FAST SUMMATION AT NONEQUISPACED KNOTS BY NFFTS

DANIEL POTTS* AND GABRIELE STEIDL†

Dedicated to Professor Manfred Tasche on the occasion of his 60th birthday

Abstract. We develop a new algorithm for the fast computation of discrete sums $f(y_j) := \sum_{k=1}^{N} \alpha_k K(y_j - x_k)$ $(j = 1, \ldots, M)$ based on the recently developed fast Fourier transform at nonequispaced knots. Our algorithm, in particular our regularization procedure, is simply structured and can be easily adapted to different kernels K. Our method utilizes the widely known FFT and can consequently incorporate advanced FFT implementations. In summary it requires $\mathcal{O}(N \log N + M)$ arithmetic operations. We prove error estimates to obtain clues about the choice of the involved parameters and present numerical examples in one and two dimensions.

2000 AMS Subject Classification: 65T40, 65T50, 65F30
Keywords: fast discrete summation, fast Fourier transform at nonequispaced knots, Toeplitz matrices, radial basis functions

1. Introduction. The fast computation of special structured discrete sums or from the linear algebra point of view of products of vectors with special structured dense matrices is a frequently appearing task in the study of particle models [17, 18], in the numerical solution of integral equations (or of partial differential equations by recasting them as integral equations) [20, 9] and in the approximation of functions by radial basis functions (RBFs) [5, 4]. There exists a rich literature on the above topics and we have only cited some typical papers here.
Various algorithms were designed to speed up the summation process. The *hierarchical multipole method* and the *fast multipole method* (FMM) are very popular in the context of particle simulation [17, 18] and of approximation by RBFs [5, 4]. For a short introduction into these methods see, e.g., the tutorial of Beatson and Greengard [3] and for a nice matrix version of the FMM [32]. The first method requires $\mathcal{O}(\log(1/\epsilon)(N+M)\log N)$ arithmetic operations, where ϵ denotes the accuracy of the computation. As convenient in literature in this field we do not use the Landau symbol in its strong sense but write instead $\mathcal{O}(\alpha\, g_1(N) + \beta\, g_2(M)) := F(M,N)$ for $N, M > 1$ if there exists $0 < c < \infty$ independent of α and β such that $|F(N,M)/(\alpha\, g_1(N) + \beta\, g_2(M))| \le c$. The FMM is quicker than the hierarchical method, namely $\mathcal{O}(\log(1/\epsilon)(N+M))$, even in practical experiments. However, the price we have to pay for this consists in a more complex structure of the algorithm and thus in a more complicated adaptation to new kernels.
Closely related to the hierarchical multipole scheme but designed for the numerical solution of integral equations is the *panel clustering method* [20]. Here the summation kernel is not explicitly given. Instead the summation coefficients have a special structure due to the analytic background they are arising from.
In recent years fast summation algorithms were introduced by Hackbusch [19] in the theory of *H–matrices* and by Tyrtyshnikov et al. [33, 16] under the name *mosaic–skeleton approximation.* The paper [9] also fits into this context.
Other algorithms for the computation of discrete sums which require only $\mathcal{O}(N \log N)$ arithmetic operations focus on the matrix compression by the discrete wavelet transform [1, 7, 21, 23].
Recently, Beylkin and Cramer [8] have proposed a fast summation algorithm with $\mathcal{O}(N \log N + M)$ arithmetic operations based on the multiresolution concept used in wavelet theory. Their hierarchical scheme is again closely related to the method of local corrections [2].

In this paper, we are interested in an algorithm for the fast computation of sums of the form

$$f(y_j) := \sum_{k=1}^{N} \alpha_k K(y_j - x_k) \qquad (j = 1, \ldots, M), \tag{1.1}$$

*University of Lübeck, Institute of Mathematics, Wallstr. 40, D–23560 Lübeck, (potts@math.uni-luebeck.de).
†University of Mannheim, Institute of Mathematics, D–68131 Mannheim, (steidl@math.uni-mannheim.de).

where K are special kernels, e.g.,

$$\frac{1}{x^2},\ \frac{1}{|x|},\ \log|x|,\ x^2\log|x|,\ \frac{1}{x}, \tag{1.2}$$

and in its multivariate generalization for RBFs $\mathcal{K}$. Of course, our algorithm can be modified for other kernels frequently used in the approximation by RBFs, e.g., the Gaussian or the (inverse) multiquadric $(x^2+c^2)^{\pm 1/2}$. This and the consideration of discrete sums with non explicitly given summation kernels will be done in a forthcoming paper. Here we propose a new summation algorithm based on the recently developed fast Fourier transform for nonequispaced knots (NFFT), see [29] and the references therein. Our algorithm, in particular our regularization procedure, is simply structured and can easily be adapted to different kernels K. Further, our method utilizes the widely known fast Fourier transform (FFT) and can incorporate advanced FFT implementations. The hierarchical structure of our algorithm is hidden in the FFT. Our algorithm requires $\mathcal{O}(N\log N+M)$ arithmetic operations for uniformly distributed points x_k and is related to the approach in [8], see Remark 3.2. Numerical experiments demonstrate the power of our algorithm both in the one–dimensional and two–dimensional settings.

The remainder of this paper is organized as follows: the original idea for our algorithm came from the consideration of (1.1) for equispaced knots x_k and y_j. In this case we have simply to compute the multiplication of a vector with a Toeplitz matrix or a block–Toeplitz–Toeplitz–block matrix in the bivariate setting. The standard algorithm to do this uses the FFT. In Section 2 we recall this standard procedure from a point of view which is suited for the generalization of the algorithm to nonequispaced knots. In one dimension this generalization is done in Section 3. In Subsection 3.1 we introduce our algorithm. In particular, we present a simple regularization procedure of the kernel which involves only solutions of small systems of linear equations with right–hand sides depending on the kernel. Thus it is easy to incorporate various kernels. Error estimates are proved in Subsection 3.2. Here we focus especially on a careful analysis of our regularization procedure. Subsection 3.3 provides a couple of numerical examples in one dimension. The generalization of our algorithm to the multivariate setting is done in Section 4, where we restrict our attention to RBFs as summation kernels. We have attached an appendix which contains the NFFT algorithms. For readers not familiar with the NFFT we recommend to read the appendix first.

2. Fast Summation at Equispaced Knots. The basic idea underlying our algorithm for the fast computation of (1.1) steams from the standard algorithm for the fast multiplication of a vector with a Toeplitz matrix. We recall this algorithm in a slightly generalized form from a point of view which carries over to the nonequispaced setting.
Let x_k $(k=1,\dots,N)$ and y_j $(j=1,\dots,M)$ be equally spaced points in $[-1/4,1/4)$, i.e., without loss of generality let $x_k := -1/4+(k-1)/(2N)$ $(k=1,\dots,N)$ and $y_j := -1/4+(j-1)(2M)$ $(j=1,\dots M)$. In the following, we agree to set $K(0):=0$ if K has a singularity at the origin. For the fast summation of

$$f(y_j) := \sum_{k=1}^{N} \alpha_k K\left(\frac{j-1}{2M}-\frac{k-1}{2N}\right) \qquad (j=1,\dots,M), \tag{2.1}$$

we want to apply the following aliasing formula.

THEOREM 2.1. *(Aliasing formula)*
Let g be a 1–periodic function with absolutely convergent Fourier series and Fourier coefficients

$$c_k(g) := \int_{-1/2}^{1/2} g(x)\,\mathrm{e}^{-2\pi\mathrm{i}kx}\,\mathrm{d}x.$$

Define an approximation

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

of $c_k(g)$ by using the trapezoidal quadrature rule. Then the following relation holds true:

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

Now let $n := 2\,\mathrm{lcm}(N,M)$. The values $K(j/n)$ $(j = -n/2+1,\dots,n/2-1)$ are determined by the entries of the matrix $\left(K\left(\frac{j-1}{2M} - \frac{k-1}{2N}\right)\right)_{j,k=1}^{M,N}$. Let K_R be any smooth 1–periodic function with

$$K_R(j/n) = K(j/n) \qquad (j = -n/2+1,\dots,n/2-1)$$

and with an arbitrary boundary value $K_R(-1/2)$. Then we have by the aliasing formula that

$$\begin{aligned} K_R(x) &= \sum_{l\in\mathbb{Z}} c_l(K_R)\,\mathrm{e}^{2\pi\mathrm{i}lx} \\ &= \sum_{l=-n/2}^{n/2-1} c_l(K_R)\,\mathrm{e}^{2\pi\mathrm{i}lx} + \sum_{l=-n/2}^{n/2-1} \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} c_{l+rn}(K_R)\,\mathrm{e}^{2\pi\mathrm{i}(l+rn)x} \\ &= \sum_{l=-n/2}^{n/2-1} b_l\,\mathrm{e}^{2\pi\mathrm{i}lx} + \sum_{l=-n/2}^{n/2-1} \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} c_{l+rn}(K_R)\,\mathrm{e}^{2\pi\mathrm{i}lx}(\mathrm{e}^{2\pi\mathrm{i}nrx} - 1), \end{aligned} \tag{2.2}$$

where

$$b_l := \frac{1}{n} \sum_{j=-n/2}^{n/2-1} K_R\left(\frac{j}{n}\right) \mathrm{e}^{-2\pi\mathrm{i}jl/n}. \tag{2.3}$$

For $x := (j-1)/(2M) - (k-1)/(2N)$, we see by definition of n that $\mathrm{e}^{2\pi\mathrm{i}nrx} - 1$ vanishes. Thus

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

and by (2.1)

$$\begin{aligned} f(y_j) &= \sum_{k=1}^{N} \alpha_k \sum_{l=-n/2}^{n/2-1} b_l\,\mathrm{e}^{2\pi\mathrm{i}l((j-1)/(2M)-(k-1)/(2N))} \\ &= \sum_{l=-n/2}^{n/2-1} b_l \left(\sum_{k=1}^{N} \alpha_k\,\mathrm{e}^{-2\pi\mathrm{i}l(k-1)/(2N)}\right) \mathrm{e}^{2\pi\mathrm{i}l(j-1)/(2M)}. \end{aligned}$$

This suggests the following algorithm for the fast summation of (2.1):

Algorithm 2.2.

Precomputation: Computation of $(b_l)_{l=-n/2}^{n/2-1}$ by (2.3).

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

$$a_l := \sum_{k=1}^{N} \alpha_k\,\mathrm{e}^{-2\pi\mathrm{i}l(k-1)/(2N)}$$

by FFT(2N) and applying that $a_{l+2Ns} = a_l$ for $s = -(n-2N)/(4N),\dots,(n-2N)/(4N)$.

