uniform resolution", i.e. where details are resolved equally well all over the sphere (see Remark 3.2). This is useful when integrating PDE on the sphere since it allows one to choose a time step that is appropriate for the grid resolution at the equator — even though the grid points are much closer together at the poles — without encountering the instabilities that would occur if the high-frequency components were not filtered out. For details on this, see [Bo01, Section 18.10].

3.4. Combining the Middle Steps

In applications such as the wavelet decomposition and the spherical filter, the forward transform is followed directly by the inverse transform, which means that the a_k^n need never be calculated explicitly. This allows us to combine the second step of the forward transform (Algorithm 3.1) with the first step of the inverse transform (Algorithm 3.2), which will, in turn, enable us to employ some fast approximate algorithms later on. Optimising the middle steps should allow us to improve the asymptotic complexity of the whole algorithm, since these steps are the most costly parts of the algorithm — they have a complexity of $\mathcal{O}(N^3)$ compared to $\mathcal{O}(N^2 \log N)$ for the outer steps.

Combining the middle steps, i.e. substituting the quadrature formula for the a_k^n (3.9) into the definition of the $\mathfrak{g}_n$ (3.11), yields (for $\sigma = 0, \ldots, N-1$)

$$\begin{aligned}\mathfrak{g}_n(\theta_\sigma) &:= \sum_{k=|n|}^{N_{\lim}} a_k^n P_k^{|n|}(\cos\theta_\sigma) \\ &= \sum_{k=|n|}^{N_{\lim}} \sum_{s=0}^{N-1} w_s P_k^{|n|}(x_s) \mathfrak{f}_n(\theta_s) P_k^{|n|}(x_\sigma) \\ &= \sum_{s=0}^{N-1} w_s \mathfrak{f}_n(\theta_s) \sum_{k=|n|}^{N_{\lim}} P_k^{|n|}(x_s) P_k^{|n|}(x_\sigma). \end{aligned} \tag{3.12}$$

We will now proceed to derive a closed form for the inner sum in this formula that will enable the application of a fast algorithm to compute the outer sum.

3.5. The Christoffel-Darboux Formula

In this section we will derive a closed form for the sum

$$S(x,y) := \sum_{k=n}^{N-1} P_k^n(x)\ P_k^n(y). \tag{3.13}$$

The closed form we will derive is known as the *Christoffel-Darboux formula*. (In fact, this name is applied to a whole family of similar formulas that can be derived for various orthogonal polynomials.) Our derivation is based on [SwSp00, Section A.3] but is more detailed.

Theorem 3.9 *The sum* (3.13) *possesses the closed form*

$$S(x,y) = \begin{cases} \dfrac{\alpha_N^n \left(P_N^n(x) P_{N-1}^n(y) - P_{N-1}^n(x) P_N^n(y)\right)}{x-y} & \text{if } x \neq y \\ \alpha_N^n \left(P_N^{n\prime}(x) P_{N-1}^n(x) - P_{N-1}^{n\prime}(x) P_N^n(x)\right) & \text{if } x = y, \end{cases} \tag{3.14}$$

where the α_k^n are the constants from the three-term recurrence (2.2).

Proof. We first examine the case where $x \neq y$. Applying the three-term recurrence (2.2) to equation (3.13) yields

$$xS(x,y) = \sum_{k=n}^{N-1} \left(\alpha_k^n P_{k-1}^n(x) + \alpha_{k+1}^n P_{k+1}^n(x)\right) P_k^n(y)$$

and

$$yS(x,y) = \sum_{k=n}^{N-1} P_k^n(x) \left(\alpha_k^n P_{k-1}^n(y) + \alpha_{k+1}^n P_{k+1}^n(y)\right).$$

Taking the difference of these two equations yields

$$\begin{aligned}
(x-y)S(x,y) = & \sum_{k=n}^{N-1} \alpha_k^n P_{k-1}^n(x)P_k^n(y) + \alpha_{k+1}^n P_{k+1}^n(x)P_k^n(y) \\
& \quad -\alpha_k^n P_k^n(x)P_{k-1}^n(y) - \alpha_{k+1}^n P_k^n(x)P_{k+1}^n(y) \\
= & \sum_{k=n}^{N-1} \alpha_k^n P_{k-1}^n(x)P_k^n(y) - \alpha_k^n P_k^n(x)P_{k-1}^n(y) \\
+ & \sum_{k=n}^{N-1} \alpha_{k+1}^n P_{k+1}^n(x)P_k^n(y) - \alpha_{k+1}^n P_k^n(x)P_{k+1}^n(y) \\
= & \sum_{k=n}^{N-1} \alpha_k^n P_{k-1}^n(x)P_k^n(y) - \alpha_k^n P_k^n(x)P_{k-1}^n(y) \\
+ & \sum_{k=n+1}^{N} \alpha_k^n P_k^n(x)P_{k-1}^n(y) - \alpha_k^n P_{k-1}^n(x)P_k^n(y) \\
= & \underbrace{\alpha_n^n P_{n-1}^n(x)P_n^n(y) - \alpha_n^n P_n^n(X)P_{n-1}^n(y)}_{=0 \text{ because } P_k^n \equiv 0 \text{ for } k<n} \\
& + \alpha_N^n P_N^n(x)P_{N-1}^n(y) - \alpha_N^n P_{N-1}^n(x)P_N^n(y) \\
= & \alpha_N^n \left(P_N^n(x)P_{N-1}^n(y) - P_{N-1}^n(x)P_N^n(y) \right).
\end{aligned}$$

Dividing by $x-y$ yields the proposition.

We now turn to the case $x = y$. Using l'Hôpital's rule, we obtain

$$\begin{aligned}
\lim_{y\to x} S(x,y) = & \lim_{y\to x} \frac{\alpha_N^n \left(P_N^n(x)P_{N-1}^n(y) - P_{N-1}^n(x)P_N^n(y) \right)}{x-y} \\
= & \lim_{y\to x} \frac{\alpha_N^n \left(P_N^n(x)P_{N-1}^{n\prime}(y) - P_{N-1}^n(x)P_N^{n\prime}(y) \right)}{-1} \\
= & \alpha_N^n \left(P_{N-1}^n(x)P_N^{n\prime}(x) - P_N^n(x)P_{N-1}^{n\prime}(x) \right).
\end{aligned}$$

■

The Christoffel-Darboux formula for the case where $x = y$ requires the evaluation of $P_k^{n\prime}$, the derivatives of the normalised associated Legendre functions. A recurrence equation for the $P_k^{n\prime}$ can be obtained by differentiating the three-term recurrence

$$xP_k^n = \alpha_k^n P_{k-1}^n + \alpha_{k+1}^n P_{n+1}^n,$$

yielding

$$P_k^n + xP_k^{n\prime} = \alpha_k^n P_{k-1}^{n\prime} + \alpha_{k+1}^n P_{k+1}^{n\prime}. \tag{3.15}$$

To start the recurrence, we need $P_n^{n\prime}$. We have

$$P_n^n(x) = \frac{1}{2^n n!}\left[\frac{2n+1}{2}(2n)!\right]^{1/2}(1-x^2)^{n/2}$$

(see Remark 2.3). Denoting $c \coloneqq \frac{1}{2^n n!}\left[\frac{2n+1}{2}(2n)!\right]^{1/2}$ for clarity, we obtain

$$\begin{aligned} P_n^{n\prime}(x) &= c\cdot\left[\frac{n}{2}(1-x^2)^{(n-2)/2}\cdot(-2x)\right] \\ &= -cnx(1-x^2)^{(n-2)/2}. \end{aligned}$$

Again, as mentioned for the P_n^n in Remark 2.3, care must be taken in evaluating the constant c to avoid overflow.

3.6. Applying the Christoffel-Darboux Formula to the Spherical Filter

Applying the Christoffel-Darboux formula (3.14) to the equation (3.12) that was derived for the middle steps yields

$$\begin{aligned} \mathfrak{g}_n(\theta_\sigma) &= \sum_{s=0}^{N-1} w_s \mathfrak{f}_n(\theta_s) \sum_{k=|n|}^{N_{\lim}} P_k^{|n|}(x_s) P_k^{|n|}(x_\sigma) \\ &= w_\sigma \mathfrak{f}_n(\theta_\sigma)\alpha_{N_{\lim}+1}^{|n|}\left(P_{N_{\lim}+1}^{|n|\prime}(x_\sigma)P_{N_{\lim}}^{|n|}(x_\sigma) - P_{N_{\lim}}^{|n|\prime}(x_\sigma)P_{N_{\lim}+1}^{|n|}(x_\sigma)\right) + \\ &\quad \sum_{\substack{s=0\\ s\neq\sigma}}^{N-1} w_s \mathfrak{f}_n(\theta_s)\frac{\alpha_{N_{\lim}+1}^{|n|}\left(P_{N_{\lim}+1}^{|n|}(x_s)P_{N_{\lim}}^{|n|}(x_\sigma) - P_{N_{\lim}}^{|n|}(x_s)P_{N_{\lim}+1}^{|n|}(x_\sigma)\right)}{x_s - x_\sigma}. \end{aligned}$$

Focusing only on the sum $\sum_{\substack{s=0\\ s\neq\sigma}}^{N-1}\ldots$, we find

$$\begin{aligned} &\sum_{\substack{s=0\\ s\neq\sigma}}^{N-1} w_s \mathfrak{f}_n(\theta_s)\frac{\alpha_{N_{\lim}+1}^{|n|}\left(P_{N_{\lim}+1}^{|n|}(x_s)P_{N_{\lim}}^{|n|}(x_\sigma) - P_{N_{\lim}}^{|n|}(x_s)P_{N_{\lim}+1}^{|n|}(x_\sigma)\right)}{x_s - x_\sigma} \\ &\qquad = \alpha_{N_{\lim}+1}^{|n|}\left(P_{N_{\lim}}^{|n|}(x_\sigma)\sum_{\substack{s=0\\ s\neq\sigma}}^{N-1}\frac{w_s \mathfrak{f}_n(\theta_s)P_{N_{\lim}+1}^{|n|}(x_s)}{x_s - x_\sigma}\right. \\ &\qquad\qquad \left. - P_{N_{\lim}+1}^{|n|}(x_\sigma)\sum_{\substack{s=0\\ s\neq\sigma}}^{N-1}\frac{w_s \mathfrak{f}_n(\theta_s)P_{N_{\lim}}^{|n|}(x_s)}{x_s - x_\sigma}\right). \end{aligned} \tag{3.16}$$

This alone has not led to a reduction in the asymptotic complexity of the middle steps — we still have a complexity of $\mathcal{O}(N_{\text{lim}}\, N^2)$: $\mathcal{O}(N_{\text{lim}})$ for the loop $n = -N_{\text{lim}}, \ldots, N_{\text{lim}}$; $\mathcal{O}(N)$ for the loop $\sigma = 0, \ldots, N-1$; and $\mathcal{O}(N)$ for the loop $s = 0, \ldots, N-1$. However, the evaluation of the sums in (3.16) can be speeded up using fast approximate algorithms. This was done using Fast Multipole Methods in [JA97]; we will use a different technique, which will be presented next.

4. A Fast Approximate Summation Algorithm

In this section, we will present an algorithm, from [PoSt02], that computes sums of the form

$$f(y_j) = \sum_{k=1}^{N} \beta_k K(y_j - x_k) \quad (j = 1, \ldots, M),$$

where K is the so-called *kernel*. To compute the sums arising in our application, we will choose $K(x) = 1/x$.

The algorithm presented here is an approximate algorithm that is substantially faster than evaluating the sum directly. It requires $\mathcal{O}(\sigma N \log(\sigma N) + mM)$ operations compared to the $\mathcal{O}(MN)$ operations that would be required for a direct evaluation. (The constants σ and m control the accuracy of the approximation.)

The algorithm is based on the Nonequispaced Fast Fourier Transform (NFFT), an approximate algorithm for computing the Discrete Fourier Transform at nonequispaced nodes. We will derive the NFFT first and then go on to develop the fast summation algorithm.

4.1. The Nonequispaced FFT

The NFFT is used for the fast evaluation of sums of the forms

$$f(\omega_j) = \sum_{k=-N/2}^{N/2-1} f_k \, \mathrm{e}^{-\mathrm{i}2\pi k\omega_j} \qquad (j = -M/2, \ldots, M/2 - 1)$$

and

$$h(k) = \sum_{j=-M/2}^{M/2-1} h_j \, \mathrm{e}^{-\mathrm{i}2\pi k\omega_j} \qquad (k = -N/2, \ldots, N/2 - 1)$$

($\omega_j \in \mathbb{R}$), known as the NDFT and NDFT$^{\mathrm{T}}$, respectively.

Note that this is not the most general case — we still have equispaced data in either the time or the frequency domain. A fast algorithm for the case where the data in both the time and frequency domains are nonequispaced is presented in [PST00].

Note also that the NDFT can be interpreted as the multiplication of the matrix $\mathbf{A} := (\mathrm{e}^{-\mathrm{i}2\pi k\omega_j})_{j=-M/2,\, k=-N/2}^{M/2-1 \quad N/2-1}$ with the vector $\mathbf{f} := (f_k)_{k=-N/2}^{N/2-1}$. The NDFT$^{\mathrm{T}}$ is equivalent to the multiplication of $\mathbf{A}^{\mathrm{T}}$ with $\mathbf{h} := (h_j)_{j=-M/2}^{M/2-1}$.

The fast algorithms used to compute the NDFT and NDFT$^{\mathrm{T}}$ are known as the NFFT and NFFT$^{\mathrm{T}}$, respectively. In the following, we will initially restrict our attention to the NFFT and then show later on how the NFFT algorithm can be modified to yield the NFFT$^{\mathrm{T}}$ algorithm.

Approximating f

We wish to evaluate

$$f(\omega) = \sum_{k=-N/2}^{N/2-1} f_k \, \mathrm{e}^{-\mathrm{i}2\pi k\omega}$$

at the nodes $\omega_j \in \mathbb{R}$ ($j = -M/2, \ldots, M/2-1$).

It is obvious that f is a one-periodic, band-limited function with Fourier coefficients f_k. (Note that, for convenience, this section uses the convention that a Fourier series is a linear combination of basis functions $\mathrm{e}^{-\mathrm{i}2\pi kx}$; the rest of this work follows the more usual convention of using the basis functions $\mathrm{e}^{\mathrm{i}2\pi kx}$, i.e. without the minus sign in the exponent.)

We will approximate f by a sum of translates of a 1-periodic function $\tilde{\varphi}$ (i.e. $\tilde{\varphi}(\omega) = \sum_{r\in\mathbb{Z}} \varphi(\omega + r)$) with good localisation in time and frequency. That is, we wish to find an approximation for f of the form

$$f(\omega) \approx g(\omega) \coloneqq \sum_{l=-\sigma N/2}^{\sigma N/2-1} g_l \, \tilde{\varphi}\left(\omega - \frac{l}{\sigma N}\right),$$

where $\sigma > 1$ is the so-called *oversampling factor* and should be chosen such that $\sigma N/2$ is an integer.

Broadly speaking, we will now proceed as follows: We will match the Fourier coefficients of g as closely as possible to the Fourier coefficients of f to ensure a good approximation; from these Fourier coefficients, we will compute the g_l; and finally, we will evaluate g at the nodes ω_j. In this last step, we will make use of the fact that $\tilde{\varphi}$ has good localisation in time. Thus, for every ω_j, only a few terms in the sum have to be evaluated to achieve a good approximation for $g(\omega_j)$.

Note that the definition of g possesses σN degrees of freedom (namely, the coefficients g_l). This means that the Fourier coefficients of g also possess σN degrees of freedom, i.e. we will only be able to match σN of these to the Fourier coefficients of f (we will choose to match the lowest-frequency coefficients). This explains why σ is referred to as the "oversampling factor".

We will now state the properties that $\tilde{\varphi}$ should have in more detail. $\tilde{\varphi}$ should have an absolute convergent Fourier series with Fourier coefficients

$$c_k(\tilde{\varphi}) \coloneqq \int_{-\frac{1}{2}}^{\frac{1}{2}} \tilde{\varphi}(\omega) \, \mathrm{e}^{\mathrm{i}2\pi k\omega} \mathrm{d}\omega,$$

it should be even (this guarantees that the $c_k(\tilde{\varphi})$ are real-valued), and the $c_k(\tilde{\varphi})$ should become small for $|k| \geq \sigma N - N/2$. (The reason for the value $\sigma N - N/2$ will become clear in a moment.) We will also require $c_k(\tilde{\varphi}) \neq 0$ for $k = -N/2, \ldots, N/2-1$.

Remark 4.1 If the Fourier transform $\hat{\varphi}(\eta) \coloneqq \int_{-\infty}^{\infty} \varphi(\omega) \, \mathrm{e}^{-\mathrm{i}2\pi\eta\omega} \mathrm{d}\omega$ of φ is known, then the $c_k(\tilde{\varphi})$ can be obtained by sampling $\hat{\varphi}$ at the frequencies $\eta = -k$, $k \in \mathbb{Z}$. This is

because

$$\begin{aligned}
c_k(\tilde{\varphi}) &= \int_{-\frac{1}{2}}^{\frac{1}{2}} \tilde{\varphi}(\omega)\ \mathrm{e}^{\mathrm{i}2\pi k\omega}\mathrm{d}\omega \\
&= \int_{-\frac{1}{2}}^{\frac{1}{2}} \left[\sum_{r\in\mathbb{Z}} \varphi(\omega+r)\right] \mathrm{e}^{\mathrm{i}2\pi k\omega}\mathrm{d}\omega \\
&= \sum_{r\in\mathbb{Z}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \varphi(\omega+r)\ \mathrm{e}^{\mathrm{i}2\pi k\omega}\mathrm{d}\omega \\
&= \sum_{r\in\mathbb{Z}} \int_{-\frac{1}{2}+r}^{\frac{1}{2}+r} \varphi(x)\ \mathrm{e}^{\mathrm{i}2\pi k(x-r)}\mathrm{d}x \\
&= \sum_{r\in\mathbb{Z}} \int_{-\frac{1}{2}+r}^{\frac{1}{2}+r} \varphi(x)\ \mathrm{e}^{\mathrm{i}2\pi kx}\mathrm{d}x \\
&= \int_{-\infty}^{\infty} \varphi(x)\ \mathrm{e}^{\mathrm{i}2\pi kx}\mathrm{d}x \\
&= \hat{\varphi}(-k).
\end{aligned}$$

□

Matching the Fourier coefficients of g to those of f

We have already mentioned that we will choose the coefficients g_l in the definition of g such that the Fourier coefficients of g match those of f as closely as possible. Since the Fourier coefficients of $\tilde{\varphi}(\cdot - \frac{l}{\sigma N})$ are

$$c_k\left(\tilde{\varphi}\left(\cdot - \frac{l}{\sigma N}\right)\right) = c_k(\tilde{\varphi}) \cdot \mathrm{e}^{\mathrm{i}2\pi kl/(\sigma N)},$$

we have

$$\begin{aligned}
c_k(g) &= \sum_{l=-\sigma N/2}^{\sigma N/2-1} g_l\ c_k(\tilde{\varphi})\ \mathrm{e}^{\mathrm{i}2\pi kl/(\sigma N)} \\
&= c_k(\tilde{\varphi})\ \hat{g}_k \qquad \text{where} \qquad \hat{g}_k \coloneqq \sum_{l=-\sigma N/2}^{\sigma N/2-1} g_l\ \mathrm{e}^{\mathrm{i}2\pi kl/(\sigma N)}.
\end{aligned}$$

We can thus write $g(\omega)$ as

$$g(\omega) = \sum_{k\in\mathbb{Z}} c_k(\tilde{\varphi})\ \hat{g}_k\ \mathrm{e}^{-\mathrm{i}2\pi k\omega}.$$

Note the relationship between the g_l and the $\hat{g}_k$: The $\hat{g}_k$ can be computed from the g_l using an IDFT; this means that the g_l can be obtained from the $\hat{g}_k$ using a DFT:

$$g_l = \frac{1}{\sigma N} \sum_{k=-\sigma N/2}^{\sigma N/2-1} \hat{g}_k \, \mathrm{e}^{-\mathrm{i}2\pi kl/(\sigma N)}.$$

What remains, thus, is to choose appropriate values for the $\hat{g}_k$, which we will do in a moment; the g_l can then be computed using a DFT (or, more efficiently, an FFT).

