# Fast Convolution with Radial Kernels at Nonequispaced Knots

Daniel Potts* Gabriele Steidl† Arthur Nieslony‡

**Abstract**

We develop a new algorithm for the fast evaluation of linear combinations of radial functions $f(\boldsymbol{y}_j) := \sum_{k=1}^{N} \alpha_k K(||\boldsymbol{y}_j - \boldsymbol{x}_k||)$ $(j = 1, \ldots, M)$ for $\boldsymbol{x}_k, \boldsymbol{y}_j \in \mathbb{R}^2$ based on the recently developed fast Fourier transform at nonequispaced knots. For smooth kernels, e.g. the Gaussian, our algorithm requires $\mathcal{O}(N + M)$ arithmetic operations. In case of singular kernels an additional regularization procedure must be incorporated and the algorithm has the arithmetic complexity $\mathcal{O}(N \log \sqrt{N} + M)$ or $\mathcal{O}(M \log \sqrt{M} + N)$ if either the points $\boldsymbol{y}_j$ or the points $\boldsymbol{x}_k$ are "reasonably uniformly distributed". We prove error estimates to obtain clues about the choice of the involved parameters and present numerical examples for various singular and smooth kernels in two dimensions.

**2000 AMS Subject Classification**: 65T40, 65T50, 65F30
**Keywords**: fast discrete summation, radial basis functions, fast Fourier transform at nonequispaced knots, fast Gauss transform, fast multipole method

## 1 Introduction

In this paper, we develop a new algorithm for the fast computation of sums of the form

$$f(\boldsymbol{y}_j) := \sum_{k=1}^{N} \alpha_k \mathcal{K}(\boldsymbol{y}_j - \boldsymbol{x}_k) \qquad (j = 1, \ldots, M) \tag{1.1}$$

at knots $\boldsymbol{y}_j, \boldsymbol{x}_k \in \mathbb{R}^2$. We will focus our attention on real–valued radially symmetric functions $\mathcal{K}$, i.e. $\mathcal{K}(\boldsymbol{x}) = K(\|\boldsymbol{x}\|)$, where $\|\cdot\| = ||\cdot||_2$ denotes the Euclidean norm in $\mathbb{R}^2$ and where $K$ is a univariate function.

Frequently applied kernels are the singularity function $K(||\boldsymbol{x}||) = \log ||\boldsymbol{x}||$ of the bivariate Laplacian which appears for example in two–dimensional particle simulation [16] and the thin–plate spline $K(||\boldsymbol{x}||) = ||\boldsymbol{x}||^2 \log ||\boldsymbol{x}||$ [8] which is often used for the scattered data approximation of surfaces. These kernels or their derivatives have singularities at zero. The best known algorithm for the efficient computation of (1.1) is the fast multipole method (FMM). The FMM was originally developed in the context of particle simulation by L. Greengard and V. Rokhlin [16, 18] and requires only $\mathcal{O}(N + M)$ arithmetic operations, where the $\mathcal{O}$ constant depends on the desired 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 $\alpha$

*University of Lübeck, Institute of Mathematics, D–23560 Lübeck, (potts@math.uni-luebeck.de).
†University of Mannheim, Faculty of Mathematics and Computer Science, D–68131 Mannheim, (steidl@math.uni-mannheim.de).
‡University of Mannheim, Faculty of Mathematics and Computer Science, D–68131 Mannheim.

and $\beta$ such that $|F(N,M)/(\alpha\, g_1(N)+\beta\, g_2(M))|\le c$. The prize one has to pay for the fast performance of the FMM consists in an expensive adaptation to new kernels. In particular, R. Beatson et al. have adapted the FMM to various kernels appearing in the scattered data approximation of functions. Meanwhile several improved and modified FMM versions were suggested, see, e.g. [30, 28, 3, 7].
Another frequently used smooth kernel which arises in the context of diffusion [17], image processing [11], fluid dynamics and finance [6], is the Gaussian $K(||\boldsymbol{x}||) = \mathrm{e}^{-\sigma||\boldsymbol{x}||^2}$. The corresponding transform (1.1) is known as discrete Gauss transform. Fast Gauss transforms were introduced in [19, 20, 2].
However, for equispaced knots, (1.1) is simply a discrete convolution and its fast computation is mainly realized by fast Fourier methods exploiting the basic property $\mathrm{e}^{2\pi\mathrm{i}(\boldsymbol{y}-\boldsymbol{x})} = \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{y}}\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{x}}$. Following these lines, we propose to compute the "convolution at nonequispaced knots" (1.1) by Fourier methods too, more precisely by fast Fourier transforms at nonequispaced knots (NFFT). The corresponding techniques were recently developed by A. Dutt and V. Rokhlin [10] and were further modified and improved by various groups, in particular by the authors [4, 27, 9, 13, 26]. Briefly speaking, for arbitrary points $\boldsymbol{w}_j \in [-1/2, 1/2)^2$ and the index set $I_n := \{\boldsymbol{l} \in \mathbb{Z}^2 : -\frac{n}{2} \le \boldsymbol{l} < \frac{n}{2}\}$ with componentwise inequalities, the NFFT$(n)$ computes sums of the form

$$f(\boldsymbol{w}_j) = \sum_{\boldsymbol{k}\in I_n} f_{\boldsymbol{k}}\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{w}_j} \qquad (j=-M/2,\dots,M/2-1) \tag{1.2}$$

and the NFFT$^{\mathrm{T}}(n)$ sums of the form

$$h(\boldsymbol{j}) = \sum_{k=-M/2}^{M/2-1} f_k\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{j}\boldsymbol{w}_k} \qquad (\boldsymbol{j}\in I_n) \tag{1.3}$$

with $\mathcal{O}((\rho n)^2\log(\rho n) + m^2 M)$ arithmetic operations, respectively. Here $\rho$ is an oversampling factor and $m$ a cut–off parameter. Both parameters have to be chosen in accordance with the desired accuracy of the NFFT computations. In general the approximation error introduced by the NFFT decreases exponentially in $m$, where the basis of the exponent depends on $\rho$. For a summary of various error estimates see the appendix in [24].
Concerning the notation NFFT and NFFT$^{\mathrm{T}}$, note that the first Fourier sum (1.2) is the multiplication of $\boldsymbol{f} := (f_{\boldsymbol{k}})_{\boldsymbol{k}\in I_n}$ with

$$\boldsymbol{A}_M := \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{w}_j}\right)_{j=-M/2,\dots,M/2-1;\boldsymbol{k}\in I_n},$$

while the second sum (1.3) describes the multiplication with $\boldsymbol{A}_M^{\mathrm{T}}$. Meanwhile there exist several free NFFT software packages for the computation of fast Fourier transforms at nonequispaced knots, e.g. the C–package of the authors [23] which uses the FFTW [14] or the MATLAB–package [12].
For smooth kernels $\mathcal{K}$ , e.g. the Gaussian, our new summation algorithm requires $\mathcal{O}(N+M)$ arithmetic operations for arbitrary distributed points $\boldsymbol{x}_k$ and $\boldsymbol{y}_j$. For kernels with singular derivatives at one point, we have to introduce an additional regularization procedure and a "near field correction". If either the knots $\boldsymbol{x}_k$ or $\boldsymbol{y}_j$ are "sufficiently uniformly distributed", a notation which we will clarify later, then our algorithm requires $\mathcal{O}(N\log\sqrt{N}+M)$, respectively $\mathcal{O}(M\log\sqrt{M}+N)$ arithmetic operations, where the $\mathcal{O}$ constant depends on the desired accuracy of the computation.

In summary, the advantage of our algorithm, which has nearly the same arithmetic complexity as the FMM algorithm consists in its simple structure which resembles ideas from the fast convolution by FFTs and allows an immediate incorporation of various kernels. Moreover one can build up on existing software packages. A drawback of our algorithm in its present form is that for singular kernels, the arithmetic complexity can be only assured if one of the point sets $\{\boldsymbol{x}_k\}$ or $\{\boldsymbol{y}_j\}$ is "reasonably uniformly distributed".
In this paper, we will focus our attention on the summation (1.1) with explicitly known kernels. In the numerical solution of integral equations (or of partial differential equations by recasting them as integral equations) one computational task consists in the evaluation of large sums with special structured but not explicitly known kernels. Here the method of panel clustering, see, e.g. [22], or more general the theory of $H$–matrices, see, e.g. [21] and mosaic–skeleton approximations, see, e.g. [29, 15] may be applied.
The remainder of this paper is organized as follows: in Section 2 we present our NFFT based summation algorithm. In particular, we present a simple regularization procedure for singular kernels which involves only solutions of small systems of linear equations with right–hand sides depending on the kernel. This makes it easy to adapt our algorithm to various kernels. Error estimates are proved in Section 3. Section 4 provides numerical examples for various singular and smooth kernels.