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

$$d_l := a_l b_l.$$

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

$$f\left(y_j\right) = \sum_{l=-n/2}^{n/2-1} d_l \, \mathrm{e}^{2\pi \mathrm{i} l(j-1)/(2M)} = \sum_{l=-M}^{M-1} \left(\sum_{s=-(n-2M)/(4M)}^{(n-2M)/(4M)} d_{l+2Ms} \right) \mathrm{e}^{2\pi \mathrm{i} l(j-1)/(2M)}$$

by IFFT($2M$).

For $M = N$ we have that $\boldsymbol{K}_N := \left(K(\frac{j-k}{2N})\right)_{j,k=1}^{N}$ is an N by N Toeplitz matrix. In this case Algorithm 2.2 coincides with the standard Toeplitz matrix – vector multiplication algorithm based on embedding $\boldsymbol{K}_N$ into an $2N$ by $2N$ circulant matrix, and than carrying out the multiplication by using the fast Fourier transform (see e.g. [10]). Note that there exist faster algorithms for the multiplication of a vector with a Toeplitz matrix based on trigonometric transforms [27]. These algorithms can be derived in a similar way as above by using Chebyshev expansions instead of Fourier expansions.

3. Fast Summation at Nonequispaced Knots.

Let

$$f(x) := \sum_{k=1}^{N} \alpha_k K(x - x_k), \tag{3.1}$$

where $|x_k| \le 1/4 - 1/32$. To keep the notation simple, we restrict our attention to even kernels K which are in C^∞ except for the origin. In the following subsection, we propose an algorithm for the fast evaluation of f at M nonequispaced knots y_j, where $|y_j| \le 1/4 - 1/32$. Note that, for some reasons which will become clear in the next subsection, we have scaled the knots so that $|y_j - x_k| \le 1/2 - 1/16 = 7/16$. Error estimates, which determine together with the desired arithmetic complexity of our algorithm the choice of the parameters, are given in Subsection 3.2. Numerical results for the even kernels in (1.2) are presented in Subsection 3.3.

3.1. The Algorithm.

The outline of our algorithm can be sketched as follows: we regularize K near 0 and near the boundary $\pm 1/2$ to obtain a 1–periodic smooth kernel K_R in the Sobolev space $H^p(\mathbb{T})$. Then we approximate K_R by the Fourier series K_{RF} given by

$$K_{RF}(x) := \sum_{l=-n/2}^{n/2-1} b_l \, \mathrm{e}^{2\pi \mathrm{i} l x}, \tag{3.2}$$

where $n \le 2N$ and

$$b_l := \frac{1}{n} \sum_{j=-n/2}^{n/2-1} K_R\left(\frac{j}{n}\right) \mathrm{e}^{-2\pi \mathrm{i} jl/n} \qquad (l = -n/2, \ldots n/2 - 1). \tag{3.3}$$

Note that one can also use the more symmetric setting

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

where $\varepsilon_l := 1/2$ if $l = \pm n/2$ and $\varepsilon_l := 1$ otherwise. This would introduce only small modifications in the following considerations.
Regarding that

$$K = (K - K_R) + (K_R - K_{RF}) + K_{RF} = K_{NE} + K_{ER} + K_{RF}, \tag{3.4}$$

where $K_{NE} := K - K_R$ and $K_{ER} := K_R - K_{RF}$ and assuming that K_{ER} becomes sufficiently small, we approximate K by $K_{NE} + K_{RF}$ and consequently f by

$$\tilde{f}(x) := f_{NE}(x) + f_{RF}(x), \tag{3.5}$$

where

$$f_{RF}(x) := \sum_{k=1}^{N} \alpha_k K_{RF}(x - x_k) \tag{3.6}$$

and where

$$f_{NE}(x) := \sum_{k=1}^{N} \alpha_k K_{NE}(x - x_k) \tag{3.7}$$

denotes similar to multipole methods the near field summation at x. Instead of f we intend to evaluate $\tilde{f}$ at the points y_j. We suppose that the knots x_k $(k = 1, \ldots, N)$ are uniformly distributed so that every interval of length $1/n$ contains at most ν points, i.e. $n \approx 2N/\nu$. Choosing K_R such that $\operatorname{supp}(K - K_R) \cap [-7/16, 7/16] \subseteq [-a/n, a/n]$ with an appropriate parameter a, the evaluation of f_{NE} at the M points y_j requires only $\mathcal{O}(a\nu M)$ arithmetic operations. Similarly, if instead the points y_j $(j = 1, \ldots, M)$ are uniformly distributed so that every interval of length $1/n$ contains at most ν of these points, i.e. $n \approx 2M/\nu$, then the evaluation of f_{NE} requires $\mathcal{O}(a\nu N)$ arithmetic operations.
Note that by our scaling of the input knots x_k and y_j we have no additional computational effort at the boundary here. By (3.2) we can rewrite (3.6) as

$$f_{RF}(x) = \sum_{k=1}^{N} \alpha_k \sum_{l=-n/2}^{n/2-1} b_l \, \mathrm{e}^{2\pi \mathrm{i} l(x - x_k)}$$

which further implies that

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

The expression in the inner brackets can be computed by an $\mathrm{NFFT}^{\mathrm{T}}(n)$, this will be followed by n multiplications with b_l and completed by an $\mathrm{NFFT}(n)$ to compute the outer summation with the complex exponentials. If m is the cut-off parameter and $\sigma = 2$ the oversampling factor of the $\mathrm{NFFT}^{\mathrm{T}}/\mathrm{NFFT}$, then the proposed evaluation of f_{RF} at the points y_j $(j = 1, \ldots, M)$ requires $\mathcal{O}(m(N + M) + n\log(n))$ arithmetic operations.

Before summarizing the above algorithm we have to describe the regularization K_R of K near the origin and the boundary more precisely.
Since K is even, we construct K_R on $[-1/2, 1/2]$ as follows:

$$K_R(x) := \begin{cases} T_I(x) & \text{if } x \in (-\frac{a}{n}, \frac{a}{n}), \\ T_B(x) & \text{if } x \in [-\frac{1}{2}, -\frac{7}{16}) \cup (\frac{7}{16}, \frac{1}{2}], \\ K(x) & \text{otherwise.} \end{cases} \tag{3.8}$$

Here we choose

$$T_I(x) := \sum_{j=0}^{p-1} a_j^I \cos \frac{\pi n j}{2a} x, \tag{3.9}$$

where the coefficients a_j^I are determined by

$$T_I^{(r)}\left(\frac{a}{n}\right) = K^{(r)}\left(\frac{a}{n}\right) \qquad (r = 0, \ldots, p-1), \tag{3.10}$$

and

$$T_B(x) := \begin{cases} \sum\limits_{j=0}^{p-1} a_j^B \cos(8j\pi(x - 1/2)) & \text{if } x \in (7/16, 1/2], \\ \sum\limits_{j=0}^{p-1} a_j^B \cos(8j\pi(x + 1/2)) & \text{if } x \in [-1/2, -7/16), \end{cases} \tag{3.11}$$

where the coefficients a_j^B are defined by

$$T_B^{(r)}(7/16) = K^{(r)}(7/16) \qquad (r = 0, \ldots, p-1). \tag{3.12}$$

Note that we can use an expansion with sine functions for odd kernels K and Fourier expansions for other kernels. The ansatz (3.9) and the conditions (3.10) lead to the following linear systems of equations

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

$$\sum_{j=1}^{\lfloor \frac{p-1}{2} \rfloor} a_{2j}^I (2j)^{2r} (-1)^{r+j} = \left(\frac{2a}{\pi n}\right)^{2r} K^{(2r)}\left(\frac{a}{n}\right) \quad (r = 1, \ldots, \left\lfloor \frac{p-1}{2} \right\rfloor) , \tag{3.13}$$

$$\sum_{j=0}^{\lfloor \frac{p-2}{2} \rfloor} a_{2j+1}^I (2j+1)^{2r+1} (-1)^{r+j} = -\left(\frac{2a}{\pi n}\right)^{2r+1} K^{(2r+1)}\left(\frac{a}{n}\right) \ (r = 0, \ldots, \left\lfloor \frac{p-2}{2} \right\rfloor) . \tag{3.14}$$

Here $\lfloor a \rfloor$ denotes the largest integer $\leq a$. Similarly, we obtain from (3.11) and (3.12) linear systems of equations with the same coefficient matrices

$$\boldsymbol{V}_{\text{even}}^{\mathrm{T}} := \left((-1)^{r+j} (2j)^{2r}\right)_{r,j=1}^{\lfloor \frac{p-1}{2} \rfloor} , \quad \boldsymbol{V}_{\text{odd}}^{\mathrm{T}} := \left((-1)^{r+j} (2j+1)^{2r+1}\right)_{r,j=0}^{\lfloor \frac{p-2}{2} \rfloor} \tag{3.15}$$

as in (3.13), (3.14) but with different right–hand sides

$$\left((8\pi)^{-2r} K^{(2r)}(7/16)\right)_{r=1}^{\lfloor \frac{p-1}{2} \rfloor} , \quad \left((8\pi)^{-(2r+1)} K^{(2r+1)}(7/16)\right)_{r=0}^{\lfloor \frac{p-2}{2} \rfloor} . \tag{3.16}$$

Thus, the computation of the coefficients a_j^I and a_j^B requires the solution of 4 small (size $\approx \frac{p}{2}$) linear systems of equations with two different mosaic Vandermonde-like coefficient matrices (3.15). These matrices will come into the play in our error estimates.

Now we can summarize our algorithm.

Algorithm 3.1.

Precomputation:

i) Computation of a_j^I, a_j^B $(j = 0, \ldots, p-1)$ *by* (3.13), (3.14) *and* (3.16).

ii) Computation of $(b_l)_{l=-n/2}^{n/2-1}$ *by* (3.3) *and* (3.8).

iii) Computation of $K_{NE}(y_j - x_k)$ *for all* $j = 1, \ldots, M$ *and* $k \in I_{n,a}^{NE}(y_j)$, *where* $I_{n,a}^{NE}(y_j) := \{k \in \{1, \ldots, N\} : |y_j - x_k| < \frac{a}{n}\}$. *See also Remark* 3.6.

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

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

by NFFT$^{\mathrm{T}}(n)$, *see Algorithm 5.2 in the appendix.*

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

$$d_l := a_l b_l .$$

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

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

by NFFT(n), see Algorithm 5.1 in the appendix.

4. *For $j = 1, \ldots, M$ compute the near field summations*

$$f_{NE}(y_j) = \sum_{k \in I_{n,a}^{NE}(y_j)} \alpha_k K_{NE}(y_j - x_k).$$

5. *For $j = 1, \ldots, M$ compute the near field corrections*

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

Our algorithm requires

$$\mathcal{O}(n \log n + m(N + M) + a\nu M) \tag{3.17}$$

arithmetic operations, where m is given by the NFFT/NFFT$^{\mathrm{T}}$ algorithms in the appendix.