Note that the $\hat{g}_k$ are, by their definition, periodic, i.e. $\hat{g}_k = \hat{g}_{k+r\sigma N}$ for any $r \in \mathbb{Z}$. This means that the $\hat{g}_k$ only have σN degrees of freedom, as has been mentioned already.

In choosing the $\hat{g}_k$, we want to match the Fourier coefficients of g as closely as possible to those of f. Since the Fourier coefficients of f are $c_k(f) = f_k$ for $k = -N/2, \ldots, N/2-1$ and $c_k(f) = 0$ for all other k, and since the Fourier coefficients of g are $\hat{g}_k \cdot c_k(\tilde{\varphi})$, we set

$$\hat{g}_k := \begin{cases} f_k / c_k(\tilde{\varphi}) & \text{for } k = -N/2, \ldots, N/2-1 \\ 0 & \text{for } k \in \{-\sigma N/2, \ldots, \sigma N/2 - 1\} \setminus \{-N/2, \ldots, N/2-1\}, \end{cases}$$

and of course the $\hat{g}_k$ are then continued periodically. The *aliasing error* is then described by

$$g(\omega) - f(\omega) = \sum_{\substack{k \in \mathbb{Z} \setminus \\ \{-\sigma N/2, \ldots, \sigma N/2 - 1\}}} \hat{g}_k \, c_k(\tilde{\varphi}) \, \mathrm{e}^{-\mathrm{i}2\pi k\omega}.$$

By the assumption that $c_k(\tilde{\varphi}) \approx 0$ for $|k| \geq \sigma N - N/2$, the aliasing error will remain small. It may not be immediately obvious why we have to require $c_k(\tilde{\varphi}) \approx 0$ only for $|k| \geq \sigma N - N/2$ and not for $|k| \geq \sigma N/2$. The reason is that for $\sigma N/2 \leq |k| < \sigma N - N/2$ we have $\hat{g}_k = 0$ (because the $\hat{g}_k$ are continued periodically). This means that the corresponding terms in the aliasing error will vanish, so that we have to require $c_k(\tilde{\varphi})$ to be small only for $|k| \geq \sigma N - N/2$.

Evaluating g

We have now determined the $\hat{g}_k$ and thus the g_l. What remains is to evaluate the sum

$$g(\omega) = \sum_{l=-\sigma N/2}^{\sigma N/2-1} g_l \, \tilde{\varphi}\left(\omega - \frac{l}{\sigma N}\right)$$

efficiently.

To this end, we will assume, as stated previously, that φ possesses good localisation in time. Assuming that φ is small outside the window $I_m := [-m/(\sigma n), m/(\sigma n)]$ (where $m \in \mathbb{N}$ is a constant with $m \ll n$), we can approximate it by the truncated version

$$\psi(\omega) := \begin{cases} \varphi(\omega) & \text{if } \omega \in I_m \\ 0 & \text{otherwise,} \end{cases}$$

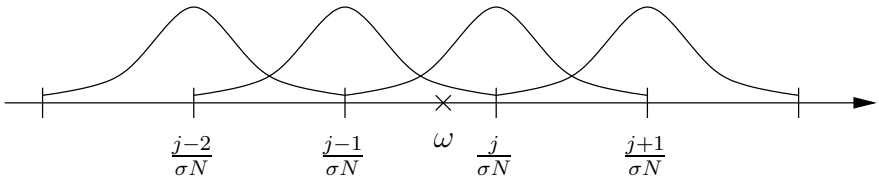
and again we set $\tilde{\psi}(\omega) := \sum_{r\in\mathbb{Z}} \psi(\omega + r)$.

We thus have

$$g(\omega) \approx \sum_{l=-\sigma N/2}^{\sigma N/2-1} g_l \, \tilde{\psi}\left(\omega - \frac{l}{\sigma N}\right)$$
$$= \sum_{l=[\omega\sigma N]-m}^{[\omega\sigma N]+m} g_l \, \tilde{\psi}\left(\omega - \frac{l}{\sigma N}\right), \tag{4.1}$$

where $[a]$ denotes the integer closest to a.

To see why this is correct, consider the following example, where $m = 1$ and $j = [\omega\sigma N]$:



The sum runs through the indexes $l = j-1, j, j+1$. Strictly speaking, the translate $\tilde{\psi}(\omega - \frac{j+1}{\sigma N})$ need not be included in the sum, since it does not make a contribution at ω. In the opposite case where ω is slightly larger than $\frac{j}{\sigma N}$, the translate $\tilde{\psi}(\omega + \frac{j+1}{\sigma N})$ could be left out of the sum. However, rather than trying to determine which of these two cases applies in a given situation, we will keep our formulas simple and always include both marginal translates in the sum.

It should be noted that, in the implementation, one can perform the small optimisation of replacing $\tilde{\psi}$ in equation (4.1) with φ. This eliminates the need to check whether $\omega \in I_m$, and at the same time the sum becomes slightly more accurate because the marginal translate that did not contribute to the sum when $\tilde{\psi}$ was used now makes a contribution.

Truncating φ introduces the so-called *truncation error*. (It should be noted that, in general, $\tilde{\varphi}(\omega) \neq \tilde{\psi}(\omega)$ even for $\omega \in I_m$. This is because, for $\omega \in I_m$, $\tilde{\varphi}(\omega) = \sum_{r\in\mathbb{Z}} \varphi(\omega+r)$ but $\tilde{\psi}(\omega) = \varphi(\omega)$, since $\tilde{\psi}$ was truncated before being made periodic.)

We thus have two errors that must be balanced against one another. The aliasing error depends on the frequency localisation of φ, and the truncation error depends on the time localisation of φ. Since an increase in time localisation leads to a decrease in frequency localisation and vice versa, a good balance must be struck in the choice of φ. Possible choices include Gaussians, cardinal central B-splines and Kaiser-Bessel functions; the latter are favoured in [PoSt02], and so we choose to use these functions. We thus have

$$\varphi(\omega) = \frac{1}{\pi} \begin{cases} \dfrac{\sinh\left(b\sqrt{m^2 - (\sigma N)^2\,\omega^2}\right)}{\sqrt{m^2 - (\sigma N)^2\,\omega^2}} & \text{for } |\omega| < \frac{m}{\sigma N} \quad (b := \pi(2 - \frac{1}{\sigma})) \\ \dfrac{\sin\left(b\sqrt{(\sigma N)^2\,\omega^2 - m^2}\right)}{\sqrt{(\sigma N)^2\,\omega^2 - m^2}} & \text{otherwise} \end{cases}$$

and

$$c_k(\tilde{\varphi}) = \begin{cases} \frac{1}{\sigma N}\, I_0\left(m\sqrt{b^2 - (2\pi k/(\sigma N))^2}\right) & \text{for } k = -\sigma N(1 - \frac{1}{2\sigma}), \ldots, \sigma N(1 - \frac{1}{2\sigma}) \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

where I_0 denotes the modified zero-order Bessel function. (These definitions were taken from [PoSt02].)

Suitable values for m and σ also have to be chosen — the former controls the truncation error, and the latter controls the aliasing error. Error estimates that provide guidance in the choice of these parameters are presented in Section 5 of [PoSt02].

We are now ready to summarise the NFFT algorithm:

Algorithm 4.1 (NFFT)

Input: $f_k \in \mathbb{C}$ $(k = -N/2, \ldots, N/2 - 1)$
(*Fourier coefficients*)

Output: $\tilde{f}(\omega_j) \in \mathbb{C}$ $(j = -M/2, \ldots, M/2 - 1)$
(*approximate values for* $f(\omega_j)$)

Constants: $M, N \in \mathbb{N}$
$\omega_j \in \mathbb{R}$ $(j = -M/2, \ldots, M/2 - 1)$
$\sigma \in \mathbb{Q}$ (*oversampling factor, must satisfy* $\sigma N/2 \in \mathbb{N}$)
$m \in \mathbb{N}$ (*parameter for the truncation of* $\tilde{\varphi}$)

Precomputation: *(i) For* $k = -N/2, \ldots, N/2 - 1$, *compute*

$$c_k(\tilde{\varphi}) := \int_{-\frac{1}{2}}^{\frac{1}{2}} \tilde{\varphi}(\omega)\, \mathrm{e}^{\mathrm{i}2\pi k\omega} \mathrm{d}\omega$$

(*A closed expression can usually be found for the* $c_k(\tilde{\varphi})$*; see Remark 4.1. For the case where* φ *is a Kaiser-Bessel function, see equation (4.2).*)

(ii) For $j = -M/2, \ldots, M/2 - 1$, *compute*

$$\psi\left(\omega_j - \frac{l}{\sigma N}\right) \qquad (l = [\omega_j \sigma N] - m, \ldots, [\omega_j \sigma N] + m)$$

1. *For* $k = -N/2, \ldots, N/2-1$, *compute*

$$\hat{g}_k := f_k / c_k(\tilde{\varphi})$$

Complexity: $\mathcal{O}(N)$

2. *For* $l = -\sigma N/2, \ldots, \sigma N/2-1$, *compute*

$$g_l := \frac{1}{\sigma N} \sum_{k=-N/2}^{N/2-1} \hat{g}_k \, \mathrm{e}^{-\mathrm{i}2\pi kl/(\sigma N)}$$

using an FFT. Complexity: $\mathcal{O}(\sigma N \log(\sigma N))$

3. *For* $j = -M/2, \ldots, M/2-1$, *compute*

$$\tilde{f}(\omega_j) := \sum_{l=[\omega_j \sigma N]-m}^{[\omega_j \sigma N]+m} g_l \, \psi\left(\omega_j - \frac{l}{\sigma N}\right)$$

Complexity: $\mathcal{O}(mM)$

The overall complexity is $\mathcal{O}(\sigma N \log(\sigma N) + mM)$.

Interpreting the NFFT as a product of matrices

It was already mentioned earlier that the NDFT can be interpreted as a matrix-vector multiplication $\mathbf{Af}$ ($\mathbf{A} \in \mathbb{C}^{M \times N}, \mathbf{f} \in \mathbb{C}^N$). In this section, we will show that the NFFT algorithm can be interpreted as an approximate factorisation of $\mathbf{A}$. Since the $\mathrm{NDFT}^{\mathrm{T}}$ is given by $\mathbf{A}^{\mathrm{T}}\mathbf{h}$ ($\mathbf{h} \in \mathbb{C}^M$), we need only transpose the factorisation of $\mathbf{A}$ to obtain an $\mathrm{NFFT}^{\mathrm{T}}$ algorithm.

$\mathbf{A}$ may be factorised approximately as follows:

$$\mathbf{A} \approx \mathbf{BFD},$$

where each of the three matrices corresponds to a step in the NFFT algorithm:

1. $\mathbf{D} \in \mathbb{R}^{N \times N}$ is a diagonal matrix:

$$\mathbf{D} := \operatorname{diag}\left(\frac{1}{c_k(\tilde{\varphi})}\right)_{k=-N/2}^{N/2-1}$$

2. $\mathbf{F} \in \mathbb{C}^{\sigma N \times N}$ is a truncated Fourier matrix:

$$\mathbf{F} := \left(\mathrm{e}^{-\mathrm{i}2\pi kl/(\sigma N)}\right)_{l=-\sigma N/2,\ k=-N/2}^{\sigma N/2-1 \quad N/2-1}$$

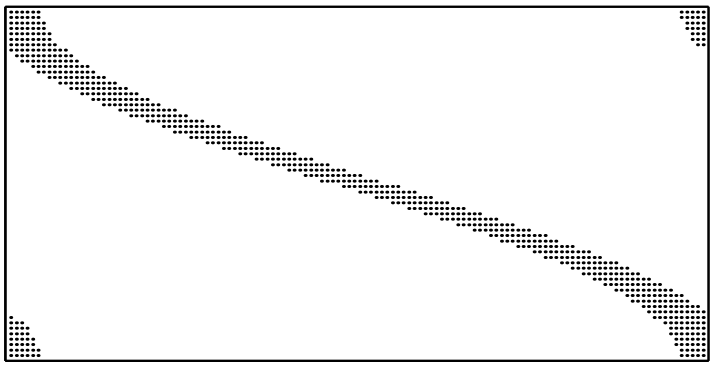


Figure 4.1: Structure of the matrix $\mathbf{B}$. Non-zero entries are indicated by dots. The row index j runs from $-M/2$ to $M/2-1$, the column index l runs from $-\sigma N/2$ to $\sigma N/2-1$. Parameters used were $M = N = 64$, $\sigma = 2$ and $m = 5$; Legendre nodes were used for the ω_j.

3. $\mathbf{B} \in \mathbb{R}^{M \times \sigma N}$ is a sparse band matrix with $2m+1$ non-zero entries per row:

$$\mathbf{B} := (b_{j\,l})_{j=-M/2,\; l=-\sigma N/2}^{M/2-1 \quad \sigma N/2-1}$$

where

$$b_{j\,l} = \begin{cases} \tilde{\psi}\left(\omega_j - \frac{l}{\sigma N}\right) & \text{if } l \in \{[\omega_j \sigma N] - m, \ldots, [\omega_j \sigma N] + m\} \\ 0 & \text{otherwise.} \end{cases}$$

(Note: For notational convenience, we are assuming that the elements of the set $\{[\omega_j \sigma N] - m, \ldots, [\omega_j \sigma N] + m\}$ will wrap around automatically to fall within the range $-\sigma N/2, \ldots, \sigma N/2 - 1$.)

A diagram illustrating the structure of the matrix $\mathbf{B}$ is given in Figure 4.1.

The NFFT$^{\mathrm{T}}$ algorithm

The factorisation that was derived for $\mathbf{A}$ allows us to derive an NFFT$^{\mathrm{T}}$ algorithm simply by transposing $\mathbf{A}$:

$$\mathbf{A}^{\mathrm{T}}\mathbf{h} \approx \mathbf{D}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{h} = \mathbf{D}\mathbf{F}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{h}.$$

This is quite straightforward; the only point which deserves to be commented on is the multiplication of the sparse matrix $\mathbf{B}^{\mathrm{T}} = (b_{j\,l})_{l=-\sigma N/2,\; j=-M/2}^{\sigma N/2-1 \quad M/2-1}$ with $\mathbf{h}$. Performing this matrix-vector multiplication in the usual way — that is, stepping down the rows of $\mathbf{B}^{\mathrm{T}}$ — is difficult to implement because there is no simple rule to tell us which columns in a given row are non-zero. It is much easier to determine which rows in a given column are non-zero — for column j, these are rows $[\omega_j \sigma N] - m, \ldots, [\omega_j \sigma N] + m$ (assuming that indexes are automatically wrapped to the appropriate range).

Because of this, we will perform the matrix-vector multiplication by stepping across $\mathbf{B}^{\mathrm{T}}$ and processing one column at a time. This means that each component of the result vector will be updated several times during the computation until it reaches its final value when the last column has been processed.

We thus propose implementing the operation $\mathbf{g} = \mathbf{B}^{\mathrm{T}}\mathbf{h}$ like this:

```
for l = -σN/2, ..., σN/2 - 1
    g_l := 0
end
for j = -M/2, ..., M/2 - 1
    for l = [ω_j σN] - m, ..., [ω_j σN] + m
        g_l := g_l + h_j b_jl
    end
end
```

We are now ready to present the NFFT$^{\mathrm{T}}$ algorithm:

Algorithm 4.2 (NFFT$^{\mathrm{T}}$)

Input:	$h_j \in \mathbb{C}$ $(j = -M/2, \ldots, M/2-1)$ (*Fourier coefficients*)
Output:	$\tilde{h}(k) \in \mathbb{C}$ $(k = -N/2, \ldots, N/2-1)$ (*approximate values for* $h(k)$)
Constants:	$M, N \in \mathbb{N}$ $\omega_j \in \mathbb{R}$ $(j = -M/2, \ldots, M/2-1)$ $\sigma \in \mathbb{Q}$ (*oversampling factor, must satisfy* $\sigma N/2 \in \mathbb{N}$) $m \in \mathbb{N}$ (*parameter for the truncation of* $\tilde{\varphi}$)
Precomputation:	*Compute* $c_k(\tilde{\varphi})$ *and* $\psi(\omega_j - \frac{l}{\sigma N})$ *as for the NFFT.*

1.
```
for l = -σN/2, ..., σN/2 - 1
    g_l := 0
end
for j = -M/2, ..., M/2 - 1
    for l = [ω_j σN] - m, ..., [ω_j σN] + m
        g_l := g_l + h_j ψ(ω_j - l/(σN))
    end
end
```

2. *For* $k = -N/2, \ldots, N/2-1$, *compute*

$$\hat{g}_k := \frac{1}{\sigma N} \sum_{l=-\sigma N/2}^{\sigma N/2-1} g_l \, \mathrm{e}^{-\mathrm{i}2\pi k \frac{l}{\sigma N}}$$

using a reduced FFT

3. For $k = -N/2, \ldots, N/2 - 1$, compute

$$\tilde{h}(k) := \hat{g}_k / c_k(\tilde{\varphi})$$

The overall complexity is $\mathcal{O}(\sigma N \log(\sigma N) + mM)$ — the same as for the NFFT, since the same operations are performed, only in a different order.

