NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2002; **00**:1–8 *Prepared using* nlaauth.cls *[Version: 2000/03/22 v1.0]*

Fast summation based on fast trigonometric transforms at nonequispaced nodes

Markus Fenn[1] and Daniel Potts[2]

[1] *University of Mannheim, Institute of Mathematics, D–68159 Mannheim, Germany*
Markus.Fenn@uni-mannheim.de
[2] *University of Lübeck, Institute of Mathematics, Wallstr. 40, D–23560 Lübeck, Germany*
potts@math.uni-luebeck.de

SUMMARY

We develop a new algorithm for the fast computation of matrix–vector–products with special matrices. More precisely we develop a method for the fast computation of sums $f(y_j) := \sum_{k=1}^{N} \alpha_k K(y_j - x_k)$ at nonequispaced nodes x_k and y_j $(j = 1, \ldots, M)$ which requires only $\mathcal{O}(N \log N + (M + N))$ arithmetic operations. Our algorithm is based on a novel approach to fast discrete trigonometric transforms at nonequispaced nodes.

KEY WORDS: fast Fourier, cosine and sine transforms at nonequispaced knots, NFFT, NDCT, NDST, fast discrete summation
Mathematics Subject Classification. 65T40, 65T50, 65F30.

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 simulations, in the numerical solution of integral equations and in the approximation of functions by radial basis functions. Various algorithms were designed to speed up the summation process, e. g., the *fast multipole method* [8, 1], the *panel clustering method* [10], the theory of *$\mathcal{H}$–matrices* [9], *mosaic–skeleton approximations* [19], *multiresolution based methods* [4] and *wavelet methods* [3]. Recently, a fast summation algorithm based on the *fast Fourier transform at nonequispaced nodes* (NFFT) was developed [14] which allows a simple incorporation of different kernels K. The objective of this paper is to improve this NFFT based summation algorithm for real input data by applying fast trigonometric transforms instead of *fast Fourier transforms* (FFT). To achieve our goal we have to develop fast algorithms for the *discrete cosine transform at nonequispaced nodes* (NDCT) and for the *discrete sine transform at nonequispaced nodes* (NDST), respectively. Our approach is based on the NFFT and seems to be easier than the Chebyshev transform based derivation in [13] and faster than the algorithm in [18] which still uses FFTs. Instead of FFTs we use fast algorithms for the *discrete cosine transform* (DCT–I) and for the *discrete sine transform* (DST–I) of type I.

Received
Revised
Accepted

The remainder of this paper is organized as follows: in Section 2 we present our fast algorithms for the NDCT and the NDST which we call NFCT and NFST, respectively. In Section 3 we apply these algorithms for the fast computation of sums of the form

$$f(y_j) = \sum_{k=1}^{N} \alpha_k K(y_j - x_k) \quad (j = 1, \ldots, M) \quad \text{resp.} \quad \boldsymbol{f} = \boldsymbol{K\alpha}, \tag{1.1}$$

where $\boldsymbol{f} = \big(f(y_j)\big)_j$, $\boldsymbol{\alpha} = \big(\alpha_k\big)_k$, $\boldsymbol{K} = \big(K(y_j - x_k)\big)_{jk}$ and K are special even kernels, e. g.,

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

Finally, we present numerical examples in Section 4.

2. Fast trigonometric transforms at nonequispaced nodes

In the following we develop fast algorithms for trigonometric transforms at nonequispaced nodes. Our approach is based on the NFFT (see e. g. [17, 15]), which we introduce first.

