Electronic Transactions on Numerical Analysis.
Volume 16, pp. 70-93, 2003.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

A FAST ALGORITHM FOR FILTERING AND WAVELET DECOMPOSITION ON THE SPHERE *

MARTIN BÖHME AND DANIEL POTTS†

Abstract. This paper introduces a new fast algorithm for uniform-resolution filtering of functions defined on the sphere. We use a fast summation algorithm based on Nonequispaced Fast Fourier Transforms, building on previous work that used Fast Multipole Methods. The resulting algorithm performs a triangular truncation of the spectral coefficients while avoiding the need for fast spherical Fourier transforms. The method requires $\mathcal{O}(N^2 \log N)$ operations for $\mathcal{O}(N^2)$ grid points.

Furthermore, we apply these techniques to obtain a fast wavelet decomposition algorithm on the sphere. We present the results of numerical experiments to illustrate the performance of the algorithms.

Key words. spherical filter, spherical Fourier transform, spherical harmonics, associated Legendre functions, fast discrete transforms, fast Fourier transform at nonequispaced knots, wavelets, fast discrete summation.

AMS subject classifications. 65Txx, 33C55, 42C10.

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 modeling [3, 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 [10]. This property is often exploited in the integration of PDEs, for example; one technique for avoiding instabilities is performing a triangular truncation in each time step.

Thus, one of the subproblems of integrating PDEs on the sphere is projecting a function onto the space of band-limited functions. This operation is referred to as the "spherical filter" [10]. 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 spherical filtering is quite a costly operation, significant effort has been expended on speeding it up. Jakob-Chien and Alpert [10] presented an $\mathcal{O}(N^2 \log N)$ algorithm that is based on Fast Multipole Methods (FMM) (see, for example, [5]) to evaluate the sums that occur in the spherical filter. This work was extended by Yarvin and Rokhlin [22], who derived a generalised FMM and applied it to the spherical filter. Swarztrauber and Spotz [20] also presented optimisations to the spherical filter but took 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 $\mathcal{O}(N^2)$.

Our algorithm is based on the ideas from [10]. However, we wish to avoid the Fast Multipole Methods employed there because we feel that they are rather technical and difficult to implement. Instead, we will use a fast summation algorithm [14] that is based on the Nonequispaced Fast Fourier Transform (NFFT) [17]. We believe that this algorithm is conceptually simpler and easier to implement, and the measurements performed in [14] demonstrate that

*Received July 29, 2002. Accepted for publication March 24, 2003. Recommended by Sven Ehrich.

†University of Lübeck, Institute of Mathematics, D–23560 Lübeck, Germany. E-mails: boehme@informatik.uni-luebeck.de, potts@math.uni-luebeck.de.

its running times are at least as good as those of FMM-type algorithms. 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 (see also [11]).

The asymptotic complexity of the NFFT-based 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 $\mathcal{O}(N \log N)$, which results in a total complexity for the spherical filter algorithm of $\mathcal{O}(N^2 \log N)$.

The structure of this paper is as follows. Section 2 presents some basic definitions, including the spherical harmonics, introduces the concept of band-limited functions, and defines the spherical filter and wavelet decomposition. This section also discusses how the two transforms can be combined in a way that allows fast summation algorithms to be employed. Section 3 presents the fast NFFT-based summation algorithm from [14] and adapts it for use with odd kernel functions. Section 4 summarises the complete spherical filter algorithm and provides a complexity estimate. Section 5 discusses an implementation of the algorithm and presents the results of some numerical experiments. Finally, Section 6 gives the conclusions of the paper.

2. The Spherical Filter and Wavelet Decomposition on the Sphere. We start with several basic definitions.

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 given by*

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

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

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

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* [20, equation A.3.1]. *Further, the P_k^n fulfill the orthogonality relation*

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

Throughout this paper, we will be dealing with functions that are defined on the unit sphere $S := \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\|_2 = 1\}$. In particular, we will focus 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 $\varphi \in [0, 2\pi)$.

DEFINITION 2.3. *The* spherical harmonics *are functions on the unit sphere defined as*

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

where $\theta \in [0,\pi]$; $\varphi \in [0,2\pi)$; $n \in \mathbb{Z}, k \in \mathbb{N}_0, k \geq |n|$. The spherical harmonics form an orthonormal basis of $L^2(S)$ with respect to the scalar product

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

ETNA
Kent State University
etna@mcs.kent.edu

We will now 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 latitude / longitude grid to Fourier space and back. And finally, we will combine these two transforms to yield an algorithm for the spherical filter.

2.1. Definition of the Spherical Filter and the Wavelet Decomposition. We give some more definitions.

DEFINITION 2.4. *A function $f \in L^2(S)$ is said to be* band-limited *with bandwidth N if it can be expressed as*

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

B_N denotes the set of all band-limited functions with bandwidth N.

For any N, B_N is a subspace of $L^2(S)$.

REMARK 2.5. *Band-limited functions possess the useful property that any rotated version of $f \in B_N$ is also in B_N (see* [10, 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.* □

DEFINITION 2.6. *The orthogonal projection of a function $f \in L^2(S)$ onto B_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.

DEFINITION 2.7. *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*

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

(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 [15] and [7]. The underlying wavelets are called *Shannon wavelets*.