# 2 Fast Summation Algorithms

We are interested in the fast evaluation of sums

$$f(\boldsymbol{x}) := \sum_{k=1}^{N} \alpha_k \mathcal{K}(\boldsymbol{x} - \boldsymbol{x}_k) = \sum_{k=1}^{N} \alpha_k K(\|\boldsymbol{x} - \boldsymbol{x}_k\|),$$

at $M$ different knots $\boldsymbol{y}_j$ $(j = 1, \ldots, M)$. The kernel function $K$ is in general a non-periodic function, while the use of Fourier methods requires to replace $K$ by a periodic version. Without loss of generality we may assume that the knots are scaled, such that $\|\boldsymbol{x}_k\|, \|\boldsymbol{y}_j\| < \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2}$ and consequently $\|\boldsymbol{y}_j - \boldsymbol{x}_k\| < \frac{1}{2} - \varepsilon_{\mathrm{B}}$. The parameter $\varepsilon_{\mathrm{B}} > 0$, which we specify later, guarantees that $K$ has to be evaluated only at points in the interval $[-\frac{1}{2} + \varepsilon_{\mathrm{B}}, \frac{1}{2} - \varepsilon_{\mathrm{B}}]$. This simplifies the later consideration of a 1-periodic version of $K$.

First we deal with kernels $\mathcal{K}$ which are $C^\infty$ except for the origin, where $\mathcal{K}$ or its derivatives may have singularities. Examples of such kernels are

$$\|\boldsymbol{x}\|^2 \log \|\boldsymbol{x}\|, \log \|\boldsymbol{x}\| \quad \text{and} \quad \frac{1}{\|\boldsymbol{x}\|^\beta} \quad (\beta \in \mathbb{N}). \tag{2.1}$$

Beyond a special treatment of $\mathcal{K}$ near the boundary we have to be concerned about the singularity of $\mathcal{K}$ at the origin. We regularize $\mathcal{K}$ near $\mathbf{0}$ and near the boundary of $\|\boldsymbol{x}\| = 1/2$ as follows

$$\mathcal{K}_{\mathrm{R}}(\boldsymbol{x}) := \begin{cases} T_{\mathrm{I}}(\|\boldsymbol{x}\|) & \text{if} \quad \|\boldsymbol{x}\| \le \varepsilon_{\mathrm{I}}, \\ T_{\mathrm{B}}(\|\boldsymbol{x}\|) & \text{if} \quad \frac{1}{2} - \varepsilon_{\mathrm{B}} < \|\boldsymbol{x}\| < \frac{1}{2}, \\ T_{\mathrm{B}}(\frac{1}{2}) & \text{if} \quad \frac{1}{2} \le \|\boldsymbol{x}\|, \\ K(\|\boldsymbol{x}\|) & \text{otherwise}, \end{cases} \tag{2.2}$$

where $0 < \varepsilon_{\mathrm{I}} < \frac{1}{2} - \varepsilon_{\mathrm{B}} < \frac{1}{2}$. The functions $T_{\mathrm{I}}$ and $T_{\mathrm{B}}$ will be chosen such that $\mathcal{K}_{\mathrm{R}}$ is in the Sobolev space $H^p(\mathbb{T}^2)$ for an appropriate parameter $p > 0$. Several regularizations of $\mathcal{K}$ are possible, e.g. by algebraic polynomials, splines or trigonometric polynomials. Here we focus our attention on trigonometric polynomials $T_{\mathrm{I}}$ and $T_{\mathrm{B}}$. We set

$$T_{\mathrm{I}}(x) := \sum_{j=0}^{p-1} a_j^{\mathrm{I}} \cos \frac{\pi j}{2\varepsilon_{\mathrm{I}}} x, \tag{2.3}$$

where the coefficients $a_j^{\mathrm{I}}$ are determined by

$$T_{\mathrm{I}}^{(r)}(\varepsilon_{\mathrm{I}}) = K^{(r)}(\varepsilon_{\mathrm{I}}) \qquad (r = 0, \ldots, p-1), \tag{2.4}$$

and

$$T_{\mathrm{B}}(x) := \sum_{j=0}^{p_{\mathrm{B}}-1} a_j^{\mathrm{B}} \cos\left(\frac{\pi j}{2\varepsilon_{\mathrm{B}}}\left(x - \frac{1}{2}\right)\right) \qquad \left(x \in \left(\frac{1}{2} - \varepsilon_{\mathrm{B}}, \frac{1}{2}\right]\right). \tag{2.5}$$

where $p_{\mathrm{B}} := p + \lfloor (p-1)/2 \rfloor$ and the coefficients $a_j^{\mathrm{B}}$ are determined by

$$\begin{aligned} T_{\mathrm{B}}^{(r)}\left(\frac{1}{2} - \varepsilon_{\mathrm{B}}\right) &= K^{(r)}\left(\frac{1}{2} - \varepsilon_{\mathrm{B}}\right) \qquad (r = 0, \ldots, p-1), \\ T_{\mathrm{B}}^{(2r)}\left(\frac{1}{2}\right) &= 0 \qquad\qquad (r = 1, \ldots, \lfloor (p-1)/2 \rfloor). \end{aligned} \tag{2.6}$$

Here $\lfloor x \rfloor$ denotes the largest integer $\leq x$. By definition (2.5) we have that $T_{\mathrm{B}}^{(2r+1)}\left(\frac{1}{2}\right) = 0$ for all $r \in \mathbb{N}_0$.

By the ansatz (2.3) and condition (2.4) the coefficients of $\boldsymbol{a}_{\mathrm{even}}^{\mathrm{I}} := (a_{2j}^{\mathrm{I}})_{j=1}^{\lfloor \frac{p-1}{2} \rfloor}$ and $\boldsymbol{a}_{\mathrm{odd}}^{\mathrm{I}} := (a_{2j+1}^{\mathrm{I}})_{j=0}^{\lfloor \frac{p-2}{2} \rfloor}$ are the solutions of the following two linear systems of equations

$$\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}} \, \boldsymbol{a}_{\mathrm{even}}^{\mathrm{I}} = \left( \left( \frac{2}{\pi \varepsilon_{\mathrm{I}}} \right)^{2r} K^{(2r)}(\varepsilon_{\mathrm{I}}) \right)_{r=1}^{\lfloor \frac{p-1}{2} \rfloor}$$

and

$$\boldsymbol{V}_{\mathrm{odd}}^{\mathrm{I}} \, \boldsymbol{a}_{\mathrm{odd}}^{\mathrm{I}} = \left( -\left( \frac{2}{\pi \varepsilon_{\mathrm{I}}} \right)^{2r+1} K^{(2r+1)}(\varepsilon_{\mathrm{I}}) \right)_{r=0}^{\lfloor \frac{p-2}{2} \rfloor},$$

where $\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}}$ and $\boldsymbol{V}_{\mathrm{odd}}^{I}$ are the mosaic Vandermonde matrices

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

In our algorithm we will only need small values $p \leq 16$ of $p$ for which these ill–conditioned small linear systems can be solved without problems.

In a similar way, we see by (2.5) and (2.6), that the coefficients of $\boldsymbol{a}_{\mathrm{even}}^{\mathrm{B}} := (a_{2j}^{\mathrm{B}})_{j=1}^{\lfloor \frac{p_{\mathrm{B}}-1}{2} \rfloor}$ and $\boldsymbol{a}_{\mathrm{odd}}^{\mathrm{B}} := (a_{2j+1}^{\mathrm{B}})_{j=0}^{\lfloor \frac{p_{\mathrm{B}}-2}{2} \rfloor}$ are the solution of the linear system

$$\begin{pmatrix} \boldsymbol{V}_{\mathrm{even}}^{\mathrm{B}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V}_{\mathrm{odd}}^{\mathrm{B}} \\ \boldsymbol{V}_{\mathrm{e}}^{\mathrm{B}} & \boldsymbol{V}_{\mathrm{o}}^{\mathrm{B}} \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_{\mathrm{even}}^{\mathrm{B}} \\ \boldsymbol{a}_{\mathrm{odd}}^{\mathrm{B}} \end{pmatrix} = \begin{pmatrix} \left(K^{(2r)}\left(\frac{1}{2} - \varepsilon_{\mathrm{B}}\right)\right)_{r=1}^{\lfloor \frac{p-1}{2} \rfloor} \\ \left(K^{(2r+1)}\left(\frac{1}{2} - \varepsilon_{\mathrm{B}}\right)\right)_{r=0}^{\lfloor \frac{p-2}{2} \rfloor} \\ \boldsymbol{0} \end{pmatrix},$$

where

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

and

$$\boldsymbol{V}_{\text{e}}^{\text{B}} := \left((-1)^{r}(2j)^{(2r)}\right)_{r=1,j=1}^{\lfloor\frac{p-1}{2}\rfloor,\lfloor\frac{p_{\text{B}}-1}{2}\rfloor}, \quad \boldsymbol{V}_{\text{o}}^{\text{B}} := \left((-1)^{r}(2j+1)^{(2r)}\right)_{r=1,j=0}^{\lfloor\frac{p-1}{2}\rfloor,\lfloor\frac{p_{\text{B}}-2}{2}\rfloor}.$$

Next we approximate the smooth function $\mathcal{K}_{\text{R}}$ by the Fourier series

$$\mathcal{K}_{\text{RF}}(\boldsymbol{x}) := \sum_{\boldsymbol{l}\in I_n} b_{\boldsymbol{l}}\, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}}, \tag{2.7}$$

where

$$b_{\boldsymbol{l}} := \frac{1}{n^2}\sum_{\boldsymbol{j}\in I_n} \mathcal{K}_{\text{R}}(\boldsymbol{j}/n)\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{j}\boldsymbol{l}/n} \quad (\boldsymbol{l}\in I_n)\,. \tag{2.8}$$

