Chapter 1

Fast Fourier transforms for nonequispaced data: A tutorial

Daniel Potts
Gabriele Steidl
Manfred Tasche

ABSTRACT In this section, we consider approximate methods for the fast computation of multivariate discrete Fourier transforms for nonequispaced data (NDFT) *in the time domain and in the frequency domain. In particular, we are interested in the approximation error as function of the arithmetic complexity of the algorithm. We discuss the robustness of* NDFT*–algorithms with respect to roundoff errors and apply* NDFT*–algorithms for the fast computation of Bessel transforms.*

1.1 Introduction

Let $\Pi^d := [-\frac{1}{2}, \frac{1}{2})^d$ and $I_N := \{k \in \mathbb{Z}^d : -\frac{N}{2} \le k < \frac{N}{2}\}$, where the inequalities hold componentwise. For $x_k \in \Pi^d$, $v_j \in N\Pi^d$, and $f_k \in \mathbb{C}$, we are interested in the fast and robust computation of the *discrete Fourier transform for nonequispaced data* (NDFT)

$$f(v_j) = \sum_{k \in I_N} f_k \, \mathrm{e}^{-2\pi \mathrm{i} x_k v_j} \qquad (j \in I_M) \,. \tag{1.1}$$

For arbitrary nodes, the direct evaluation of (1.1) takes $\mathcal{O}(N^d M^d)$ arithmetical operations, too much for practical purposes. For equispaced nodes $x_k := \frac{k}{N}$ $(k \in I_N)$ and $v_j := j$ $(j \in I_N)$, the values $f(v_j)$ can be computed by the well-known *fast Fourier transform* (FFT) with only $\mathcal{O}(N^d \log N)$ arithmetical operations. However, the FFT requires sampling on an equally spaced grid, which represents a significant limitation for many applications. For example, most geographical data are sampled at individual observation points or by fast moving measuring devises along lines. Using the ACT method (*a*daptive weight, *c*onjugate gradient acceleration, *T*oeplitz matrices) [12] for the approximation of functions from scattered data [22], one has to solve a system of linear equations with a block Toeplitz matrix as system matrix. The entries of this Toeplitz matrix are of the form (1.1) and should be computed by efficient NDFT algorithms. Further applica-

tions of the NDFT range from frequency analysis of astronomical data [19] to modelling and imaging [4, 23].

Often either the nodes in time or in frequency domain lie on an equispaced grid, i.e., we have to compute

$$f(v_j) = \sum_{k \in I_N} f_k \, \mathrm{e}^{-2\pi \mathrm{i} k v_j / N} \qquad (j \in I_M) \tag{1.2}$$

or

$$h(k) := \sum_{j \in I_N} f_j \, \mathrm{e}^{-2\pi \mathrm{i} k v_j / N} \qquad (k \in I_M). \tag{1.3}$$

Several methods were introduced for the fast evaluation of (1.2) and (1.3). It may be useful to compare the different approaches by using the following matrix-vector notation: For the univariate setting $d = 1$ and $M = N$, the equations (1.2) can be written as

$$\hat{f} = A_N \, f \tag{1.4}$$

with

$$\hat{f} := (f(v_j))_{j=-N/2}^{N/2-1} \, , \; f := (f_k)_{k=-N/2}^{N/2-1} \, , \; A_N := \left(\mathrm{e}^{-2\pi \mathrm{i} k v_j / N} \right)_{j,k=-N/2}^{N/2-1} .$$

Note that (1.3) is simply the "transposed" version of (1.2), i.e.

$$\hat{h} = A_N^{\mathrm{T}} \, f \tag{1.5}$$

with $\hat{h} := (h(k))_{k=-N/2}^{N/2-1}$.

The different NDFT algorithms can be divided into three groups:

I. Based on local series expansions of the complex exponentials like Taylor series [2] or series of prolate spheroidal functions [18] algorithms were developed which are more efficient than the direct evaluation of the NDFT. In matrix-vector notation, the matrix A_N was approximated by the Hadamard (componentwise) product $\circ$ of the *classical Fourier matrix*

$$F_N := \left(\mathrm{e}^{-2\pi \mathrm{i} k j / N} \right)_{j,k=-N/2}^{N/2-1}$$

and a low rank matrix E, i.e.

$$A_N \approx F_N \circ E \, .$$

The rank of E determines the approximation error. Multiplication of the right-hand side with a vector means to perform rank(E) times an FFT of length N.

II. A quite different idea was introduced in [9] for the univariate case. Note that the idea can be generalized for "line settings". Using Lagrange interpolation, the matrix A_N can be *exactly* splitted as

$$A_N = D_1 C D_2 F_N \,, \tag{1.6}$$

where D_i $(i = 1, 2)$ are diagonal matrices and where

$$C := (1/\varphi((v_j/N) - (k/n))_{j,k=-N/2}^{N/2-1}$$

with a monotone decreasing function φ. A. Dutt and V. Rokhlin [9] originally have used $\varphi(x) := \tan(\pi x) - \mathrm{i}$. For real input data, a more efficient modified version with $\varphi(x) := 1/x$ was suggested in [6]. See Figure 1.1. The multiplication with C can be *approximately* realized by the *fast multipole method* [13].