ETNA
Kent State University
etna@mcs.kent.edu

2.2. Discrete Spherical Fourier Transform. To apply the spherical filter or the wavelet decomposition to a function defined by its values on the sphere, the Fourier coefficients a_k^n have to be computed from the function values:

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

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

$$\mathfrak{f}_n(\theta) := \frac{1}{2\pi} \int_0^{2\pi} f(\theta,\varphi) \mathrm{e}^{-\mathrm{i}n\varphi}\ \mathrm{d}\varphi \tag{2.3}$$

and

$$a_k^n = \int_0^\pi P_k^{|n|}(\cos\theta) \mathfrak{f}_n(\theta) \sin\theta\ \mathrm{d}\theta. \tag{2.4}$$

It is well-known that if f is a band-limited function, the integrals in (2.3) and (2.4) can be computed using the DFT and Gauss-Legendre quadrature, respectively. This yields the following algorithm for the computation of the a_k^n.

ALGORITHM 1 (Discrete Spherical Fourier Transform (DSFT)).

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

It is convenient to split up the inverse operation (computation of $f(\theta_s,\varphi_t)$ from Fourier coefficients a_k^n) into two phases as well. This yields the following algorithm (the parameter N_{lim} controls how many Fourier terms are included in the sum).

ALGORITHM 2 (Inverse Discrete Spherical Fourier Transform (IDSFT)).

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

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

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

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

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

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

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

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

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

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".

In recent years, several fast algorithms for the DSFT and IDSFT have been developed. Driscoll and Healy present an $\mathcal{O}(N^2 \log^2 N)$ algorithm [4], and this algorithm was modified by several authors (e.g. [9, 16, 19]) to improve its stability. Mohlenkamp [13] presents two algorithms, one with a complexity of $\mathcal{O}(N^{5/2} \log N)$ and the other with a complexity of $\mathcal{O}(N^2 \log^2 N)$. It appears, though, that asymptotically fast DSFT algorithms still present some difficulties, e.g. instabilities or high overhead, and so research is still going on into fast $\mathcal{O}(N^3)$ algorithms; see, for example, [20].

2.3. 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. Jakob-Chien and Alpert [10] make use of this fact by combining the second step of the forward transform with the first step of the inverse transform and then applying an FMM to evaluate the resulting sum quickly.

By combining the second step of Algorithm 1 and the first step of Algorithm 2 and rearranging we obtain (for $n = -N_{\mathrm{lim}}, \ldots, N_{\mathrm{lim}}$ and $s = 0, \ldots, N-1$)

$$\mathfrak{g}_n(\theta_\sigma) = \sum_{s=0}^{N-1} w_s \mathfrak{f}_n(\theta_s) \sum_{k=|n|}^{N_{\mathrm{lim}}} P_k^{|n|}(x_s) P_k^{|n|}(x_\sigma). \tag{2.5}$$

The inner sum in this formula has a closed form known as the Christoffel-Darboux formula.

THEOREM 2.8 (Christoffel-Darboux Formula). *The sum*

$$S(x, y) := \sum_{k=n}^{N-1} P_k^n(x)\; P_k^n(y) \tag{2.6}$$

ETNA
Kent State University
etna@mcs.kent.edu

possesses the closed form

$$S(x,y) = \begin{cases} \frac{\alpha_N^n \left(P_N^n(x)P_{N-1}^n(y) - P_{N-1}^n(x)P_N^n(y)\right)}{x-y} & \text{if } x \neq y \\ \alpha_N^n \left(P_N^{n\prime}(x)P_{N-1}^n(x) - P_{N-1}^{n\prime}(x)P_N^n(x)\right) & \text{if } x = y, \end{cases} \tag{2.7}$$

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

REMARK 2.9. *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*

$$P_k^n + xP_k^{n\prime} = \alpha_k^n P_{k-1}^{n\prime} + \alpha_{k+1}^n P_{k+1}^{n\prime}. \tag{2.8}$$

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

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

Denoting $c := \frac{1}{2^n n!}\left[\frac{2n+1}{2}(2n)!\right]^{1/2}$ *we obtain*

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

□

Applying the Christoffel-Darboux formula (2.7) to equation (2.5) yields

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

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

$$\begin{aligned}\sum_{\substack{s=0\\ s\neq\sigma}}^{N-1} w_s\mathfrak{f}_n(\theta_s)&\frac{\alpha_{N_{\mathrm{lim}}+1}^{|n|}\left(P_{N_{\mathrm{lim}}+1}^{|n|}(x_s)P_{N_{\mathrm{lim}}}^{|n|}(x_\sigma) - P_{N_{\mathrm{lim}}}^{|n|}(x_s)P_{N_{\mathrm{lim}}+1}^{|n|}(x_\sigma)\right)}{x_s - x_\sigma} \\ &= \alpha_{N_{\mathrm{lim}}+1}^{|n|}\left(P_{N_{\mathrm{lim}}}^{|n|}(x_\sigma)\sum_{\substack{s=0\\ s\neq\sigma}}^{N-1}\frac{w_s\mathfrak{f}_n(\theta_s)P_{N_{\mathrm{lim}}+1}^{|n|}(x_s)}{x_s - x_\sigma}\right. \\ &\qquad \left. - P_{N_{\mathrm{lim}}+1}^{|n|}(x_\sigma)\sum_{\substack{s=0\\ s\neq\sigma}}^{N-1}\frac{w_s\mathfrak{f}_n(\theta_s)P_{N_{\mathrm{lim}}}^{|n|}(x_s)}{x_s - x_\sigma}\right).\end{aligned} \tag{2.9}$$

This alone has not led to a reduction in the asymptotic complexity of the middle steps — we still have a complexity of $\mathcal{O}(N_{\mathrm{lim}}\, N^2)$: $\mathcal{O}(N_{\mathrm{lim}})$ for the loop $n = -N_{\mathrm{lim}}, \ldots, N_{\mathrm{lim}}$; $\mathcal{O}(N)$ 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 (2.3) can be speeded up using fast approximate algorithms. This was done using Fast Multipole Methods in [10]; we will use a different technique, which we present next.