2.1. The NFFT

The NDFT consists in the evaluation of the 1–periodic trigonometric polynomial

$$f^{\mathrm{F}}(v) := \sum_{k=-n}^{n-1} \hat{f}_k^{\mathrm{F}}\, \mathrm{e}^{-2\pi \mathrm{i} k v} \tag{2.1}$$

at the nodes $v_j \in [-1/2, 1/2)$ $(j = 1, \ldots, M)$. We want to present a fast approximative algorithm for the NDFT. Let φ be an even *window function* so that its *1–periodic* version $\tilde{\varphi}(v) = \sum_{r \in \mathbb{Z}} \varphi(v + r)$ has an absolute convergent Fourier series. We introduce the *oversampling factor* $\sigma > 1$ and approximate f^{F} by

$$s_1(v) := \sum_{l=-\sigma n}^{\sigma n - 1} g_l\, \tilde{\varphi}\left(v - \frac{l}{2\sigma n}\right),$$

i. e., we want to define g_l such that $s_1 \approx f^{\mathrm{F}}$. We rewrite s_1 as Fourier series, such that

$$\begin{aligned} s_1(v) &= \sum_{k \in \mathbb{Z}} \hat{g}_k\, c_k(\tilde{\varphi})\, \mathrm{e}^{-2\pi \mathrm{i} k v} \\ &= \sum_{k=-\sigma n}^{\sigma n - 1} \hat{g}_k\, c_k(\tilde{\varphi})\, \mathrm{e}^{-2\pi \mathrm{i} k v} + \sum_{r \in \mathbb{Z}\setminus\{0\}} \sum_{k=-\sigma n}^{\sigma n - 1} \hat{g}_k\, c_{k+2\sigma n r}(\tilde{\varphi})\, \mathrm{e}^{-2\pi \mathrm{i}(k + 2\sigma n r) v}, \end{aligned} \tag{2.2}$$

where

$$\hat{g}_k := \sum_{l=-\sigma n}^{\sigma n - 1} g_l\, \mathrm{e}^{\pi \mathrm{i} k l/(\sigma n)}, \qquad g_l = \frac{1}{2\sigma n} \sum_{k=-\sigma n}^{\sigma n - 1} \hat{g}_k\, \mathrm{e}^{-\pi \mathrm{i} k l/(\sigma n)} \tag{2.3}$$

and

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

Suppose that the Fourier coefficients $c_k(\tilde{\varphi})$ become sufficiently small for $|k| \geq \sigma n$ and that $c_k(\tilde{\varphi}) \neq 0$ for $k = -n, \ldots, n-1$. Then, comparing (2.2) with (2.1), we set for $r \in \mathbb{Z}$

$$\hat{g}_k := \hat{g}_{k+2\sigma n r} = \begin{cases} \hat{f}_k^{\mathrm{F}}/c_k(\tilde{\varphi}) & k = -n, \ldots, n-1, \\ 0 & k = -\sigma n, \ldots, -n-1;\ k = n, \ldots, \sigma n - 1. \end{cases} \tag{2.4}$$

Now the values g_l can be obtained from (2.3) by a (reduced) FFT of size $2\sigma n$. Assume further that $\tilde{\varphi}$ is also well–localized in time domain such that $s_1(v_j)$ $(j = 1, \ldots, M)$ can be approximated by the truncated sum

$$f^{\mathrm{F}}(v_j) \approx s_1(v_j) \approx s(v_j) := \sum_{l=\lfloor 2\sigma n v_j \rfloor - (m-1)}^{\lceil 2\sigma n v_j \rceil + (m-1)} g_l \, \tilde{\varphi}\left(v_j - \frac{l}{2\sigma n}\right), \tag{2.5}$$

which contains at most $2m - 1$ nonzero summands. In summary, the NFFT approximates $f^{\mathrm{F}}(v_j)$ by computing $s(v_j)$ via (2.4), (2.3) and (2.5) with $\mathcal{O}(\sigma n \log(\sigma n) + mM)$ arithmetic operations.

To keep the error $|f^{\mathrm{F}}(v_j) - s(v_j)|$ small, several window functions φ with good localization in time and frequency domain were proposed, e. g., the *Gaussian* [6, 17, 5], *cardinal central B–splines* [2, 17], *sinc functions* [12] or *Kaiser–Bessel functions* [11, 7]. A detailed analysis of the approximation errors can be found in the corresponding papers. See also [12] for a numerical comparison of the NFFT with different window functions and different choices for the parameters σ and m.