Remark 3.2. (Relation to the approach of Beylkin and Cramer [8])
Recently, Beylkin and Cramer have proposed a fast summation algorithm based on the multiresolution concept. This algorithm requires also $\mathcal{O}(N \log N + N)$ arithmetic operations. We want to sketch its relation to our approach. Shortly speaking Beylkin's and Cramer's idea is the following: for $|y - x| > R$ they showed that the kernel K can be approximated with accuracy dependent on R by

$$K(x - y) \approx \sum_{r,s} t_{r-s} \, \varphi\left(x - \frac{r}{2^j}\right) \varphi\left(y - \frac{s}{2^j}\right), \tag{3.18}$$

where φ denotes a (dilated) scaling function with small support. As in [6] the authors prefer (dilated) cardinal B–splines as φ, see (5.10). The coefficients t_k are simply kernel values for large indices k and must be computed by solving a linear system of equations which involves the mask of the two–scale relation of the autocorrelation function of φ for smaller indices k. Then (1.1) can be approximated by

$$\begin{aligned} f(y) \approx & \sum_{k=1}^{N} \alpha_k \sum_{r,s} t_{r-s} \, \varphi\left(x_k - \frac{r}{2^j}\right) \varphi\left(y - \frac{s}{2^j}\right) \\ & - \sum_{k \in I_y} \alpha_k \sum_{r,s} t_{r-s} \, \varphi\left(x_k - \frac{r}{2^j}\right) \varphi\left(y - \frac{s}{2^j}\right) + \sum_{k \in I_y} \alpha_k K(y - x_k), \end{aligned}$$

where $I_y := \{x_k : |y - x_k| \le R\}$. The computation of the last two sums requires as Step 4 of Algorithm 3.1 only $\mathcal{O}(M)$ arithmetic operations, where the constant in $\mathcal{O}(M)$ depends on R. The first sum can be rewritten as

$$\sum_{s} \sum_{r} t_{r-s} \sum_{k=1}^{N} \alpha_k \, \varphi\left(x_k - \frac{r}{2^j}\right) \, \varphi\left(y - \frac{s}{2^j}\right),$$

which implies the following computation scheme:

- Computation of

$$g_r := \sum_{k=1}^{N} \alpha_k \, \varphi\left(x_k - \frac{r}{2^j}\right)$$

  for all r. Since φ has small support, this requires for fixed r only a small number of additions and for all r only $\mathcal{O}(N)$ arithmetic operations.

- Computation of $\tilde{g}_s := \sum_r g_r t_{r-s}$ for all s.

  This is a Toeplitz matrix times vector multiplication which can be realized as in Section 2 by FFTs of length 2 times the size of the Toeplitz matrix, i.e. in $\mathcal{O}(N \log N)$ arithmetic operations.
- Computation of $\sum_s \tilde{g}_s \, \varphi\left(y - \frac{s}{2^J}\right)$.

  By the small support of φ this requires $\mathcal{O}(M)$ arithmetic operations.

Let us turn to our algorithm. By (5.8) in the appendix, we see that our NFFT algorithm uses the approximation

$$\mathrm{e}^{-2\pi i k x} \approx \frac{1}{\sigma n \hat{\varphi}(k)} \sum_r \tilde{\psi}\left(x - \frac{r}{\sigma n}\right) \mathrm{e}^{-2\pi i k r/(\sigma n)},$$

with appropriate real–valued even functions φ and ψ. If φ has compact support, e.g. in case of (dilated) B-splines φ, then $\psi = \varphi$. Using this in (3.2), we obtain for $|y - x| > a/n$ (and $|y - x| \leq 1/2 - a/n$) with accuracy depending on a/n the approximation

$$\begin{aligned} K_{RF}(y-x) &\approx \sum_{l=-n/2}^{n/2-1} b_l \, \mathrm{e}^{-2\pi i l x} \, \mathrm{e}^{2\pi i l y} \\ &\approx \sum_{l=-n/2}^{n/2-1} \frac{b_l}{(\sigma n)^2 |\hat{\varphi}(l)|^2} \sum_{r,s} \tilde{\psi}\left(x - \frac{r}{\sigma n}\right) \mathrm{e}^{-2\pi i l r/(\sigma n)} \, \tilde{\psi}\left(y - \frac{s}{\sigma n}\right) \mathrm{e}^{2\pi i l s/(\sigma n)} \\ &= \sum_{r,s} \tilde{t}_{r-s} \tilde{\psi}\left(x - \frac{r}{\sigma n}\right) \tilde{\psi}\left(y - \frac{s}{\sigma n}\right), \end{aligned}$$

where

$$\tilde{t}_j := \sum_{l=-n/2}^{n/2-1} \frac{b_l}{(\sigma n)^2 |\hat{\varphi}(l)|^2} \, \mathrm{e}^{-2\pi i l j/(\sigma n)}.$$

This shows the relation to (3.18). However, we think that the details of our algorithm, in particular our regularization procedure, are much simpler than those in [8]. Moreover, our φ is not restricted to functions satisfying a two–scale relation. □

3.2. Error Estimates. In this section, we show that the error involved in our algorithm can be kept sufficiently small, where we have to pay for smaller errors with a larger amount of arithmetic operations and thus with a slower speed of the performance.

Beyond the well–known errors appearing in the NFFT computations, see appendix, our algorithm introduces the errors $|f(y_j) - \tilde{f}(y_j)|$ $(j = 1, \ldots, M)$. By (3.4), (3.5) and (3.8), we see for $|y| \leq 1/4 - 1/32$ that

$$|f(y) - \tilde{f}(y)| = |\sum_{k=1}^{N} \alpha_k K_{ER}(y - x_k)|$$

and by Hölder's inequality with $1/q + 1/q' = 1$ $(1 \leq q \leq \infty)$ that

$$|f(y) - \tilde{f}(y)| \leq ||\boldsymbol{\alpha}||_q \, ||(K_{ER}(y - x_k))_{k=1}^N||_{q'} \tag{3.19}$$

where $||\boldsymbol{\alpha}||_q := \left(\sum_{k=1}^N |\alpha_k|^q\right)^{1/q}$ for $1 \leq q < \infty$ and $||\boldsymbol{\alpha}||_\infty := \max_{k=1,\ldots,N} |\alpha_k|$. In the following, we restrict our attention in (3.19) to $q = 1$. Note that this estimate is sharp if $\alpha_k = 0$ for all indices k except for this one which corresponds to the largest value of $|K_{ER}(y - x_k)|$. If the values α_k are nearly equal for all k it would be better to choose $q = \infty$. This will be considered in a forthcoming paper. In case $q = 1$, it remains to estimate $||K_{ER}||_\infty := \max_{x \in [-1/2, 1/2]} |K_{ER}(x)|$.

THEOREM 3.3. *Let K be an even function which is C^∞ except for the origin. Assume that K satisfies*

$$|K^{(r)}(x)| \le C\,\frac{(r+\alpha-1)!}{|x|^{r+\alpha}} \quad (x \in [-1/2,1/2]\backslash\{0\}), \tag{3.20}$$

for all $r = 1,\dots,p$ ($p \ge 2$) and some nonnegative constants $C \in \mathbb{R}$ and $\alpha \in \mathbb{N}_0$ depending only on K. Then, for $2 \le p \le 50$, the following estimate holds true

$$||K_{ER}||_\infty \le C\,\frac{n^\alpha}{a^{\alpha-1}}\,\frac{(p+\alpha+\delta_{0,\alpha}-2)!}{p-1}\left(\frac{\beta\pi}{4a}\right)^p,$$

where $\beta := \sqrt{1+(2/\pi)^2}$ and $\delta_{0,\alpha} := \max\{0, 1-\alpha\}$.

Note that condition (3.20) is fulfilled for our homogeneous kernels, e.g., we have $\alpha = 1$ for $K(x) = \frac{1}{x}$ and $\alpha = 2$ for $K(x) = \frac{1}{x^2}$. Further we see that $\alpha = 0$ for $K(x) = \log|x|$. The theorem can be enlarged for $\alpha \in \mathbb{R}_+$ by carefully handling the factorial.
We conjecture that the above restriction $p \le 50$ is not necessary. However, our proof incorporates some numerical computations up to $p = 50$.

Proof. By the same procedure as in the previous section (see (2.2)), i.e., by Fourier expansion of K_R and application of the aliasing formula, we obtain that

$$K_{ER}(x) = \sum_{k=-n/2}^{n/2-1} \sum_{\substack{r\in\mathbb{Z}\\ r\ne 0}} c_{k+rn}(K_R)\,\mathrm{e}^{2\pi\mathrm{i}kx}(\,\mathrm{e}^{2\pi\mathrm{i}rnx}-1).$$

Note that by an appropriate choice of n, this error was zero for equispaced knots. For arbitrary $x \in [-1/2,1/2]$, we can further estimate

$$|K_{ER}(x)| \le 2\left(2\sum_{k=n/2+1}^{\infty} |c_k(K_R)| + |c_{n/2}(K_R)|\right).$$

By construction we have that $K_R \in H^p(\mathbb{T})$ which implies that

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

so that

$$|K_{ER}(x)| \le 2\left(2\sum_{k=n/2+1}^{\infty} (2\pi k)^{-p} + (\pi n)^{-p}\right)\int_{-1/2}^{1/2} |K_R^{(p)}(x)|\,\mathrm{d}x.$$

For $p \ge 2$ the above sum can be estimated by an upper integral. This leads to

$$|K_{ER}(x)| \le \frac{C}{(p-1)\pi^p n^{p-1}}\int_{-1/2}^{1/2} |K_R^{(p)}(x)|\,\mathrm{d}x$$

and since K and T_I, T_B are even to

$$|K_{ER}(x)| \le \frac{C}{(p-1)\pi^p n^{p-1}}\left(\int_0^{\frac{a}{n}} |T_I^{(p)}(x)|\,\mathrm{d}x + \int_{\frac{a}{n}}^{\frac{7}{16}} |K^{(p)}(x)|\,\mathrm{d}x + \int_{\frac{7}{16}}^{\frac{1}{2}} |T_B^{(p)}(x)|\,\mathrm{d}x\right). \tag{3.21}$$

Now we have to take the definition of the kernel K into consideration. By (3.20), the integral in the middle of (3.21) can be estimated by

$$\begin{aligned} \int_{\frac{a}{n}}^{\frac{7}{16}} |K^{(p)}(x)|\,\mathrm{d}x &\le C\,(p+\alpha-1)! \int_{\frac{a}{n}}^{\frac{7}{16}} x^{-(p+\alpha)}\,\mathrm{d}x \\ &\le C\,(p+\alpha-2)!\,\left(\frac{n}{a}\right)^{p+\alpha-1}. \end{aligned} \tag{3.22}$$