4.2. A Fast Summation Algorithm Based on the NFFT

As stated already in the introduction to this section, we will present a fast summation algorithm for sums of the form

$$f(y_j) = \sum_{k=1}^{N} \beta_k K(y_j - x_k) \quad (j = 1, \ldots, M), \tag{4.3}$$

where $\beta_k \in \mathbb{C}$, $x_k \in \mathbb{R}$ $(k = 1, \ldots, N)$, $y_j \in \mathbb{R}$ $(j = 1, \ldots, M)$.

The kernel K must be in C^∞, except for the origin, which may (and typically does) exhibit a singularity. If so, we agree to set $K(0) := 0$ so that we will be able to evaluate K for all $x \in \mathbb{R}$. (The effect of this is that any terms for which $y_j = x_k$ will simply be left out of the sum.) The kernel that we will use in our application is $K(x) = 1/x$.

The general idea is to regularise the kernel and make it periodic so that it can be approximated well by a Fourier series. Replacing the regularised kernel by the Fourier series then allows the NFFT to be used to speed up the computation but has two consequences that must be taken into consideration. Firstly, the Fourier series will not approximate K well near the origin, because of the singularity. For this reason, so-called near-field corrections have to be applied for values of $y_j - x_k$ that are close to zero. Secondly, since the kernel has been made periodic, the values of $y_j - x_k$ have to lie within the period that spans the origin, which may mean that the x_k and y_j have to be scaled to a suitable range. This will be discussed in more detail later on.

Regularising the kernel

In the following, we will derive K_R, a 1-periodic version of the kernel that is regularised near 0 and $\pm 1/2$. This regularisation is carried out using a cosine series for even kernels, a sine series for odd kernels, and a Fourier series for other kernels. Since our application uses the kernel $1/x$, we will only examine the regularisation of odd kernels.

An example of a regularised kernel is shown in Figure 4.2. To demonstrate that the kernel has been made periodic, two periods are shown.

We will parameterise our regularised kernel by the constant a, which governs the width of the intervals on which the kernel is regularised. Specifically, the kernel will be replaced by a sine series on the intervals $\left[-\frac{1}{2}, -\frac{1}{2} + \frac{a}{n}\right)$, $\left(-\frac{a}{n}, \frac{a}{n}\right)$ and $\left(\frac{1}{2} - \frac{a}{n}, \frac{1}{2}\right]$, where n is the number of terms that will be used later on in the Fourier series approximation of K_R. Thus, on $x \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, we have

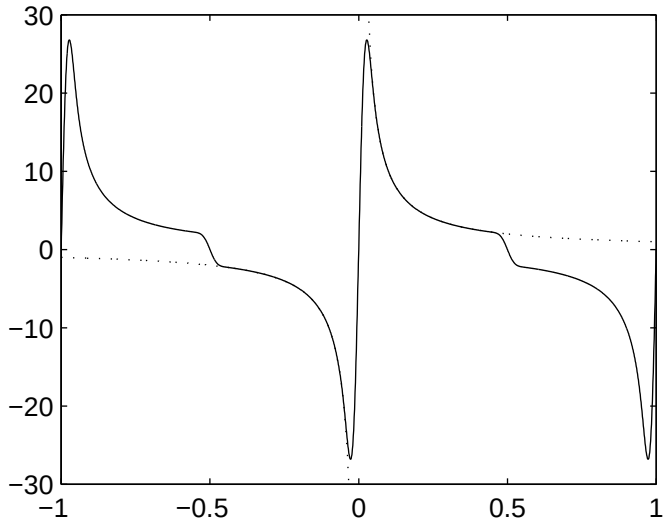


Figure 4.2: The kernel $K(x) = 1/x$ (dotted) and its 1-periodic regularised version (solid). Parameters used were $n = 128$, $a = p = 12$.

$$K_R(x) := \begin{cases} T_I(x) & \text{if } x \in \left(-\frac{a}{n}, \frac{a}{n}\right) \\ T_B(x) & \text{if } x \in \left[-\frac{1}{2}, -\frac{1}{2} + \frac{a}{n}\right) \cup \left(\frac{1}{2} - \frac{a}{n}, \frac{1}{2}\right] \\ K(x) & \text{otherwise} \end{cases}$$

with

$$T_I(x) := \sum_{j=1}^{p} a_j^I \sin \frac{\pi n j}{2a} x \quad \text{and}$$

$$T_B(x) := \begin{cases} \displaystyle\sum_{j=1}^{p} a_j^B \sin \frac{\pi n j}{2a}(x - 1/2) & \text{if } x \in \left(\frac{1}{2} - \frac{a}{n}, \frac{1}{2}\right] \\ \displaystyle\sum_{j=1}^{p} a_j^B \sin \frac{\pi n j}{2a}(x + 1/2) & \text{if } x \in \left[-\frac{1}{2}, -\frac{1}{2} + \frac{a}{n}\right), \end{cases}$$

where p is a parameter that controls how smoothly the sine series blends into the kernel. For $x \notin \left[-\frac{1}{2}, \frac{1}{2}\right]$, $K_R(x)$ is continued periodically.

To determine the coefficients a_j^I and a_j^B, we will demand that T_I and T_B should match the value and the first $p - 1$ derivatives of the kernel at the joins, i.e.

$$T_I^{(r)}\left(\frac{a}{n}\right) = K^{(r)}\left(\frac{a}{n}\right) \qquad (r = 0, \ldots, p-1) \tag{4.4}$$

and

$$T_B^{(r)}\left(-\frac{1}{2}+\frac{a}{n}\right) = K^{(r)}\left(-\frac{1}{2}+\frac{a}{n}\right) \quad (r=0,\ldots,p-1). \tag{4.5}$$

From (4.4), we obtain

$$\sum_{j=1}^{p} a_j^I \left(\frac{\pi n j}{2a}\right)^r (-1)^{r/2} \sin\frac{\pi j}{2} = K^{(r)}\left(\frac{a}{n}\right) \tag{4.6}$$

for even r and

$$\sum_{j=1}^{p} a_j^I \left(\frac{\pi n j}{2a}\right)^r (-1)^{(r-1)/2} \cos\frac{\pi j}{2} = K^{(r)}\left(\frac{a}{n}\right) \tag{4.7}$$

for odd r $(r=0,\ldots,p-1)$.

Note that the system matrix of this system of equations exhibits a chessboard-like structure, allowing it to be split into the two systems of equations

$$\sum_{j=0}^{\lfloor (p-1)/2\rfloor} a_{2j+1}^I (2j+1)^{2r} (-1)^r (-1)^j = \left(\frac{2a}{\pi n}\right)^{2r} K^{(2r)}\left(\frac{a}{n}\right) \qquad (r=0,\ldots,\lfloor\tfrac{(p-1)}{2}\rfloor)$$

and

$$\sum_{j=0}^{\lfloor p/2\rfloor} a_{2j}^I (2j)^{2r+1} (-1)^r (-1)^j = \left(\frac{2a}{\pi n}\right)^{2r+1} K^{(2r+1)}\left(\frac{a}{n}\right) \qquad (r=0,\ldots,\lfloor\tfrac{(p-2)}{2}\rfloor).$$

Turning now to equation (4.5), the condition for regularisation around $\pm 1/2$, we obtain

$$\sum_{j=1}^{p} a_j^B \left(\frac{\pi n j}{2a}\right)^r (-1)^{r/2} \sin\left[\frac{\pi n j}{2a}\left(-\frac{1}{2}+\frac{a}{n}+\frac{1}{2}\right)\right] = K^{(r)}\left(-\frac{1}{2}+\frac{a}{n}\right)$$

$$\Leftrightarrow$$

$$\sum_{j=1}^{p} a_j^B \left(\frac{\pi n j}{2a}\right)^r (-1)^{r/2} \sin\frac{\pi j}{2} = K^{(r)}\left(-\frac{1}{2}+\frac{a}{n}\right).$$

for even r. Note that this equation has the same left-hand side as equation (4.6). The case for odd r proceeds in a similar way, yielding an equation with the same left-hand

side as equation (4.7). The result of this is that we can compute the a_j^B using the same system matrices as for the a_j^I.

Approximating K_R by a Fourier Series

We approximate K_R by the Fourier series

$$K_{RF}(x) \coloneqq \sum_{l=-n/2}^{n/2-1} b_l \, \mathrm{e}^{\mathrm{i}2\pi l x},$$

where $n \approx N$ is the desired number of terms in the Fourier series and the Fourier coefficients are given by

$$b_l \coloneqq \frac{1}{n} \sum_{j=-n/2}^{n/2-1} K_R\left(\frac{j}{n}\right) \mathrm{e}^{-\mathrm{i}2\pi j l/n} \quad (l = -n/2, \ldots, n/2-1).$$

We now split up K like this:

$$K = (K - K_R) + (K_R - K_{RF}) + K_{RF} \tag{4.8}$$

and denote $K - K_R$ by K_{NE} (this is the so-called near-field correction) and $K_R - K_{RF}$ by K_{ER} (this is the error introduced by the Fourier series approximation). Since K_R is smooth, we expect that K_{ER} will almost vanish, and so we replace K by $K_{NE} + K_{RF}$ in the sum we wish to compute:

$$\begin{aligned} \tilde{f}(y_j) &\coloneqq \sum_{k=1}^{N} \beta_k (K_{NE} + K_{RF})(y_j - x_k) \quad (j = 1, \ldots, M) \\ &= \sum_{k=1}^{N} \beta_k K_{NE}(y_j - x_k) + \sum_{k=1}^{N} \beta_k K_{RF}(y_j - x_k). \end{aligned} \tag{4.9}$$

We denote the first sum by $f_{NE}(y_j)$ and the second by $f_{RF}(y_j)$.

With a view to evaluating $f_{NE}(y_j)$ efficiently, we now return to the question of which range the x_k and y_j should be constrained to. We note that $K_{NE}(x) = 0$ for $x \in \left[-\frac{1}{2} + \frac{a}{n}, \frac{1}{2} - \frac{a}{n}\right] \setminus \left(-\frac{a}{n}, \frac{a}{n}\right)$. By requiring $y_j - x_k \in \left[-\frac{1}{2} + \frac{a}{n}, \frac{1}{2} - \frac{a}{n}\right]$, we can eliminate all of the terms from $\sum_{k=1}^{N} \beta_k K_{NE}(y_j - x_k)$ for which $y_j - x_k \notin \left(-\frac{a}{n}, \frac{a}{n}\right)$. If the x_k and y_j are distributed evenly, this means that only a few terms will be left in the sum.

To satisfy the condition $y_j - x_k \in \left[-\frac{1}{2} + \frac{a}{n}, \frac{1}{2} - \frac{a}{n}\right]$, we will require $|x_k|, |y_j| \leq \frac{1}{4} - \frac{a}{2n}$. If the x_k and y_j do not lie within the required range, they can be scaled by the factor

$$s \coloneqq \frac{\frac{1}{4} - \frac{a}{2n}}{\max\{|x_1|, \ldots, |x_N|, |y_1|, \ldots, |y_M|\}}.$$

Of course, this will require the kernel $K(x)$ to be replaced with $K(\frac{x}{s})$.

Having seen how to compute f_{NE} efficiently, we turn our attention to f_{RF}. We have

$$
\begin{aligned}
f_{RF}(y_j) &= \sum_{k=1}^{N} \beta_k K_{RF}(y_j - x_k) \\
&= \sum_{k=1}^{N} \beta_k \sum_{l=-n/2}^{n/2-1} b_l \, \mathrm{e}^{\mathrm{i}2\pi l(y_j - x_k)} \\
&= \sum_{l=-n/2}^{n/2-1} b_l \left(\sum_{k=1}^{N} \beta_k \mathrm{e}^{-\mathrm{i}2\pi l x_k} \right) \mathrm{e}^{\mathrm{i}2\pi l y_j}.
\end{aligned}
$$

The expression in brackets can be computed efficiently using an NFFT$^{\mathrm{T}}$. We then multiply with the b_l and compute the outer sum using an NFFT.

We are now ready to summarise the NFFT summation algorithm:

Algorithm 4.3 (NFFT Summation)

Input: $\beta_k \in \mathbb{C}$ $(k = 1, \ldots, N)$
(*Coefficients for the sum*)

Output: $\tilde{f}(y_j)$ $(j = 1, \ldots, M)$
(*approximate values for* $f(y_j)$)

Constants: $M, N \in \mathbb{N}$
$x_k \in \mathbb{R}$ $(k = 1, \ldots, N)$
$y_j \in \mathbb{R}$ $(j = 1, \ldots, M)$
$(|x_k - y_j| \leq \frac{1}{4} - \frac{a}{2n}$ $(k = 1, \ldots, N,\ j = 1, \ldots, M))$
$a, p \in \mathbb{N}$ (*parameters for the regularisation of the kernel*)
$n \in \mathbb{N}$ (*number of terms in the Fourier series* K_{RF})
$m \in \mathbb{N}, \sigma \in \mathbb{Q}$ (*parameters for the NFFT*)

Precomputation:

(i) *Compute coefficients* a_j^I *and* a_j^B $(j = 1, \ldots, p)$ *for the regularised kernel by solving the systems of equations derived from* (4.4) *and* (4.5).

(ii) *For* $l = -n/2, \ldots, n/2 - 1$, *compute*

$$
b_l := \frac{1}{n} \sum_{j=-n/2}^{n/2-1} K_R\left(\frac{j}{n}\right) \mathrm{e}^{-\mathrm{i}2\pi j l/n}
$$

using an FFT

(iii) *For* $j = 1, \ldots, M$, *compute*

$$
K_{NE}(y_j - x_k)
$$

for those $k \in \{1, \ldots, N\}$ *that satisfy* $|y_j - x_k| < \frac{a}{n}$

1. *For* $l = -n/2, \ldots, n/2-1$, *compute*

$$a_l := \sum_{k=1}^{N} \beta_k \, \mathrm{e}^{-\mathrm{i}2\pi l x_k}$$

using an NFFT^T. *Complexity:* $\mathcal{O}(\sigma n \log(\sigma n) + mN)$

2. *For* $l = -n/2, \ldots, n/2-1$, *compute*

$$d_l := a_l \, b_l$$

Complexity: $\mathcal{O}(n)$

3. *For* $j = 1, \ldots, M$, *compute*

$$f_{RF}(y_j) := \sum_{l=-n/2}^{n/2-1} d_l \, \mathrm{e}^{\mathrm{i}2\pi l y_j}$$

using an NFFT. Complexity: $\mathcal{O}(\sigma n \log(\sigma n) + mM)$

4. *For* $j = 1, \ldots, M$, *compute*

$$f_{NE}(y_j) := \sum_{\substack{k \in \{1,\ldots,N\}, \\ |y_j - x_k| < a/n}} \beta_k \, K_{NE}(y_j - x_k)$$

Complexity: $\mathcal{O}(a\nu M)$

5. *For* $j = 1, \ldots, M$, *compute*

$$\tilde{f}(y_j) := f_{RF}(y_j) + f_{NE}(y_j)$$

Complexity: $\mathcal{O}(M)$

The constant ν in the complexity estimate for step 4 is the maximum number of x_k in an interval of width $1/n$. Obviously, the more evenly the x_k are distributed, the smaller ν will become. The overall complexity is $\mathcal{O}(\sigma n \log(\sigma n) + m(M+N) + a\nu M)$.

4.3. Error Estimate for the Fast Summation Algorithm

We will now provide an error estimate for the fast summation algorithm (Algorithm 4.3) just presented. The error estimate is based on [PoSt02, Section 3.2], but in contrast to the results given there, we estimate the error not for the general case of an even kernel but for the special case of the odd kernel $K(x) = \frac{s}{x}$ $(s \in \mathbb{R}, s > 0)$. Also, our error

estimate still contains the coefficients a_j^I and a_j^B which occur in the regularised kernel, whereas the magnitude of these coefficients is also estimated in [PoSt02].

In the following, we will deal solely with the error introduced by Algorithm 4.3. The additional error introduced by the NFFT has already been investigated thoroughly, see Section 5 of [PoSt02] and the literature cited there. Basically, the error in the NFFT has been shown to decay exponentially with the parameter m for several choices of φ, including the Kaiser-Bessel functions we use.

Let us now examine the approximation error in Algorithm 4.3. In Section 4.2, the kernel K was split up into three components:

$$K = K_{NE} + K_{ER} + K_{RF}$$

(see equation (4.8)), where K_{NE} was the near-field correction, K_{ER} was the error introduced by the Fourier series approximation, and K_{RF} was the Fourier series itself. The kernel was then approximated as $K \approx K_{NE} + K_{RF}$. By comparing the definition of f (4.3) and the definition of the approximation $\tilde{f}$ (4.9), we see that

$$|f(y_j) - \tilde{f}(y_j)| = \left| \sum_{k=1}^{N} \beta_k K_{ER}(y_j - x_k) \right| \quad (j = 1, \ldots, M),$$

and by applying Hölder's inequality we obtain

$$|f(y_j) - \tilde{f}(y_j)| \leq \left\| (\beta_k)_{k=1}^N \right\|_p \ \left\| \left(K_{ER}(y_j - x_k) \right)_{k=1}^N \right\|_q, \tag{4.10}$$

where $\frac{1}{p} + \frac{1}{q} = 1$ $(1 \leq p \leq \infty)$. (The p-norm of a vector $\mathbf{v} = (v_k)_{k=1}^N$ is defined as $\|\mathbf{v}\|_p := \left(\sum_{k=1}^N |v_k|^p \right)^{\frac{1}{p}}$ for $1 \leq p < \infty$ and $\|\mathbf{v}\|_\infty := \max_{k=1,\ldots,N} |v_k|$.) As in [PoSt02], we will only examine (4.10) for $p = 1$ but note that the choice $p = \infty$ would be preferable if all of the β_k are nearly equal.

For $p = 1$ we have $q = \infty$, and so we need to estimate $\left\| \left(K_{ER}(y_j - x_k) \right)_{k=1}^N \right\|_\infty \leq \max_{x \in [-1/2, 1/2]} |K_{ER}(x)|$. An estimate for this is given by the following theorem.