2.2. The NFCT

Let us turn to the NDCT. For given real data $\hat{f}_k^{\mathrm{C}} \in \mathbb{R}$ and arbitrary nodes $v_j \in [0, 1/2]$ $(j = 1, \ldots, M)$ we are interested in the fast and robust computation of

$$f^{\mathrm{C}}(v_j) = f_j^{\mathrm{C}} := \sum_{k=0}^{n-1} \hat{f}_k^{\mathrm{C}} \cos(2\pi k v_j). \tag{2.6}$$

Choosing $\hat{f}_k^{\mathrm{F}} \in \mathbb{R}$ $(k = 0, \ldots, n-1)$ with $\hat{f}_k^{\mathrm{F}} = \hat{f}_{-k}^{\mathrm{F}}$ and $\hat{f}_{-n}^{\mathrm{F}} = 0$ in (2.1) we obtain

$$f^{\mathrm{F}}(v) = \sum_{k=-n}^{n-1} \hat{f}_k^{\mathrm{F}} \, \mathrm{e}^{-2\pi \mathrm{i} k v} = \sum_{k=0}^{n-1} 2\varepsilon_{n,k} \hat{f}_k^{\mathrm{F}} \cos(2\pi k v),$$

where $\varepsilon_{n,0} = \varepsilon_{n,n} := 1/2$ and $\varepsilon_{n,k} := 1$ $(k = 1, \ldots, n-1)$. Consequently, we have for $\hat{f}_k^{\mathrm{C}} = 2\varepsilon_{n,k} \hat{f}_k^{\mathrm{F}}$ that $f^{\mathrm{C}}(v) = f^{\mathrm{F}}(v)$. Since $\tilde{\varphi}$ is even, we verify that $c_k(\tilde{\varphi}) = c_{-k}(\tilde{\varphi})$ and further by (2.4) that $\hat{g}_k = \hat{g}_{-k}$ $(k = 1, \ldots, \sigma n - 1)$. Using this symmetry and (2.3), we get for $l = 0, \ldots, \sigma n$ that

$$g_l = \frac{1}{2\sigma n} \sum_{k=-\sigma n}^{\sigma n - 1} \hat{g}_k \, \mathrm{e}^{-\pi \mathrm{i} k l/(\sigma n)} = \frac{1}{\sigma n} \sum_{k=0}^{\sigma n} \varepsilon_{\sigma n,k} \, \hat{g}_k \cos\left(\frac{\pi k l}{\sigma n}\right). \tag{2.7}$$

Note that $g_{2\sigma n r - l} = g_l$ $(r \in \mathbb{Z})$. Finally, we compute as in (2.5) the sums

$$f^{\mathrm{C}}(v_j) \approx s(v_j) = \sum_{l=\lfloor 2\sigma n v_j \rfloor - (m-1)}^{\lceil 2\sigma n v_j \rceil + (m-1)} g_l \, \tilde{\varphi}\left(v_j - \frac{l}{2\sigma n}\right). \tag{2.8}$$

In summary, we obtain the following algorithm for the fast computation of (2.6) with arithmetic complexity $\mathcal{O}(\sigma n \log(\sigma n) + mM)$:

Algorithm 2.1. (NFCT)
Input: $n, M \in \mathbb{N}$, $\sigma > 1$, $\hat{f}_k^{\mathrm{C}} \in \mathbb{R}$ $(k = 0, \ldots, n-1)$, $v_j \in [0, 1/2]$ $(j = 1, \ldots, M)$.
Precomputation: $c_k(\tilde{\varphi})$ $(k = 0, \ldots, n-1)$,
$\tilde{\varphi}\left(v_j - \frac{l}{2\sigma n}\right)$ $(j = 1, \ldots, M; l = \lfloor 2\sigma n v_j \rfloor - (m-1), \ldots, \lceil 2\sigma n v_j \rceil + (m-1))$
1. For $k = 0, \ldots, n-1$ *compute* $\hat{g}_k := \dfrac{\hat{f}_k^{\mathrm{C}}}{2\varepsilon_{n,k} c_k(\tilde{\varphi})}$ *and for* $k = n, \ldots, \sigma n$ *set* $\hat{g}_k := 0$.
2. For $l = 0, \ldots, \sigma n$ *compute* g_l *by* (2.7) *by a fast* DCT–I *of length* σn.
3. For $j = 1, \ldots, M$ *compute* $s(v_j)$ *by* (2.8).
Output: $s(v_j)$ *approximate values for* $f^{\mathrm{C}}(v_j)$.