For the estimation of the first integral in (3.21) we restrict our attention to even $p \ge 2$. Similar estimates can be obtained for odd p. Set $q := p/2 - 1$. Then we see by (3.9) that

$$\begin{aligned} |T_I^{(p)}(x)| &= \left(\frac{\pi n}{2a}\right)^p \left| \sum_{j=1}^{p-1} j^p a_j^I \cos \frac{\pi n j}{2a} x \right|, \\ &\le \left(\frac{\pi n}{2a}\right)^p \sum_{j=1}^{p-1} j^p |a_j^I| \end{aligned}$$

and consequently

$$\int_0^{\frac{a}{n}} |T_I^{(p)}(x)|\,\mathrm{d}x \le \left(\frac{\pi}{2}\right)^p \left(\frac{n}{a}\right)^{p-1} \sum_{j=1}^{p-1} j^p |a_j^I|. \tag{3.23}$$

The coefficients a_{2j}^I and a_{2j+1}^I are determined by the linear systems (3.13) and (3.14) with corresponding coefficient matrices (3.15). By considering

$$\boldsymbol{V}_{\text{even}}\,(c_{k,l})_{k,l=1}^q = \boldsymbol{I}_q$$

where $\boldsymbol{I}_q$ denotes the $q \times q$ identity matrix, and regarding the relation between Vandermonde matrices and polynomial interpolation, we conclude that $c_{k,l}$ are just the coefficients of the even Lagrange polynomials

$$L_{2l,p}(x) = \sum_{k=1}^q (-1)^{k+l} c_{k,l}\, x^{2k} := \frac{x^2}{(2l)^2} \prod_{\substack{s=1\\ s\neq l}}^q \frac{x^2-(2s)^2}{(2l)^2-(2s)^2}. \tag{3.24}$$

Using Vieta's Theorem, we see that the coefficients $c_{k,l}$ are positive. Together with $(-1)^{k+l}\,c_{k,l} = L_{2l,p}^{(2k)}(0)/(2k)!$ this implies $c_{k,l} = |L_{2l,p}^{(2k)}(0)|/(2k)!$ and consequently

$$(\boldsymbol{V}_{\text{even}}^{\mathrm{T}})^{-1} = \left(\frac{|L_{2l,p}^{(2k)}(0)|}{(2k)!} \right)_{l,k=1}^q.$$

In the same way, we see that

$$(\boldsymbol{V}_{\text{odd}}^{\mathrm{T}})^{-1} = \left(\frac{|L_{2l+1,p}^{(2k+1)}(0)|}{(2k+1)!} \right)_{l,k=0}^q,$$

where

$$L_{2l+1,p}(x) = \frac{x}{(2l+1)} \prod_{\substack{s=0\\ s\neq l}}^q \frac{x^2-(2s+1)^2}{(2l+1)^2-(2s+1)^2}. \tag{3.25}$$

Thus, by (3.13),

$$(a^I_{2j})_{j=1}^q = (\boldsymbol{V}^{\mathrm{T}}_{\text{even}})^{-1} \left(\left(\frac{2a}{\pi n}\right)^{2r} K^{(2r)} \left(\frac{a}{n}\right) \right)_{r=1}^q \tag{3.26}$$

which implies together with (3.20) that

$$\begin{aligned} |a^I_{2j}| &\leq C \left(\frac{n}{a}\right)^{\alpha} \sum_{k=1}^{q} \left(\frac{2}{\pi}\right)^{2k} \frac{(2k+\alpha-1)!}{(2k)!} \, |L^{(2k)}_{2j,p}(0)| \\ &\leq C \, \frac{(p-3+\alpha+\delta_{0,\alpha})!}{(p-2)!} \left(\frac{n}{a}\right)^{\alpha} \sum_{k=1}^{q} \left(\frac{2}{\pi}\right)^{2k} |L^{(2k)}_{2j,p}(0)|. \end{aligned}$$

The analogue for the coefficients with odd index is

$$|a^I_{2j+1}| \;\leq\; C \, \frac{(p-2+\alpha+\delta_{0,\alpha})!}{(p-1)!} \left(\frac{n}{a}\right)^{\alpha} \sum_{k=0}^{q} \left(\frac{2}{\pi}\right)^{2k+1} |L^{(2k+1)}_{2j+1,p}(0)|.$$

Now we obtain by (3.23) that

$$\int_0^{\frac{a}{n}} |T^{(p)}_I(x)| \, \mathrm{d}x \leq C \left(\frac{\pi}{2}\right)^{p} \left(\frac{n}{a}\right)^{p-1+\alpha} \frac{(p+\alpha-2+\delta_{0,\alpha})!}{(p-1)!} \left(S_{\text{even}}(p) + S_{\text{odd}}(p)\right),$$

where

$$S_{\text{even}}(p) := \sum_{k=1}^{q} \left(\frac{2}{\pi}\right)^{2k} \sum_{j=1}^{q} (2j)^p \, |L^{(2k)}_{2j,p}(0)| \, , \tag{3.27}$$

$$S_{\text{odd}}(p) := \sum_{k=0}^{q} \left(\frac{2}{\pi}\right)^{2k+1} \sum_{j=0}^{q} (2j+1)^p |L^{(2k+1)}_{2j+1,p}(0)| \, . \tag{3.28}$$

The values $S_{\text{even}}(p)$ and $S_{\text{odd}}(p)$ can be estimated by Lemma 3.4. Using these results, we get

$$\int_0^{\frac{a}{n}} |T^{(p)}_I(x)| \, \mathrm{d}x \;\leq\; C \, (p-2+\alpha+\delta_{0,\alpha})! \left(\frac{\pi}{2}\right)^{p} \left(\frac{n}{a}\right)^{p-1+\alpha} \left(\frac{\beta\pi}{2}\right)^{p} . \tag{3.29}$$

Finally, the third integral on the right–hand side of (3.21) can be estimated in a similar way. We finish the proof by combining this result with (3.29), (3.22) and (3.21). ■

LEMMA 3.4. *Let $p \geq 2$ be even and let $q := p/2 - 1$. Further, let $L_{2j,p}$ and $L_{2j+1,p}$ be given by (3.24) and (3.25), respectively. Then the sums $S_{\text{even}}(p)$ and $S_{\text{odd}}(p)$ in* (3.27) *and* (3.28) *can be estimated by*

$$S_{\text{even}}(p) \leq \left(\frac{2}{\pi}\right)^{2} \frac{1}{2} \left(\frac{\beta}{2}\right)^{p-4} \sum_{k=0}^{q-1} (p-2k-2)^p \, \frac{\left[\frac{2^k q!}{(q-k)!}\right]^2}{k! \frac{(p-k-2)!}{(p-2k-2)!}} \, , \tag{3.30}$$

$$S_{\text{odd}}(p) \leq \left(\frac{2}{\pi}\right) \left(\frac{\beta}{2}\right)^{p-2} \sum_{k=0}^{q} (p-2k-1)^p \, \frac{\left[\frac{(p-1)!(q-k)!}{2^k (p-2k-1)! q!}\right]^2}{k! \frac{(p-k-1)!}{(p-2k-1)!}} \, . \tag{3.31}$$

For $p \leq 50$, we have

$$S_{\text{even/odd}}(p) \;\leq\; C \left(\frac{\beta\pi}{2}\right)^{p} (p-1)! \tag{3.32}$$

with a constant $C < 1/2$.

Proof. We restrict our attention to $S_{\text{even}}(p)$. The upper bound for $S_{\text{odd}}(p)$ can be deduced in a similar way.
First we see by (3.24) that

$$L_{p-2,p}(x) = \frac{x^2}{(p-2)^2} \prod_{s=1}^{q-1} \frac{x^2-(2s)^2}{(p-2)^2-(2s)^2} = \frac{1}{(p-2)!} \frac{1}{2^{p-3}} \prod_{s=0}^{q-1} (x^2-(2s)^2)$$

and that

$$L_{2j,p}(x) = L_{2j,p-2}(x) \frac{x^2-(p-2)^2}{(2j)^2-(p-2)^2} \qquad (j = 1, \dots, q-1)\,.$$

For $k = 1, \dots, q$, we obtain by Leibniz's rule

$$\begin{aligned} L_{2j,p}^{(2k)}(x) = L_{2j,p-2}^{(2k)}(x) \frac{x^2-(p-2)^2}{(2j)^2-(p-2)^2} + \frac{2x\,2k}{(2j)^2-(p-2)^2} L_{2j,p-2}^{(2k-1)}(x) \\ + \frac{(2k)(2k-1)}{(2j)^2-(p-2)^2} L_{2j,p-2}^{(2k-2)}(x) \end{aligned}$$

and in particular for $x := 0$

$$L_{2j,p}^{(2k)}(0) = L_{2j,p-2}^{(2k)}(0) \frac{(p-2)^2}{(p-2)^2-(2j)^2} - L_{2j,p-2}^{(2k-2)}(0) \frac{2k(2k-1)}{(p-2)^2-(2j)^2}\,. \tag{3.33}$$

Substituting this in $S_{\text{even}}(p)$ and regarding that $L_{2j,p-2}^{(2k)}(0)$ and $L_{2j,p-2}^{(2k-2)}(0)$ have opposite signs, we get

$$\begin{aligned} S_{\text{even}}(p) &= \sum_{k=1}^{q} \left(\frac{2}{\pi}\right)^{2k} (p-2)^p |L_{p-2,p}^{(2k)}(0)| \\ &+ (p-2)^2 \sum_{k=1}^{q} \left(\frac{2}{\pi}\right)^{2k} \sum_{j=1}^{q-1} \frac{(2j)^p}{(p-2)^2-(2j)^2} |L_{2j,p-2}^{(2k)}(0)| \\ &+ \sum_{k=1}^{q} \left(\frac{2}{\pi}\right)^{2k} 2k(2k-1) \sum_{j=1}^{q-1} \frac{(2j)^p}{(p-2)^2-(2j)^2} |L_{2j,p-2}^{(2k-2)}(0)| \\ &\le (p-2)^p \sum_{k=1}^{q} \left(\frac{2}{\pi}\right)^{2k} |L_{p-2,p}^{(2k)}(0)| \\ &+ \left(1 + \left(\frac{2}{\pi}\right)^2\right) (p-2)^2 \sum_{k=1}^{q-1} \left(\frac{2}{\pi}\right)^{2k} \sum_{j=1}^{q-1} \frac{(2j)^p}{(p-2)^2-(2j)^2} |L_{2j,p-2}^{(2k)}(0)|\,. \end{aligned}$$