Theorem 4.2 *For the kernel $K(x) = \frac{s}{x}$ $(s \in \mathbb{R},\ s > 0)$, we have*

$$|K_{ER}(x)| \leq 4 \frac{p - 1 + n}{(p-1)\pi^p a^{p-1} n} \left[\frac{s(p-1)!\, n}{a} + \left(\frac{\pi}{2} \right)^p \sum_{j=1}^{p} j^p (|a_j^I| + |a_j^B|) \right]$$

for $x \in [-\frac{1}{2}, \frac{1}{2}]$, where a, p and n are the parameters for the NFFT summation algorithm (Algorithm 4.3) and a_j^I and a_j^B $(j = 1, \ldots, p)$ are the coefficients for the regularised kernel K_R which are obtained by solving the systems of equations derived from (4.4) and (4.5).

Proof. The coefficients b_l of the Fourier series K_{RF} were defined as

$$b_l := \frac{1}{n} \sum_{j=-n/2}^{n/2-1} K_R \left(\frac{j}{n} \right) \mathrm{e}^{-\mathrm{i}2\pi jl/n} \quad (l = -n/2, \ldots, n/2 - 1).$$

This allows us to apply Theorem 3.5 (the aliasing theorem for Fourier series), yielding

$$b_l = c_k(K_R) + \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} c_{k+rn}(K_R).$$

This means that

$$\begin{aligned}
K_{ER}(x) &= K_R(x) - K_{RF}(x)\\
&= \sum_{l\in\mathbb{Z}} c_l(K_R)\,\mathrm{e}^{\mathrm{i}2\pi lx} - \sum_{l=-n/2}^{n/2-1} b_l\,\mathrm{e}^{\mathrm{i}2\pi lx}\\
&= \sum_{l\in\mathbb{Z}} c_l(K_R)\,\mathrm{e}^{\mathrm{i}2\pi lx} - \sum_{l=-n/2}^{n/2-1} c_l(K_R)\,\mathrm{e}^{\mathrm{i}2\pi lx} - \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}}\sum_{l=-n/2}^{n/2-1} c_{l+rn}(K_R)\,\mathrm{e}^{\mathrm{i}2\pi lx}\\
&= \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}}\sum_{l=-n/2}^{n/2-1} c_{l+rn}(K_R)\,\mathrm{e}^{\mathrm{i}2\pi(l+rn)x} - \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}}\sum_{l=-n/2}^{n/2-1} c_{l+rn}(K_R)\,\mathrm{e}^{\mathrm{i}2\pi lx}\\
&= \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}}\sum_{l=-n/2}^{n/2-1} c_{l+rn}(K_R)\,\mathrm{e}^{\mathrm{i}2\pi lx}(\mathrm{e}^{\mathrm{i}2\pi rnx} - 1),
\end{aligned}$$

and we can estimate

$$\begin{aligned}
|K_{ER}(x)| &\le 2\sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}}\sum_{l=-n/2}^{n/2-1} |c_{l+rn}(K_R)|\\
&= 2\left(|c_{n/2}(K_R)| + 2\sum_{k=n/2+1}^{\infty} |c_k(K_R)|\right).
\end{aligned}$$

For the $c_k(K_R)$ we have

$$c_k(K_R) = (\mathrm{i}2\pi k)^{-p} c_k(K_R^{(p)}),$$

which we see by partial integration:

$$\begin{aligned}
c_k(K_R) &= \int_{-\frac{1}{2}}^{\frac{1}{2}} K_R(x)\,\mathrm{e}^{-\mathrm{i}2\pi kx}\mathrm{d}x\\
&= \left[K_R(x)\left(-\frac{1}{\mathrm{i}2\pi k}\right)\,\mathrm{e}^{-\mathrm{i}2\pi kx}\right]_{-\frac{1}{2}}^{\frac{1}{2}} + \frac{1}{\mathrm{i}2\pi k}\int_{-\frac{1}{2}}^{\frac{1}{2}} K_R'(x)\,\mathrm{e}^{-\mathrm{i}2\pi kx}\mathrm{d}x\\
&= 0 + \frac{1}{\mathrm{i}2\pi k} c_k(K_R').
\end{aligned}$$

Iteration then yields $c_k(K_R) = (\mathrm{i}2\pi k)^{-p} c_k(K_R^{(p)})$. Using this, we obtain

$$\begin{aligned}|K_{ER}(x)| &\le 2\left((\pi n)^{-p}\left|c_{n/2}(K_R^{(p)})\right| + 2\sum_{k=n/2+1}^{\infty}(2\pi k)^{-p}\left|c_k(K_R^{(p)})\right|\right)\\ &\le 2\left((\pi n)^{-p} + 2\sum_{k=n/2+1}^{\infty}(2\pi k)^{-p}\right)\int_{-\frac{1}{2}}^{\frac{1}{2}}\left|K_R^{(p)}(x)\right|\mathrm{d}x,\end{aligned}$$

because $c_k(K_R^{(p)}) = \int_{-1/2}^{1/2} K_R^{(p)}(x)\,\mathrm{e}^{-\mathrm{i}2\pi kx}\mathrm{d}x$ and thus $\left|c_k(K_R^{(p)})\right| \le \int_{-1/2}^{1/2}\left|K_R^{(p)}(x)\right|\mathrm{d}x$. The sum $\sum_{k=n/2+1}^{\infty}(2\pi k)^{-p}$ can be estimated as the lower sum of an integral:

$$\begin{aligned}\sum_{k=n/2+1}^{\infty}(2\pi k)^{-p} &\le \frac{1}{2\pi}\int_{\pi n}^{\infty} x^{-p}\mathrm{d}x\\ &= \frac{1}{2\pi}\left[\frac{x^{-p+1}}{1-p}\right]_{\pi n}^{\infty}\\ &= \frac{1}{2\pi(p-1)}(\pi n)^{-p+1}\\ &= \frac{1}{2(p-1)\pi^p n^{p-1}}.\end{aligned}$$

Using this, we obtain

$$\begin{aligned}|K_{ER}(x)| &\le 2\left((\pi n)^{-p} + \frac{1}{(p-1)\pi^p n^{p-1}}\right)\int_{-\frac{1}{2}}^{\frac{1}{2}}\left|K_R^{(p)}(x)\right|\mathrm{d}x\\ &= 2\frac{p-1+n}{(p-1)(\pi n)^p}\int_{-\frac{1}{2}}^{\frac{1}{2}}\left|K_R^{(p)}(x)\right|\mathrm{d}x.\end{aligned}$$

What remains is to estimate the integral in this equation. Since K_R is odd, we have

$$\begin{aligned}\int_{-\frac{1}{2}}^{\frac{1}{2}}\left|K_R^{(p)}(x)\right|\mathrm{d}x &\le 2\int_0^{\frac{1}{2}}\left|K_R^{(p)}(x)\right|\mathrm{d}x\\ &= 2\left(\int_0^{\frac{a}{n}}\left|T_I^{(p)}\right|\mathrm{d}x + \int_{\frac{a}{n}}^{\frac{1}{2}-\frac{a}{n}}\left|K^{(p)}(x)\right|\mathrm{d}x + \int_{\frac{1}{2}-\frac{a}{n}}^{\frac{1}{2}}\left|T_B^{(p)}\right|\mathrm{d}x\right).\end{aligned} \tag{4.11}$$

Since $K(x) = \frac{s}{x}$, we have $K^{(p)}(x) = s(-1)^p\frac{p!}{x^{p+1}}$, and so for the middle integral in (4.11)

we have

$$\begin{aligned}
\int_{\frac{a}{n}}^{\frac{1}{2}-\frac{a}{n}} \left|K^{(p)}(x)\right| \mathrm{d}x &= \int_{\frac{a}{n}}^{\frac{1}{2}-\frac{a}{n}} s\frac{p!}{x^{p+1}}\mathrm{d}x \\
&= \left[-s\frac{(p-1)!}{x^p}\right]_{\frac{a}{n}}^{\frac{1}{2}-\frac{a}{n}} \\
&= s(p-1)!\left(\frac{1}{\left(\frac{a}{n}\right)^p} - \frac{1}{\left(\frac{1}{2}-\frac{a}{n}\right)^p}\right) \\
&= s(p-1)!\left(\frac{n}{a}\right)^p\left(1 - \frac{1}{\left(\frac{n}{2a}-1\right)^p}\right) \\
&\leq s(p-1)!\left(\frac{a}{n}\right)^p.
\end{aligned}$$

For the left integral in (4.11), we have

$$\begin{aligned}
\left|T_I^{(p)}(x)\right| &= \left(\frac{\pi n}{2a}\right)^p \left|\sum_{j=1}^{p} j^p a_j^I \sin\frac{\pi n j}{2a}x\right| \qquad (4.12) \\
&\leq \left(\frac{\pi n}{2a}\right)^p \sum_{j=1}^{p} j^p |a_j^I| \\
\int_0^{\frac{a}{n}} \left|T_I^{(p)}(x)\right|\mathrm{d}x &\leq \left(\frac{\pi}{2}\right)^p \left(\frac{n}{a}\right)^{p-1} \sum_{j=1}^{p} j^p |a_j^I|
\end{aligned}$$

for even p. For odd p, we have a cosine instead of the sine in (4.12) and arrive at the same estimate. Similarly, for the right integral in (4.11), we have

$$\int_{\frac{1}{2}-\frac{a}{n}}^{\frac{1}{2}} \left|T_B^{(p)}(x)\right|\mathrm{d}x \leq \left(\frac{\pi}{2}\right)^p \left(\frac{n}{a}\right)^{p-1} \sum_{j=1}^{p} j^p |a_j^B|.$$

By assembling these various estimates, we obtain

$$\begin{aligned}
|K_{ER}(x)| &\leq 2\frac{p-1+n}{(p-1)\pi^p n^p} \cdot 2\left[s(p-1)!\left(\frac{n}{a}\right)^p + \left(\frac{\pi}{2}\right)^p \left(\frac{n}{a}\right)^{p-1} \sum_{j=1}^{p} j^p (|a_j^I| + |a_j^B|)\right] \\
&= 4\frac{p-1+n}{(p-1)\pi^p a^{p-1} n}\left[\frac{s(p-1)!\,n}{a} + \left(\frac{\pi}{2}\right)^p \sum_{j=1}^{p} j^p (|a_j^I| + |a_j^B|)\right]
\end{aligned}$$

■

Note that Stirling's formula can be used to show that the $\pi^p a^{p-1}$ in the denominator will dominate the $(p-1)!$ in the numerator if a is sufficiently large compared to p. However, Theorem 4.2 still requires the values of the coefficients a_j^I and a_j^B to be calculated

for every combination of parameters a, p, and n, and because the dependence of a_j^I and a_j^B on these parameters has not been quantified, the theorem does not tell us anything about the asymptotic behaviour of the error for varying a and p.

Potts and Steidl [PoSt02] go on to estimate a_j^I and a_j^B and show that the error decays exponentially in p if a is sufficiently large compared to p (for example, $a = p$). Their proof is for even kernels, but we expect that it can easily be adapted to odd kernels.

5. The Fast Spherical Filter Algorithm

In this section, we will summarise the fast spherical filter algorithm and show that it has an asymptotic complexity of $\mathcal{O}(N^2 \log N)$ by proving that the NFFT summation algorithm has a complexity of $\mathcal{O}(N \log N)$ when used with Legendre nodes.

5.1. Algorithm

Having discussed all of the components of the fast spherical filter algorithm, we are ready to present it in its entirety.

Algorithm 5.1 (Fast Spherical Filter)

Input:	$f(\theta_s, \varphi_t) \in \mathbb{C}$ $(s = 0, \dots, N-1,\ t = 0, \dots, 2N-1)$ *(function values on the spherical grid)*
Output:	$\tilde{f}^{N_{\lim}}(\theta_s, \varphi_t) \in \mathbb{C}$ $(s = 0, \dots, N-1,\ t = 0, \dots, 2N-1)$ *(approximate values for* $f^{N_{\lim}}(\theta_s, \varphi_t)$*)*
Constants:	$N, N_{\lim} \in \mathbb{N}$
	$x_s, w_s \in \mathbb{R}$ $(s = 0, \dots, N-1)$ *(nodes and weights for Gauss-Legendre quadrature)*
	$\theta_s := \arccos x_s$ $(s = 0, \dots, N-1)$, $\varphi_t := \frac{t\pi}{N}$ $(t = 0, \dots, 2N-1)$ *(grid points)*
	$a, p, m \in \mathbb{N}, \sigma \in \mathbb{Q}$ *(parameters for the NFFT summation algorithm)*
Precomputation:	*For* $n = 0, \dots, N_{\lim}$, $s = 0, \dots, N-1$ *compute* $P^n_{N_{\lim}}(x_s)$, $P^n_{N_{\lim}+1}(x_s)$, $P^{n\prime}_{N_{\lim}}(x_s)$ *and* $P^{n\prime}_{N_{\lim}+1}(x_s)$ *using the recurrences (2.2) and (3.15).*

1. *For* $s = 0, \dots, N-1$ *compute*

$$\mathfrak{f}_n(\theta_s) := \frac{1}{2N} \sum_{t=0}^{2N-1} f(\theta_s, \varphi_t)\, \mathrm{e}^{-\mathrm{i}n\varphi_t} \qquad (n = -(N-1), \dots, N-1)$$

   using an FFT. Complexity: $\mathcal{O}(N^2 \log N)$

2. *For* $n = -N_{\lim}, \dots, N_{\lim}$ *compute*

$$\mathfrak{S}_1(n, \sigma) := \sum_{\substack{s=0 \\ s \neq \sigma}}^{N-1} \frac{w_s \mathfrak{f}_n(\theta_s) P^{|n|}_{N_{\lim}+1}(x_s)}{x_s - x_\sigma} \qquad (\sigma = 0, \dots, N-1)$$

and

$$\mathfrak{S}_2(n,\sigma) := \sum_{\substack{s=0\\ s\neq\sigma}}^{N-1} \frac{w_s \mathfrak{f}_n(\theta_s) P_{N_{\lim}}^{|n|}(x_s)}{x_s - x_\sigma} \qquad (\sigma = 0,\dots,N-1)$$

using the NFFT summation algorithm, and then set

$$\mathfrak{g}_n(\theta_\sigma) := \alpha_{N_{\lim}+1}^{|n|}\Big(P_{N_{\lim}}^{|n|}(x_\sigma)\big(\mathfrak{S}_1(n,\sigma) + w_s\mathfrak{f}_n(\theta_\sigma)P_{N_{\lim}+1}^{|n|\prime}(x_\sigma)\big) - P_{N_{\lim}+1}^{|n|}(x_\sigma)\big(\mathfrak{S}_2(n,\sigma) + w_s\mathfrak{f}_n(\theta_\sigma)P_{N_{\lim}}^{|n|\prime}(x_\sigma)\big)\Big).$$

Complexity: $\mathcal{O}\big(N_{\lim}\cdot(\sigma N\log(\sigma N) + (m + a\log N)N)\big)$

3. *For* $s = 0,\dots,N-1$ *compute*

$$\tilde{f}^{N_{\lim}}(\theta_s,\varphi_t) := \sum_{n=-N_{\lim}}^{N_{\lim}} \mathrm{e}^{\mathrm{i}n\varphi_t}\mathfrak{g}_n(\theta_s) \qquad (t = 0,\dots,2N-1)$$

using an FFT. Complexity: $\mathcal{O}(N^2\log N)$

Note that the complexity estimates assume that the parameter n for the NFFT summation algorithm is set to N. This seems to be a sensible choice that is also used in [PoSt02] for the numerical experiments.

Note also that the reasoning behind the complexity estimate for step 2 is not entirely straightforward — it will be given in the next section. The overall complexity of the algorithm is $\mathcal{O}\big(\sigma N^2\log(\sigma N) + (m + a\log N)N^2\big)$ (where $N_{\lim}$ and N have been combined for clarity).

A point that merits discussion is the precomputation of the P_k^n and $P_k^{n\prime}$. It is important to do this as a precomputation, since the time required is $\mathcal{O}(N_{\lim}^2 N)$ — there are N points for which to evaluate these functions, $N_{\lim}+1$ different values for n, and the evaluation of the functions for $k = N_{\lim}$ and $k = N_{\lim}+1$ requires $\mathcal{O}(N_{\lim})$ applications of the three-term recurrence. If these values were not precomputed, the time required to evaluate them would dominate the running time of the algorithm.

Since the values of the P_k^n are only required for two different values of k, it would seem preferable to use a recurrence over n instead of recurrence (2.2), which runs over k. A recurrence over n does exist (see [MaOb43, Chapter 4, §3]), but it is unsuitable for numerical work because it is unstable when applied forwards and prone to underflow when applied backwards.

5.2. Complexity Estimate

We will now examine the asymptotic complexity of Algorithm 5.1 and justify the complexity estimate of $\mathcal{O}\big(N_{\lim}\cdot(\sigma N\log(\sigma N) + (m + a\log N)N)\big)$ for step 2 of the algorithm.

In this step, the NFFT summation algorithm (Algorithm 4.3) is applied $\mathcal{O}(N_{\text{lim}})$ times. The asymptotic complexity for this algorithm is $\mathcal{O}(\sigma n \log(\sigma n) + m(M+N) + a\nu M)$, and in our application we have $M = N$ and $n = N$, yielding a complexity of $\mathcal{O}(\sigma N \log(\sigma N) + (m + a\nu)N)$. The variable ν denotes the maximum number of nodes x_k in an interval of width $1/N$ and was used in Section 4.2 to estimate the number of times the near field has to be evaluated in step 4 of the algorithm.

What value should we assign to this variable? In the spherical filter algorithm, the x_k are Legendre nodes. The distances between Legendre nodes are proportional to $1/N$ in the middle of the interval $[-1, 1]$ and proportional to $1/N^2$ near the borders [Fu92, Section 3.1]. This means that an interval of width $1/N$ contains $\mathcal{O}(1)$ nodes if it is located near the middle of $[-1, 1]$, and $\mathcal{O}(N)$ nodes if it is located near the borders. Since ν is the maximum number of nodes in an interval of width $1/N$, it would appear that one should set $\nu = N$, giving a complexity estimate of $\mathcal{O}(N^3)$.

However, this estimate is too pessimistic. This is because the node distance of $1/N^2$ only applies quite close to the borders of the interval. Over most of the interval, the nodes are spaced much further apart. For this reason, we do not obtain a very precise complexity estimate if we use the minimum node distance to estimate the number of near-field evaluations.

We will now show that for the case of Legendre nodes x_k, the number of near-field evaluations performed in step 4 of the NFFT summation algorithm is bounded by $\mathcal{O}(aN \log N)$. To do this, we will introduce the requirement that $\frac{a}{n} < \frac{1}{\sqrt{2}}$; note that $\frac{a}{n}$ is always far less than this value in practice. To simplify the proof, we will also require N to be an integer multiple of 4.