2.3. The NFST

Now we are interested in the fast and robust computation of the NDST. For given real data $\hat{f}_k^{\mathrm{S}} \in \mathbb{R}$ and arbitrary nodes $v_j \in [0, 1/2]$ $(j = 1, \ldots, M)$ we have to compute

$$f^{\mathrm{S}}(v_j) = f_j^{\mathrm{S}} := \sum_{k=1}^{n-1} \hat{f}_k^{\mathrm{S}} \sin(2\pi k v_j) \,. \tag{2.9}$$

We consider again (2.1) and assume that $\hat{f}_k^{\mathrm{F}} \in \mathbb{R}$ with $\hat{f}_{-k}^{\mathrm{F}} = -\hat{f}_k^{\mathrm{F}}$ $(k = 1, \ldots, n-1)$ and that $\hat{f}_0^{\mathrm{F}} = \hat{f}_{-n}^{\mathrm{F}} = 0$. Then

$$f^{\mathrm{F}}(v) = \sum_{k=-n}^{n-1} \hat{f}_k^{\mathrm{F}} \,\mathrm{e}^{-2\pi \mathrm{i} k v} = -\mathrm{i} \sum_{k=1}^{n-1} 2\hat{f}_k^{\mathrm{F}} \sin(2\pi k v).$$

Consequently, we have for $\hat{f}_k^{\mathrm{S}} = 2\hat{f}_k^{\mathrm{F}}$ that $f^{\mathrm{S}}(v) = \mathrm{i} f^{\mathrm{F}}(v)$. This time, equation (2.4) yields $\hat{g}_k = -\hat{g}_{-k}$ $(k = 1, \ldots, \sigma n - 1)$. Thus, for $l = 0, \ldots, \sigma n$ (2.7) becomes

$$\mathrm{i} g_l = \frac{\mathrm{i}}{2\sigma n} \sum_{k=-\sigma n}^{\sigma n - 1} \hat{g}_k \,\mathrm{e}^{-\pi \mathrm{i} k l/(\sigma n)} = \frac{1}{\sigma n} \sum_{k=1}^{\sigma n - 1} \hat{g}_k \sin\left(\frac{\pi k l}{\sigma n}\right) . \tag{2.10}$$

Note that $g_{2\sigma n r - l} = -g_l$ $(r \in \mathbb{Z})$. Finally, we compute as in (2.5) the sums

$$f^{\mathrm{S}}(v_j) = \mathrm{i} f^{\mathrm{F}}(v_j) \approx \mathrm{i} s(v_j) = \sum_{l = \lfloor 2\sigma n v_j \rfloor - (m-1)}^{\lceil 2\sigma n v_j \rceil + (m-1)} \mathrm{i} g_l \,\tilde{\varphi}\left(v_j - \frac{l}{2\sigma n}\right) . \tag{2.11}$$

In summary, we obtain the following algorithm for the fast computation of (2.9) with arithmetic complexity $\mathcal{O}(\sigma n \log(\sigma n) + mM)$:

Algorithm 2.2. (NFST)
Input: $n, M \in \mathbb{N}$, $\sigma > 1$, $\hat{f}_k^{\mathrm{S}} \in \mathbb{R}$ $(k = 1, \ldots, n-1)$, $v_j \in [0, 1/2]$ $(j = 1, \ldots, M)$.
Precomputation: $c_k(\tilde{\varphi})$ $(k = 1, \ldots, n-1)$,
$\tilde{\varphi}\left(v_j - \frac{l}{2\sigma n}\right)$ $(j = 1, \ldots, M; l = \lfloor 2\sigma n v_j \rfloor - (m-1), \ldots, \lceil 2\sigma n v_j \rceil + (m-1))$
1. For $k = 1, \ldots, n-1$ *compute* $\hat{g}_k := \dfrac{\hat{f}_k^{\mathrm{S}}}{2 c_k(\tilde{\varphi})}$ *and for* $k = 0, n, \ldots, \sigma n$ *set* $\hat{g}_k := 0$.

2. For $l = 0, \dots, \sigma n$ compute g_l by (2.10) by a fast DST–I *of length σn.*
3. For $j = 1, \dots, M$ compute $\mathrm{i}s(v_j)$ *by (2.11).*
Output: $\mathrm{i}s(v_j)$ *approximate values for $f^{\mathrm{S}}(v_j)$.*

Since we have derived the fast algorithms for the NDCT and NDST from the NFFT, the analysis of the approximation error is straightforward.

The NDCT and the NDST can be interpreted as matrix–vector multiplication with the matrices $\boldsymbol{C_v} := (\cos 2\pi k v_j)_{jk}$ and $\boldsymbol{S_v} := (\sin 2\pi k v_j)_{jk}$ with $\boldsymbol{v} := \left(v_j\right)_j$.
In a similar way as for the NFFT [15] we can develop fast algorithms for the computation of

$$\begin{aligned}\hat{h}_k^{\mathrm{C}} &:= \sum_{j=1}^{M} f_j^{\mathrm{C}} \cos(2\pi k v_j) \quad (k = 0, \dots, n),\\ \hat{h}_k^{\mathrm{S}} &:= \sum_{j=1}^{M} f_j^{\mathrm{S}} \sin(2\pi k v_j) \quad (k = 1, \dots, n-1),\end{aligned}$$

i. e. , for the matrix–vector multiplication with the transposed matrices $\boldsymbol{C}_{\boldsymbol{v}}^{\mathrm{T}}$ and $\boldsymbol{S}_{\boldsymbol{v}}^{\mathrm{T}}$. We refer to these algorithms as NFCT$^{\mathrm{T}}$ and NFST$^{\mathrm{T}}$, respectively.

3. Fast summation

In this section, we present an algorithm for the fast computation of sums of the form (1.1). We follow the approach presented in [14]. We suppose that $x_k, y_j \in [0, 1/2 - a/(2n)]$, where we specify the parameter a later. We regularize K near 0 and near the boundary 1/2 to obtain a smooth 1–periodic even kernel

$$K_{\mathrm{R}}(x) := \begin{cases} T_{\mathrm{I}}(x) & \text{if } x \in [0, \frac{a}{2n}),\\ T_{\mathrm{B}}(x) & \text{if } x \in (\frac{1}{2} - \frac{a}{2n}, \frac{1}{2}],\\ K(x) & \text{otherwise,}\end{cases} \tag{3.1}$$

where

$$T_{\mathrm{I}}(x) := \sum_{j=0}^{p-1} a_j^{\mathrm{I}} \cos \frac{\pi n j}{a} x, \qquad T_{\mathrm{B}}(x) := \sum_{j=0}^{p-1} a_j^{\mathrm{B}} \cos \frac{\pi n j}{a} \left(x - \frac{1}{2}\right)$$

and where $p \in \mathbb{N}$ and $n \le N$, are appropriate parameters, which will be chosen later. The coefficients a_j^{I} and a_j^{B} are determined by

$$T_{\mathrm{I}}^{(r)}\left(\tfrac{a}{2n}\right) = K^{(r)}\left(\tfrac{a}{2n}\right) \quad \text{and} \quad T_{\mathrm{B}}^{(r)}\left(\tfrac{1}{2} - \tfrac{a}{2n}\right) = K^{(r)}\left(\tfrac{1}{2} - \tfrac{a}{2n}\right) \quad (r = 0, \dots, p-1)$$