Then our original kernel splits as

$$\mathcal{K} = (\mathcal{K}-\mathcal{K}_{\text{R}}) + (\mathcal{K}_{\text{R}}-\mathcal{K}_{\text{RF}}) + \mathcal{K}_{\text{RF}} = \mathcal{K}_{\text{NE}} + \mathcal{K}_{\text{ER}} + \mathcal{K}_{\text{RF}}, \tag{2.9}$$

where $\mathcal{K}_{\text{NE}} := \mathcal{K}-\mathcal{K}_{\text{R}}$ and $\mathcal{K}_{\text{ER}} := \mathcal{K}_{\text{R}}-\mathcal{K}_{\text{RF}}$. Since $\mathcal{K}_{\text{R}}$ is smooth, its Fourier approximation $\mathcal{K}_{\text{RF}}$ should introduce only a small error $\mathcal{K}_{\text{ER}}$. We neglect this error and approximate $f$ by

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

where

$$f_{\text{NE}}(\boldsymbol{x}) := \sum_{k=1}^{N} \alpha_k \mathcal{K}_{\text{NE}}(\boldsymbol{x}-\boldsymbol{x}_k)\,, \tag{2.10}$$

$$f_{\text{RF}}(\boldsymbol{x}) := \sum_{k=1}^{N} \alpha_k \mathcal{K}_{\text{RF}}(\boldsymbol{x}-\boldsymbol{x}_k)\,. \tag{2.11}$$

Instead of $f$ we evaluate $\tilde{f}$ at the points $\boldsymbol{y}_j$. If either the points $\boldsymbol{x}_k$ or the points $\boldsymbol{y}_j$ are "sufficiently uniformly distributed" this can indeed be done in a fast way, namely:

- *Near field computation* (2.10)

  By definition (2.2), the function $\mathcal{K}_{\text{NE}}$ has a small support contained in the ball of radius $\varepsilon_{\text{I}}$ around $\mathbf{0}$ and in the neighborhood of the boundary. The boundary is not interesting for us since $\|\boldsymbol{x}_k-\boldsymbol{y}_j\| \le 1/2-\varepsilon_{\text{B}}$. To achieve the desired complexity of our algorithm we suppose that either the $N$ points $\boldsymbol{x}_k$ or the $M$ points $\boldsymbol{y}_j$ are "sufficiently uniformly distributed", i.e. we suppose that there exists a small constant $\nu\in\mathbb{N}$ such that every ball of radius $\varepsilon_{\text{I}}$ contains at most $\nu$ of the points $\boldsymbol{x}_k$ or of the points $\boldsymbol{y}_j$, respectively. This implies that $\varepsilon_I$ depends linearly on $1/\sqrt{N}$, respectively $1/\sqrt{M}$. In the following, we restrict our attention to the case

  $$\varepsilon_{\text{I}} \sim \sqrt{\frac{\nu}{N}}. \tag{2.12}$$

  Then the sum (2.10) contains for fixed $\boldsymbol{y}_j$ not more than $\nu$ summands so that its evaluation at $M$ knots requires only $\mathcal{O}(\nu M)$ arithmetic operations.

We remark that also $\mathcal{O}(\log \sqrt{M})$, respectively $\mathcal{O}(\log \sqrt{N})$ points instead of $\mathcal{O}(1)$ points per ball will keep a complexity of $\mathcal{O}(M \log \sqrt{M} + N \log \sqrt{N})$ of our whole algorithm. This is in particular the case for inhomogeneously distributed Legendre knots which play an important role in computations on the 2–sphere [5].

- *Far field summation* (2.11) *by* $\mathrm{NFFT}^{\mathrm{T}}/\mathrm{NFFT}$

  Substituting (2.7) for $\mathcal{K}_{\mathrm{RF}}$ we obtain

$$f_{\mathrm{RF}}(\boldsymbol{y}_j) = \sum_{k=1}^{N} \alpha_k \sum_{\boldsymbol{l} \in I_n} b_{\boldsymbol{l}}\, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{l}(\boldsymbol{y}_j - \boldsymbol{x}_k)} = \sum_{\boldsymbol{l} \in I_n} b_{\boldsymbol{l}} \left( \sum_{k=1}^{N} \alpha_k\, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{l} \boldsymbol{x}_k} \right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{l} \boldsymbol{y}_j}$$

  The expression in the inner brackets can be computed by a bivariate $\mathrm{NFFT}^{\mathrm{T}}(n)$. This is followed by $n^2$ multiplications with $b_{\boldsymbol{l}}$ and completed by a bivariate $\mathrm{NFFT}(n)$ to compute the outer sum with the complex exponentials. If $m$ is the cut-off parameter and $\rho = 2$ the oversampling factor of the $\mathrm{NFFT}^{\mathrm{T}}/\mathrm{NFFT}$, then the proposed evaluation of $f_{\mathrm{RF}}$ at the points $\boldsymbol{y}_j$ $(j = 1, \ldots, M)$ requires $\mathcal{O}(m^2(N+M) + (\rho n)^2 \log(\rho n))$ arithmetic operations. The relation between $M, N$ and $n$ is determined by the approximation error of the algorithm and will be specified in Section 3.1.

In summary we obtain the following algorithm:

**Algorithm 2.1**

*Precomputation:*

*i) Computation of* $a_j^{\mathrm{I}}$ $(j = 0, \ldots, p-1)$ *and* $a_j^{\mathrm{B}}$ $(j = 0, \ldots, p_{\mathrm{B}} - 1)$.

*ii) Computation of* $(b_{\boldsymbol{l}})_{\boldsymbol{l} \in I_n}$ *by* (2.8) *and* (2.2).

*iii) Computation of* $K_{\mathrm{NE}}(\boldsymbol{y}_j - \boldsymbol{x}_k)$ *for all* $(j = 1, \ldots, M)$ *and* $k \in I_{n,a}^{\mathrm{NE}}(j)$, *where* $I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j) := \{k \in \{1, \ldots, N\} : \|\boldsymbol{y}_j - \boldsymbol{x}_k\| < \varepsilon_{\mathrm{I}}\}$. *See also Remark 4.3.*

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 the bivariate* $\mathit{NFFT}^{\mathrm{T}}(n)$.

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

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

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

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

   *by the bivariate NFFT(n).*

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

$$f_{\mathrm{NE}}(\boldsymbol{y}_j) = \sum_{k \in I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)} \alpha_k \mathcal{K}_{\mathrm{NE}}(\boldsymbol{y}_j - \boldsymbol{x}_k).$$

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

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

**Remark 2.2** (**Matrix version of Algorithm 2.1**)

Let $\boldsymbol{K} := \left(\mathcal{K}(\boldsymbol{y}_j - \boldsymbol{x}_k)\right)_{j=1,k=1}^{M,N}$ and $\boldsymbol{\alpha} := (\alpha_k)_{k=1}^N$. Then Algorithm 2.1 reads in matrix–vector notation

$$\boldsymbol{K}\boldsymbol{\alpha} \approx \left(\bar{\boldsymbol{A}}_M \boldsymbol{D}_{\mathcal{K}_{\mathrm{R}}} \boldsymbol{A}_N^{\mathrm{T}} + \boldsymbol{K}_{\mathrm{NE}}\right)\boldsymbol{\alpha},$$

where $\boldsymbol{D}_{\mathcal{K}_{\mathrm{R}}} := \operatorname{diag}(b_{\boldsymbol{l}})_{\boldsymbol{l}\in I_n}$, $\boldsymbol{A}_N := \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}_k}\right)_{k=1,\ldots,N;\boldsymbol{l}\in I_n}$, $\boldsymbol{A}_M := \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{y}_j}\right)_{j=1,\ldots,M;\boldsymbol{l}\in I_n}$ and $\boldsymbol{K}_{\mathrm{NE}} := \left(\mathcal{K}_{\mathrm{NE}}(\boldsymbol{y}_j - \boldsymbol{x}_k)\right)_{j=1,k=1}^{M,N}$. Using the matrix–vector notation of our NFFT/NFFT$^{\mathrm{T}}$

$$\boldsymbol{A}_M \approx \boldsymbol{B}_M \boldsymbol{F}_{\rho n} \boldsymbol{D}_n$$

with a diagonal matrix $\boldsymbol{D}_n$, a sparse matrix $\boldsymbol{B}_M$ having at most $(2m+1)^2$ nonzero entries in each row and column and the "equispaced" Fourier matrix $\boldsymbol{F}_{\rho n} := \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{l}/n}\right)_{\boldsymbol{k}\in I_n;\boldsymbol{l}\in I_{\rho n}}$, this can be rewritten as

$$\boldsymbol{K}\boldsymbol{\alpha} \approx (\bar{\boldsymbol{B}}_M \boldsymbol{T} \boldsymbol{B}_N^{\mathrm{T}} + \boldsymbol{K}_{\mathrm{NE}})\boldsymbol{\alpha},$$

where $\boldsymbol{T} := \boldsymbol{F}_{\rho n}\boldsymbol{D}_n\boldsymbol{D}_{\mathcal{K}_{\mathrm{R}}}\boldsymbol{D}_n\boldsymbol{F}_{\rho n}^{\mathrm{T}}$ is a block–Toeplitz–Toeplitz–block matrix. Note that we can avoid the complex arithmetic introduced by the FFT by using fast Toeplitz matrix times vector multiplications based on trigonometric transforms (see Algorithm 3 of [25]). □