Now we apply (3.33) with $p-2$ instead of p in the last sum and iterate this scheme. Setting

$$R(p) := \sum_{k=1}^{q} \left(\frac{2}{\pi}\right)^{2k} |L_{p-2,p}^{(2k)}(0)|,$$

we obtain in this way

$$\begin{aligned}
S_{\text{even}}(p) &\le (p-2)^p R(p) + (p-2)^2\,\beta^2\,\frac{(p-4)^p}{(p-2)^2-(p-4)^2}R(p-2) \\
&+ (p-2)^2(p-4)^2\,\beta^4\,\frac{(p-6)^p}{((p-2)^2-(p-6)^2)((p-4)^2-(p-6)^2)}R(p-4) + \ldots \\
&+ (p-2)^2\ldots 4^2\,\beta^{p-4}\,\frac{2^p}{((p-2)^2-2^2)((p-4)^2-2^2)\ldots(4^2-2^2)}R(4) \\
&= \sum_{k=0}^{q-1}\beta^{2k}\,(p-2k-2)^p\,R(p-2k)\,\frac{\left[\frac{q!}{(q-k)!}\right]^2}{k!\,\frac{(p-k-2)!}{(p-2k-2)!}}\,. \qquad (3.34)
\end{aligned}$$

Next, we consider

$$R(p) = \frac{1}{(p-2)!}\frac{1}{2^{p-3}}\sum_{k=1}^{q}\left(\frac{2}{\pi}\right)^{2k}|w_p^{(2k)}(0)|\,,$$

where

$$w_p(x) = \prod_{s=0}^{q-1}(x^2-(2s)^2)\,.$$

Since

$$w_p(x) = w_{p-2}(x)(x^2-(p-4)^2)\,,$$

we obtain by applying the Leibniz rule again

$$w_p^{(2k)}(0) = -w_{p-2}^{(2k)}(0)(p-4)^2 + 2k(2k-1)w_{p-2}^{(2k-2)}(0).$$

Hence the sum $R(p)$ can be rewritten as

$$\begin{aligned}
R(p) &\le \frac{1}{(p-2)!}\frac{1}{2^{p-3}}\Bigg((p-4)^2\sum_{k=1}^{q-1}\left(\frac{2}{\pi}\right)^{2k}|w_{p-2}^{(2k)}(0)| \\
&\quad + (p-2)(p-3)\left(\frac{2}{\pi}\right)^2\sum_{k=1}^{q-1}\left(\frac{2}{\pi}\right)^{2k}|w_{p-2}^{(2k)}(0)|\Bigg) \\
&\le \frac{1}{(p-4)!}\frac{\beta^2}{2^{p-3}}\sum_{k=1}^{q-1}\left(\frac{2}{\pi}\right)^{2k}|w_{p-2}^{(2k)}(0)| \;=\; \left(\frac{\beta}{2}\right)^2 R(p-2) \\
&\le \left(\frac{\beta}{2}\right)^{p-4} R(4) \;=\; \left(\frac{\beta}{2}\right)^{p-4}\left(\frac{2}{\pi}\right)^2\frac{1}{2}\,.
\end{aligned}$$

Substituting the above expression for $R(p-2k)$ in (3.34), we get (3.30).
The estimates (3.32) can be obtained from (3.30) and (3.31) by straightforward computation on the computer. This completes the proof. ■

By (3.19), Theorem 3.3 and Stirling's formula in the form

$$n! < 1.1\left(\frac{n}{\mathrm{e}}\right)^n\sqrt{2\pi n}$$

we obtain the following corollary:

COROLLARY 3.5. Under the assumptions of Theorem 3.3 the error introduced by Algorithm 3.1 can be estimated by

$$|f(y_j) - \tilde{f}(y_j)| \le C\, \|\boldsymbol{\alpha}\|_1\, n^\alpha \, \frac{\sqrt{p+\alpha+\delta_{0,\alpha}-2}}{a^{1-\delta_{0,\alpha}}(p-1)} \left(\frac{\beta\pi}{4\,\mathrm{e}} \, \frac{p+\alpha+\delta_{0,\alpha}-2}{a} \right)^{p+\alpha+\delta_{0,\alpha}-2} . \tag{3.35}$$

Thus, the error decays exponentially with p if

$$\frac{\beta\pi}{4\,\mathrm{e}} \, \frac{p+\alpha+\delta_{0,\alpha}-2}{a} < 1 \qquad \left(\frac{\beta\pi}{4\,\mathrm{e}} \approx \frac{1}{3} \right).$$

□

In our numerical examples we simply choose $a = p$. Of course one obtains better approximation results for larger values of a. However, by (3.17), this also enlarges the arithmetic complexity of the algorithm.

3.3. Numerical Results. In this section, we present a couple of numerical examples. All algorithms were implemented in C and tested on a SGI O2 using double precision arithmetic.
We have computed (1.1) with $M = N$ and $y_j = x_j$ $(j = 1, \dots, N)$ i.e.,

$$f(x_j) := \sum_{\substack{k=1 \\ k \ne j}}^{N} \alpha_k K(x_j - x_k) \qquad (j = 1, \dots, N)$$

for the different kernels in (1.2) by our Algorithm 3.1. In all computations we have used $n = N$ and randomly chosen uniformly distributed knots x_k in $[-1/4 + 1/32, 1/4 - 1/32]$ so that $\nu \approx 2$ in (3.17). Then, by (3.17), our algorithm has the arithmetic complexity $\mathcal{O}(N \log N + 4mN + 4aM)$.
We always apply the NFFT/NFFT$^{\mathrm{T}}$ with Kaiser–Bessel function φ (see appendix (5.12)) and oversampling factor $\sigma = 2$. The FFT implementation was taken from the "Numerical Recipes in C" [30, p. 523 f]. By using a more sophisticated FFT implementation the speed of our algorithm can be further improved.
The coefficients α_k were randomly distributed in $[0, 1]$. Moreover, every figure presents the arithmetic mean of 20 runs of the algorithm. We are interested in the behavior of the error

$$E := \max_{j=1,\dots,N} \frac{|f(x_j) - \tilde{f}(x_j)|}{|f(x_j)|}. \tag{3.36}$$

First we consider the kernels $K(x) = 1/|x|^\alpha$ $(\alpha = 1, 2)$ in more detail. Note that similar results also happen for the other kernels in (1.2). Figure 3.1 shows the error $\log_{10} E$ of Algorithm 3.1 for $(p, a) \in \{1, \dots, 9\}^2$ and $N = 512$. Here we have set $m = 12$ to ensure that the choice of this parameter doesn't create an additional error in the NFFT/NFFT$^{\mathrm{T}}$. The figure suggests to choose simply $a = p$. This was also recommended from the theoretical point of view in Subsection 3.2.
Next we are interested in a minimal choice of the parameter m since this influences the arithmetical complexity of the NFFT/NFFT$^{\mathrm{T}}$. Figure 3.2 shows the error $\log_{10} E$ for $a = p \in \{1, \dots, 9\}$ and $m \in \{1, \dots, 9\}$ for the kernels $K(x) = 1/|x|$ (left) and $K(x) = 1/|x|^2$ (right), where $N = 512$. We see that for $p \le 5$ one should use the simple setting $a = p = m$, while for larger values of p the choice $m = 5$ leads also to good results.
Figure 3.3 incorporates the other even kernels (1.2). We consider the error E as function of $p = a = m$. The figure confirms the exponential decay of E with increasing p.
Finally, Table 3.1 compares the CPU time of our algorithm and with those of the FMM proposed in [12], Algorithm 3.2. Our implementation of the FMM uses the Chebyshev expansion of order 5 to obtain an accuracy $E \le 10^{-5}$. Note that recently an algorithm which is roughly twice as fast as the used FMM algorithm was developed in [34]. Table 3.1 presents the CPU times as well as the approximation errors. By t_{slow}, t_{apr} and t_{FMM} we denote the computational time for the straightforward algorithm, Algorithm 3.1 and the Algorithm based on the FMM, respectively. Furthermore, Column 3 shows the computational time t_{NE} for Step 4 of Algorithm 3.1 based on Remark 3.6. The Columns 6 and 7 contain the approximation error of Algorithm 3.1 and of the FMM, respectively.

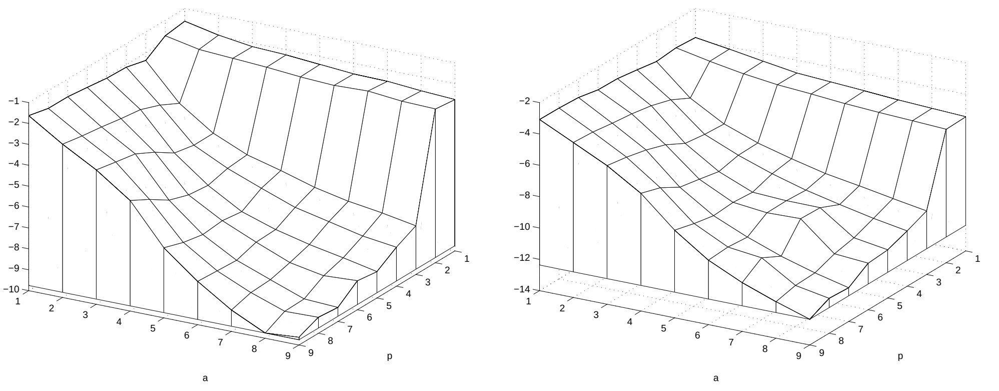

FIG. 3.1. *Error* $\log_{10} E$ *for* $K(x) = 1/|x|$ *(left) and* $K(x) = 1/|x|^2$ *(right) for* $N = 512$*,* $m = 12$ *and* $(p, a) \in \{1, \dots, 9\}^2$.

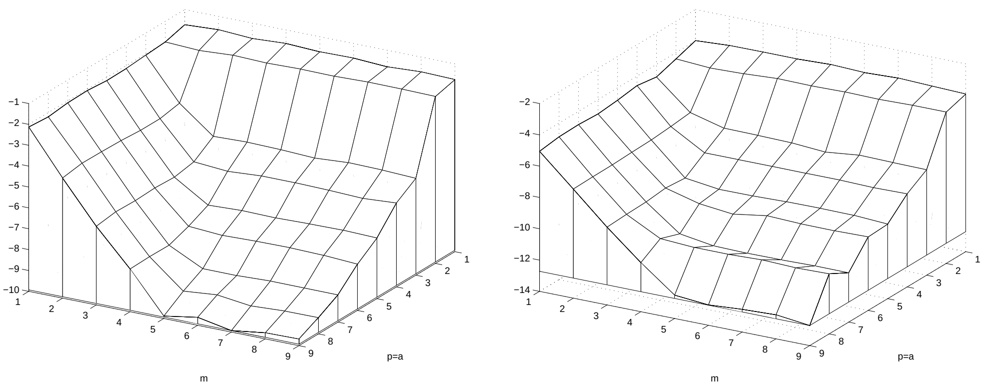

FIG. 3.2. *Error* $\log_{10} E$ *for* $K(x) = 1/|x|$ *(left) and* $K(x) = 1/|x|^2$ *(right) for* $(a, m) \in \{1, \dots, 9\}^2$ *and* $a = p$, $N = 512$.