and can be computed by solving small linear systems of equations. Then we approximate K_{R} by the cosine series K_{RC} given by

$$K_{\mathrm{RC}}(x) = \sum_{l=0}^{n-1} b_l \cos(2\pi l x), \qquad \text{where} \qquad b_l := \frac{2\varepsilon_{n,l}}{n} \sum_{j=0}^{n} \varepsilon_{n,j} K_{\mathrm{R}}\left(\frac{j}{2n}\right) \cos\left(\frac{\pi l j}{n}\right). \tag{3.2}$$

Regarding that $K = K_{\mathrm{NE}} + K_{\mathrm{ER}} + K_{\mathrm{RC}}$ with $K_{\mathrm{NE}} := K - K_{\mathrm{R}}$, $K_{\mathrm{ER}} := K_{\mathrm{R}} - K_{\mathrm{RC}}$ and assuming that K_{ER} becomes sufficiently small, we approximate K by $K_{\mathrm{NE}} + K_{\mathrm{RC}}$ and consequently f by $\tilde{f}(y) := f_{\mathrm{NE}}(y) + f_{\mathrm{RC}}(y)$, where

$$f_{\mathrm{RC}}(y) := \sum_{k=1}^{N} \alpha_k K_{\mathrm{RC}}(y - x_k) \quad \text{and} \quad f_{\mathrm{NE}}(y) := \sum_{k=1}^{N} \alpha_k K_{\mathrm{NE}}(y - x_k).$$

These sums can be evaluated in an efficient way as follows: supposing that every interval of length $1/(2n)$ contains at most ν points x_k, the evaluation of f_{NE} at the M points y_j requires only $\mathcal{O}(a\nu M)$ arithmetic operations. We can rewrite f_{RC} as

$$\begin{aligned} f_{\mathrm{RC}}(y) &= \sum_{k=1}^{N} \alpha_k \sum_{l=0}^{n-1} b_l \cos\big(2\pi l(y - x_k)\big) \\ &= \sum_{k=1}^{N} \alpha_k \sum_{l=0}^{n-1} b_l \big(\cos(2\pi l y)\cos(2\pi l x_k) + \sin(2\pi l y)\sin(2\pi l x_k)\big). \end{aligned}$$

The expressions in the inner brackets of

$$f_{\mathrm{RC}}(y_j) = \sum_{l=0}^{n-1} b_l \left(\sum_{k=1}^{N} \alpha_k \cos(2\pi l x_k)\right) \cos(2\pi l y_j) + \sum_{l=1}^{n-1} b_l \left(\sum_{k=1}^{N} \alpha_k \sin(2\pi l x_k)\right) \sin(2\pi l y_j)$$

can be computed by $\mathrm{NFCT}^{\mathrm{T}}/\mathrm{NFST}^{\mathrm{T}}$. This will be followed by $2n$ multiplications with b_l and completed by NFCT/NFST to compute the outer summations.

In summary we obtain the following fast summation algorithm:

Algorithm 3.1.
Input: $N, n, M \in \mathbb{N}$, $\sigma > 1$, α_k, x_k $(k = 1\dots, N)$, y_j $(j = 1, \dots, M)$.
Precomputation: a_j^{I}, a_j^{B} $(j = 0, \dots, p-1)$, b_l $(l = 0, \dots, n-1)$ *by* (3.2) *and* (3.1),
$K_{\mathrm{NE}}(y_j - x_k)$ *for all* $j = 1, \dots, M$, $k \in I_{n,a}^{\mathrm{NE}}(y_j)$,
where $I_{n,a}^{\mathrm{NE}}(y_j) := \{k \in \{1, \dots, N\} : |y_j - x_k| < \frac{a}{2n}\}$.