Next we are interested in smooth kernels, e.g. in the Gaussian $\mathrm{e}^{-\sigma x^2}$. Here no regularization at the neighborhood of $\mathbf{0}$ is necessary and our computation does not require a "near field correction". If the kernel $K$ is very small at the boundary, e.g. for large values $\sigma$ of the Gaussian, we also do not need a regularization at the boundary. In this case we set $\mathcal{K}_{\mathrm{R}} := \mathcal{K}$. Otherwise we use

$$\mathcal{K}_{\mathrm{R}}(\boldsymbol{x}) = \begin{cases} T_B(\|\boldsymbol{x}\|) & \text{if } \frac{1}{2} - \varepsilon_{\mathrm{B}} < \|\boldsymbol{x}\| < \frac{1}{2}, \\ T_B(\frac{1}{2}) & \text{if } \frac{1}{2} \le \|\boldsymbol{x}\|, \\ K(\|\boldsymbol{x}\|) & \text{otherwise.} \end{cases} \tag{2.13}$$

Then Algorithm 2.1 simplifies to its first three steps. Moreover we will see in Section 3.2 that the lack of the "near field correction" implies that the size $n$ of the NFFT/ NFFT$^{\mathrm{T}}$ does not depend on the number $N$ or $M$ of the knots. Thus the Algorithm 2.1 (Step 1 – 3) requires for nonsingular kernels only $\mathcal{O}((\rho n)^2 \log(\rho n) + m^2(M+N)) = \mathcal{O}(M+N)$ arithmetic operations. Applied to the Gaussian we call Algorithm 2.1 (Step 1 – 3) *fast Gauss transform*.

# 3 Error estimates

Beyond the well–known errors appearing in the NFFT computations, our algorithm produces the error

$$|f(\boldsymbol{y}_j) - \tilde{f}(\boldsymbol{y}_j)| = \left|\sum_{k=1}^N \alpha_k \mathcal{K}_{\mathrm{ER}}(\boldsymbol{y}_j - \boldsymbol{x}_k)\right| \qquad (j = 1, \ldots, M).$$

By Hölders inequality these errors can be estimated by

$$|f(\boldsymbol{y}_j) - \tilde{f}(\boldsymbol{y}_j)| \le \|\boldsymbol{\alpha}\|_p \left\|\left(\mathcal{K}_{\mathrm{ER}}(\boldsymbol{y}_j - \boldsymbol{x}_k)\right)_{k=1}^N\right\|_q, \tag{3.1}$$

where $1/p + 1/q = 1$ $(1 \le p, q \le \infty)$ and

$$\|\boldsymbol{\alpha}\|_p := \begin{cases} \left(\sum\limits_{k=1}^{N} |\alpha_k|^p\right)^{1/p} & 1 \le p < \infty, \\ \max\limits_{k=1,\ldots,N} |\alpha_k| & p = \infty\,. \end{cases}$$

In the following, we restrict our attention to the case $p = 1$, i.e. we use $\|\boldsymbol{\alpha}\|_1$ in (3.1), so that it remains to estimate $||\mathcal{K}_{\mathrm{ER}}||_\infty := \max\limits_{\|\boldsymbol{x}\| \le \frac{1}{2} - \varepsilon_{\mathrm{B}}} |\mathcal{K}_{\mathrm{ER}}(\boldsymbol{x})|$. This kind of estimate is also given for FMMs. Note that (3.1) is sharp if $\alpha_k = 0$ for all indices $k$ except for this one which corresponds to the largest value $|\mathcal{K}_{\mathrm{ER}}(\boldsymbol{y}_j - \boldsymbol{x}_k)|$.
Our error estimates are based on the following well–known aliasing result:

**Lemma 3.1** *(Aliasing error) Let $f \in L_2(\mathbb{T}^2)$ be a function with absolutely convergent Fourier series*

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^2} c_{\boldsymbol{k}}(f)\, \mathrm{e}^{2\pi i \boldsymbol{k}\boldsymbol{x}}\,, \quad c_{\boldsymbol{k}}(f) = \int_{\mathbb{T}^2} f(\boldsymbol{x})\, \mathrm{e}^{-2\pi i \boldsymbol{k}\boldsymbol{x}}\, d\boldsymbol{x}$$

*and let an approximation of $f$ be given by*

$$f_F(\boldsymbol{x}) := \sum_{\boldsymbol{l} \in I_n} b_{\boldsymbol{l}}\, \mathrm{e}^{2\pi i \boldsymbol{l}\boldsymbol{x}}\,, \quad b_{\boldsymbol{l}} = \frac{1}{n^2} \sum_{\boldsymbol{j} \in I_n} f\left(\frac{\boldsymbol{j}}{n}\right) \mathrm{e}^{-2\pi i \boldsymbol{j}\boldsymbol{l}/n}\,.$$

*Then*

$$b_{\boldsymbol{l}} = c_{\boldsymbol{l}}(f) + \sum_{\boldsymbol{r} \in \mathbb{Z}^2 \setminus (0,0)} |c_{\boldsymbol{l}+n\boldsymbol{r}}(f)|$$

*and the approximation error can be estimated for all $\boldsymbol{x} \in \mathbb{T}^2$ by*

$$|f(\boldsymbol{x}) - f_F(\boldsymbol{x})| \le 2 \sum_{\boldsymbol{r} \in \mathbb{Z}^2 \setminus (0,0)} \sum_{\boldsymbol{l} \in I_n} |c_{\boldsymbol{l}+n\boldsymbol{r}}(f)|.$$

## 3.1 Singular Kernels

In the following we restrict our attention to kernels $K = K_\beta$ $(\beta \in \mathbb{N}_0)$ which are $C^\infty$ except for the origin and which satisfy for $k = 1, \ldots, p$ the relation

$$|K^{(k)}(x)| \le C_K \frac{(k + \beta - 1)!}{(\beta - 1)!} |x|^{-(k+\beta)} \quad (x \neq 0), \tag{3.2}$$

with a constant $C_K \ge 0$. In case $\beta = 0$ we set $(-1)! := 1$. For example, $K(x) = 1/|x|^\beta$ $(\beta \in \mathbb{N})$ fulfills (3.2) with $C_K = 1$. Further, $K(x) = \log|x|$ satisfies (3.2) with $C_K = 1$ and $\beta = 0$.

The following lemma was proved by the authors in [24].

**Lemma 3.2** *Let $K$ satisfy* (3.2) *and let $T_{\mathrm{I}}$ be the trigonometric polynomial associated with $K$ by* (2.3) *and* (2.4). *Then, for $2 \le p \le 50$, the following estimate holds true*

$$\sum_{j=1}^{p-1} j^p |a_j^{\mathrm{I}}| \;\le\; C_K \frac{(p + \beta + \delta_{0,\beta} - 2)!}{(\beta - 1)!\, \varepsilon_{\mathrm{I}}^\beta} \left(\frac{\gamma\pi}{2}\right)^p$$

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

We conjecture that Lemma 3.2 is true for arbitrary integers $p \geq 2$. However, our proof in [24] contains some numerical computations which were done only up to $p = 50$.

Now we can prove the following theorem.

**Theorem 3.3** *Let $K$ be a function which is $C^\infty$ except for the origin. Assume that $K$ satisfies (3.2). For the radial function $\mathcal{K}(\boldsymbol{x}) := K(\|\boldsymbol{x}\|)$, we define $\mathcal{K}_{\mathrm{ER}}$ by (2.9). Then, for $3 \leq p \leq 50$, $p \ll n$ and $\varepsilon_{\mathrm{I}} \leq \varepsilon_{\mathrm{B}}$, the following estimate holds true*

$$\|\mathcal{K}_{\mathrm{ER}}\|_\infty < C_K \, \frac{10\,\pi\,\sqrt{p}\,(p+\beta+\delta_{0,\beta}-2)!}{(\beta-1)!(p-2)\,n^{p-2}\,\varepsilon_{\mathrm{I}}^{p+\beta-2}}.$$

**Proof:** By Lemma 3.1, we obtain that

$$\begin{aligned}
\|\mathcal{K}_{\mathrm{ER}}\|_\infty &\leq 2\sum_{\boldsymbol{l}\in I_n}\sum_{\boldsymbol{r}\in\mathbb{Z}^2\setminus(0,0)} |c_{\boldsymbol{l}+n\boldsymbol{r}}(\mathcal{K}_{\mathrm{R}})| \\
&\leq 2\sum_{k=-n/2}^{n/2} \left(|c_{(n/2,k)}(\mathcal{K}_{\mathrm{R}})| + |c_{(k,n/2)}(\mathcal{K}_{\mathrm{R}})|\right) \; + \; 2 \sum_{\|\boldsymbol{k}\|_\infty \geq n/2+1} |c_{\boldsymbol{k}}(\mathcal{K}_{\mathrm{R}})|\,. \qquad (3.3)
\end{aligned}$$

To keep the notation simple we restrict our attention to even $p$ and set $q := \frac{p}{2}$. Since $\mathcal{K}_{\mathrm{R}} \in H^p(\mathbb{T}^2)$, we have for any bivariate polynomial

$$P(x_1,x_2) = \sum_{\|\boldsymbol{\mu}\|_1\leq p} a_{\boldsymbol{\mu}} x_1^{\mu_1} x_2^{\mu_2}$$

of total degree $\leq p$ that

$$c_{\boldsymbol{k}}(\mathcal{K}_{\mathrm{R}}) = \frac{1}{P(2\pi \mathrm{i}\boldsymbol{k})} c_{\boldsymbol{k}}(P(D)\mathcal{K}_{\mathrm{R}}),$$

where

$$P(D) := \sum_{\|\boldsymbol{\mu}\|_1\leq p} a_{\boldsymbol{\mu}} \frac{\partial^{\|\boldsymbol{\mu}\|_1}}{\partial x_1^{\mu_1}\partial x_2^{\mu_2}}\,.$$