REMARK 3.6. Up to now we have assumed that the values $K_{NE}(y_j - x_k) = K(y_j - x_k) - K_R(y_j - x_k) = K(y_j - x_k) - T_I(y_j - x_k)$ $(k \in I_{n,a}^{NE}(y_j);\ j = 1, \dots, N)$ in Step 4 of Algorithm 3.1 were precomputed. Alternatively, one can use the following procedure: splitting

$$f_{NE}(y_j) = \sum_{k \in I_{n,a}^{NE}(y_j)} \alpha_k K(y_j - x_k) - \sum_{k \in I_{n,a}^{NE}(y_j)} \alpha_k T_I(y_j - x_k)\,,$$

we evaluate the first sum with $\mathcal{O}(2a\nu M)$ arithmetic operations first. By definition (3.9) of T_I the computation of the second sum would require $\mathcal{O}(2pa\nu M)$ arithmetic operations. We approximate this sum as follows: we precompute $T_I(\frac{a}{n}\frac{s}{ap})$ $(s = -ap, \dots, ap)$ and approximate $T_I(y_j - x_k)$ by cubic spline interpolation. Now the (approximate) computation of the second sum requires nearly the same number of arithmetic operations as the computation the first one. □

4. Fast Summation at Multivariate Nonequispaced Knots. In this section we briefly explain how to extend our univariate scheme to higher dimensions.
Of course, the generalization of Algorithm 3.1 for the tensor product setting, i.e., for the fast summation

$$f(\boldsymbol{y}_j) := \sum_{k=1}^{N} \alpha_k \prod_{s=1}^{d} K^s(y_j^s - x_k^s) \qquad (j = 1, \dots, M),$$

where $\boldsymbol{x}_k := (x_k^s)_{s=1}^d$, $\boldsymbol{y}_k := (y_k^s)_{s=1}^d$ and K^s are some univariate kernels considered in the previous section, is straightforward. With respect to Section 2 we only note that our fast summation algorithm on a regular two–dimensional grid coincides with an algorithm based on the embedding the corresponding block–Toeplitz–Toeplitz–block matrix in a block circulant matrix with circulant blocks (see [10]).

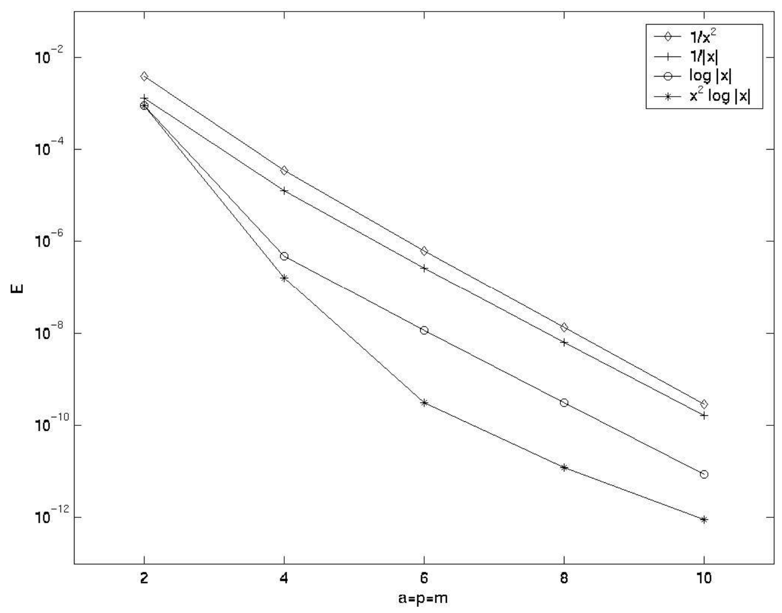

FIG. 3.3. *Error E for various kernels and $a = p = m \in \{2,4,6,8,10\}$, where $N = 512$.*

	Computational Time				Error	
$N = n$	t_{slow}	t_{NE}	t_{apr}	t_{FMM}	E	E_{FMM}
64	1.152e-03	1.193e-03	2.952e-03	1.584e-03	1.634e-06	1.060e-05
128	4.608e-03	2.429e-03	5.911e-03	3.243e-03	6.778e-06	1.580e-05
256	1.849e-02	4.938e-03	1.194e-02	6.738e-03	4.521e-06	1.223e-05
512	7.396e-02	9.557e-03	2.419e-02	1.450e-02	6.366e-06	5.0164e-06
1024	3.010e-01	1.956e-02	5.034e-02	3.940e-02	9.184e-06	4.010e-06
2048	1.524e+00	3.953e-02	1.171e-01	8.093e-02	9.483e-06	5.676e-06
4096	6.702e+00	7.843e-02	2.559e-01	2.263e-01	4.256e-06	1.770e-06
8192	2.730e+01	1.590e-01	5.997e-01	1.112e+00	5.449e-06	1.372e-06

TABLE 3.1

Comparison of the computational time and of the approximation error of Algorithm 3.1 with $a = p = m = 4$ and of the FMM for $K(x) = 1/|x|$.

Other algorithms for the fast multiplication of vectors with block–Toeplitz–Toeplitz–block matrices based on trigonometric transforms can be found e.g. in [28].

In this paper we are interested in rotation-invariant kernels, i.e., in radial basis functions $\mathcal{K}(\boldsymbol{x}) = K(\|\boldsymbol{x}\|)$, where we denote by $\|\cdot\|$ the Euclidean norm in $\mathbb{R}^d$. Again we assume that K is even. We focus on the fast computation of

$$f(\boldsymbol{y}_j) := \sum_{k=1}^{N} \alpha_k K(\|\boldsymbol{y}_j - \boldsymbol{x}_k\|) \qquad (j = 1, \ldots, M)\,, \tag{4.1}$$

where $\boldsymbol{x}_k, \boldsymbol{y}_j \in \mathbb{R}^d$ are with $\|\boldsymbol{x}_k\| \le \frac{1}{4} - \frac{1}{32}$ and $\|\boldsymbol{y}_k\| \le \frac{1}{4} - \frac{1}{32}$. Note that we have by the above scaling that $\|\boldsymbol{y}_j - \boldsymbol{x}_k\| \le \frac{1}{2} - \frac{1}{16} = \frac{7}{16}$. An advantage of rotation-invariant kernels is that we can use nearly the same regularization procedure as in the univariate case. Only the NFFT/NFFT$^{\text{T}}$ computations require really a multivariate setting.

4.1. The Algorithm. Similar as in Subsection 3.1 we regularize $\mathcal{K}$ near $\mathbf{0}$ and near the boundary of $\Pi^d := [-\frac{1}{2}, \frac{1}{2})^d$ to obtain a smooth kernel $\mathcal{K}_R$. Then we approximate $\mathcal{K}_R$ by the Fourier series

$$\mathcal{K}_{RF}(\boldsymbol{x}) := \sum_{\boldsymbol{l} \in I_n} b_{\boldsymbol{l}} \, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{l} \boldsymbol{x}},$$

where $I_n := \{\boldsymbol{l} \in \mathbb{Z}^d : -\frac{n}{2} \le \boldsymbol{l} < \frac{n}{2}\}$ with componentwise inequalities and

$$b_{\boldsymbol{l}} := \frac{1}{n^d} \sum_{\boldsymbol{j} \in I_n} \mathcal{K}_R(\boldsymbol{j}/n) \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{j} \boldsymbol{l}/n} \quad (\boldsymbol{l} \in I_n) \,. \tag{4.2}$$

We construct the regularized kernel $\mathcal{K}_R$ for $\boldsymbol{x} \in \Pi^d$ as follows:

$$\mathcal{K}_R(\boldsymbol{x}) := \begin{cases} T_I(\|\boldsymbol{x}\|) & \text{if } \|\boldsymbol{x}\| \le \frac{a}{n}, \\ T_{\tilde{B}}(\|\boldsymbol{x}\|) & \text{if } \frac{7}{16} < \|\boldsymbol{x}\| < \frac{1}{2}, \\ T_{\tilde{B}}(\frac{1}{2}) & \text{if } \|\boldsymbol{x}\| \ge \frac{1}{2}, \\ K(\|\boldsymbol{x}\|) & \text{otherwise.} \end{cases} \tag{4.3}$$

Here we choose $T_I(x)$ as in (3.10) and

$$T_{\tilde{B}}(x) := \sum_{j=0}^{p+\lfloor \frac{p-1}{2} \rfloor - 1} a_j^{\tilde{B}} \cos(8\pi j (x - 1/2)) \qquad (x \in (7/16, 1/2]) \,,$$

where the coefficients $a_j^{\tilde{B}}$ are defined by (3.12) and

$$T_{\tilde{B}}^{(2r)}\left(\frac{1}{2}\right) = 0 \quad \left(r = 1, \dots, \left\lfloor \frac{p-1}{2} \right\rfloor\right).$$

This leads to the following algorithm:

ALGORITHM 4.1.

Precomputation:

i) Computation of a_j^I $(j = 0, \dots, p-1)$ *and* $a_j^{\tilde{B}}$ $(j = 0, \dots, p - 1 + \lfloor p/2 \rfloor)$.

ii) Computation of $(b_{\boldsymbol{l}})_{\boldsymbol{l} \in I_n}$ *by* (4.2) *and* (4.3).

iii) Computation of $K_{NE}(\boldsymbol{y}_j - \boldsymbol{x}_k)$ *for all* $j = 1, \dots, M$ *and* $k \in I_{n,a}^{NE}(j)$, *where* $I_{n,a}^{NE}(j) := \{k \in \{1, \dots, N\} : \|\boldsymbol{y}_j - \boldsymbol{x}_k\| < \frac{a}{n}\}$. *See also Remark* 3.6.

1. *For* $\boldsymbol{l} \in I_n$ *compute*

$$a_{\boldsymbol{l}} := \sum_{k=1}^{N} \alpha_k \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{l} \boldsymbol{x}_k}$$

by d–variate NFFT$^{\mathrm{T}}$*(n), see Algorithm 1.2 in* [29].

2. *For* $\boldsymbol{l} \in I_n$ *compute the products*

$$d_{\boldsymbol{l}} := a_{\boldsymbol{l}} b_{\boldsymbol{l}}.$$

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

$$f_{RF}(\boldsymbol{y}_j) := \sum_{\boldsymbol{l} \in I_n} d_{\boldsymbol{l}} \, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{l} \boldsymbol{y}_j}$$

by d–variate NFFT(n), see Algorithm 1.1 in [29].