1. *For* $l = 0, \dots, n-1$ *compute by* $\mathrm{NFCT}^{\mathrm{T}}$ *resp.* $\mathrm{NFST}^{\mathrm{T}}$

$$a_l^{\mathrm{C}} := \sum_{k=1}^{N} \alpha_k \cos(2\pi l x_k) \quad \text{and} \quad a_l^{\mathrm{S}} := \sum_{k=1}^{N} \alpha_k \sin(2\pi l x_k).$$

2. *For* $l = 0, \dots, n-1$ *compute the products* $d_l^{\mathrm{C}} := a_l^{\mathrm{C}} b_l$ *and* $d_l^{\mathrm{S}} := a_l^{\mathrm{S}} b_l$.
3. *For* $j = 1, \dots, M$ *compute by* NFCT *resp.* NFST

$$f^{\mathrm{C}}(y_j) := \sum_{l=0}^{n-1} d_l^{\mathrm{C}} \cos(2\pi l y_j) \quad \text{and} \quad f^{\mathrm{S}}(y_j) := \sum_{l=1}^{n-1} d_l^{\mathrm{S}} \sin(2\pi l y_j).$$

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

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

5. *For* $j = 1, \dots, M$ *compute the sums* $\tilde{f}(y_j) = f^{\mathrm{C}}(y_j) + f^{\mathrm{S}}(y_j) + f_{NE}(y_j)$.

Output: $\tilde{f}(y_j)$ *approximate values for* $f(y_j)$.

Our algorithm requires $\mathcal{O}(N \log N + M)$ arithmetic operations. Error estimates can be obtained in the same way as in [14]. It follows that $a = p$ and $n = N/2$ is a good choice for the parameters, where p is responsible for the approximation error (see Figure 2).

With $\boldsymbol{D}_K := \mathrm{diag}(\mathrm{b}_l)_l$ and $\boldsymbol{K}_{\mathrm{NE}} := \big(K_{\mathrm{NE}}(y_j - x_k)\big)_{jk}$ Algorithm 3.1 reads in matrix–form

$$\boldsymbol{K\alpha} \approx \left(\boldsymbol{C_y D_K C_x^{\mathrm{T}}} + \boldsymbol{S_y D_K S_x^{\mathrm{T}}} + \boldsymbol{K}_{\mathrm{NE}}\right)\boldsymbol{\alpha}.$$

4. Numerical Examples

In this section, we present numerical examples for the fast summation based on our NFCT/NFST. We always apply the NFCT/NFST with Kaiser-Bessel function φ and oversampling factor $\sigma = 2$. All algorithms were implemented in C and tested on an AMD Athlon(tm) XP1800+, 512MB RAM, SuSe–Linux, using double precision arithmetic. The DCT/DST implementation was taken from the "Numerical Recipes in C" [16]. The coefficients α_k were randomly distributed in $[0,1]$ and the nodes $x_k = y_k$ $(M = N)$ were randomly chosen in $[0, \frac{1}{2} - \frac{a}{2n}]$. In Algorithm 3.1 we have set $n = N/2$. Every entry presents the arithmetic mean of 20 runs of the algorithm.

Figure 4.1 compares the CPU time of our Algorithm 3.1 and of Algorithm 3.1 in [14] based on the NFFT for $K(x) = 1/|x|$ and $a = p = m = 4$ as function of the problem size N. The approximation error

$$E_\infty := \max_{j=1,\dots,M} \frac{|f(y_j) - \tilde{f}(y_j)|}{|f(y_j)|}$$

of the fast approximative algorithms for these computations is about 10^{-5}.

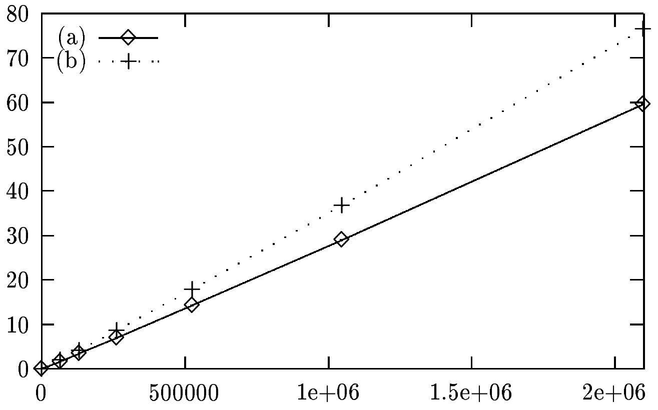

Figure 4.1. CPU time (in sec) of (a) Algorithm 3.1 and (b) Algorithm 3.1 in [14] in dependence on N.