Lemma 5.1 *If the NFFT summation algorithm (Algorithm 4.3) is used with Legendre nodes x_k ($k = 1, \ldots, N$, with N an integer multiple of 4), if $M = N$ and $y_k = x_k$ ($k = 1, \ldots, N$) and the parameters a and n satisfy $\frac{a}{n} < \frac{1}{\sqrt{2}}$, then the number of near-field evaluations required in step 4 of the algorithm is not greater than*

$$\frac{a(2N+1)^2}{n}\left(\frac{7}{24} + \frac{1}{2\sqrt{2}\pi}\ln\frac{4N+2}{3\pi}\right)$$

Proof. In the following, we will use the notation $\#S$ to denote the number of elements of the finite set S.

A near-field evaluation has to be carried out for all x_k and x_j with $|x_k - x_j| < \frac{a}{n}$. We want to count the number of such (ordered) pairs, and we will call this number C.

If we introduce an asymmetry between the x_k and x_j, we can restate the problem as follows: For all x_k, count the x_j with $x_j \in (x_k - \frac{a}{n}, x_k + \frac{a}{n})$, i.e.

$$C = \sum_{k=1}^{N} \#\left\{x_j : x_j \in \left(x_k - \frac{a}{n}, x_k + \frac{a}{n}\right)\right\}.$$

We can split up this problem into three parts,

$$C_1 := \sum_{k=1}^{N} \# \left\{ x_j : x_j \in \left(x_k, x_k + \frac{a}{n} \right) \right\}$$
$$C_2 := \sum_{k=1}^{N} \# \left\{ x_j : x_j \in \left(x_k - \frac{a}{n}, x_k \right) \right\}$$
$$C_3 := \sum_{k=1}^{N} \#\{ x_j : x_j = x_k \} = N,$$

and we then have $C = C_1 + C_2 + C_3$.

The x_k are symmetric about the origin, i.e. if x_k is a Legendre node, then $-x_k$ is also a Legendre node. (This is because the x_k are the roots of P_N and P_N is an even polynomial for even N, see Definition 2.1). Because of the symmetry of the x_k, the count that we obtain in C_1 for a certain x_k is equal to the count that we obtain in C_2 for $-x_k$. This means that we can obtain $C_1 + C_2$ as follows:

$$C_1 + C_2 = 2 \left(C_1^{<0} + C_2^{<0} \right),$$

where

$$C_1^{<0} := \sum_{\substack{k=1 \\ x_k<0}}^{N} \# \left\{ x_j : x_j \in \left(x_k, x_k + \frac{a}{n} \right) \right\}$$
$$C_2^{<0} := \sum_{\substack{k=1 \\ x_k<0}}^{N} \# \left\{ x_j : x_j \in \left(x_k - \frac{a}{n}, x_k \right) \right\}.$$

(Since we required N to be even, we have $x_k \neq 0$ for all k; this makes things easier, because if there were an $x_k = 0$, we would have to take care of the fact that it is symmetric to itself.)

If we consider that

$$\begin{aligned} x_j \in \left(x_k - \frac{a}{n}, x_k \right) \quad &\Leftrightarrow \\ x_k - \frac{a}{n} < x_j \quad \wedge \quad x_j < x_k \quad &\Leftrightarrow \\ x_k < x_j + \frac{a}{n} \quad \wedge \quad x_j < x_k \quad &\Leftrightarrow \\ x_k \in \left(x_j, x_j + \frac{a}{n} \right), & \end{aligned}$$

we see that $C_2^{<0}$ can also be written as

$$C_2^{<0} = \sum_{j=1}^{N} \# \left\{ x_k : x_k < 0 \ \wedge \ x_k \in \left(x_j, x_j + \frac{a}{n} \right) \right\}.$$

If $x_k < 0$, then $x_k \in \left(x_j, x_j + \frac{a}{n}\right)$ is only possible if $x_j < 0$, and so we can add this additional constraint without changing $C_2^{<0}$:

$$\begin{aligned} C_2^{<0} &= \sum_{\substack{j=1 \\ x_j<0}}^{N} \# \left\{ x_k : x_k < 0 \ \wedge \ x_k \in \left(x_j, x_j + \frac{a}{n}\right) \right\} \\ &\leq \sum_{\substack{j=1 \\ x_j<0}}^{N} \# \left\{ x_k : x_k \in \left(x_j, x_j + \frac{a}{n}\right) \right\} \\ &= C_1^{<0}. \end{aligned}$$

From $C_1 + C_2 = 2\left(C_1^{<0} + C_2^{<0}\right)$ and $C_2^{<0} \leq C_1^{<0}$ we obtain the estimate

$$\begin{aligned} C_1 + C_2 &\leq 2\left(C_1^{<0} + C_1^{<0}\right) \\ &= 4 \sum_{\substack{k=1 \\ x_k<0}}^{N} \# \left\{ x_j : x_j \in \left(x_k, x_k + \frac{a}{n}\right) \right\}, \end{aligned}$$

and assuming that the x_k are ordered we have

$$C_1 + C_2 \leq 4 \sum_{k=1}^{N/2} \# \left\{ x_j : x_j \in \left(x_k, x_k + \frac{a}{n}\right) \right\}. \tag{5.1}$$

Before continuing further, we need the following inequality, which gives us an estimate for the location of the x_k:

$$-1 \leq -\cos \frac{k - \frac{1}{2}}{N + \frac{1}{2}}\pi \leq x_k \leq -\cos \frac{k}{N + \frac{1}{2}}\pi \leq 1 \quad (k = 1, \ldots, N) \tag{5.2}$$

(see [Fu92, equation 3.1.1]).

Let us now turn to the problem of estimating how many x_j lie in the interval $(x_k, x_k + \frac{a}{n})$. Let $c(k)$ denote the smallest index i for which $x_{k+i} \geq x_k + \frac{a}{n}$. If we constrain ourselves to $k \leq \frac{N}{2}$, then our requirement $\frac{a}{n} < \frac{1}{\sqrt{2}}$ implies that such an index always exists, i.e. that $c(k)$ is well-defined. We can see this if we consider that

$$x_N \geq -\cos \frac{N - 1/2}{N + 1/2}\pi = -\cos \frac{1 - 1/(2N)}{1 + 1/(2N)}\pi$$

by equation (5.2), which implies

$$x_N \geq -\cos \frac{1 - 1/8}{1 + 1/8}\pi = -\cos \frac{7}{9}\pi \geq -\cos \frac{3}{4}\pi = \frac{1}{\sqrt{2}}$$

because $N \geq 4$. On the other hand, $x_k \leq 0$ because of $k \leq \frac{N}{2}$, and so $x_N \geq \frac{1}{\sqrt{2}} \geq x_k + \frac{a}{n}$.

By the definition of $c(k)$ is it clear that the x_j in the interval $(x_k, x_k + \frac{a}{n})$ are just those for $j = k+1, \ldots, k+c(k)-1$, which means that there are exactly $c(k)-1$ such x_j. We will construct an upper bound $\tilde{c}(k) \geq c(k)$ $(k = 1, \ldots, \frac{N}{2})$ for $c(k)$, and in this way we immediately obtain an upper bound for the number of x_j in $(x_k, x_k + \frac{a}{n})$.

We set

$$\tilde{c}(k) := \begin{cases} \min\left(\frac{N}{2},\ \frac{a}{n} \cdot \frac{2N+1}{4\sqrt{2}} \cdot \left(\sin \frac{2k-\frac{1}{2}}{2N+1}\pi\right)^{-1} + \frac{1}{2}\right) & \text{for } k = 1, \ldots, \frac{N}{4} \\ \min\left(\frac{N}{2},\ \frac{a}{n} \cdot \frac{2N+1}{4} + \frac{1}{2}\right) & \text{for } k = \frac{N}{4}+1, \ldots, \frac{N}{2}, \end{cases} \tag{5.3}$$

and we will show that this is an upper bound for $c(k)$. Note that $\frac{1}{2} \leq \tilde{c}(k) \leq \frac{N}{2}$ for $k = 1, \ldots, \frac{N}{2}$; this property will be needed in the following.

Claiming that $\tilde{c}(k) \geq c(k)$ is equivalent to claiming

$$x_{k+\tilde{c}(k)} \geq x_k + \frac{a}{n} \tag{5.4}$$

because $c(k)$ was defined as the smallest i for which $x_{k+i} \geq x_k + \frac{a}{n}$. We will now investigate under which conditions inequality (5.4) is satisfied. This inequality holds if

$$-\cos\left(\frac{k+\tilde{c}(k)-\frac{1}{2}}{N+\frac{1}{2}}\pi\right) \geq -\cos\left(\frac{k}{N+\frac{1}{2}}\pi\right) + \frac{a}{n}$$

(because of inequality (5.2)), and this is equivalent to

$$\cos\left(\frac{k+\tilde{c}(k)-\frac{1}{2}}{N+\frac{1}{2}}\pi\right) \leq \cos\left(\frac{k}{N+\frac{1}{2}}\pi\right) - \frac{a}{n} \quad \Leftrightarrow$$

$$\cos\left(\frac{k+\tilde{c}(k)-\frac{1}{2}}{N+\frac{1}{2}}\pi\right) - \cos\left(\frac{k}{N+\frac{1}{2}}\pi\right) \leq -\frac{a}{n}.$$

With $\cos(\alpha) - \cos(\beta) = -2\sin\left(\frac{\alpha+\beta}{2}\right)\sin\left(\frac{\alpha-\beta}{2}\right)$, this is equivalent to

$$-2\sin\left(\frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi\right)\sin\left(\frac{\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi\right) \leq -\frac{a}{n} \quad \Leftrightarrow$$

$$2\sin\left(\frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi\right)\sin\left(\frac{\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi\right) \geq \frac{a}{n}$$

and this is satisfied if

$$2\sin\left(\frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi\right)\frac{2\sqrt{2}}{\pi} \cdot \frac{\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi \geq \frac{a}{n}$$

because $\sin(x) \geq \frac{2\sqrt{2}}{\pi}x$ for $0 \leq x \leq \frac{\pi}{4}$ and $\frac{1}{2} \leq \tilde{c}(k) \leq \frac{N}{2}$. The above is equivalent to

$$\tilde{c}(k) \geq \frac{a}{n} \cdot \frac{2N+1}{4\sqrt{2}}\left(\sin\frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi\right)^{-1} + \frac{1}{2}. \tag{5.5}$$

We will now show that $\tilde{c}(k)$ satisfies $\tilde{c}(k) \geq c(k)$, and we can do this either directly or by showing that $\tilde{c}(k)$ satisfies inequality (5.5), which implies $\tilde{c}(k) \geq c(k)$. We will consider two cases.

Case 1 $k = 1, \ldots, \frac{N}{4}$

(This corresponds to the first case in (5.3), the definition of the $\tilde{c}(k)$.)

Assume first that $\tilde{c}(k) = \frac{N}{2}$. By inequality (5.2) we have

$$x_k \leq -\cos \frac{\frac{N}{4}}{N+\frac{1}{2}}\pi \leq -\cos\frac{\pi}{4} = -\frac{1}{\sqrt{2}},$$

and we know by the symmetry of the x_k that $x_{N/2+1} \geq 0$. Thus $x_k + \frac{a}{n} \leq -\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} = 0 \leq x_{N/2+1}$, so by setting $i = \frac{N}{2} + 1 - k$ we have $x_{k+i} \geq x_k + \frac{a}{n}$ and thus $c(k) \leq i \leq \frac{N}{2} = \tilde{c}(k)$.

Assume conversely that

$$\tilde{c}(k) = \frac{a}{n} \cdot \frac{2N+1}{4\sqrt{2}} \cdot \left(\sin \frac{2k-\frac{1}{2}}{2N+1}\pi\right)^{-1} + \frac{1}{2}.$$

Because $k \leq \frac{N}{4}$ and $\tilde{c}(k) \leq \frac{N}{2}$, we have $2k + \tilde{c}(k) \leq N$ and thus $\sin \frac{2k-\frac{1}{2}}{2N+1}\pi \leq \sin \frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi$. This means that $\tilde{c}(k)$ satisfies inequality (5.5).

Case 2 $k = \frac{N}{4} + 1, \ldots, \frac{N}{2}$.

(This corresponds to the second case in (5.3).)

Again, assume first that $\tilde{c}(k) = \frac{N}{2}$. By the symmetry of the x_k we know that $x_k \leq 0$, and by inequality (5.2) we have

$$x_{3N/4+1} \geq -\cos \frac{\frac{3N}{4}+\frac{1}{2}}{N+\frac{1}{2}}\pi \geq -\cos\frac{3\pi}{4} = \frac{1}{\sqrt{2}}.$$

Thus $x_k + \frac{a}{n} \leq \frac{1}{\sqrt{2}} \leq x_{3N/4+1}$, so by setting $i = \frac{3N}{4} + 1 - k$ we have $x_{k+i} \geq x_k + \frac{a}{n}$ and thus $c(k) \leq i \leq \frac{3N}{4} + 1 - (\frac{N}{4} + 1) = \frac{N}{2} = \tilde{c}(k)$.

Assume conversely that $\tilde{c}(k) = \frac{a}{n} \cdot \frac{2N+1}{4} + \frac{1}{2}$. Because $\frac{N}{4} + 1 \leq k \leq \frac{N}{2}$ and $\frac{1}{2} \leq \tilde{c}(k) \leq \frac{N}{2}$, we have $\frac{N}{2} + \frac{5}{2} \leq 2k + \tilde{c}(k) \leq \frac{3N}{2}$ and thus $\frac{1}{\sqrt{2}} = \sin(\frac{\pi}{4}) \leq \sin \frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi$. This means that $\tilde{c}(k)$ satisfies inequality (5.5).

We have thus proved that $\tilde{c}(k)$ is an upper bound for $c(k)$. Remembering that $c(k) - 1$ is the number of x_j in $(x_k, x_k + \frac{a}{n})$, we have:

$$\begin{aligned}
\#\left\{x_j : x_j \in \left(x_k, x_k + \frac{a}{n}\right)\right\} &\leq \tilde{c}(k) - 1 \\
&\leq \begin{cases} \frac{a}{n} \cdot \frac{2N+1}{4\sqrt{2}} \cdot \left(\sin \frac{2k-\frac{1}{2}}{2N+1}\pi\right)^{-1} - \frac{1}{2} & \text{for } k = 1, \ldots, \frac{N}{4} \\ \frac{a}{n} \cdot \frac{2N+1}{4} - \frac{1}{2} & \text{for } k = \frac{N}{4}+1, \ldots, \frac{N}{2}. \end{cases}
\end{aligned}$$

Using this, we can now estimate $C_1 + C_2$ (see equation (5.1)):

$$C_1 + C_2 \le 4 \sum_{k=1}^{N/2} \# \left\{ x_j : x_j \in \left(x_k, x_k + \frac{a}{n} \right) \right\} \le 4(C_l + C_r)$$

where

$$C_l := \sum_{k=1}^{N/4} \left[\frac{a}{n} \cdot \frac{2N+1}{4\sqrt{2}} \cdot \left(\sin \frac{2k - \frac{1}{2}}{2N+1} \pi \right)^{-1} - \frac{1}{2} \right]$$

and

$$C_r := \sum_{k=N/4+1}^{N/2} \left[\frac{a}{n} \cdot \frac{2N+1}{4} - \frac{1}{2} \right].$$

It is obvious that

$$C_r = \frac{N}{4} \left(\frac{a}{n} \cdot \frac{2N+1}{4} - \frac{1}{2} \right) = \frac{aN(2N+1)}{16n} - \frac{N}{8}.$$

Turning to C_l, we see that

$$C_l = \frac{a(2N+1)}{4\sqrt{2}n} \left(\left(\sin \frac{\frac{3}{2}}{2N+1} \pi \right)^{-1} + \sum_{k=2}^{N/4} \left(\sin \frac{2k - \frac{1}{2}}{2N+1} \pi \right)^{-1} \right) - \frac{N}{8}$$

and we can estimate

$$\left(\sin \frac{\frac{3}{2}}{2N+1} \pi \right)^{-1} \le \frac{\pi}{2\sqrt{2}} \cdot \frac{4N+2}{3\pi} = \frac{2N+1}{3\sqrt{2}}$$

using $\sin x \ge \frac{2\sqrt{2}}{\pi} x$ for $0 \le x \le \frac{\pi}{4}$. (We have $\frac{3/2}{2N+1} \pi \le \frac{1}{4} \pi$ because $N \ge 4$.)

To estimate the sum $S := \sum_{k=2}^{N/4} \left(\sin \frac{2k - \frac{1}{2}}{2N+1} \pi \right)^{-1}$, we note that it is the lower sum for an integral:

$$\begin{aligned}
\frac{2\pi}{2N+1} S &\le \int_{\frac{3\pi}{4N+2}}^{\frac{N/2 - 1/2}{2N+1} \pi} \frac{1}{\sin x} \mathrm{d}x \\
&\le \int_{\frac{3\pi}{4N+2}}^{\frac{\pi}{4}} \frac{1}{\sin x} \mathrm{d}x \\
&= \left[\ln \tan \frac{x}{2} \right]_{x = \frac{3\pi}{4N+2}}^{\frac{\pi}{4}} \\
&= \ln \tan \frac{\pi}{8} - \ln \tan \frac{3\pi}{8N+4} \\
&\le \ln \tan \frac{\pi}{8} - \ln \frac{3\pi}{8N+4} \\
&\le \ln \frac{1}{2} + \ln \frac{8N+4}{3\pi} \\
&= \ln \frac{4N+2}{3\pi}.
\end{aligned}$$

With this, we obtain

$$\begin{aligned} C_l &\leq \frac{a(2N+1)}{4\sqrt{2}n}\left(\frac{2N+1}{3\sqrt{2}}+\frac{2N+1}{2\pi}\ln\frac{4N+2}{3\pi}\right)-\frac{N}{8} \\ &= \frac{a(2N+1)^2}{4\sqrt{2}n}\left(\frac{1}{3\sqrt{2}}+\frac{1}{2\pi}\ln\frac{4N+2}{3\pi}\right)-\frac{N}{8} \end{aligned}$$

and thus

$$\begin{aligned} C_1+C_2 \leq 4(C_l+C_r) &\leq \frac{a(2N+1)^2}{\sqrt{2}n}\left(\frac{1}{3\sqrt{2}}+\frac{1}{2\pi}\ln\frac{4N+2}{3\pi}\right)+\frac{aN(2N+1)}{4n}-N \\ &\leq \frac{a(2N+1)^2}{n}\left(\frac{1}{6}+\frac{1}{8}+\frac{1}{2\sqrt{2}\pi}\ln\frac{4N+2}{3\pi}\right)-N \\ &= \frac{a(2N+1)^2}{n}\left(\frac{7}{24}+\frac{1}{2\sqrt{2}\pi}\ln\frac{4N+2}{3\pi}\right)-N. \end{aligned}$$

In all, we have

$$C = C_1+C_2+C_3 = C_1+C_2+N \leq \frac{a(2N+1)^2}{n}\left(\frac{7}{24}+\frac{1}{2\sqrt{2}\pi}\ln\frac{4N+2}{3\pi}\right).$$

■

If we set $n = N$, as was done in the spherical filter algorithm, then Lemma 5.1 tells us that step 4 of the NFFT summation algorithm (Algorithm 4.3) has a complexity of $\mathcal{O}(aN\log N)$. This implies an overall complexity for the NFFT summation of $\mathcal{O}\big(\sigma N\log(\sigma N)+(m+a\log N)N\big)$.