4. *For* $j = 1, \dots, M$ *compute the near field summations*

$$f_{NE}(\boldsymbol{y}_j) = \sum_{k \in I_{n,a}^{NE}(j)} \alpha_k K_{NE}(\boldsymbol{y}_j - \boldsymbol{x}_k).$$

5. *For* $j = 1, \dots, M$ *compute the near field corrections*

$$\tilde{f}(\boldsymbol{y}_j) = f_{NE}(\boldsymbol{y}_j) + f_{RF}(\boldsymbol{y}_j).$$

Our algorithm requires $\mathcal{O}(n^d \log n + m^d(M + N) + a^d M)$ arithmetic operations.

4.2. Numerical Results. We have implemented Algorithm 4.1 in two dimensions. In the following we present numerical results for the kernel $1/\|\boldsymbol{x}\|$. We have used the same general settings as in Subsection 3.3, i.e., the coefficients α_k were randomly chosen in $[0, 1]$, the knots $\boldsymbol{y}_j = \boldsymbol{x}_j \in \mathbb{R}^2$ $(j = 1, \dots, N = M)$ were randomly distributed in a disk with radius $\leq 1/4 - 1/32$ and the parameters for the NFFT are tensor products of Kaiser–Bessel functions as φ and $\sigma = 2$ as oversampling factor.
Table 4.1 presents the CPU time as well as the approximation error of Algorithm 4.1 for the computation of the sum (4.1) for increasing transform lengths N and fixed $m = a = p = 4$. The Columns 3 – 5 of Table 4.1 show the CPU times t_{slow} of the straightforward algorithm, t_{NE} of the near field correction including Remark 3.6 and t_{apr} of the whole Algorithm 4.1, respectively. Note that t_{NE} includes the computational time for the search of all points in the near field, which can be done in $\mathcal{O}(\log N)$. The last column contains the approximation error

$$E := \max_{j=1,\dots,N} \frac{|f(\boldsymbol{x}_j) - \tilde{f}(\boldsymbol{x}_j)|}{|f(\boldsymbol{x}_j)|}.$$

		Computational Time			Error
n	N	t_{slow}	t_{NE}	t_{apr}	E
32	1024	3.300e-01	6.600e-01	6.900e-01	5.246e-06
64	4096	5.050e+00	1.940e+00	2.060e+00	1.379e-05
128	16384	8.571e+01	6.850e+00	7.450e+00	1.158e-05
256	65536	1.534e+03	2.652e+01	3.094e+01	2.070e-05
512	262144	2.942e+04	1.052e+02	1.253e+02	1.595e-05
512	550000	1.326e+05	4.316e+02	4.575e+02	1.442e-05

TABLE 4.1
Computational time and approximation error for the kernel $K(\boldsymbol{x}) = 1/\|\boldsymbol{x}\|$ and $m = a = p = 4$.

5. Appendix: NFFT.

The main tools in our fast summation Algorithm 3.1 are the NFFT$^{\mathrm{T}}$ and the NFFT in Step 1 and 3 of the algorithm. Details concerning NFFT algorithms can be found for example in [29] and a software package can be found in [24]. In the following, we briefly describe the basic idea of these algorithms and cite error estimates.
We are interested in the fast computation of the sums

$$f(w_j) = \sum_{k=-n/2}^{n/2-1} f_k \, \mathrm{e}^{-2\pi \mathrm{i} k w_j} \qquad (j = -M/2, \dots, M/2-1) \tag{5.1}$$

and

$$h(j) = \sum_{k=-M/2}^{M/2-1} f_k \, \mathrm{e}^{-2\pi \mathrm{i} j w_k} \qquad (j = -n/2, \dots, n/2-1), \tag{5.2}$$

where $w_j \in [-1/2, 1/2)$. In Step 1 of Algorithm 3.1 we have to compute (5.2) with $M = N$ and $w_k = x_{k+M/2+1}$ and in Step 3 we have to calculate (5.1) with $w_j = -y_{j+M/2+1}$. Since (5.1) is in matrix–vector form the multiplication of $\boldsymbol{f} := (f_k)_{k=-n/2}^{n/2-1}$ with $\boldsymbol{A} := \left(\mathrm{e}^{-2\pi \mathrm{i} k w_j}\right)_{j=-M/2,k=-n/2}^{M/2-1,n/2-1}$ and (5.2) describes the multiplication with $\boldsymbol{A}^{\mathrm{T}}$, every algorithm for the fast computation of (5.1) implies a fast algorithm for the computation of (5.2), for details see e.g. [29]. In this sense we refer to the first algorithm as NFFT or NFFT(n) if we want to emphasize the transform length and to the second one as NFFT$^{\mathrm{T}}$. We restrict our attention to the computation of (5.1) by the NFFT(n).

Problem (5.1) is equivalent to the evaluation of the 1–periodic function

$$f(w) = \sum_{k=-n/2}^{n/2-1} f_k \, \mathrm{e}^{-2\pi \mathrm{i} k w} \tag{5.3}$$

at the knots w_j $(j = -M/2, \dots, M/2-1)$. This can be realized in an efficient way by approximating f by the sum of translates of a 1–periodic function $\tilde{\varphi}$ with good localization in time and frequency. Let $\varphi \in L^2(\mathbb{R})$ be given such that its 1–periodic version

$$\tilde{\varphi}(w) = \sum_{r \in \mathbb{Z}} \varphi(w + r)$$

has an absolute convergent Fourier series. We further assume that φ is even so that $\hat{\varphi}$ is real–valued. In the following, we choose $\tilde{\varphi}$ by the above periodization so that its Fourier coefficients are given by

$$\hat{\varphi}(k) = \int_{\mathbb{R}} \varphi(w) \mathrm{e}^{2\pi \mathrm{i} k w} \mathrm{d}w = \int_{-1/2}^{1/2} \tilde{\varphi}(w) \mathrm{e}^{2\pi \mathrm{i} k w} \, \mathrm{d}w \quad (k \in \mathbb{Z})$$

and

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

Let $\sigma > 1$ be a so–called *oversampling factor*. We want to approximate f by the sum f_1 of translates of $\tilde{\varphi}$

$$f_1(w) = \sum_{l=-\sigma n/2}^{\sigma n/2-1} g_l \, \tilde{\varphi}\left(w - \frac{l}{\sigma n}\right),$$

i.e., we ask for appropriate coefficients g_l. Switching to the Fourier series representation of the right–hand side, this can be rewritten as

$$\begin{aligned} f_1(w) &= \sum_{k \in \mathbb{Z}} \hat{g}_k \, \hat{\varphi}(k) \, \mathrm{e}^{-2\pi \mathrm{i} k w} \\ &= \sum_{k=-\sigma n/2}^{\sigma n/2-1} \hat{g}_k \, \hat{\varphi}(k) \, \mathrm{e}^{-2\pi \mathrm{i} k w} + \sum_{r \in \mathbb{Z} \setminus \{0\}} \sum_{k=-\sigma n/2}^{\sigma n/2-1} \hat{g}_k \, \hat{\varphi}(k + \sigma n r) \, \mathrm{e}^{-2\pi \mathrm{i}(k + \sigma n r) w}, \end{aligned}$$

where

$$\hat{g}_k := \sum_{l=-\sigma n/2}^{\sigma n/2-1} \hat{g}_l \, \mathrm{e}^{2\pi \mathrm{i} k l/(\sigma n)}. \tag{5.4}$$

Note that $\hat{g}_k = \hat{g}_{k+\sigma r n}$ for all $r \in \mathbb{Z}$. Comparing this equation with (5.3) and supposing that $\hat{\varphi}(k) \neq 0$ for $k = -n/2, \dots, n/2-1$ we propose to set

$$\hat{g}_k := \begin{cases} f_k / \hat{\varphi}(k) & k = -n/2, \dots, n/2-1, \\ 0 & k = -\sigma n/2, \dots, -n/2-1; n/2, \dots, \sigma n/2-1 \end{cases}. \tag{5.5}$$

By (5.4) this implies

$$g_l = \frac{1}{\sigma n} \sum_{k=-n/2}^{n/2-1} \hat{g}_k \, \mathrm{e}^{-2\pi \mathrm{i} k l/(\sigma n)}. \tag{5.6}$$

Then we obtain

$$f_1(w) = f(w) + \sum_{r\in\mathbb{Z}\setminus\{0\}} \sum_{k=-n/2}^{n/2-1} \hat{g}_k\, \hat{\varphi}(k+\sigma n r)\, \mathrm{e}^{-2\pi\mathrm{i}(k+\sigma n r)w}.$$

In general the approximation f_1 of f introduces a so–called *aliasing error* given by the double sum on the right–hand side. Obviously, the aliasing error becomes small if the Fourier coefficients $\hat{\varphi}(k)$ of $\tilde{\varphi}$ are small for $|k| \geq (\sigma n/2)(2 - 1/\sigma)$. In case that φ is bandlimited in $[-\sigma n + n/2, \sigma n - n/2]$, e.g., for suitably dilated Kaiser–Bessel window functions or for suitably dilated powers of the sinc function, do we have $f = f_1$.
We further suppose that φ is small (relative to $\varphi(0)$) outside some interval $I_m := [-m/(\sigma n), m/(\sigma n)]$ $(m \ll n)$ so that we can approximate it by a compactly supported function $\psi = \mathbf{1}_{I_m}\varphi$, where $\mathbf{1}_I$ denotes the characteristic function of the interval I. Let $\tilde{\psi}$ be the 1–periodic version of ψ. Then we obtain instead of f_1 the sparse sums

$$f_2(w_j) := \sum_{l=[w_j\sigma n]-m}^{[w_j\sigma n]+m} g_l\, \tilde{\psi}\left(w_j - \frac{l}{\sigma n}\right) \quad (j = -M/2, \ldots, M/2-1). \tag{5.7}$$

Here $[a]$ denotes the integer nearest to a. In general the approximation f_2 of f_1 introduces a *truncation error.* Only in case that φ is supported in I_m, e.g., for suitably dilated cardinal B–splines φ, do we have $\psi = \varphi$ and thus $f_2 = f_1$.

In summary, the whole algorithm for the fast approximate computation of (5.1) consists in the computation of the n multiplications (5.5), the computation of (5.6) by a (reduced) FFT(σn) and the sparse summations (5.7) and requires $\mathcal{O}(n + \sigma n \log(\sigma n) + (2m+1)M) = \mathcal{O}(n\log n + mM)$ arithmetic operations.