Figure 4.2 presents the error E_∞ introduced by our Algorithm 3.1 as function of the parameters $a = p$ for the different kernels (1.2). We have chosen $N = 1024$ and $m = 8$ to ensure that the NFCT/NFST don't introduce additional errors.

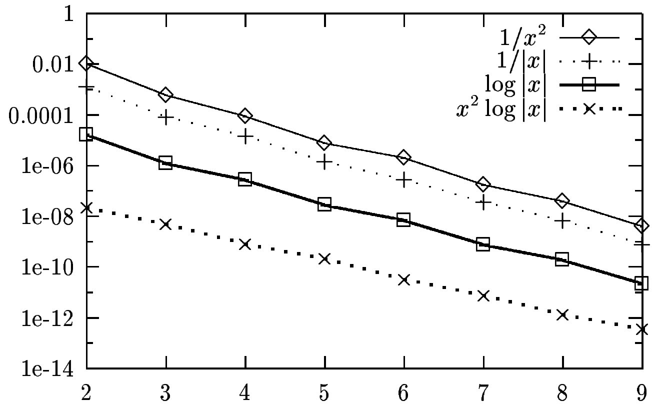

Figure 4.2. Error E_∞ of Algorithm 3.1 in dependence on $a = p$ for the kernels (1.2).

REFERENCES

1. R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: I. *J. Computers Math. Appl.*, 24:7 – 19, 1992.
2. G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2:363 – 381, 1995.
3. G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure and Appl. Math.*, 44:141 – 183, 1991.
4. G. Beylkin and R. Cramer. A multiresolution approach to regularization of singular operators and fast summation. SIAM *J. Sci. Comput.*, 24:81–117, 2002.
5. A. J. W. Duijndam and M. A. Schonewille. Nonuniform fast Fourier transform. *Geophysics*, 64:539 – 551, 1999.
6. A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. SIAM *J. Sci. Stat. Comput.*, 14:1368 – 1393, 1993.
7. K. Fourmont. Schnelle Fourier–Transformation bei nichtäquidistanten Gittern und tomographische Anwendungen. PhD thesis, University of Münster, 1999.
8. L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325 – 348, 1987.
9. W. Hackbusch. A sparse matrix arithmetic based on H–matrices, Part I: introduction to H–matrices. *Computing*, 62:89 – 108, 1999.
10. 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.
11. J. I. Jackson. Selection of a convolution function for Fourier inversion using gridding. *IEEE Trans. Medical Imaging*, 10:473 – 478, 1991.
12. S. Kunis and D. Potts. NFFT, Softwarepackage, C subroutine library. http://www.math.uni-luebeck.de/potts/nfft, 2002.
13. D. Potts. Fast algorithms for discrete polynomial transforms on arbitrary grids. *Linear Algebra Appl.*, to appear.
14. D. Potts and G. Steidl. Fast summation at nonequispaced knots by NFFTs. *Preprint MU-Luebeck, A02-02*, 2002.
15. 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.
16. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1992.
17. G. Steidl. A note on fast Fourier transforms for nonequispaced grids. *Adv. Comput. Math.*, 9:337 – 353, 1998.
18. B. Tian and Q. H. Liu. Nonuniform fast cosine transform and Chebyshev PSTD algorithm. *J. Electromagnet. Waves Appl*, 14:797 – 798, 2000.
19. E. E. Tyrtyshnikov. Mosaic–skeleton approximations. *Calcolo*, 33:47 – 57, 1996.