In contrast to the other NDFT methods, the splitting (1.6) works also for the inverse matrix, i.e. $A_N^{-1} = \bar{F}_N D_3 C^{\mathrm{T}} D_4$, such that for suitably distributed nodes v_j, the same algorithm can be used for the inverse transform. For clustered nodes, the inverse approach does not work, since the entries of the diagonal matrices differ extremely. See [6].

III. The most efficient NDFT algorithms for the computation of (1.2) and (1.3) we have noticed the first time in [8, 4]. See also [5, 20]. In [25], we proposed a unified approach to the efficient computations of (1.2) and (1.3), which includes [8, 4]. In matrix-vector notation our approach to (1.2) can be written as

$$A_N \approx B F_n D \quad (n := \alpha N;\, \alpha > 1)\,, \tag{1.7}$$

with a modified diagonal matrix D and with a sparse matrix B with nonzero entries $\psi((v_j/N) - (l/n))$, where ψ approximates φ. The approaches in [8, 4] differ only by the choice of the window function φ. Now we have learned that this algorithm with several window functions φ is a remake of a *gridding algorithm*, which was known in image processing context years ago [23, 15, 19]. The gridding method is simply the transposed version of (1.7) for the efficient computation of (1.3).

However, the dependence of speed and accuracy of the algorithm on the choice of the oversampling factor α and the window width of φ was firstly investigated in [8, 4]. In [25] the error estimates from [8] were improved, which leads to citeria for the choice of the parameters of the algorithm.

In this paper, we repeat the unified approach (1.7) for the fast computation of (1.2) from [25] and show the relation to the gridding algorithm for the fast computation of (1.3). We will see that our approach is also closely connected with interpolation methods by "translates of one function" known from approximation theory. We include estimates of the approximation error.

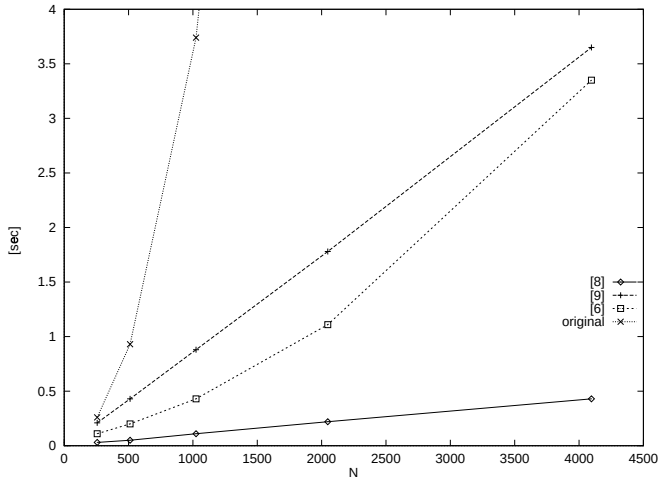

FIGURE 1.1. Time (in seconds) for straightforward computation of NDFT, NDFT–algorithm in [8], FMM–based NDFT–algorithms in [9] and [6] without consideration of initialization (precomputation) steps.

Section 1.3 contains an algorithm for the efficient computation of the general NDFT (1.1).

In Section 1.4, we demonstrate another advantage of Algorithm 1.1, namely its robustness with respect to roundoff errors, a feature which is well-known from the classical FFT [21, 26].

Finally, we apply the NDFT in a fast Bessel transform algorithm.

1.2 NDFT for nonequispaced data either in time or frequency domain

Let us begin with the computation of (1.2). Only for notational reasons, we restrict our attention to the case $M = N$. We have to evaluate the 1-periodic trigonometric polynomial

$$f(v) := \sum_{k \in I_N} f_k \, \mathrm{e}^{-2\pi \mathrm{i} k v} \tag{1.8}$$

at the nodes $w_j := v_j/N \in \Pi^d$ $(j \in I_N)$. We introduce the *oversampling factor* $\alpha > 1$ and set $n := \alpha N$.

Let φ be a 1-periodic function with uniformly convergent Fourier series. We approximate f by

$$s_1(v) := \sum_{l \in I_n} g_l \, \varphi(v - \frac{l}{n}). \tag{1.9}$$

Switching to the frequency domain, we obtain

$$s_1(v) = \sum_{k\in\mathbb{Z}^d} \hat{g}_k\, c_k(\varphi)\, \mathrm{e}^{-2\pi \mathrm{i} k v} \tag{1.10}$$

$$= \sum_{k\in I_n} \hat{g}_k\, c_k(\varphi)\, \mathrm{e}^{-2\pi \mathrm{i} k v} + \sum_{r\in\mathbb{Z}^d\setminus\{0\}} \sum_{k\in I_n} \hat{g}_k\, c_{k+nr}(\varphi)\, \mathrm{e}^{-2\pi \mathrm{i}(k+nr)v}$$

with

$$\hat{g}_k := \sum_{l\in I_n} g_l\, \mathrm{e}^{2\pi \mathrm{i} k l/n}, \tag{1.11}$$