In particular, we obtain for

$$P(x_1,x_2) = (x_1^2+x_2^2)^q = \|\boldsymbol{x}\|^p$$

that

$$|c_{\boldsymbol{k}}(\mathcal{K}_{\mathrm{R}})| \leq \frac{1}{(2\pi)^p\|\boldsymbol{k}\|^p}\,|c_{\boldsymbol{k}}(\Delta^q\mathcal{K}_{\mathrm{R}})|\,,$$

where $\Delta u = \frac{\partial^2}{\partial x_1^2}u + \frac{\partial^2}{\partial x_2^2}u$ and $\Delta^q u = \Delta(\Delta^{q-1}u)$. Thus, by (3.3),

$$\|\mathcal{K}_{\mathrm{ER}}\|_\infty \leq \frac{2}{(2\pi)^p}\left(2\sum_{k=-n/2}^{n/2}\left(\left(\frac{n}{2}\right)^2+k^2\right)^{-q} + \sum_{\|\boldsymbol{k}\|_\infty\geq n/2+1}\frac{1}{\|\boldsymbol{k}\|^p}\right)\int_{\mathbb{T}^2}|\Delta^q\mathcal{K}_{\mathrm{R}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x}\,.$$

The first sum can be estimated directly by

$$\sum_{k=-n/2}^{n/2}\left(\left(\frac{n}{2}\right)^2+k^2\right)^{-q} \;\leq\; n\left(\frac{n^2}{4}\right)^{-q} \;=\; \frac{2^p}{n^{p-1}}$$

and the second sum by an upper integral

$$\sum_{\|\boldsymbol{k}\|_\infty \geq n/2+1} \frac{1}{\|\boldsymbol{k}\|^p} \;\leq\; \int\limits_{||\boldsymbol{x}|| \geq n/2} \frac{1}{||\boldsymbol{x}||^p}\,\mathrm{d}\boldsymbol{x} \,+\, 4\int\limits_{n/2}^{\infty} \frac{1}{x}\,\mathrm{d}x \;\leq\; \frac{2^{p-2}}{n^{p-2}(p-2)}\left(2\pi + \frac{8}{n}\right).$$

In summary we obtain for $n \gg p$ that

$$||\mathcal{K}_{\mathrm{ER}}||_\infty \leq \frac{5\pi}{4(p-2)\pi^p n^{p-2}} \int\limits_{\mathbb{T}^2} |\Delta^q \mathcal{K}_{\mathrm{R}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x}\,. \tag{3.4}$$

By definition of $\mathcal{K}_{\mathrm{R}}$, in particular, since $\mathcal{K}_{\mathrm{R}}(\boldsymbol{x})$ is constant for $\|\boldsymbol{x}\| \geq 1/2$, it follows that

$$\begin{aligned} \int\limits_{\mathbb{T}^2} |\Delta^q \mathcal{K}_{\mathrm{R}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} \;&=\; \int\limits_{\|\boldsymbol{x}\| \leq \varepsilon_{\mathrm{I}}} |\Delta^q T_{\mathrm{I}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} \,+\, \int\limits_{\varepsilon_{\mathrm{I}} \leq \|\boldsymbol{x}\| \leq \frac{1}{2}-\varepsilon_{\mathrm{B}}} |\Delta^q K(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} \\ &\quad+ \int\limits_{\frac{1}{2}-\varepsilon_{\mathrm{B}} \leq \|\boldsymbol{x}\| \leq \frac{1}{2}} |\Delta^q T_{\mathrm{B}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} \end{aligned}$$

and further by using polar coordinates

$$\begin{aligned} \int\limits_{\mathbb{T}^2} |\Delta^q \mathcal{K}_{\mathrm{R}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} \;&=\; 2\pi\Bigg(\int\limits_0^{\varepsilon_{\mathrm{I}}} r|\Delta_r^q T_{\mathrm{I}}(r)|\,\mathrm{d}r \,+\, \int\limits_{\varepsilon_{\mathrm{I}}}^{\frac{1}{2}-\varepsilon_{\mathrm{B}}} r|\Delta_r^q K(r)|\,\mathrm{d}r \\ &\quad+ \int\limits_{\frac{1}{2}-\varepsilon_{\mathrm{B}}}^{\frac{1}{2}} r|\Delta_r^q T_{\mathrm{B}}(r)|\,\mathrm{d}r\Bigg) \end{aligned} \tag{3.5}$$

where $\Delta_r u = u_{rr} + \frac{1}{r}u_r$. Note that

$$\Delta_r^q u \;=\; \sum_{k=1}^{p} \frac{c_{p,k}}{r^{p-k}} \frac{\mathrm{d}^k}{\mathrm{d}r^k}\,u\,, \tag{3.6}$$

where the coefficients $c_{p,k}$ can be obtained recursively by

$$\begin{aligned} c_{p,p} \;&=\; 1\,,\;\; c_{p,p-1} = p/2\,, \\ c_{p,k} \;&=\; (p-2-k)^2 c_{p-2,k} + (3-2(p-k))c_{p-2,k-1} + c_{p-2,k-2} \quad (k = 3,\ldots,p-2)\,, \\ c_{p,2} \;&=\; -(p-3)^2 c_{p-2,1}\,,\;\; c_{p,1} = (p-3)^2 c_{p-2,1}\,. \end{aligned}$$

Using this relation, it is easy to check by induction that

$$|c_{p,k}| = (-1)^{k+1}\,c_{p,k} \qquad (k = 1,\ldots,p-1). \tag{3.7}$$

By definition (2.3) of $T_{\mathrm{I}}$, we see that

$$\begin{aligned} \Delta_r^q T_{\mathrm{I}}(r) \;&=\; \sum_{j=0}^{p-1} a_j^{\mathrm{I}} \sum_{k=1}^{p} \frac{c_{p,k}}{r^{p-k}} \left(\frac{\pi j}{2\varepsilon_{\mathrm{I}}}\right)^k \cos^{(k)}\left(\frac{\pi j}{2\varepsilon_{\mathrm{I}}}\,r\right) \\ &=\; \left(\frac{\pi}{2\varepsilon_{\mathrm{I}}}\right)^p \sum_{j=0}^{p-1} j^p a_j^{\mathrm{I}} \sum_{k=1}^{p} \frac{c_{p,k}}{\left(\frac{\pi j}{2\varepsilon_{\mathrm{I}}}\,r\right)^{p-k}} \cos^{(k)}\left(\frac{\pi j}{2\varepsilon_{\mathrm{I}}}\,r\right) \\ &=\; \left(\frac{\pi}{2\varepsilon_{\mathrm{I}}}\right)^p \sum_{j=0}^{p-1} j^p a_j^{\mathrm{I}}\;(\Delta_r^q \cos)\left(\frac{\pi j}{2\varepsilon_{\mathrm{I}}}\,r\right)\,. \end{aligned} \tag{3.8}$$

Since

$$\Delta_x^q x^j = \begin{cases} 0 & j < p, \\ 2^p\,(q!)^2 & j = p, \\ 2^p\,\left(\frac{(j/2)!}{(j/2-q)!}\right)^2\, x^{j-p} & j > p, \end{cases}$$

we obtain by the Taylor series of $\cos x$ that

$$|\Delta_x^q \cos x| \le |(\Delta_x^q \cos)(0)| = 2^p\,(q!)^2/p!$$

and by using Stirling's formula for $p \ge 3$

$$\sqrt{2\pi p}\left(\frac{p}{\mathrm{e}}\right)^p < p! < 1.03\,\sqrt{2\pi p}\left(\frac{p}{\mathrm{e}}\right)^p \tag{3.9}$$

that

$$|\Delta_x^q \cos x| \le \frac{4}{3}\,\sqrt{p}.$$

Applying this inequality and Lemma 3.2 in (3.8), we get

$$\int_0^{\varepsilon_{\mathrm{I}}} r|\Delta_r^q T_{\mathrm{I}}|\,\mathrm{d}r \le C_K\,\frac{2\,\sqrt{p}\,(p+\beta+\delta_{0,\beta}-2)!}{3\,(\beta-1)!\,\varepsilon_{\mathrm{I}}^{p+\beta-2}}\,\left(\frac{\gamma\pi^2}{4}\right)^p. \tag{3.10}$$

In a similar way we can estimate the third integral on the right-hand side of (3.5). Next we have by assumption (3.2) on $K$ and by (3.7) that

$$\begin{aligned} |\Delta_r^q K(r)| &= \left|\sum_{k=1}^{p} \frac{c_{p,k}}{r^{p-k}}\,K^{(k)}(r)\right| \le C_K \sum_{k=1}^{p} \frac{|c_{p,k}|}{r^{p-k}}\,\frac{(k+\beta-1)!}{(\beta-1)!\,r^{k+\beta}} \\ &\le \frac{C_K}{r^{p+\beta}}\left(\frac{(p+\beta-1)!}{(\beta-1)!} - \sum_{k=1}^{p-1}(-1)^k c_{p,k}\,\frac{(k+\beta-1)!}{(\beta-1)!}\right). \end{aligned}$$