3. A Fast Approximate Summation Algorithm. In this section, we will present an algorithm, from [14], that computes sums of the form

$$f(y_j) = \sum_{\substack{k=1 \\ k\neq j}}^{N} \beta_k K(y_j - x_k) \quad (j = 1, \dots, M), \tag{3.1}$$

where $\beta_k \in \mathbb{C}$, $x_k \in \mathbb{R}$ ($k = 1, \dots, N$), $y_j \in \mathbb{R}$ ($j = 1, \dots, M$). The function K is the so-called *kernel*. K must be in C^∞, except for the origin, which may exhibit a singularity. If so, we agree to set $K(0) := 0$ so that we will be able to evaluate K for all $x \in \mathbb{R}$. 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 $\mathcal{O}\left(N \log N + \log\left(\frac{1}{\varepsilon}\right) M\right)$ operations (where ε is the desired accuracy) compared to the $\mathcal{O}(MN)$ operations that would be required for a direct evaluation.

The algorithm is based on the Nonequispaced Fast Fourier Transform (NFFT), an approximate algorithm for computing the Discrete Fourier Transform at nonequispaced nodes (see, for example, [6, 1, 18, 17]). A software-package is available from [11]. We will refer to the algorithm that computes

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

($\omega_j \in \mathbb{R}$) as the NFFT and to the algorithm that computes

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

as the NFFT$^{\mathrm{T}}$. The complexity of both algorithms is $\mathcal{O}(\sigma N \log(\sigma N) + mM)$, where σ and m are constants that control the accuracy of the approximation.

The general idea of the fast summation algorithm 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. For even kernels this was worked out in [14]. Since our application uses the kernel $1/x$, we need a regularisation for odd kernels.

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,

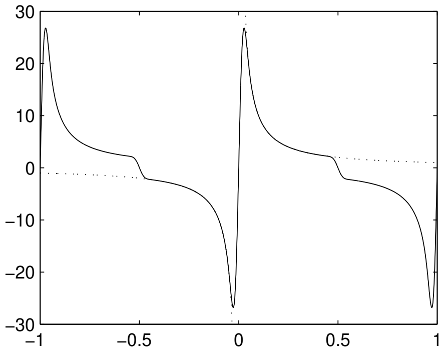

FIG. 3.1. *The kernel $K(x) = 1/x$ (dotted) and its 1-periodic regularised version (solid). Parameters are $n = 128$, $a = p = 12$.*

on $x \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, we have

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

with

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

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

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.

An example of a regularised kernel is shown in Figure 3.1. To demonstrate that the kernel has been made periodic, two periods are shown.

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.

$$T_I^{(r)}\left(\frac{a}{n}\right) = K^{(r)}\left(\frac{a}{n}\right) \quad (r = 0, \ldots, p-1) \tag{3.2}$$

and

$$T_B^{(r)}\left(-\frac{1}{2} + \frac{a}{n}\right) = K^{(r)}\left(-\frac{1}{2} + \frac{a}{n}\right) \quad (r = 0, \ldots, p-1). \tag{3.3}$$

From (3.2), we obtain

$$\sum_{j=1}^{p} a_j^I \left(\frac{\pi n j}{2a}\right)^r (-1)^{r/2} \sin \frac{\pi j}{2} = K^{(r)}\left(\frac{a}{n}\right) \tag{3.4}$$

for even r and

$$\sum_{j=1}^{p} a_j^I \left(\frac{\pi n j}{2a}\right)^r (-1)^{(r-1)/2} \cos\frac{\pi j}{2} = K^{(r)}\left(\frac{a}{n}\right) \tag{3.5}$$

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

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

and

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

Turning now to equation (3.3), 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 (3.4). The case for odd r proceeds in a similar way, yielding an equation with the same left-hand side as equation (3.5). 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

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

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

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

We now split up K like this:

$$K = (K - K_R) + (K_R - K_{RF}) + K_{RF} \tag{3.6}$$

ETNA
Kent State University
etna@mcs.kent.edu

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:

$$\begin{aligned}\tilde{f}(y_j) &:= \sum_{k=1}^{N} \beta_k (K_{NE} + K_{RF})(y_j - x_k) \quad (j = 1, \ldots, M) \\ &= \sum_{k=1}^{N} \beta_k K_{NE}(y_j - x_k) + \sum_{k=1}^{N} \beta_k K_{RF}(y_j - x_k).\end{aligned} \tag{3.7}$$

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 the sum $\sum_{k=1}^{N} \beta_k K_{NE}(y_j - x_k)$ 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 $|x_k|, |y_j| \le \frac{1}{4} - \frac{a}{2n}$. If the x_k and y_j do not lie within the required range, they can be scaled by the factor

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

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 $\mathrm{NFFT}^{\mathrm{T}}$. We then multiply with the b_l and compute the outer sum using an NFFT.