Since step 2 of the fast spherical filter (Algorithm 5.1) calls the NFFT summation $\mathcal{O}(N_{\lim})$ times, it has a complexity of $\mathcal{O}\big(N_{\lim}\cdot(\sigma N\log(\sigma N)+(m+a\log N)N)\big)$, as stated in Section 5.1.

6. Implementation and Numerical Experiments

In this section, we present an implementation of the fast spherical filter algorithm. Numerical experiments will be performed on this implementation to test its accuracy and speed. We will also discuss a characteristic type of error produced by the NFFT summation algorithm that was discovered while testing the wavelet decomposition.

6.1. Implementation

The spherical filter algorithm and the auxiliary algorithms were implemented in C++ using double-precision arithmetic. The linear algebra library CLAPACK 3.0 (from http://www.netlib.org) was used to solve the eigenproblem that occurs in the computation of the Gauss-Legendre nodes and weights (see Appendix A) and to solve the system of equations that occurs in the NFFT summation algorithm (Algorithm 4.3). The FFT library FFTW 2.1.3 (from http://www.fftw.org) was used to evaluate the FFTs that occur in the various algorithms. The compiler used was GCC 2.95.2 (all optimisations turned on). Numerical experiments were run under Linux 2.2.14 on a 550 MHz Pentium III with 256 MB of memory.

6.2. Numerical Experiments

We will first examine the errors that are generated by the approximate algorithms. Besides testing the spherical filter as a whole, we also performed tests on the NFFT summation algorithm in isolation to assess the accuracy of the error analysis that was carried out in Section 4.3. To test the algorithm under the same conditions under which it is used in the spherical filter, we set $M = N$ and $n = N$ and used Legendre nodes for the x_k. We also replaced the NFFT with an NDFT to make sure we were measuring only the errors occurring in the summation algorithm itself, since these are the errors that were estimated in Section 4.3. The coefficients β_k were set to $\beta_k := 0$ if $k \neq N/2$ and $\beta_{N/2} := 1$. This was intended to minimise the estimation error from Hölder's inequality in equation (4.10) and to allow us to focus on the accuracy of the estimate for $|K_{ER}(x)|$, which is the core part of the error analysis and the place where the parameters a and p come into play.

Figure 6.1 shows the predicted and actual maximum errors for $N = 1024$ and $a = p = 4, \ldots, 12$. Both errors decay exponentially, but the predicted error is too pessimistic by a factor of about 10^3 and does not decay quite as quickly the actual error.

Next, we will examine the influence of the parameters a, p and m on the accuracy of the spherical filter algorithm in its entirety. Potts and Steidl [PoSt02] recommend setting $a = p = m$ for $p \leq 5$ and $a = p$, $m = 5$ for $p > 5$.

Experiments were performed to assess whether these recommendations are also valid in the context of the spherical filter. The relative error between the exact solution and the approximate solution was computed as

$$E := \frac{\|\mathbf{F}^{N_{\text{lim}}} - \tilde{\mathbf{F}}^{N_{\text{lim}}}\|_{\text{F}}}{\|\mathbf{F}^{N_{\text{lim}}}\|_{\text{F}}}.$$

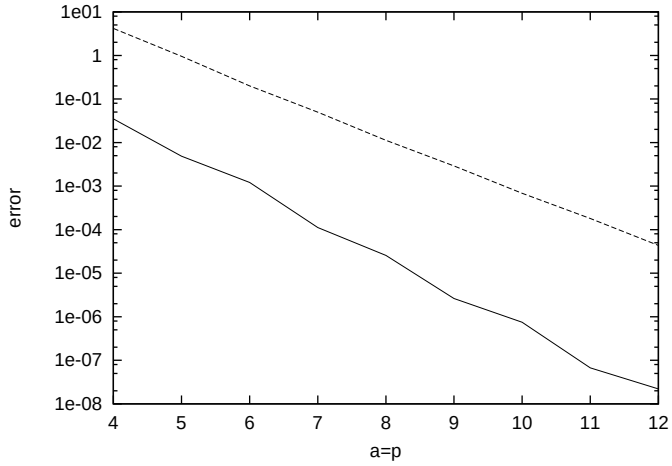


Figure 6.1: Predicted maximum error (dashed) and actual maximum error (solid) for the NFFT summation algorithm (N=1024).

($\mathbf{F}^{N_{\text{lim}}} := \left(f^{N_{\text{lim}}}(\theta_s, \varphi_t)\right)_{s=0,\ t=0}^{N-1\ 2N-1}$ is the exact solution, $\tilde{\mathbf{F}}^{N_{\text{lim}}} := \left(\tilde{f}^{N_{\text{lim}}}(\theta_s, \varphi_t)\right)_{s=0,\ t=0}^{N-1\ 2N-1}$ is the approximate solution as computed by Algorithm 5.1, and $\|\cdot\|_{\text{F}}$ is the Frobenius norm.) The input data for the spherical filter were random values taken from $[0, 1]$, the grid resolution was $N = 512$, and the error was averaged over ten runs with different input data. The influence of the oversampling factor σ was not tested; the setting $\sigma = 2$ was used in all cases, which is the same value used by Potts and Steidl in their experiments.

Figure 6.2 shows the error on a logarithmic scale for $1 \leq a, p \leq 12$, with $m = 12$ to keep the error from the NFFT small. The behaviour of the error with varying a and p agrees well with the theoretical prediction about the behaviour of the NFFT summation algorithm. The figure supports the recommendation by Potts and Steidl to set $a = p$.

Figure 6.3 shows the error for $1 \leq a, m \leq 12$ with $p = a$. For fixed a, the error decreases exponentially in m, but only up to about $m = a - 3$; therefore, we recommend setting $m = a - 3$, since higher values do not seem to lead to higher precision.

Next, we will examine the execution time of Algorithm 5.1 compared to the exact algorithm that is obtained by computing the sums $\mathfrak{S}_1$ and $\mathfrak{S}_2$ in step 2 of Algorithm 5.1 exactly. For the approximate algorithm, we set $a = p = 8$ and $m = 5$, which, according to the results above, results in a relative error of about $2 \cdot 10^{-8}$. Figure 6.4 shows the execution times for the exact algorithm and the approximate algorithm for different values of N. The two dashed lines show time complexities of $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$. The $\mathcal{O}(N^3)$ time complexity of the exact algorithm is obvious, whereas the execution times of the approximate algorithm grow only slightly faster than $\mathcal{O}(N^2)$. For this reason, though the exact algorithm is faster for small N, the execution times are about equal for $N = 64$, and for large N, the approximate algorithm is significantly faster — 142 seconds for $N = 1024$ compared to 1902 seconds for the exact algorithm.

Finally, we will compare the individual times required by the three steps in Algorithm

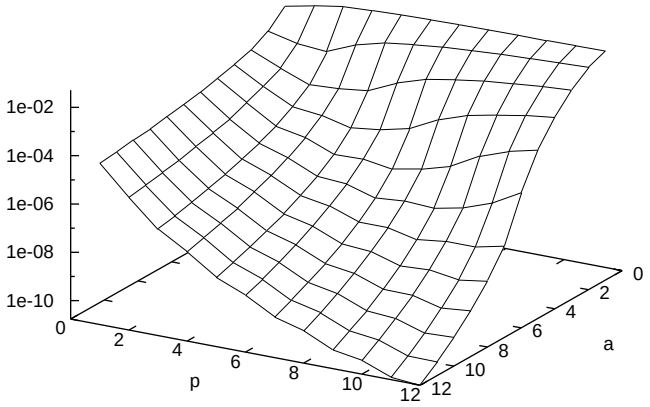


Figure 6.2: Relative error E in the result of the fast spherical filter algorithm for $1 \leq a, p \leq 12$, $N = 512$. The NFFT parameter m was set to 12 in all cases.

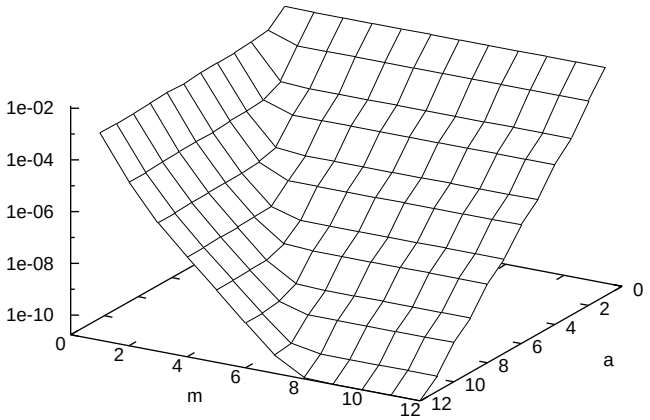


Figure 6.3: Relative error E in the result of the fast spherical filter algorithm for $1 \leq a, m \leq 12$ with $p = a$, $N = 512$.

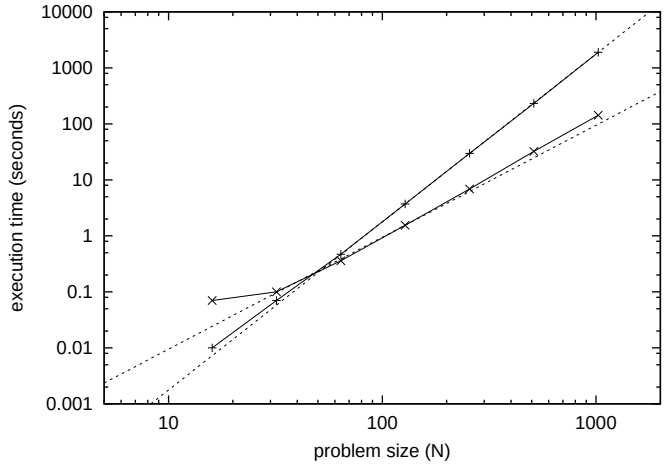


Figure 6.4: Execution times of the exact algorithm (plus signs) and the approximate algorithm (crosses). The dashed lines show time complexities of $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$; they intercept the plots at $N = 128$.

5.1. Steps 1 and 3 compute FFTs, whereas step 2 uses the NFFT summation algorithm to evaluate the Christoffel-Darboux formula. All three steps have an asymptotic complexity of $\mathcal{O}(N^2 \log N)$, but we are of course also interested in the hidden constant. Figure 6.5 shows two plots, one for the time required in step 2 (solid) and the other for the time required in steps 1 and 3 combined (dashed). While both the FFT and the NFFT summation algorithm show the same asymptotic behaviour, the latter is substantially more expensive to compute — the times differ by a factor of about 40.

6.3. Wavelet Decomposition

As remarked in Section 3.1, a wavelet decomposition on the sphere can be computed using the spherical filter. The low-frequency component is simply $\tilde{f}^{N-1}$, and the wavelet component is $\tilde{g} = \tilde{f}^{N-1} - \tilde{f}^{N/2-1}$.

Since the wavelet component usually has quite a small magnitude compared to the low-frequency component, a lot of cancellation occurs in the subtraction $\tilde{f}^{N-1} - \tilde{f}^{N/2-1}$. This can lead to numerical problems since a small relative error in $\tilde{f}^{N-1}$ or $\tilde{f}^{N/2-1}$ causes a much larger relative error in the wavelet component $\tilde{g}$, and as we will see, this problem is very real when wavelet decompositions are computed using the fast spherical filter.

To test the wavelet decomposition, the function

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

was used; this function was taken from [PST96]. If the function values are interpreted as distances from the origin, then this function describes a half-sphere that is joined to

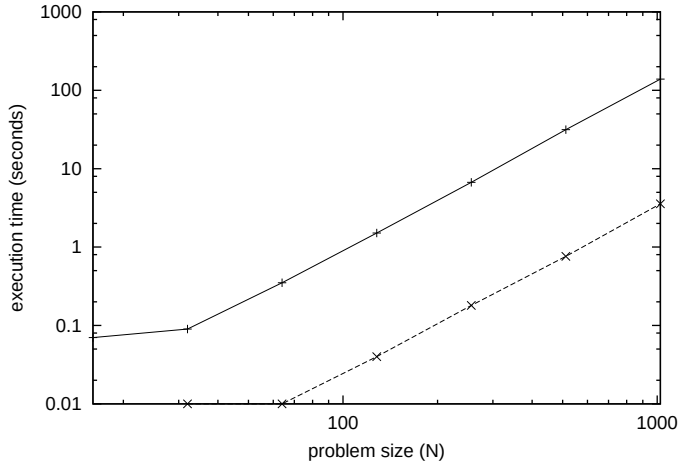


Figure 6.5: Execution times for the individual steps in the fast spherical filter (Algorithm 5.1). The solid line plots the time taken by step 2, the dashed line plots the time for steps 1 and 3 combined.

a half-ellipsoid. It is smooth everywhere except at the equator; this discontinuity should show up in the wavelet component.

A wavelet decomposition was carried out on the function f_{test} rotated around the x_1 axis by an angle of $\pi/6$. (A grid resolution of $N = 1024$ was used, and the NFFT summation parameters were set to $a = p = 8$ and $m = 5$.) The corresponding wavelet component is shown in Figure 6.6. Two things are apparent. Firstly, the discontinuity shows up in the wavelet component, as expected. Secondly, and rather unexpectedly, there are prominent bands at $\theta = \frac{\pi}{4}$ and $\theta = \frac{3\pi}{4}$. These are artefacts caused by the NFFT summation algorithm.

Note the scale to the right of the image. The amplitude of the wavelet component is about 10^{-6}, and that of the artefacts about $2 \cdot 10^{-7}$. It is evident that, as discussed above, a lot of cancellation is taking place in the computation $\tilde{g} = \tilde{f}^{N-1} - \tilde{f}^{N/2-1}$, and this is causing the errors from the NFFT summation algorithm, which have about the same magnitude as the wavelet component, to become apparent.

Note also that the rotation of the discontinuity has not transferred to the artefacts — they run parallel to the equator. It appears that the artefacts are quite independent of the properties of the wavelet component; in fact, almost the same artefacts were observed when a wavelet decomposition was carried out on the constant function $f(\theta, \varphi) = 1$.

The amplitude of these artefacts can of course be reduced by increasing the NFFT summation parameters a and p, and the results obtained by doing this will be presented in a moment. (The parameter m, which controls the accuracy of the NFFT, was found to have no visible effect on the artefacts. This indicates that they are caused by the NFFT summation algorithm and not by the underlying NFFT.)

First, however, we will examine the artefacts in a little more detail. Of course, the fact that an error is occurring is not surprising — after all, we are using an approximate algorithm. Nor is the magnitude of the error excessive — an error of 10^{-7} is within

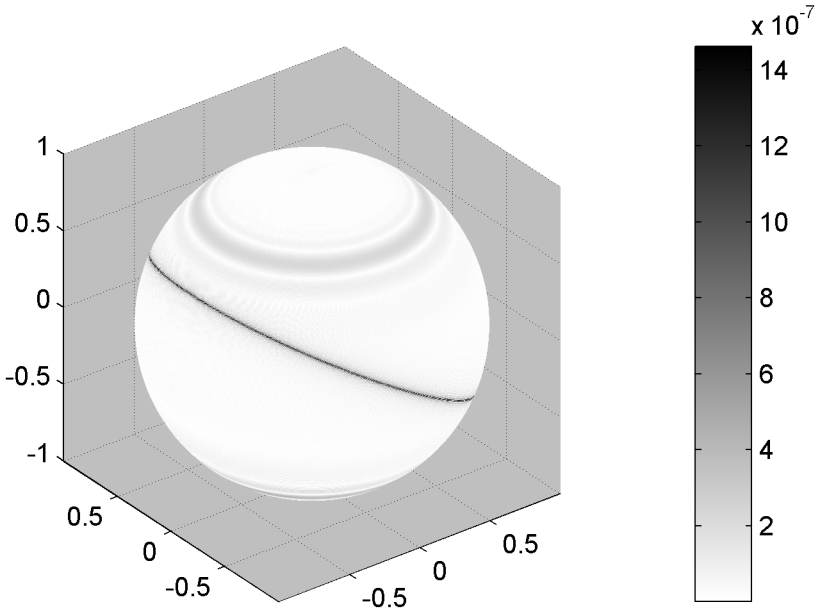


Figure 6.6: Wavelet component (absolute values) of the function f_{test} rotated around the x_1 axis by an angle of $\pi/6$. The grid resolution was $N = 1024$, and the NFFT summation parameters were $a = p = 8$ and $m = 5$. (The slight Moiré effect was caused by the software used to render the image and not by the wavelet decomposition.) The prominent bands at around $\theta = \frac{\pi}{4}$ and $\theta = \frac{3\pi}{4}$ are artefacts caused by the NFFT summation algorithm.

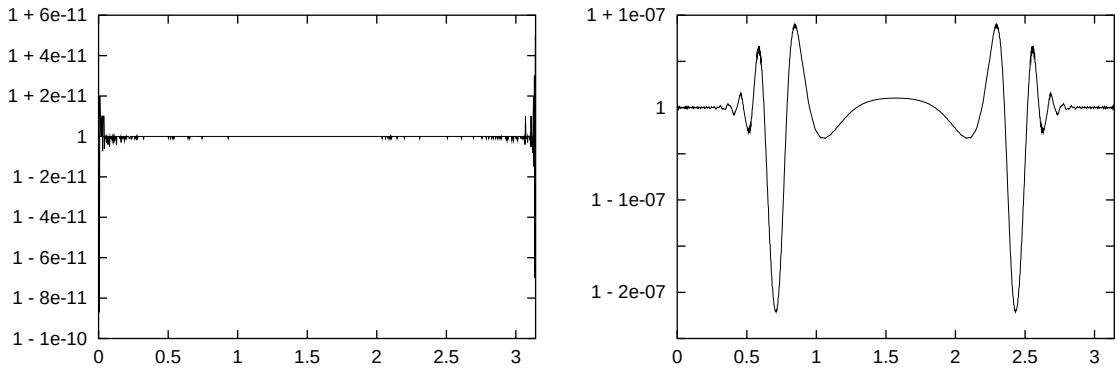


Figure 6.7: $\mathfrak{g}_0^{N-1}(\theta)$ (left) and $\mathfrak{g}_0^{N/2-1}(\theta)$ (right), the results that were obtained by applying step 2 of the fast spherical filter to the constant function $\mathfrak{f}_0(\theta) = 1$. The NFFT was replaced by an NDFT. Parameter values were $a = p = 8$, the resolution was $N = 1024$.

the range that the results from Section 6.2 would lead one to expect for a setting of $a = p = 8$ and $m = 5$. What is surprising, however, is that the error has a such a regular structure and that it is so strong even for very simple inputs like $f(\theta, \varphi) = 1$.