For $x > 0$ and $\beta \in \mathbb{N}$, we obtain by direct computation that

$$\Delta_x^q(x^{-\beta}) = \beta^2(\beta+2)^2\ldots(\beta+p-2)^2\,x^{-(\beta+p)}$$

and by (3.6) that

$$\Delta_x^q(x^{-\beta}) = \sum_{k=1}^{p} c_{p,k}(-1)^k\,\frac{(k+\beta-1)!}{(\beta-1)!}\,x^{-(\beta+p)}.$$

Thus,

$$|\Delta_r^q K(r)| \le \frac{C_K}{r^{p+\beta}}\left(\frac{(p+\beta-1)!}{(\beta-1)!} - \left(\beta^2(\beta+2)^2\ldots(\beta+p-2)^2 - \frac{(p+\beta-1)!}{(\beta-1)!}\right)\right).$$

For $\beta = 0$ we have to consider $\log x$ instead of $x^{-\beta}$ to obtain a similar result. For $\beta \in \mathbb{N}_0$, we get in summary

$$|\Delta_r^q K(r)| \le \frac{2\,C_K\,(p+\beta-1)!}{(\beta-1)!}\,r^{-(p+\beta)}$$

and consequently

$$\int_{\varepsilon_{\mathrm{I}}}^{\frac{1}{2}-\varepsilon_{\mathrm{B}}} r|\Delta_r^q K(r)|\,\mathrm{d}r \le \frac{2\,C_K\,(p+\beta-1)!}{(p+\beta-2)\,(\beta-1)!}\,\varepsilon_{\mathrm{I}}^{-(p+\beta-2)}. \tag{3.11}$$

By (3.10), (3.11), (3.5) and (3.4) we obtain finally

$$||\mathcal{K}_{\mathrm{ER}}||_\infty < \frac{C_K\, 10\,\pi\,\sqrt{p}\,(p+\beta+\delta_{0,\beta}-2)!}{(\beta-1)!\,(p-2)n^{p-2}\,\varepsilon_{\mathrm{I}}^{p+\beta-2}}\,.$$

■

We will use the estimates in the Theorem 3.3 to specify the parameters $\varepsilon_I, p$ and $n$ of our algorithm. Using the Stirling formula (3.9) we can rewrite our error estimate as

$$|f(\boldsymbol{y}_j) - \tilde{f}(\boldsymbol{y}_j)| \le 10\,\pi\, C_K\, \|\boldsymbol{\alpha}\|_1\, n^{\beta+\delta_{0,\beta}}\, \varepsilon_{\mathrm{I}}^{\delta_{0,\beta}}\, \frac{\sqrt{2\pi p(p+\beta+\delta_{0,\beta}-2)}}{(p-2)(\beta-1)!} \left(\frac{p+\beta+\delta_{0,\beta}-2}{\mathrm{e}\, n\, \varepsilon_{\mathrm{I}}}\right)^{p+\beta+\delta_{0,\beta}-2}.$$

Thus, choosing $\varepsilon_{\mathrm{I}}$ such that $\frac{p+\beta+\delta_{0,\beta}-2}{\mathrm{e}\,n\,\varepsilon_{\mathrm{I}}} < 1$, our error decays exponentially in $p$. In our numerical examples we choose

$$\varepsilon_{\mathrm{I}} = \frac{p}{n}. \tag{3.12}$$

While (3.12) steers the error, condition (2.12) on $\varepsilon_{\mathrm{I}}$ is necessary to keep the near field computation linear in $M$. Now (3.12) and (2.12) together imply that

$$n \sim p\sqrt{\frac{N}{\nu}}.$$

If $M = N$, then the near field computation requires approximately $\nu N$ arithmetic operations and the NFFT computations

$$n^2 \log n + \mathcal{O}(N) = (Np^2/\nu)\,\log(p\sqrt{N/\nu}) \;+\; \mathcal{O}(N)$$

arithmetic operations. One should choose $\nu$ such that both operation counts are balanced. It seems that $\nu \sim p$, i.e.

$$n \sim \sqrt{p\,N}$$

is a good choice.

## 3.2 The Gaussian

In this subsection we have a closer look at the error introduced by our fast Gauss transform without boundary regularization. By (3.1) it remains to estimate the error between $\mathcal{K}$ and its finite Fourier series $\mathcal{K}_{\mathrm{RF}}$.

**Theorem 3.4** *Let $\mathcal{K}(\boldsymbol{x}) := \mathrm{e}^{-\sigma\|\boldsymbol{x}\|^2}$ $(\sigma > 0)$ be a bivariate Gaussian and let $\mathcal{K}_{\mathrm{ER}} := \mathcal{K} - \mathcal{K}_{\mathrm{RF}}$, where $\mathcal{K}_{\mathrm{RF}}$ denotes the finite Fourier series (2.7) of $\mathcal{K}$. Then, for $\eta := \frac{\pi n}{2\sqrt{\sigma}} \ge 1$, the following estimate holds true*

$$\|\mathcal{K}_{\mathrm{ER}}\|_\infty \le 4\, C_\sigma \left(\mathrm{e}^{-\eta^2}\left(\sqrt{\frac{\pi}{\sigma}} + \frac{1}{\eta}\right) \;+\; \mathrm{e}^{-\sigma/4}\,\frac{2\sqrt{\sigma}}{\eta}\right), \tag{3.13}$$

*where $C_\sigma := \left(1 + \frac{\sigma}{3}\,\mathrm{e}^{-\sigma/4} + \sqrt{\frac{\pi}{\sigma}}\right)$.*

**Proof:** The Fourier transform of the univariate Gaussian is given by

$$\int_{-\infty}^{\infty} \mathrm{e}^{-\sigma x^2}\ \mathrm{e}^{-2\pi \mathrm{i} k x}\,\mathrm{d}x = \sqrt{\frac{\pi}{\sigma}}\ \mathrm{e}^{-k^2\pi^2/\sigma}.$$

Further we will use the following simple estimates:

$$\sum_{k=1}^{n/2} \frac{1}{k^2} \;\leq\; \sum_{k=1}^{\infty} \frac{1}{k^2} \;=\; \frac{\pi^2}{6}, \tag{3.14}$$

$$\sum_{k=1}^{n/2} \mathrm{e}^{-k^2\pi^2/\sigma} \;\leq\; \int_0^{\infty} \mathrm{e}^{-x^2\pi^2/\sigma}\,\mathrm{d}x \;=\; \frac{1}{2}\sqrt{\frac{\sigma}{\pi}}, \tag{3.15}$$

$$\sum_{k=n/2+1}^{\infty} \frac{1}{k^2} \;\leq\; \int_{n/2}^{\infty} \frac{1}{x^2}\,\mathrm{d}x \;=\; \frac{2}{n}, \tag{3.16}$$

$$\sum_{k=n/2+1}^{\infty} \mathrm{e}^{-k^2\pi^2/\sigma} \;\leq\; \int_{n/2}^{\infty} \mathrm{e}^{-x^2\pi^2/\sigma}\,\mathrm{d}x \;\leq\; \frac{\sigma}{n\pi^2}\ \mathrm{e}^{-\pi^2 n^2/(4\sigma)}, \tag{3.17}$$

where the last inequality follows by

$$\int_a^{\infty} \mathrm{e}^{-cx^2}\,\mathrm{d}x \leq \int_0^{\infty} \mathrm{e}^{-c(x+a)^2}\,\mathrm{d}x \leq \mathrm{e}^{-ca^2} \int_0^{\infty} \mathrm{e}^{-2acx}\,\mathrm{d}x = \frac{\mathrm{e}^{-ca^2}}{2ac}.$$

By applying two times integration by parts, we obtain for the univariate Gaussian and $k \neq 0$ that

$$\begin{aligned}
c_k\left(\mathrm{e}^{-\sigma x^2}\right) \;&:=\; \int_{-1/2}^{1/2} \mathrm{e}^{-\sigma x^2}\ \mathrm{e}^{-2\pi \mathrm{i} k x}\,\mathrm{d}x \\
&=\; (-1)^{k+1}\frac{\sigma}{2\pi^2 k^2}\ \mathrm{e}^{-\sigma/4} \;+\; \frac{\sigma}{2\pi^2 k^2} \int_{-1/2}^{1/2} (1 - 2\sigma x^2)\ \mathrm{e}^{-\sigma x^2}\ \mathrm{e}^{-2\pi \mathrm{i} k x}\,\mathrm{d}x \\
&=\; (-1)^{k+1}\frac{\sigma}{2\pi^2 k^2}\ \mathrm{e}^{-\sigma/4} \;-\; \frac{1}{4\pi^2 k^2} \int_{-\infty}^{\infty} (\mathrm{e}^{-\sigma x^2})''\ \mathrm{e}^{-2\pi \mathrm{i} k x}\,\mathrm{d}x \\
&\quad\; + \frac{1}{2\pi^2 k^2} \int_{1/2}^{\infty} (\mathrm{e}^{-\sigma x^2})''\ \cos(2\pi k x)\,\mathrm{d}x
\end{aligned}$$

and consequently

$$\begin{aligned}
|c_k\left(\mathrm{e}^{-\sigma x^2}\right)| \;&\leq\; \frac{\sigma}{2\pi^2 k^2}\ \mathrm{e}^{-\sigma/4} \;+\; \sqrt{\frac{\pi}{\sigma}}\ \mathrm{e}^{-k^2\pi^2/\sigma} \;+\; \frac{1}{2\pi^2 k^2} \int_{1/2}^{\infty} (\mathrm{e}^{-\sigma x^2})''\,\mathrm{d}x \\
&=\; \sqrt{\frac{\pi}{\sigma}}\ \mathrm{e}^{-k^2\pi^2/\sigma} \;+\; \frac{\sigma}{\pi^2 k^2}\ \mathrm{e}^{-\sigma/4}.
\end{aligned} \tag{3.18}$$