We are now ready to summarise the NFFT-based summation algorithm:

ALGORITHM 3 (NFFT-based 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: $M, N \in \mathbb{N}$
$x_k \in \mathbb{R}$ $(k = 1, \ldots, N)$
$y_j \in \mathbb{R}$ $(j = 1, \ldots, M)$
$(|x_k - y_j| \le \frac{1}{4} - \frac{a}{2n}$ $(k = 1, \ldots, N, j = 1, \ldots, M))$
$a, p \in \mathbb{N}$ (*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 controlling the accuracy of the NFFT*)

Precomputation:

(i) *Compute coefficients* a_j^I *and* a_j^B $(j = 1, \dots, p)$ *for the regularised kernel by solving the systems of equations derived from* (3.2) *and* (3.3).

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

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

using an FFT

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

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

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

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

$$d_l := a_l \; b_l$$

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

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

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

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

5. *For* $j = 1, \dots, M$, *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 $\mathcal{O}(\sigma n \log(\sigma n) + m(M + N) + a\nu M)$.

3.1. Error Estimate for the Fast Summation Algorithm. We will now provide an error estimate for the fast summation algorithm (Algorithm 3) just presented. The error estimate is based on [14, 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 [14].

In the following, we will deal solely with the error introduced by Algorithm 3. The additional error introduced by the NFFT has already been investigated thoroughly, see Section 5 of [14] and the literature cited there. Basically, the error in the NFFT has been shown to decay exponentially with the parameter m.

Let us now examine the approximation error in Algorithm 3. In Section 3, the kernel K was split up into three components:

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

(see equation (3.6)), 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 (3.1) and the definition of the approximation $\tilde{f}$ (3.7), 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

$$|f(y_j) - \tilde{f}(y_j)| \leq \left\| (\beta_k)_{k=1}^N \right\|_p \ \left\| \left(K_{ER}(y_j - x_k) \right)_{k=1}^N \right\|_q, \tag{3.8}$$

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 $\|\mathbf{v}\|_\infty := \max_{k=1,\ldots,N} |v_k|$.)

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 follows from [14] and is given by the following theorem, proved in [2].

THEOREM 3.1. *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 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 (3.2) and (3.3).

4. The Fast Spherical Filter Algorithm. In this section, we will summarise the fast spherical filter algorithm (Algorithm 4) and show that it has an asymptotic complexity of $\mathcal{O}(N^2 \log N)$. We will do this by proving (in Theorem 4.1) that the NFFT summation algorithm (Algorithm 3) has a complexity of $\mathcal{O}(N \log N)$ when used with Legendre nodes. A separate and unfortunately rather technical proof is necessary for this case because the complexity analysis for the general case only gives a complexity bound of $\mathcal{O}(N^2)$. This is discussed in detail in Section 4.2.

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

ALGORITHM 4 (Fast Spherical Filter).

Input: $f(\theta_s, \varphi_t) \in \mathbb{C}$ $(s = 0, \ldots, N-1,\ t = 0, \ldots, 2N-1)$
(*function values on the spherical grid*)

Output: $\tilde{f}^{N_{\mathrm{lim}}}(\theta_s, \varphi_t) \in \mathbb{C}$ $(s = 0, \ldots, N-1,\ t = 0, \ldots, 2N-1)$
(*approximate values for* $f^{N_{\mathrm{lim}}}(\theta_s, \varphi_t)$)

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

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

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

$a, p, m \in \mathbb{N}, \sigma \in \mathbb{Q}$
(*parameters for the NFFT summation algorithm*)

Precomputation: *For* $n = 0, \ldots, N_{\mathrm{lim}}$, $s = 0, \ldots, N-1$ *compute* $P^n_{N_{\mathrm{lim}}}(x_s)$, $P^n_{N_{\mathrm{lim}}+1}(x_s)$, $P^{n\,\prime}_{N_{\mathrm{lim}}}(x_s)$ *and* $P^{n\,\prime}_{N_{\mathrm{lim}}+1}(x_s)$ *using the recurrences* (2.2) *and* (2.8).

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

$$\mathfrak{f}_n(\theta_s) := \frac{1}{2N} \sum_{t=0}^{2N-1} f(\theta_s, \varphi_t)\, \mathrm{e}^{-\mathrm{i}n\varphi_t} \qquad (n = -(N-1), \ldots, N-1) \tag{4.1}$$