ALGORITHM 5.1. *(NFFT)*

Precomputation:

i) Computation of the Fourier coefficients $\hat{\varphi}(k)$ $(k = -n/2, \ldots, n/2-1)$

ii) Computation of the function values $\tilde{\psi}(w_j - \frac{l}{\sigma n})$ $(l = [w_j\sigma n] - m, \ldots, [w_j\sigma n] + m; j = -M/2, \ldots, M/2-1)$

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

$$\hat{g}_k := f_k/\hat{\varphi}(k).$$

2. For $l = -\sigma n/2, \ldots, \sigma n/2 - 1$ *compute by reduced FFT(σn)*

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

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

$$f_2(w_j) := \sum_{l=[w_j\sigma n]-m}^{[w_j\sigma n]+m} g_l\, \tilde{\psi}\left(w_j - \frac{l}{\sigma n}\right).$$

The corresponding "transposed" algorithm for the fast computation of (5.2) reads as follows:

ALGORITHM 5.2. *(NFFT$^{\mathrm{T}}$)*

Precomputation:

i) Computation of the Fourier coefficients $\hat{\varphi}(j)$ $(j = -n/2, \ldots, n/2-1)$

ii) Computation of the function values $\tilde{\psi}(w_k - \frac{l}{\sigma n})$ $(k \in J_m(l);\ l = -\sigma n/2, \ldots, \sigma n/2-1)$, *where* $J_m(l) := \{k \in \{-M/2, \ldots, M/2-1\} : l - m \leq \sigma n w_k \leq l + m\}$.

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

$$\tilde{g}_l := \sum_{j \in J_m(l)} f_j \, \tilde{\psi}\left(w_j - \frac{l}{\sigma n}\right).$$

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

$$\hat{g}_j := \frac{1}{\sigma n} \sum_{l=-\sigma n/2}^{\sigma n/2-1} \tilde{g}_l \, \mathrm{e}^{-2\pi \mathrm{i} k l/(\sigma n)}.$$

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

$$h(j) \approx \tilde{h}(j) := \hat{g}_j / \hat{\varphi}(j).$$

Setting $\boldsymbol{f} := (\delta(\cdot - k))_{-n/2}^{n/2-1}$, we note that the above algorithms use the approximation

$$\mathrm{e}^{-2\pi \mathrm{i} k x} \approx \frac{1}{\sigma n \, \hat{\varphi}(k)} \sum_l \tilde{\psi}\left(x - \frac{l}{\sigma n}\right) \mathrm{e}^{-2\pi \mathrm{i} k l/(\sigma n)}. \tag{5.8}$$

To keep the aliasing error and the truncation error small, several window functions φ with good localization in time and frequency were proposed, e.g. the (dilated) *Gaussian* [13, 31, 11]

$$\begin{aligned} \varphi(w) &= (\pi b)^{-1/2} \, \mathrm{e}^{-(\sigma n w)^2/b} \qquad \left(b := \frac{2\sigma}{2\sigma - 1} \frac{m}{\pi}\right), \\ \hat{\varphi}(k) &= \frac{1}{\sigma n} \, \mathrm{e}^{-(\frac{\pi k}{\sigma n})^2 b}, \end{aligned} \tag{5.9}$$

(dilated) *cardinal central B–splines* [6, 31]

$$\begin{aligned} \varphi(w) &= M_{2m}(\sigma n w), \\ \hat{\varphi}(k) &= \begin{cases} \frac{1}{\sigma n} & k = 0, \\ \frac{1}{\sigma n} \operatorname{sinc}^{2m}\left(\frac{k\pi}{\sigma n}\right) & \text{otherwise,} \end{cases} \end{aligned} \tag{5.10}$$

(dilated) *powers of the sinc function*

$$\begin{aligned} \varphi(v) &= \frac{n\,(2\sigma - 1)}{2m} \operatorname{sinc}^{2m}\left(\frac{\pi n v\,(2\sigma - 1)}{2m}\right), \\ \hat{\varphi}(k) &= M_{2m}\left(\frac{2mk}{(2\sigma - 1)\,n}\right) \end{aligned} \tag{5.11}$$

and (dilated) *Kaiser–Bessel functions* [22, 15]

$$\varphi(w) = \frac{1}{\pi} \begin{cases} \dfrac{\sinh(b\sqrt{m^2 - (\sigma n)^2 w^2})}{\sqrt{m^2 - (\sigma n)^2 w^2}} & \text{for } |w| \le \frac{m}{\sigma n} \qquad \left(b := \pi \left(2 - \frac{1}{\sigma}\right)\right), \\ \dfrac{\sin(b\sqrt{(\sigma n)^2 w^2 - m^2})}{\sqrt{(\sigma n)^2 w^2 - m^2}} & \text{otherwise,} \end{cases} \tag{5.12}$$

$$\hat{\varphi}(k) = \begin{cases} \frac{1}{\sigma n} I_0\left(m\sqrt{b^2 - (2\pi k/(\sigma n))^2}\right) & \text{for } k = -\sigma n\left(1 - \frac{1}{2\sigma}\right), \dots, \sigma n\left(1 - \frac{1}{2\sigma}\right), \\ 0 & \text{otherwise,} \end{cases}$$

where I_0 denotes the modified zero–order Bessel function. For these functions φ it was proved that

$$|f(w_j) - \tilde{f}(w_j)| \le C(\sigma, m) ||\boldsymbol{f}||_1$$

where

$$C(\sigma,m) := \begin{cases} 4\,\mathrm{e}^{-m\pi(1-1/(2\sigma-1))} & \text{for (5.9) [31],} \\ 4\left(\frac{1}{2\sigma-1}\right)^{2m} & \text{for (5.10) [31],} \\ \frac{1}{m-1}\left(\frac{2}{\sigma^{2m}} + \left(\frac{\sigma}{2\sigma-1}\right)^{2m}\right) & \text{for (5.11),} \\ 5\pi^2 m^{3/2}\sqrt[4]{1-\frac{1}{\sigma}}\,\mathrm{e}^{-m2\pi\sqrt{1-1/\sigma}} & \text{for (5.12).} \end{cases}$$

Thus, for fixed $\sigma > 1$, the approximation error introduced by the NFFT decays exponentially with the number m of summands in (5.7). On the other hand, the complexity of the NFFT increases with m. Beylkin et al. [6, 8] prefer B–splines and Rokhlin et al. [13] Gaussians. Further approaches based on scaling vectors [25], based on minimizing the Frobenius norm of certain error matrices [26] or based on min–max interpolation [14] were proposed.
By the results in [15, 26, 14] we prefer to apply the Algorithms 5.1 and 5.2 with Kaiser–Bessel functions. Using the tensor product approach our algorithms and the error estimates can be generalized to the multivariate setting.

REFERENCES

[1] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Waveletlike bases for the fast solution of second–kind integral equations. SIAM *J. Sci. Comput.*, 14:159 – 184, 1993.

[2] C. Anderson. A method of local corrections for computing the velocity field due to a distribution of vortex blobs. *J. Comput. Phys.*, 62:111 – 123, 1986.

[3] R. K. Beatson and L. Greengard. A short course on fast multipole methods. In M. Ainsworth, J. Levesley, W. A. Light, and M. Marletta, editors, *Wavelets, Multilevel Methods and Elliptic PDEs.* Clarendon Press, 1997.

[4] R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: methods for 2–dimensional polyharmonic splines. IMA *J. Numer. Anal.*, 17:343 – 372, 1997.

[5] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: I. *Comput. Math. Appl.*, 24:7 – 19, 1992.

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

[7] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure and Appl. Math.*, 44:141 – 183, 1991.

[8] G. Beylkin and R. Cramer. A multiresolution approach to regularization of singular operators and fast summation. SIAM *J. Sci. Comput.*, 24:81 – 117, 2002.

[9] R. H. Chan, F. R. Lin, and C. F. Chan. A fast solver for Fredholm equations of the second kind with weakly singular kernels. *J. Numer. Math.*, 10:13 – 36, 2002.

[10] R. H. Chan and M. K. Ng. Conjugate gradient methods of Toeplitz systems. SIAM *Rev.*, 38:427 – 482, 1996.

[11] A. J. W. Duijndam and M. A. Schonewille. Nonuniform fast Fourier transform. *Geophysics*, 64:539 – 551, 1999.

[12] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration and differentiation. SIAM *J. Numer. Anal.*, 33:1689 – 1711, 1996.

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

[14] J. A. Fessler and B. P. Sutton. Nonuniform fast Fourier transforms using min-max interpolation. IEEE *Trans. Signal Process.*, 2002. in press.

[15] K. Fourmont. Schnelle Fourier–Transformation bei nichtäquidistanten Gittern und tomographische Anwendungen. Dissertation, Universität Münster, 1999.

[16] S. A. Goreinov, E. E. Tyrtyshnikov, and E. E. Yeremin. Matrix–free iterative solution strategies for large dense systems. *Num. Lin. Alg. Appl.*, 4:273 – 294, 1997.

[17] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems.* MIT Press, Cambridge, 1988.

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

[19] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$–matrices, Part I: introduction to $\mathcal{H}$–matrices. *Computing*, 62:89 – 108, 1999.

[20] W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math.*, 54:463 – 491, 1989.

[21] A. Harten and I. Yad-Shalom. Fast multiresolution algorithms for matrix–vector multiplications. *SIAM J. Numer. Anal.*, 31:1191 – 1218, 1994.

[22] J. I. Jackson. Selection of a convolution function for Fourier inversion using gridding. IEEE *Trans. Med. Imag.*, 10:473 – 478, 1991.

[23] M. Konik, R. Schneider, and G. Steidl. Matrix sparsification by discrete multiscale methods. In C. K. Chui and L. L. Schumaker, editors, *Approximation Theory VIII: Approximation and Decomposition.* World Scientific Publishing, 1995.

[24] S. Kunis and D. Potts. NFFT, Softwarepackage, C subroutine library. http://www.math.uni-luebeck.de/potts/nfft, 2002.

[25] N. Nguyen and Q. H. Liu. The regular Fourier matrices and nonuniform fast Fourier transforms. SIAM *J. Sci. Comput.*, 21:283 – 293, 1999.

[26] A. Nieslony and G. Steidl. Approximate factorizations of Fourier matrices with nonequispaced knots. *Linear Algebra Appl.*, to appear.

[27] D. Potts and G. Steidl. Optimal trigonometric preconditioners for nonsymmetric Toeplitz systems. *Linear Algebra Appl.*, 281:265 – 292, 1998.

[28] D. Potts, G. Steidl, and M. Tasche. Trigonometric preconditioners for block Toeplitz systems. In G. Nürnberger, J. W. Schmidt, and G. Walz, editors, *Multivariate Approximation and Splines*, pages 219 – 234, Birkhäuser, Basel, 1997.

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

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

[31] G. Steidl. A note on fast Fourier transforms for nonequispaced grids. *Adv. Comput. Math.*, 9:337 – 353, 1998.

[32] X. Sun and N. P. Pitsianis. A matrix version of the fast multipole method. SIAM *Rev.*, 43:289–300, 2001.

[33] E. E. Tyrtyshnikov. Mosaic–skeleton approximations. *Calcolo*, 33:47 – 57, 1996.

[34] N. Yarvin and V. Rokhlin. An improved fast multipole algorithm for potential fields on the line. SIAM *J. Numer. Anal.*, 36:629 – 666, 1999.