By (3.3) we have

$$\begin{aligned}\|\mathcal{K}_{\mathrm{ER}}\|_\infty &\le 2\sum_{k=-n/2}^{n/2}\left(|c_{(n/2,k)}(\mathcal{K})|+|c_{(k,n/2)}(\mathcal{K})|\right) + 2\sum_{\|\boldsymbol{k}\|_\infty\ge n/2+1}|c_{\boldsymbol{k}}(\mathcal{K})| \\ &=: 2\,S_1 + 2\,S_2.\end{aligned}$$

Using the tensor product structure of the bivariate Gaussian, i.e. the splitting $c_{\boldsymbol{k}}(\mathcal{K}) = c_{k_1}(\mathrm{e}^{-\sigma x_1^2})\cdot c_{k_2}(\mathrm{e}^{-\sigma x_2^2})$, where $\boldsymbol{k} := (k_1,k_2)^{\mathrm{T}}$ and $\boldsymbol{x} := (x_1,x_2)^{\mathrm{T}}$, and (3.18) we get for the first sum

$$S_1 \le 2\left(\sqrt{\frac{\pi}{\sigma}}\,\mathrm{e}^{-n^2\pi^2/(4\sigma)} + \frac{4\sigma}{\pi^2 n^2}\,\mathrm{e}^{-\sigma/4}\right)\left(\sum_{\substack{k=-n/2\\k\ne 0}}^{n/2}\left(\sqrt{\frac{\pi}{\sigma}}\,\mathrm{e}^{-k^2\pi^2/\sigma} + \frac{\sigma}{\pi^2 k^2}\,\mathrm{e}^{-\sigma/4}\right) + \sqrt{\frac{\pi}{\sigma}}\right)$$

and further by (3.14) and (3.15)

$$S_1 \le 2\,C_\sigma\left(\sqrt{\frac{\pi}{\sigma}}\,\mathrm{e}^{-\eta^2} + \frac{1}{\eta^2}\,\mathrm{e}^{-\sigma/4}\right).$$

The second sum splits as

$$S_2 \le 4\sum_{k_1=n/2+1}^{\infty}\sum_{k_2=n/2+1}^{\infty}|c_{(k_1,k_2)}(\mathcal{K})| + 4\sum_{k_1=-n/2}^{n/2}\sum_{k_2=n/2+1}^{\infty}|c_{(k_1,k_2)}(\mathcal{K})|.$$

Estimating the right–hand side by (3.18) and (3.14) – (3.17) we arrive at

$$S_2 \le 4\,A(n,\sigma)\,\left(A(n,\sigma)+C_\sigma\right),$$

where

$$A(n,\sigma) := \frac{1}{2\sqrt{\pi}\eta}\,\mathrm{e}^{-\eta^2} + \frac{\sqrt{\sigma}}{\pi\eta}\,\mathrm{e}^{-\sigma/4}.$$

Note that $A(n,\sigma) \le 0.38$. In summary we obtain

$$\|\mathcal{K}_{\mathrm{ER}}\|_\infty \le 4\,C_\sigma\left(\mathrm{e}^{-\eta^2}\left(\sqrt{\frac{\pi}{\sigma}}+\frac{1}{\eta}\right) + \mathrm{e}^{-\sigma/4}\,\frac{2\sqrt{\sigma}}{\eta}\right).$$

■

The first summand in (3.13) decreases with increasing $\eta$. The second summand is negligible for larger $\sigma$, e.g. we have $\sqrt{\sigma}\,\mathrm{e}^{-\sigma/4} < 2.7\cdot 10^{-6}$ for $\sigma \ge 60$.

# 4 Numerical Examples

We have implemented Algorithms 2.1 in C and tested on an AMD Atlon xp1800+ 512MB RAM, SuSe-Linux 8.0 using double precision arithmetic.
Throughout our experiments we have applied the NFFT/NFFT$^{\mathrm{T}}$ package [23] with Kaiser–Bessel functions, oversampling factor $\rho = 2$ and several cut–off parameters $m$ which will be specified in the examples.

For simplicity we have chosen $M = N$ and knots $\boldsymbol{y}_j = \boldsymbol{x}_j$ $(j = 1, \dots, N)$ in $\{\boldsymbol{x} : ||\boldsymbol{x}|| \le \frac{7}{32}\}$, i. e. $\varepsilon_B = \frac{1}{16}$. In case of singular kernels we have computed

$$f(\boldsymbol{x}_j) := \sum_{\substack{k=1 \\ k \neq j}}^{N} \alpha_k \mathcal{K}(\boldsymbol{x}_j - \boldsymbol{x}_k) \qquad (j = 1, \dots, N)$$

instead of (1.1). The coefficients $\alpha_k$ were randomly distributed in [0,1]. Moreover, we set $\varepsilon_{\mathrm{I}} = \frac{p}{n}$.
Every figure presents the arithmetic mean of 20 computations.

**Example 4.1** Figure 4.1 presents the error

$$E := \max_{j=1,\dots,N} \frac{|f(\boldsymbol{x}_j) - \tilde{f}(\boldsymbol{x}_j)|}{|f(\boldsymbol{x}_j)|}\,. \tag{4.1}$$

introduced by our algorithm as function of the regularization parameter $p$ for the kernels in (2.1). We have used $N = 512^2$ randomly distributed knots $\boldsymbol{x}_j$ as depicted in Figure 4.2 (left). Further, we have applied two dimensional NFFTs of size $n = 512$ with a large cut–off parameter $m = 8$ to ensure that the NFFT does not introduce an additional error.
As expected the error decreases exponentially with increasing $p$. Further the error increases with an increasing order of the singularity of $\mathcal{K}$. □

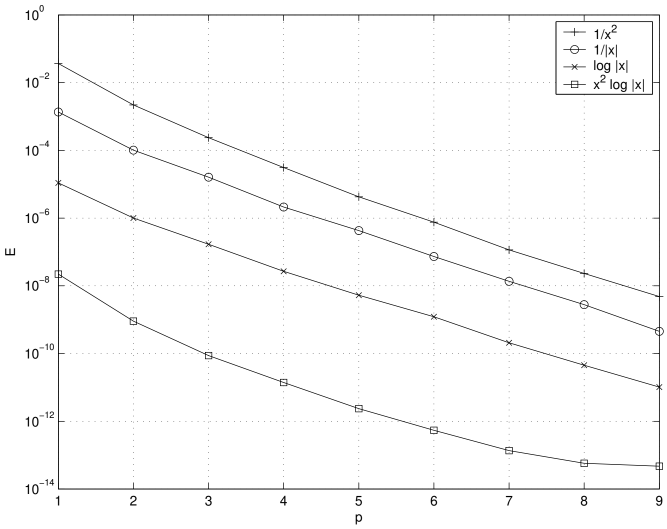


Figure 4.1: Error $E$ of Algorithm 2.1 in dependence on the regularization parameter $p$ for various kernels, $N = 512^2$ randomly distributed knots and NFFTs of size $n = 256$.

**Example 4.2** Next we are interested in computation times. To compare our method with the FMM we consider the kernel $\mathcal{K}(\boldsymbol{x}) = \log \|\boldsymbol{x}\|$. The FMM Toolbox$^{TM}$ v1.0 from MADMAX Optics$^{TM}$ [1] was specifically designed for this kernel. We applied the routine `fmmcoul2d` with parameter `IPREC=1` from this toolbox. Table 4.1 compares the time for the direct computation

with those for the fast summation by using the FMM–package and our implementation for randomly distributed knots. In our algorithm we have set $m = 4$ and $p = 3$ to achieve single precision, i.e. a maximum error $E$ in (4.1) smaller than 1.0e−6. The FMM produces nearly the same error. Note that the computation time for our fast algorithm includes the computation time for the search of all points in the near field, which can be done in $\mathcal{O}(\log N)$. The FMM performs only slightly better than our NFFT–based program. Since this is also true for the direct computation we hope to obtain further improvements by a more advanced implementation.
For highly inhomogeneous point distributions as the distribution in Figure 4.2 our algorithm requires about one and a half times the computation time reported in Table 4.1 for our algorithm. □

| parameter | | FMM | | NFFT | |
|---|---|---|---|---|---|
| $N$ | $n$ | direct | fast | direct | fast |
| $32^2$ | 88 | 1.90e−1 | 5.00e−2 | 2.00e−1 | 4.00e−2 |
| $64^2$ | 156 | 2.85e+0 | 1.30e−1 | 3.32e+0 | 1.60e−1 |
| $128^2$ | 312 | 4.73e+1 | 5.50e−1 | 5.46e+1 | 7.60e−1 |
| $256^2$ | 588 | 7.57e+2 | 2.30e+0 | 8.75e+2 | 3.46e+0 |
| $512^2$ | 980 | – | 9.40e+0 | ∼ 1.39e+4 | 1.69e+1 |
| $1024^2$ | 1960 | – | 3.85e+1 | ∼ 2.20e+5 | 7.97e+1 |

Table 4.1: Comparison of the computation time for the direct computation, the FMM and our NFFT–based Algorithm 2.1 for the kernel $\mathcal{K}(\boldsymbol{x}) = \log ||\boldsymbol{x}||$ and randomly distributed knots.