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

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

$$\mathfrak{S}_1(n, \sigma) := \sum_{\substack{s=0\\ s\neq\sigma}}^{N-1} \frac{w_s \mathfrak{f}_n(\theta_s) P^{|n|}_{N_{\mathrm{lim}}+1}(x_s)}{x_s - x_\sigma} \qquad (\sigma = 0, \ldots, N-1) \tag{4.2}$$

and

$$\mathfrak{S}_2(n, \sigma) := \sum_{\substack{s=0\\ s\neq\sigma}}^{N-1} \frac{w_s \mathfrak{f}_n(\theta_s) P^{|n|}_{N_{\mathrm{lim}}}(x_s)}{x_s - x_\sigma} \qquad (\sigma = 0, \ldots, N-1) \tag{4.3}$$

using the NFFT summation algorithm, and then set

$$\begin{aligned}\mathfrak{g}_n(\theta_\sigma) := \alpha^{|n|}_{N_{\mathrm{lim}}+1} \Big(& P^{|n|}_{N_{\mathrm{lim}}}(x_\sigma)\big(\mathfrak{S}_1(n,\sigma) + w_s \mathfrak{f}_n(\theta_\sigma) P^{|n|\,\prime}_{N_{\mathrm{lim}}+1}(x_\sigma)\big) \\ & - P^{|n|}_{N_{\mathrm{lim}}+1}(x_\sigma)\big(\mathfrak{S}_2(n,\sigma) + w_s \mathfrak{f}_n(\theta_\sigma) P^{|n|\,\prime}_{N_{\mathrm{lim}}}(x_\sigma)\big)\Big).\end{aligned}$$

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

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

$$\tilde{f}^{N_{\mathrm{lim}}}(\theta_s, \varphi_t) := \sum_{n=-N_{\mathrm{lim}}}^{N_{\mathrm{lim}}} \mathrm{e}^{\mathrm{i}n\varphi_t}\, \mathfrak{g}_n(\theta_s) \qquad (t = 0, \ldots, 2N-1) \tag{4.4}$$

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

ETNA
Kent State University
etna@mcs.kent.edu

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 [14] for the numerical experiments.

Note 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 $\mathcal{O}(N_{\mathrm{lim}}^2 N)$ — there are N points for which to evaluate these functions, $N_{\mathrm{lim}} + 1$ different values for n, and the evaluation of the functions for $k = N_{\mathrm{lim}}$ and $k = N_{\mathrm{lim}} + 1$ requires $\mathcal{O}(N_{\mathrm{lim}})$ 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 [12, Chapter 4, Section 3]), but it is unsuitable for numerical work because it is unstable when applied forwards and prone to underflow when applied backwards.

4.2. Complexity Estimate. We will now examine the asymptotic complexity of Algorithm 4 and justify the complexity estimate of $\mathcal{O}\big(N_{\mathrm{lim}}(\sigma N \log(\sigma N) + (m + a \log N)N)\big)$ for step 2 of the algorithm.

In this step, the NFFT summation algorithm (Algorithm 3) is applied $\mathcal{O}(N_{\mathrm{lim}})$ 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 3 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 [8, Section 3.1]. 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 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. As we will show, 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 $\mathcal{O}(aN \log N)$, resulting in an overall complexity for the spherical filter of $\mathcal{O}(N^2 \log N)$. To prove 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.

THEOREM 4.1. *If the NFFT summation algorithm (Algorithm 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 Algorithm 3 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,

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

and we 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 obvious from the definition of the P_k because P_k is either an even or an odd polynomial.) Because of the symmetry of the x_k, the summand that we obtain in C_1 for a certain x_k is equal to the summand that we obtain in C_2 for $-x_k$. This means that we have $C_1 + C_2 = 2\left(C_1^{<0} + C_2^{<0}\right)$, where

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

(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 $x_j \in \left(x_k - \frac{a}{n}, x_k\right) \Leftrightarrow x_k \in \left(x_j, x_j + \frac{a}{n}\right)$, 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

ETNA
Kent State University
etna@mcs.kent.edu

additional constraint without changing $C_2^{<0}$:

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

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

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

and assuming that the x_k are ordered we have

$$C_1 + C_2 \le 4 \sum_{k=1}^{N/2} \#\left\{x_j : x_j \in \left(x_k, x_k + \frac{a}{n}\right)\right\}. \tag{4.5}$$

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

$$-1 \le -\cos \frac{k - \frac{1}{2}}{N + \frac{1}{2}}\pi \le x_k \le -\cos \frac{k}{N + \frac{1}{2}}\pi \le 1 \quad (k = 1, \ldots, N) \tag{4.6}$$

(see [8, 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 $x_{k+i} \ge x_k + \frac{a}{n}$. If we constrain ourselves to $k \le \frac{N}{2}$, 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

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

by equation (4.6), which implies

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

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

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 $\tilde{c}(k) \ge c(k)$ $(k = 1, \ldots, \frac{N}{2})$ 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

$$\tilde{c}(k) := \begin{cases} \min\left(\frac{N}{2},\ \frac{a}{n} \cdot \frac{2N+1}{4\sqrt{2}} \cdot \left(\sin \frac{2k - \frac{1}{2}}{2N+1}\pi\right)^{-1} + \frac{1}{2}\right) & \text{for } k = 1, \ldots, \frac{N}{4} \\ \min\left(\frac{N}{2},\ \frac{a}{n} \cdot \frac{2N+1}{4} + \frac{1}{2}\right) & \text{for } k = \frac{N}{4} + 1, \ldots, \frac{N}{2}, \end{cases} \tag{4.7}$$

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

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

$$x_{k+\tilde{c}(k)} \ge x_k + \frac{a}{n} \tag{4.8}$$

because $c(k)$ was defined as the smallest i for which $x_{k+i} \ge x_k + \frac{a}{n}$. We will now investigate under which conditions inequality (4.8) is satisfied. This inequality holds if

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

(because of inequality (4.6)), and this is equivalent to

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

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

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

and this is satisfied if

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

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

$$\tilde{c}(k) \ge \frac{a}{n}\cdot\frac{2N+1}{4\sqrt{2}}\left(\sin\frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi\right)^{-1} + \frac{1}{2}. \tag{4.9}$$

We will now show that $\tilde{c}(k)$ satisfies $\tilde{c}(k) \ge c(k)$, and we can do this either directly or by showing that $\tilde{c}(k)$ satisfies inequality (4.9), which implies $\tilde{c}(k) \ge c(k)$. We will consider two cases.