The exact cause for the artefacts has not been determined yet, but it is clear that they originate in the NFFT summation algorithm. It seems that the higher-derivative discontinuities at the join points of the regularised kernel K_R lead to oscillations in the Fourier series K_{RF}, and that under certain circumstances the oscillations from the various translates of K_{RF} can add up to generate the artefacts we have seen; apparently, this happens when the distance between the nodes x_s is about the same as the period of the oscillations in K_{RF} or an integer multiple of this period.

We will examine step 2 of Algorithm 5.1 in isolation, since this is the step where the NFFT summation algorithm is employed. Additionally, we will set $n = 0$, thus obtaining an algorithm that operates on one-dimensional functions $\mathfrak{f}_0(\theta)$ and yields the output $\mathfrak{g}_0^{N_{\text{lim}}}(\theta)$ — the superscript indicates different values of N_{lim}. (This is equivalent to applying the original spherical filter algorithm to functions $f(\theta, \varphi)$ that depend only on θ.) For all of our tests, we will replace the NFFT with an NDFT to eliminate any additional errors caused by the NFFT.

Since the wavelet decomposition involves applying the spherical filter twice, once for $N_{\text{lim}} = N - 1$ and once for $N_{\text{lim}} = N/2 - 1$, we will first look at the results of these two operations separately. Figure 6.7 shows the results $\mathfrak{g}_0^{N-1}(\theta)$ and $\mathfrak{g}_0^{N/2-1}(\theta)$ that were obtained by applying step 2 of the algorithm to the constant function $\mathfrak{f}_0(\theta) = 1$. What is striking is that artefacts are only visible in $\mathfrak{g}_0^{N/2-1}(\theta)$, whereas $\mathfrak{g}_0^{N-1}(\theta)$ merely contains small numerical errors. This reason for this has not been established, and one might hope that a better understanding of this phenomenon could allow the artefacts in $\mathfrak{g}_0^{N/2-1}(\theta)$ to be eliminated as well. As we will see, however, it is doubtful whether $\mathfrak{g}_0^{N-1}(\theta)$ will be free of artefacts for general $\mathfrak{f}_0(\theta)$. In any case, we will now concentrate on $\mathfrak{g}_0^{N/2-1}(\theta)$, since this is where the artefacts occur in our case.

Figure 6.8 shows the "wavelet component" $\mathfrak{g}_0^{N-1}(\theta) - \mathfrak{g}_0^{N/2-1}(\theta)$ of $\mathfrak{f}_0(\theta) = 1$ for parameter values of $a = 8$, $p = 4, \ldots, 11$. Increasing p reduces the size of the artefacts, as expected, but what is interesting is the effect that p has on the shape of the artefacts: For even p, the artefacts are smooth, while for odd p, they are quite "noisy". A second pattern is that for $p = 4, 5, 8, 9$, the largest peak is positive, while for $p = 6, 7, 10, 11$, it is negative.

The pattern in the shape of the artefacts, which repeats after four steps, is quite reminiscent of the pattern exhibited by the sine function and its derivatives (sin, cos, $-\sin$, $-\cos$, sin, ...), and one may suspect that this pattern is caused by the sine series used in the regularised kernel K_R in Section 4.2 — particularly since p is the parameter that controls the number of terms in the sine series.

This suspicion is strengthened even more when one examines K_{ER}, the difference between the regularised kernel K_R and the Fourier series K_{RF} used to approximate it. Figure 6.9 shows plots of K_{ER} for $a = 8$ and $p = 4, \ldots, 7$. The same kind of pattern that was observed in the artefacts is observed here, too; again, there is a similarity in shape between the plots with even p on the one hand and those with odd p on the other.

The similarity between the properties of the artefacts and those of K_{ER} is not surprising when one considers that the error incurred in the NFFT summation algorithm is

$$\sum_{k=1}^{N} \beta_k K_{ER}(y - x_k).$$

We conjecture that for certain constellations of coefficients β_k, nodes x_k, and position of evaluation y, the terms in this sum add up to produce a large error. For example, if the distances between those x_k that lie close to y are about equal to the distance between the peaks in K_{ER} (or an integer multiple thereof), then the peaks of the various translates $K_{ER}(y - x_k)$ will coincide, and a large error can result. Of course, whether or not all of the small errors do add up to give a large error depends on the values of the β_k.

It seems plausible that this mechanism is responsible for the artefacts that were observed in the wavelet decompositions. The distances between the Legendre nodes change continuously over the interval $[-1, 1]$. The mechanism described would mean that in places where the node distances are close to the distance between the peaks in K_{ER} (or an integer multiple thereof), a large error results, whereas in between, the error is small. This would explain the large oscillations in the error that were observed, instead of an error that is of fairly uniform magnitude over the whole interval.

Of course, whether or not artefacts occur also depends critically on the values of the β_k — this is the only difference between $\mathfrak{g}_n^{N/2-1}(\theta)$ and $\mathfrak{g}_n^{N-1}(\theta)$, of which the former exhibits artefacts while the latter does not. In our application, we have $\beta_k := w_k \mathfrak{f}_n(\theta_k) P_{N_{\text{lim}}+1}^{|n|}(x_k)$ for the first sum and $\beta_k := w_k \mathfrak{f}_n(\theta_k) P_{N_{\text{lim}}}^{|n|}(x_k)$ for the second sum. Inspecting these formulas seems to suggest that artefacts can also occur in $\mathfrak{g}_n^{N-1}(\theta)$ for some $\mathfrak{f}_n(\theta)$, for in choosing $\mathfrak{f}_n(\theta)$, we have complete control over the coefficients β_k. By choosing $\mathfrak{f}_n(\theta)$ suitably, we could make the coefficients for $N_{\text{lim}} = N - 1$ exhibit exactly the same undesirable behaviour that produced the artefacts in the case $N_{\text{lim}} = N/2 - 1$.

Because of this, is does not seem as if the artefacts can be removed through some

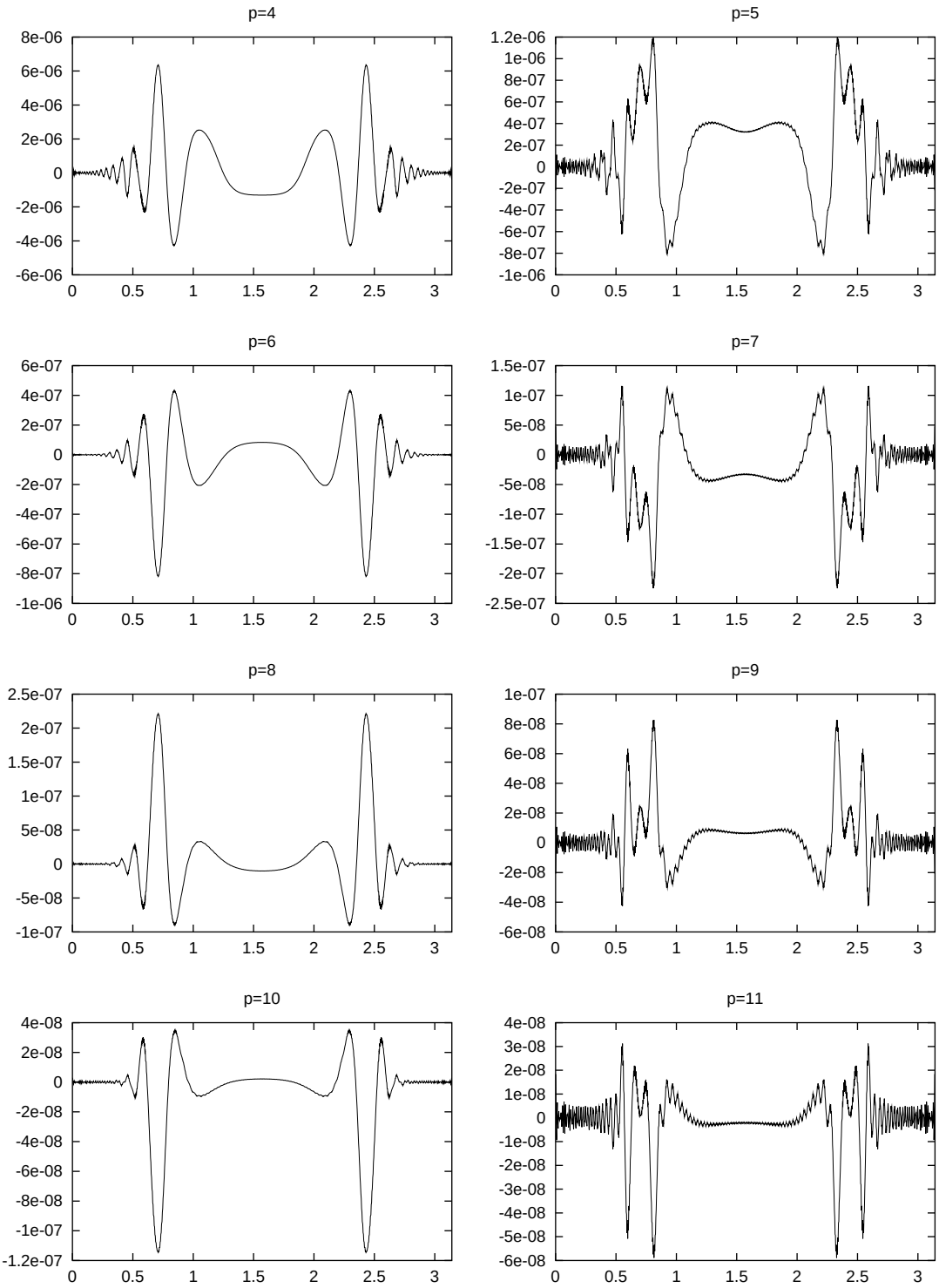


Figure 6.8: "Wavelet component" $\mathfrak{g}_0^{N-1}(\theta) - \mathfrak{g}_0^{N/2-1}(\theta)$ that was obtained by applying step 2 of the spherical filter algorithm to $\mathfrak{f}_0(\theta) = 1$. The NFFT was replaced by an NDFT. Parameter values were $a = 8$ and $p = 4, \ldots, 11$, the resolution was $N = 1024$.

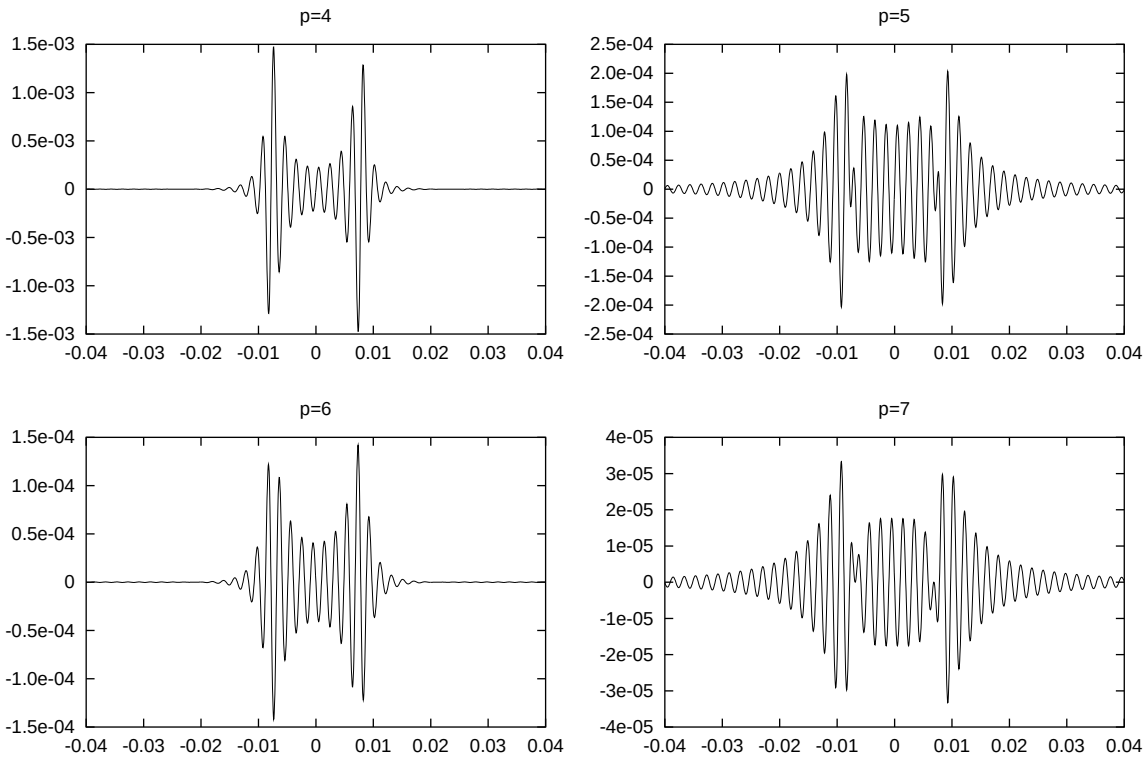


Figure 6.9: K_{ER}, the difference between the regularised kernel K_R and the Fourier series K_{RF} used to approximate it, for $a = 8$, $p = 4, \ldots, 7$ and $N = 1024$.

simple measure. It appears that they are a direct consequence of the oscillations in the approximation error K_{ER}, and it is unlikely that this problem can be fixed without sacrificing the basic idea of approximating the regularised kernel K_R by a Fourier series. It would be interesting to investigate whether other fast approximate summation algorithms, such as Fast Multipole Methods, produce more satisfying results. If the NFFT summation algorithm is to be used, however, we have no real remedy to suggest except to increase a and p.

One fix that was tried was to apply the spherical filter a second time to the already filtered $\tilde{f}^{N-1}$ and $\tilde{f}^{N/2-1}$, hoping that the same artefacts would occur again. Taking the difference between the two results would leave only the artefacts, which could then be subtracted from the wavelet component. Thus, instead of $\tilde{g} := \tilde{f}^{N-1} - \tilde{f}^{N/2-1}$, we computed

$$\tilde{g} := \tilde{f}^{N-1} - \tilde{f}^{N/2-1} - \left((\tilde{f}^{N-1})^{N-1} - \tilde{f}^{N-1}\right) + \left((\tilde{f}^{N/2-1})^{N/2-1} - \tilde{f}^{N/2-1}\right).$$

This appeared to work well, though we can give no mathematical justification apart from the observation that, if the spherical filter were being evaluated exactly, $\tilde{f}^{N-1} - (\tilde{f}^{N-1})^{N-1}$ and $\tilde{f}^{N/2-1} - (\tilde{f}^{N/2-1})^{N/2-1}$ would be identical to zero because the spherical filter is an idempotent operation. This modification almost eliminates the artefacts in the examples we examined, but it doubles the execution time because the spherical filter has to be applied twice for both values of N_{lim}. For this reason, it does not really make sense to use this modification in practice, because we found that increasing a and p reduced the artefacts just as well with less increase in execution time.

One small improvement that we can offer over the NFFT summation algorithm as presented in [PoSt02] is described in the following remark.

Remark 6.1 If the definition of $K_{NE} := K - K_R$ in the NFFT summation algorithm is modified to $K_{NE} := K - K_{RF}$, then the Fourier series approximation error $K_{ER} := K_R - K_{RF}$ is eliminated within the near field without incurring additional execution time; only the precomputation time is increased slightly because K_{RF} is more expensive to evaluate than K_R. □

Figure 6.10 shows that the modified K_{NE} reduces the size of the artefacts considerably.

To close this section, we will present results that were obtained by using the modified K_{NE} combined with increased values for the parameters a, p and m. Figure 6.11 shows the result of a wavelet decomposition on the same function as in Figure 6.6, the function f_{test} rotated around the x_1 axis by an angle of $\pi/6$. This time, however, the modified K_{NE} was used, and the NFFT summation parameters were set to $a = p = 10$ and $m = 7$. These combined measures have reduced the magnitude of the error so that it is no longer visible.

As a final example, consider Figure 6.12. This time, the function f_{test} was rotated around the x_1 axis by an angle of $\pi/2$, causing the discontinuity to run through the poles. (The parameters were the same as for the last image.) This example illustrates that discontinuities at the poles are also resolved well by the wavelet decomposition. This underlines the usefulness of the property of uniform resolution which the band-limited functions possess.

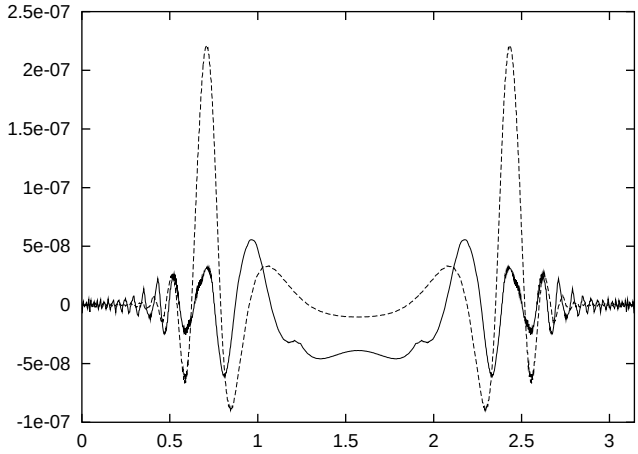


Figure 6.10: Effect of modifying K_{NE} (see Remark 6.1). The graph shows the "wavelet component" of $\mathfrak{f}_0(\theta) = 1$ for the original K_{NE} (dashed) and the modified K_{NE} (solid). Parameter values were $a = p = 8$, and the resolution was $N = 1024$. The NFFT was replaced by an NDFT.