**Remark 4.3** (**"near field correction"**) Up to now we have assumed that the values $\mathcal{K}_{\mathrm{NE}}(\boldsymbol{y}_j - \boldsymbol{x}_k) = \mathcal{K}(\boldsymbol{y}_j - \boldsymbol{x}_k) - \mathcal{K}_{\mathrm{R}}(\boldsymbol{y}_j - \boldsymbol{x}_k) = \mathcal{K}(\boldsymbol{y}_j - \boldsymbol{x}_k) - T_{\mathrm{I}}(\|\boldsymbol{y}_j - \boldsymbol{x}_k\|_2)$ ($k \in I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(\boldsymbol{y}_j)$; $j = 1, \ldots, N$) in Step 4 of Algorithm 2.1 were precomputed. Alternatively, one can use the following procedure: splitting

$$f_{\mathrm{NE}}(\boldsymbol{y}_j) = \sum_{k \in I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(\boldsymbol{y}_j)} \alpha_k \mathcal{K}(\boldsymbol{y}_j - \boldsymbol{x}_k) - \sum_{k \in I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(\boldsymbol{y}_j)} \alpha_k T_{\mathrm{I}}(\|\boldsymbol{y}_j - \boldsymbol{x}_k\|),$$

we evaluate the first sum with $\mathcal{O}(pM)$ arithmetic operations first. By definition (2.3) of $T_{\mathrm{I}}$ the computation of the second sum would require $\mathcal{O}(p^2 M)$ arithmetic operations. We approximate this sum as follows: we precompute $T_{\mathrm{I}}(\varepsilon_{\mathrm{I}} \frac{s}{p^2})$ ($s = -p^2, \ldots, p^2$) and approximate $T_{\mathrm{I}}(\|\boldsymbol{y}_j - \boldsymbol{x}_k\|_2)$ by cubic spline interpolation. Now the (approximate) computation of the second sum requires nearly the same number of arithmetic operations as the computation of the first one. □

**Example 4.4** We consider the Gaussian $\mathcal{K}(\boldsymbol{x}) = \mathrm{e}^{-\sigma\|2\boldsymbol{x}\|^2}$. We have introduced the factor 2 because the computations in [19] were done with respect to $||\boldsymbol{x}|| \le 1/2$ instead of $||\boldsymbol{x}|| \le 1/4$. Since the performance of our algorithm does not depend on the point distribution, we have chosen the inhomogeneous point distribution in Figure 4.2 (right). For randomly distributed points, we obtain exactly the same results. In the NFFT computations we have set $m = 4$. Moreover, following the error estimate in Theorem 3.4, we have chosen $n$ and $\sigma$ such that $n/\sqrt{\sigma}$ is constant. Here $n/\sqrt{\sigma} \approx 5$ which ensures an error $E \le 5.5\text{e}{-8}$. Figure 4.3 presents the CPU time of our simplified Algorithm 2.1 (Step 1 – 3, see (2.13)). As expected we see that the computational complexity is only $\mathcal{O}(N)$.

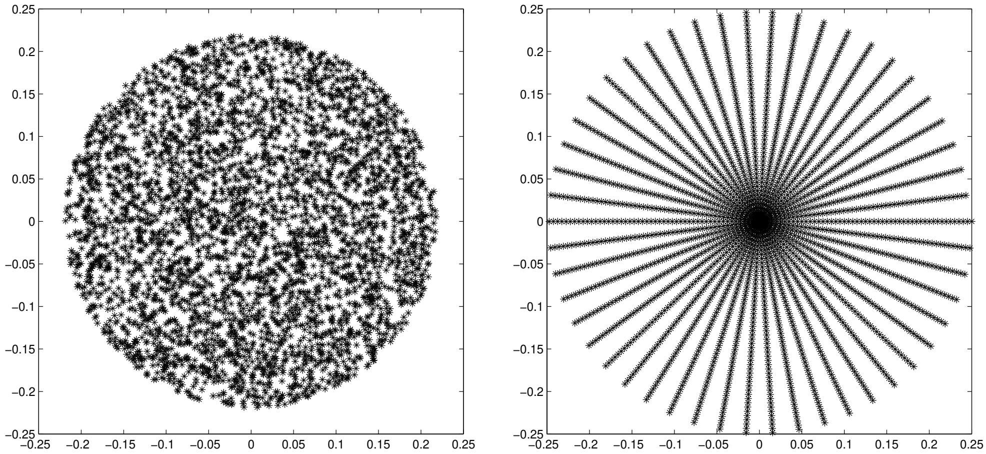

Figure 4.2: Left: homogenous point distribution. Right: highly inhomogeneous point distribution.

For small values of $\sigma$ one has to add a boundary regularization to obtain arbitrary small errors as shown in Table 4.2. Here $p = 0$ denotes the algorithm without boundary regularization. The complexity depends on $n$ as in Figure 4.3. $\square$

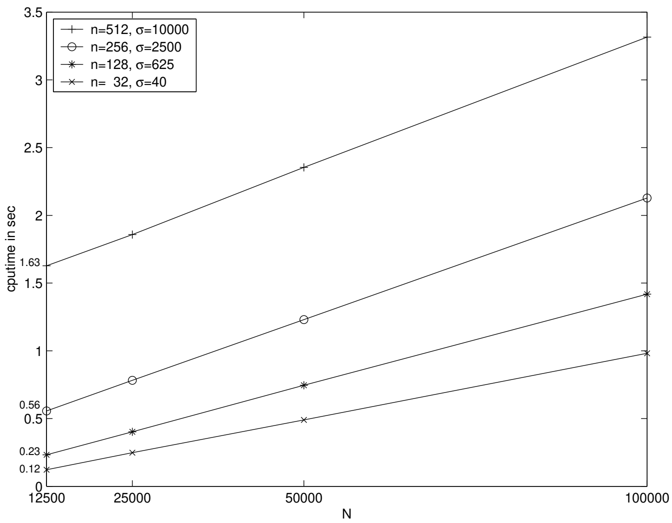


Figure 4.3: Computation time of Algorithm 2.1 for $\mathcal{K}(\boldsymbol{x}) = \mathrm{e}^{-\sigma\|2\boldsymbol{x}\|_2^2}$ with various values $\sigma$ and corresponding NFFT sizes $n$ in dependence on the number $N$ of (inhomogeneousely distributed) points.

# References

[1] MadMax Optics, FMM Toolbox$^{TM}$. http://www.madmaxoptics.com, 2003.

| $p$ | 0 | 2 | 4 | 6 | 8 |
|---|---|---|---|---|---|
| $n$ | 32 | 32 | 64 | 128 | 256 |
| $E$ | 3.659e-05 | 6.418e-06 | 1.666e-07 | 1.474e-08 | 3.739e-12 |

Table 4.2: Error of Algorithm 2.1 (Step 1 – 3) with boundary regularization for $\mathcal{K}(\boldsymbol{x}) = \mathrm{e}^{-\|2\boldsymbol{x}\|_2^2}$, i.e. $\sigma = 1$ and $N = 10000$.


[2] 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.

[3] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: Moment based methods. SIAM *J. Sci. Comput.*, 19:1428 – 1449, 1998.

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

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

[6] M. Broadie and Y. Yamamoto. Application of the fast Gauss transform to option pricing. *Management Science*, 49:1071 – 1088, 2003.

[7] J. B. Cherrie, R. K. Beatson, and G. N. Newsam. Fast evaluation of radial basis functions: Methods for generalized multiquadrics in $R^n$. SIAM *J. Sci. Comput.*, 23:1549 – 1571, 2002.

[8] J. Duchon. Fonctions splines et vecteurs aleatoires. Technical report, Seminaire d'Analyse Numerique, Universite Scientifique et Medicale, Grenoble, 1975.

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

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

[11] A. Elgammal, R. Duraiswami, and L. S. Davis. Efficient non-parametric adaptive color modeling using fast Gauss transform. Technical report, The Univ. of Maryland, 2001.

[12] J. Fessler and B. Sutton. NUFFT - nonuniform FFT toolbox for Matlab. http://www.eecs.umich.edu/˜fessler/code/index.html, 2002.

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

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

[15] S. A. Goreinov, E. E. Tyrtyshnikov, and E. E. Yeremin. Matrix–free iterative solution strategies for large dense systems. *Numer. Linear Algebra Appl.*, 4:273 – 294, 1997.

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

[17] L. Greengard and P. Lin. Spectral approximation of the free–space heat kernel. *Appl. Comput. Harmon. Anal.*, 9:83 – 97, 2000.

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

[19] L. Greengard and J. Strain. The fast Gauss transform. SIAM *J. Sci. Stat. Comput.*, 12:79 – 94, 1991.

[20] L. Greengard and X. Sun. A new version of the fast Gauss transform. *Doc. Math. J. DMV*, 3:575 – 584, 1998.

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

[22] 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.

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

[24] D. Potts and G. Steidl. Fast summation at nonequispaced knots by NFFTs. SIAM *J. Sci. Comput.*, 24:2013 – 2037, 2003.

[25] 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.

[26] 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. Birkhäuser.

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

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

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

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