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

(This corresponds to the first case in (4.7), the definition of the $\tilde{c}(k)$.)

Assume first that $\tilde{c}(k) = \frac{N}{2}$. By inequality (4.6) we have

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

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

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 $k \le \frac{N}{4}$ and $\tilde{c}(k) \le \frac{N}{2}$, we have $2k + \tilde{c}(k) \le N$ and thus $\sin\frac{2k-\frac{1}{2}}{2N+1}\pi \le \sin\frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi$. This means that $\tilde{c}(k)$ satisfies inequality (4.9).

ETNA
Kent State University
etna@mcs.kent.edu

Case 2 $k = \frac{N}{4}+1,\dots,\frac{N}{2}$.
(This corresponds to the second case in (4.7).)
Again, assume first that $\tilde{c}(k) = \frac{N}{2}$. By the symmetry of the x_k we know that $x_k \le 0$, and by inequality (4.6) we have

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

Thus $x_k + \frac{a}{n} \le \frac{1}{\sqrt{2}} \le x_{3N/4+1}$, so by setting $i = \frac{3N}{4}+1-k$ we have $x_{k+i} \ge x_k + \frac{a}{n}$ and thus $c(k) \le i \le \frac{3N}{4}+1-(\frac{N}{4}+1) = \frac{N}{2} = \tilde{c}(k)$.
Assume conversely that $\tilde{c}(k) = \frac{a}{n}\cdot\frac{2N+1}{4}+\frac{1}{2}$. Because $\frac{N}{4}+1 \le k \le \frac{N}{2}$ and $\frac{1}{2} \le \tilde{c}(k) \le \frac{N}{2}$, we have $\frac{N}{2}+\frac{5}{2} \le 2k+\tilde{c}(k) \le \frac{3N}{2}$ and thus $\frac{1}{\sqrt{2}} = \sin(\frac{\pi}{4}) \le \sin\frac{2k+\tilde{c}(k)-\frac{1}{2}}{2N+1}\pi$. This means that $\tilde{c}(k)$ satisfies inequality (4.9).

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:

$$\#\left\{x_j : x_j \in \left(x_k, x_k+\frac{a}{n}\right)\right\} \le \tilde{c}(k)-1 \le \begin{cases} \frac{a}{n}\cdot\frac{2N+1}{4\sqrt{2}}\cdot\left(\sin\frac{2k-\frac{1}{2}}{2N+1}\pi\right)^{-1}-\frac{1}{2} & \text{for } k=1,\dots,\frac{N}{4} \\ \frac{a}{n}\cdot\frac{2N+1}{4}-\frac{1}{2} & \text{for } k=\frac{N}{4}+1,\dots,\frac{N}{2}. \end{cases} \tag{4.10}$$

Using this, we can now estimate $C_1 + C_2$ (see equation (4.5)):

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

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

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

and thus

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

In all, we have

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

□

If we set $n = N$, as was done in the spherical filter algorithm, then Lemma 4.1 tells us that step 4 of the NFFT summation algorithm (Algorithm 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 4) calls the NFFT summation $\mathcal{O}(N_{\mathrm{lim}})$ times, it has a complexity of $\mathcal{O}\big(N_{\mathrm{lim}}\cdot(\sigma N\log(\sigma N) + (m + a\log N)N)\big)$, as stated in Section 4.1.

5. Implementation and Numerical Experiments. In this section, we present an implementation of the fast spherical filter algorithm. Numerical experiments will be performed to test its accuracy and speed.

5.1. Implementation. The spherical filter algorithm and the auxiliary algorithms were implemented in C++ using double-precision arithmetic. The linear algebra library CLAPACK 3.0 available from
http://www.netlib.org
was used to solve the eigenproblem that occurs in the computation of the Gauss-Legendre nodes and weights and to solve the system of equations that occurs in the NFFT summation

ETNA
Kent State University
etna@mcs.kent.edu

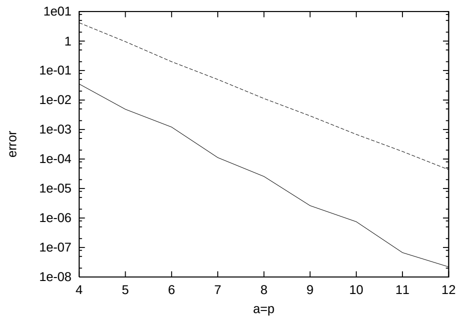

FIG. 5.1. *Predicted maximum error (dashed) and actual maximum error (solid) for the NFFT summation algorithm (N=1024).*

algorithm (Algorithm 3). The FFT library FFTW 2.1.3 available 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.