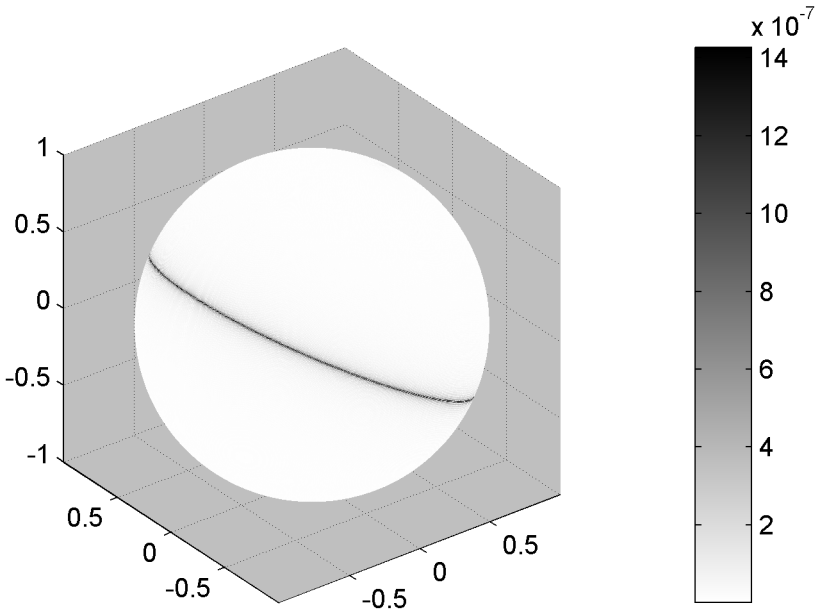


Figure 6.11: Wavelet component (absolute values) of the function f_{test} rotated around the x_1 axis by an angle of $\pi/6$. The NFFT summation parameters were increased to $a = p = 10$ and $m = 7$, and the modified K_{NE} was used. The artefacts are no longer visible. (Grid resolution is $N = 1024$.)

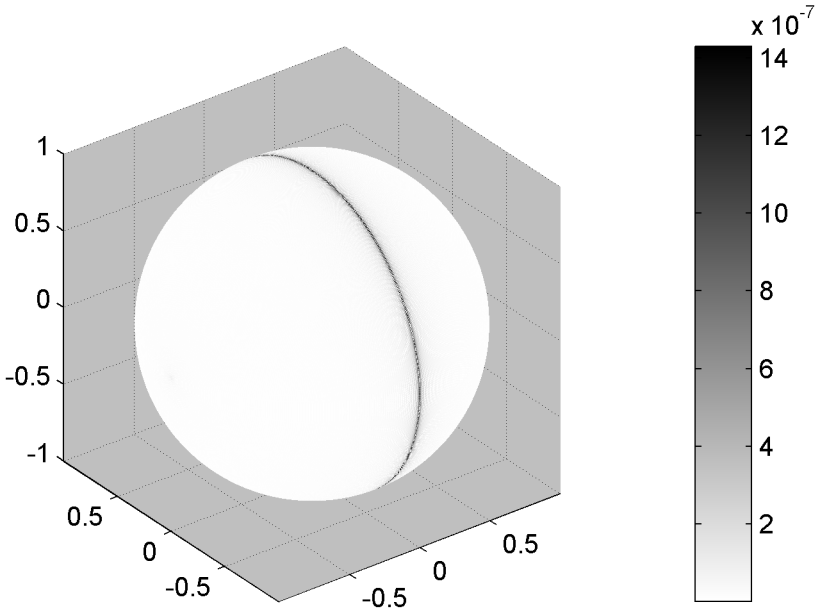


Figure 6.12: Wavelet component (absolute values) of the function f_{test} rotated around the x_1 axis by an angle of $\pi/2$. This example illustrates that discontinuities are resolved equally well at the poles as on the rest of the sphere. NFFT summation parameters were $a = p = 10$ and $m = 7$, and the modified K_{NE} was used.

7. Conclusion

We have presented a fast algorithm for the spherical filter and for wavelet decomposition on the sphere. The algorithm employs a fast approximate summation scheme, resulting in a sizeable reduction in execution times at high resolutions compared to the straightforward approach. As we have proved, the algorithm has a time complexity of $\mathcal{O}(N^2 \log N)$, which is a considerable improvement over the $\mathcal{O}(N^3)$ complexity of the straightforward algorithm. Numerical experiments show that the error produced by the approximate algorithm can be controlled well using various parameters, allowing the tradeoff between accuracy and execution time to be tuned to the requirements of the problem at hand. Controlling the accuracy of the algorithm is especially important when it is used to perform wavelet decompositions because a lot of cancellation occurs in the calculation of the wavelet component, thus magnifying the numerical errors.

The fast spherical filter algorithm makes use of the fact that the Christoffel-Darboux formula can be applied when the forward and reverse discrete spherical Fourier transforms are combined. Accelerating the forward or reverse transform alone is more difficult. Schemes have been proposed which would, in theory, yield $\mathcal{O}(N^2 \log^2 N)$ algorithms. Driscoll and Healy present such an algorithm [DH94], but it suffers from numerical instability, and modifications designed to stabilise the algorithm (e.g. [HRKM98], [PST98]) appear to increase the computational complexity again. Mohlenkamp [Moh99] presents two algorithms, one with a complexity of $\mathcal{O}(N^{5/2} \log N)$ and the other with a complexity of $\mathcal{O}(N^2 \log^2 N)$. Both appear to be numerically stable, but on the grid resolutions tested by Mohlenkamp (up to $N = 2048$), the latter does not perform as well as the asymptotic complexity suggests and is actually slower than the $\mathcal{O}(N^{5/2} \log N)$ algorithm. Because of the difficulties encountered in developing asymptotically fast spherical transforms, research is still going on into fast $\mathcal{O}(N^3)$ algorithms; see, for example, [SwSp00].

For the spherical filter, however, the last few years have brought several reliable $\mathcal{O}(N^2 \log N)$ algorithms. It would be interesting to compare execution times for the algorithms based on Fast Multipole Methods (FMM) ([JA97], [YaRo98]) with the Nonequispaced Fast Fourier Transform (NFFT) summation approach discussed here. In terms of conceptual complexity and ease of implementation, the NFFT summation approach seems to be simpler (and more pleasing mathematically) than the rather technical FMM algorithms.

Both schemes are beginning to rival $\mathcal{O}(N^3)$ algorithms on the grid sizes currently in use. As grid resolution increases in the years to come, $\mathcal{O}(N^2 \log N)$ algorithms will only become more attractive.

Acknowledgements

I thank Dr. Daniel Potts for his generous support and guidance.

A. Determining Nodes and Weights in Gaussian Quadrature

Gaussian quadrature allows the exact integration of polynomials of degree $2n+1$ by sampling them at $n+1$ points. In this section, we will derive a method for computing the nodes and weights that occur in Gaussian quadrature. The derivation is based on [Sto99, Section 3.6] but is more detailed and, in the author's opinion, easier to follow because of this.

We will begin by stating, without proof, the principal result of Gaussian quadrature. We will assume for the rest of this section that $w : [-1,1] \to \mathbb{R}$ is a positive weight function that is continuous on $[-1,1]$, that $p_j(x)$ $(j = 0, 1, \ldots)$ are the orthogonal polynomials with respect to the scalar product $\langle f, g\rangle := \int_{-1}^{1} f(x)\, g(x)\, w(x)\, \mathrm{d}x$ and that the three-term recurrence they satisfy is given by

$$\beta_{n-1} p_{n-1} + \alpha_n p_n + \gamma_{n+1} p_{n+1} = x p_n \tag{A.1}$$

(with $p_{-1}(x) := 0$ and $p_0(x) := 1$).

Theorem A.1 *Let $x_j \in \mathbb{R}$ $(j = 1, \ldots, n)$ be the roots of p_n, then there exists a uniquely determined set of weights w_j $(j = 1, \ldots, n)$ such that*

$$\sum_{j=1}^{n} p(x_j)\, w_j = \int_{-1}^{1} p(x)\, w(x)\, \mathrm{d}x$$

for all polynomials p of degree $2n-1$ or less.

(This theorem is proved in [Sto99, Section 3.6].) The roots (or *nodes*) x_j and the weights w_j can be determined by solving an eigenproblem, as we will now show.

Theorem A.2 *The roots x_j $(j = 1, \ldots, n)$ of p_n are the eigenvalues of the symmetric $n \times n$ matrix*

$$\tilde{\mathbf{A}} = \begin{pmatrix} \alpha_0 & \sqrt{\beta_0\gamma_1} & & & \\ \sqrt{\beta_0\gamma_1} & \alpha_1 & \sqrt{\beta_1\gamma_2} & & \\ & \sqrt{\beta_1\gamma_2} & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \sqrt{\beta_{n-2}\gamma_{n-1}} \\ & & & \sqrt{\beta_{n-2}\gamma_{n-1}} & \alpha_{n-1} \end{pmatrix},$$

and the weights w_j $(j = 1, \ldots, n)$ in Theorem A.1 can be obtained through $w_j = \langle p_0, p_0\rangle \left(\frac{v_j^{(1)}}{\|\mathbf{v}_j\|_2}\right)^2$, where $\mathbf{v}_j$ is the eigenvector of $\tilde{\mathbf{A}}$ for the eigenvalue x_j and $v_j^{(1)}$ is the first component of $\mathbf{v}_j$.

Proof. The three-term recurrence (A.1) for $n = 0, \ldots, n-1$ yields a system of equations that can be written as follows in matrix-vector form.

$$\begin{pmatrix} \alpha_0 & \gamma_1 & & & \\ \beta_0 & \alpha_1 & \gamma_2 & & \\ & \beta_1 & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \gamma_{n-1} \\ & & & \beta_{n-2} & \alpha_{n-1} \end{pmatrix} \begin{pmatrix} p_0(x) \\ p_1(x) \\ p_2(x) \\ \vdots \\ p_{n-1}(x) \end{pmatrix} + \gamma_n \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ p_n(x) \end{pmatrix} = x \begin{pmatrix} p_0(x) \\ p_1(x) \\ p_2(x) \\ \vdots \\ p_{n-1}(x) \end{pmatrix}.$$

We will denote the tridiagonal matrix in this equation by $\mathbf{A}$ and the vector $(p_k(x))_{k=0}^{n-1}$ by $\mathbf{p}(x)$. Let $x_1, \ldots, x_n$ be the roots of p_n. Setting $x = x_j$ $(j = 1, \ldots, n)$ in the equation above yields $\mathbf{A}\mathbf{p}(x_j) = x_j \mathbf{p}(x_j)$, i.e. the x_j are the eigenvalues of $\mathbf{A}$ with eigenvectors $\mathbf{p}(x_j)$. (The $\mathbf{p}(x_j)$ are all non-zero because $p_0(x_j) = 1$.)

$\mathbf{A}$ may be factorised as follows.

$$\mathbf{A} = \mathbf{D}^{-1}\tilde{\mathbf{A}}\mathbf{D} = \begin{pmatrix} 1 & & & & \\ & \sqrt{\frac{\beta_0}{\gamma_1}} & & & \\ & & \sqrt{\frac{\beta_0\beta_1}{\gamma_1\gamma_2}} & & \\ & & & \ddots & \\ & & & & \sqrt{\frac{\beta_0 \cdot \ldots \cdot \beta_{n-2}}{\gamma_1 \cdot \ldots \cdot \gamma_{n-1}}} \end{pmatrix}$$

$$\cdot \begin{pmatrix} \alpha_0 & \sqrt{\beta_0\gamma_1} & & & \\ \sqrt{\beta_0\gamma_1} & \alpha_1 & \sqrt{\beta_1\gamma_2} & & \\ & \sqrt{\beta_1\gamma_2} & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \sqrt{\beta_{n-2}\gamma_{n-1}} \\ & & & \sqrt{\beta_{n-2}\gamma_{n-1}} & \alpha_{n-1} \end{pmatrix}$$

$$\cdot \begin{pmatrix} 1 & & & & \\ & \sqrt{\frac{\gamma_1}{\beta_0}} & & & \\ & & \sqrt{\frac{\gamma_1\gamma_2}{\beta_0\beta_1}} & & \\ & & & \ddots & \\ & & & & \sqrt{\frac{\gamma_1 \cdot \ldots \cdot \gamma_{n-1}}{\beta_0 \cdot \ldots \cdot \beta_{n-2}}} \end{pmatrix}$$

From $\mathbf{A}\mathbf{p}(x_j) = x_j\mathbf{p}(x_j)$ we have $\mathbf{D}^{-1}\tilde{\mathbf{A}}\mathbf{D}\mathbf{p}(x_j) = x_j\mathbf{p}(x_j)$ and thus $\tilde{\mathbf{A}}\mathbf{D}\mathbf{p}(x_j) = x_j\mathbf{D}\mathbf{p}(x_j)$, i.e. $\mathbf{D}\mathbf{p}(x_j)$ is an eigenvector of $\tilde{\mathbf{A}}$ with the eigenvalue x_j. Since $\tilde{\mathbf{A}}$ is symmetric, its eigenvalues x_j are real and its eigenvectors $\mathbf{D}\mathbf{p}(x_j)$ $(j = 1, \ldots, n)$ are orthogonal. In the following, we will denote $\mathbf{D}\mathbf{p}(x_j)$ by $\mathbf{v}_j$.

We have proved the first part of the theorem — that the roots of p_n are the eigenvalues of $\tilde{\mathbf{A}}$. To prove the second part of the theorem, which states how to obtain the weights, we will now derive a system of equations that has the weights as its unknowns.

Because Gaussian quadrature integrates polynomials up to a degree of $2n-1$ exactly, we have

$$\sum_{j=1}^{n} p_k(x_j)\, w_j = \int_{-1}^{1} p_k(x)\, w(x)\, \mathrm{d}x \quad (k = 0, \ldots, n-1),$$

and we note that the integral is just $\langle p_0, p_k \rangle$, which is zero for $k \neq 0$ because the polynomials p_k are orthogonal. This gives us

$$\sum_{j=1}^{n} p_0(x_j)\, w_j = \langle p_0, p_0 \rangle$$

and

$$\sum_{j=1}^{n} p_k(x_j)\, w_j = 0 \quad (k = 1, \ldots, n-1).$$

By writing this system of equations in matrix-vector form we obtain

$$\begin{pmatrix} p_0(x_1) & \cdots & p_0(x_n) \\ \vdots & & \vdots \\ \vdots & & \vdots \\ p_{n-1}(x_1) & \cdots & p_{n-1}(x_n) \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} < p_0, p_0 > \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

We observe that the columns of the matrix contain the vectors $\mathbf{p}(x_1), \ldots, \mathbf{p}(x_n)$. Multiplying $\mathbf{D}$ onto the left side of this equation yields

$$(\mathbf{v}_1, \ldots, \mathbf{v}_n) \begin{pmatrix} w_1 \\ \vdots \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} < p_0, p_0 > \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

because $\mathbf{D}\mathbf{p}(x_j) = \mathbf{v}_j$. Since the $\mathbf{v}_j$ are orthogonal, multiplying $\mathbf{v}_j^{\mathrm{T}}$ onto the left side of this equation yields

$$\mathbf{v}_j^{\mathrm{T}}\mathbf{v}_j w_j = \langle p_0, p_0\rangle v_j^{(1)},$$

where $v_j^{(1)}$ is the first component of the vector $\mathbf{v}_j$. Since $v_j^{(1)} = p_0(x_j) = 1$, we have $v_j^{(1)} = (v_j^{(1)})^2$ and thus

$$w_j = \langle p_0, p_0\rangle \frac{(v_j^{(1)})^2}{\mathbf{v}_j^{\mathrm{T}}\mathbf{v}_j} = \langle p_0, p_0\rangle \left(\frac{v_j^{(1)}}{\|\mathbf{v}_j\|_2}\right)^2.$$

■

References

[Bo01] J. P. Boyd. *Chebyshev and Fourier Spectral Methods*, Second Edition. Dover, New York, 2001.

[DH94] J. R. Driscoll and D. M. Healy Jr. Computing Fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics* **15**, 202-250 (1994).

[DGR96] A. Dutt, M. Gu, and V. Rokhlin. Fast Algorithms for Polynomial Interpolation, Integration and Differentiation. *SIAM Journal on Numerical Analysis* **33**(5), 1689–1711 (1996).

[FGS98] W. Freeden, T. Gervens, and M. Schreiner. *Constructive Approximation on the Sphere*. Oxford University Press, Oxford, 1998.

[Fu92] D. Funaro. *Polynomial Approximation of Differential Equations*. Springer, Heidelberg, 1992.

[HRKM98] D. M. Healy Jr., D. Rockmore, P. J. Kostelec, and S. S. B. Moore. FFTs for the 2-Sphere — Improvements and Variations. *Advances in Applied Mathematics* (to appear). Preprint available at http://www.cs.dartmouth.edu/~geelong/publications.

[JA97] R. Jakob-Chien and B. K. Alpert. A Fast Spherical Filter with Uniform Resolution. *Journal of Computational Physics* **136**, 580–584 (1997).

[MaOb43] W. Magnus and F. Oberhettinger. *Formeln und Sätze für die speziellen Funktionen der mathematischen Physik*. Springer, Berlin, 1943.

[Moh99] M. J. Mohlenkamp. A Fast Transform for Spherical Harmonics. *Journal of Fourier Analysis and Applications* **5**(2/3), 159-184 (1999).

[PoSt02] D. Potts and G. Steidl. Fast Summation at Nonequispaced Knots by NFFTs. Preprint, Medical University of Lübeck, 2002.

[PST96] D. Potts, G. Steidl, and M. Tasche. Kernels of Spherical Harmonics and Spherical Frames. In F. Fontanella, K. Jetter and P.-J. Laurent (eds.), *Advanced Topics in Multivariate Approximation*, pp. 287–301. World Scientific Publ., Singapore, 1996.

[PST98] D. Potts, G. Steidl, and M. Tasche. Fast and stable algorithms for discrete spherical Fourier transforms. *Linear Algebra and its Applications* **275**, 433-450 (1998).

[PST00] 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 (eds.), *Modern Sampling Theory: Mathematics and Applications*, pp. 251–274, Birkhäuser, Boston, 2000.

[Sto99] J. Stoer. *Numerische Mathematik I*, 8$^{\text{th}}$ Edition. Springer, Berlin / Heidelberg / New York, 1999.

[SwSp00] P. N. Swarztrauber and W. F. Spotz. Generalized Discrete Spherical Harmonic Transforms. *Journal of Computational Physics* **159**, 213–230 (2000).

[YaRo98] N. Yarvin and V. Rokhlin. A Generalized One-Dimensional Fast Multipole Method with Application to Filtering of Spherical Harmonics. *Journal of Computational Physics* **147**, 594–609 (1998).