5.2. 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 3.1. 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 3.1. 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 (3.8) 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 5.1 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. In [14], the authors 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

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

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, $\mathbf{F}^{N_{\mathrm{lim}}} := \left(f^{N_{\mathrm{lim}}}(\theta_s, \varphi_t)\right)_{s=0,\ t=0}^{N-1\ 2N-1}$ is the solution obtained by computing the sums in step 2 of Algorithm 4 exactly, and $\tilde{\mathbf{F}}^{N_{\mathrm{lim}}} := \left(\tilde{f}^{N_{\mathrm{lim}}}(\theta_s, \varphi_t)\right)_{s=0,\ t=0}^{N-1\ 2N-1}$ is the approximate solution as computed by Algorithm 4. 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.

ETNA
Kent State University
etna@mcs.kent.edu

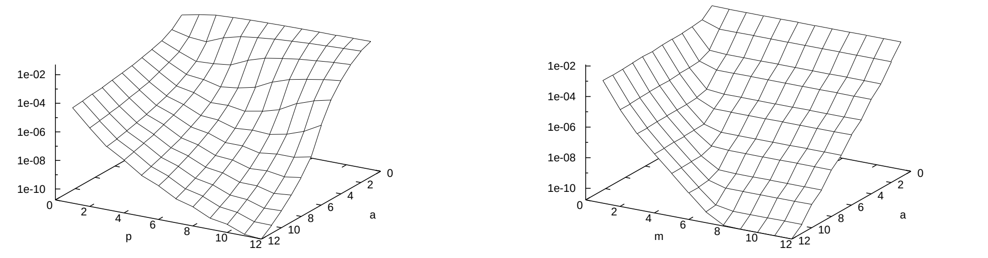

FIG. 5.2. *Relative error E in the result of the fast spherical filter algorithm. Left: $1 \leq a, p \leq 12$, $m = 12$, $N = 512$. Right: $1 \leq a, m \leq 12$ with $p = a$, $N = 512$.*

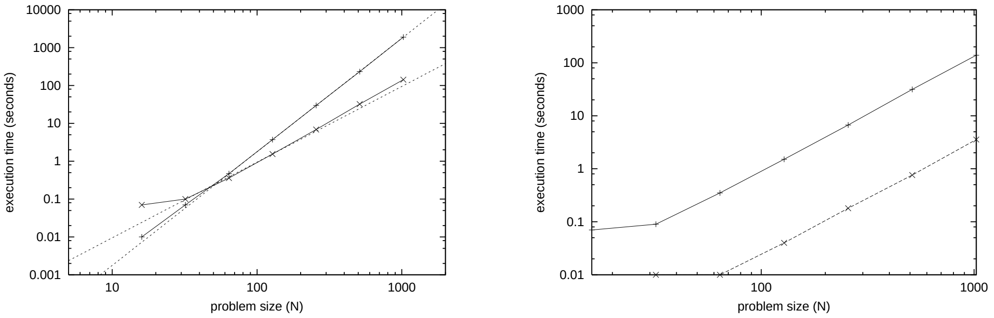

FIG. 5.3. *Left: Execution times of the exact algorithm (plus signs) and the approximate algorithm (crosses). The dashed lines show time complexities of $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$; they intercept the plots at $N = 128$. Right: Execution times for the individual steps in the fast spherical filter (Algorithm 4). The solid line plots the time taken by step 2, the dashed line plots the time for steps 1 and 3 combined.*

Figure 5.2 (left) shows the error on a logarithmic scale for $1 \leq a, p \leq 12$, with $m = 12$ and $\sigma = 2$ 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 in [14] to set $a = p$.

Figure 5.2 (right) 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 4 compared to the algorithm that is obtained by computing the sums in step 2 of Algorithm 4 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 5.3 (left) 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 4. 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 5.3 (right) 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.

One small improvement that we can offer over the NFFT summation algorithm as presented in [14] is described in the following remark.

REMARK 5.1. *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.* □

5.3. Wavelet Decomposition. As remarked in Section 2.1, 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 $\tilde{g}$. Special care should thus be taken in choosing parameters for the approximate algorithms to guarantee sufficient accuracy.

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 [15]. If the function values are interpreted as distances from the origin, then this function describes a half-sphere that is joined to 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 $a = p = 10$ and $m = 7$. The modifications discussed in Remark 5.1 were applied.) The corresponding wavelet component is shown in Figure 5.4 (left).

As a second example, consider Figure 5.4 (right). 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.

6. 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.

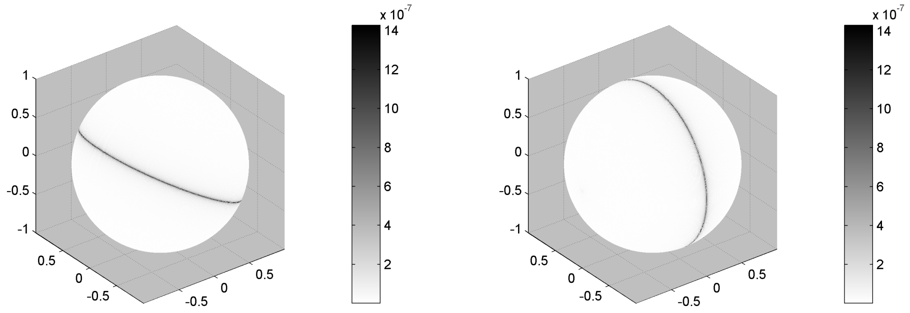

FIG. 5.4. *Wavelet component (absolute values) of the function f_{test} rotated around the x_1 axis by an angle of $\pi/6$ (left) and $\pi/2$ (right). The NFFT summation parameters were set to $a = p = 10$ and $m = 7$, and the modified K_{NE} was used. (Grid resolution is $N = 1024$.)*

The algorithm is based on previous work by Jakob-Chien and Alpert [10] and Yarvin and Rokhlin [22]. The Fast Multipole Methods (FMM) used in these approaches were replaced by a fast summation algorithm based on the Nonequispaced Fast Fourier Transform (NFFT). It would be interesting to compare execution times for the FMM-based spherical filter algorithms with the NFFT summation approach discussed here. In [14], a performance comparison showed the NFFT-based summation algorithm to be slightly faster than the FMM algorithm it was compared with. For this reason, we would expect the spherical filter algorithm presented here to offer similar performance to FMM-based algorithms. 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.

Performance measurements show that the algorithm presented here behaves as its $\mathcal{O}(N^2 \log N)$ complexity predicts. This makes it faster than the straightforward $\mathcal{O}(N^3)$ algorithm for $N \geq 64$. Of course, more efficient $\mathcal{O}(N^3)$ algorithms exist, and these may still have a slight performance advantage; for a comparison, see [21]. However, as grid resolution increases, $\mathcal{O}(N^2 \log N)$ algorithms will become more and more attractive.

Acknowledgments. The authors thank the referees for their very helpful comments and suggestions.

REFERENCES

[1] G. BEYLKIN, *On the fast Fourier transform of functions with singularities*, Appl. Comput. Harmon. Anal., 2 (1995), pp. 363 – 381.

[2] M. BÖHME, *A fast algorithm for filtering and wavelet decomposition on the sphere*. Diplomarbeit, Universität zu Lübeck, Report B-02-08, 2002.

[3] J. P. BOYD, *Chebyshev and Fourier Spectral Methods*, Dover Press, New York, second ed., 2000.

[4] J. DRISCOLL AND D. HEALY, *Computing Fourier transforms and convolutions on the 2–sphere*, Adv. in Appl. Math., 15 (1994), pp. 202 – 250.

[5] A. DUTT, M. GU, AND V. ROKHLIN, *Fast algorithms for polynomial interpolation, integration and differentiation*, SIAM J. Numer. Anal., 33 (1996), pp. 1689 – 1711.

[6] A. DUTT AND V. ROKHLIN, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput., 14 (1993), no. 6, pp. 1368 – 1393.

[7] W. FREEDEN, T. GERVENS, AND M. SCHREINER, *Constructive Approximation on the Sphere*, Oxford University Press, Oxford, 1998.

[8] D. FUNARO, *Polynomial Approximation of Differential Equations*, Springer–Verlag, Berlin, 1992.

[9] D. HEALY, P. KOSTELEC, S. S. B. MOORE, AND D. ROCKMORE, *FFTs for the 2-sphere – Improvements and variations*, J. Fourier Anal. Appl., 9 (2003).

[10] R. JAKOB-CHIEN AND B. K. ALPERT, *A fast spherical filter with uniform resolution*, J. Comput. Phys., 136 (1997), pp. 580 – 584.

[11] S. KUNIS AND D. POTTS, *NFFT, Softwarepackage, C subroutine library*. (2002) http://www.math.uni-luebeck.de/potts/nfft.

[12] W. MAGNUS AND F. OBERHETTINGER, *Formeln und Sätze für die speziellen Funktionen der mathematischen Physik*, Springer, Berlin, 1943.

[13] M. J. MOHLENKAMP, *A fast transform for spherical harmonics*, J. Fourier Anal. Appl., 5 (1999), pp. 159 – 184.

[14] D. POTTS AND G. STEIDL, *Fast summation at nonequispaced knots by NFFTs*, SIAM J. Sci. Comput., (to appear).

[15] D. POTTS, G. STEIDL, AND M. TASCHE, *Kernels of Spherical Harmonics and Spherical Frames*, in Advanced Topics in Multivariate Approximation, F. Fontanella, K. Jetter, and P.-J. Laurent, eds., World Scientific Publ., Singapore, 1996, pp. 287 – 301.

[16] ———, *Fast and stable algorithms for discrete spherical Fourier transforms*, Linear Algebra Appl., 275 (1998), pp. 433 – 450.

[17] ———, *Fast Fourier transforms for nonequispaced data: A tutorial*, in Modern Sampling Theory: Mathematics and Applications, J. J. Benedetto and P. J. S. G. Ferreira, eds., Boston, 2001, Birkhäuser, pp. 247 – 270.

[18] G. STEIDL, *A note on fast Fourier transforms for nonequispaced grids*, Adv. Comput. Math., 9 (1998), pp. 337 – 353.

[19] R. SUDA AND M. TAKAMI, *A fast spherical harmonics transform algorithm*, Math. Comp., 71 (2002), no. 238, pp. 703 – 715, (electronic).

[20] P. N. SWARZTRAUBER AND W. F. SPOTZ, *Generalized discrete spherical harmonic transforms*, J. Comput. Phys., 159 (2000), pp. 213 – 230.

[21] ———, *A performance comparison of associated Legendre projections*, J. Comput. Phys., 168 (2001), pp. 339 – 355.

[22] N. YARVIN AND V. ROKHLIN, *A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics*, J. Comput. Phys., 147 (1998), pp. 549 – 609.