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

If the Fourier coefficients $c_k(\varphi)$ become sufficiently small for $k \in \mathbb{Z}^d\setminus I_n$ and if $c_k(\varphi) \neq 0$ for $k \in I_N$, then we suggest by comparing (1.8) with (1.10) to set

$$\hat{g}_k := \begin{cases} f_k/c_k(\varphi) & k\in I_N, \\ 0 & k\in I_n\setminus I_N. \end{cases} \tag{1.12}$$

Now the values g_l can be obtained from (1.11) by the reduced inverse d-variate FFT of size αN. If φ is also well-localized in time domain such that it can be approximated by a 1-periodic function ψ with $\operatorname{supp}\psi \cap \Pi^d \subseteq \frac{2m}{n}\Pi^d$ $(2m \ll n)$, then

$$f(w_j) \approx s_1(w_j) \approx s(w_j) = \sum_{l\in I_{n,m}(w_j)} g_l\, \psi(w_j - \frac{l}{n}) \tag{1.13}$$

with $I_{n,m}(w_j) := \{l \in I_n : nw_j - m \leq l \leq nw_j + m\}$. For fixed $w_j \in \Pi^d$, the above sum contains at most $(2m+2)^d$ nonzero summands.

In summary, we obtain the following algorithm for the fast computation of (1.2) with $\mathcal{O}((\alpha N)^d \log(\alpha N))$ arithmetical operations:

Algorithm 1.1. (Fast computation of NDFT (1.2))

Input: $N \in \mathbb{N}$, $\alpha > 1$, $n := \alpha N$, $w_j \in \Pi^d$, $f_k \in \mathbb{C}$ $(j, k \in I_N)$.

0. *Precompute* $c_k(\varphi)$ $(k \in I_N)$, $\psi(w_j - \frac{l}{n})$ $(j \in I_N, l \in I_{n,m}(w_j))$.
1. *Form* $\hat{g}_k := f_k/c_k(\varphi)$ $(k \in I_N)$.
2. *Compute by d–variate* FFT

$$g_l := n^{-d} \sum_{k\in I_N} \hat{g}_k\, \mathrm{e}^{-2\pi \mathrm{i} k l/n} \quad (l \in I_n).$$

3. *Set*

$$s(w_j) := \sum_{l\in I_{n,m}(w_j)} g_l\, \psi(w_j - \frac{l}{n}) \quad (j \in I_N)\,.$$

Output: $s(w_j)$ *approximate value of* $f(w_j)$ $(j \in I_N)$.

In the univariate setting $d = 1$, Algorithm 1.1 reads in matrix-vector notation as

$$A_N f \approx B F_n D f \,,$$

where B denotes the sparse matrix

$$B := \left(\psi(w_j - \frac{l}{n}) \right)_{j=-N/2,l=-n/2}^{N/2-1\,,n/2-1} \tag{1.14}$$

and where

$$D := \left(O \,|\, \mathrm{diag}(1/(n\, c_k(\varphi)))_{k=-N/2}^{N/2-1} \,|\, O \right)^{\mathrm{T}} \tag{1.15}$$

with $(N,(n-N)/2)$-zero matrices O.

If only few nodes v_j differ from the equispaced nodes j, then the approximating function s_1 of f can be alternatively constructed by requiring exact interpolation at the nodes j/n, i.e. $f(j/n) = s_1(j/n)$ $(j \in I_n)$. As consequence we have only to replace $c_k(\varphi)$ in step 1 of Algorithm 1.1 by the discrete Fourier sum of $(\varphi(l/n))_{l \in I_n}$. The approximation of a function f by an interpolating function s_1 which is a linear combinations of translates of one given function φ on a grid was considered in various papers in approximation theory. See for example [16] and also [4].

Let us turn to the gridding algorithm for the fast computation of (1.3). It seems to be useful to introduce the gridding idea by avoiding the convenient notation with delta distributions. In short: For

$$g(x) := \sum_{j \in I_N} f_j \, \varphi(x + w_j),$$

we obtain that

$$c_k(g) = \sum_{j \in I_N} f_j \, \mathrm{e}^{-2\pi \mathrm{i} k w_j} \, c_k(\varphi) = h(k)\, c_k(\varphi) \quad (k \in \mathbb{Z}^d)$$

such that $h(k)$ in (1.3) can be computed, if $c_k(g)$ is known. Now we determine

$$c_k(g) = \int_{\Pi^d} \sum_{j \in I_N} f_j \, \varphi(x + w_j) \, \mathrm{e}^{2\pi \mathrm{i} k x} \, \mathrm{d}x$$

approximately by the trapezoidal rule

$$c_k(g) \approx \frac{1}{n^d} \sum_{l \in I_n} \sum_{j \in I_N} f_j \, \varphi(w_j - \frac{l}{n}) \, \mathrm{e}^{-2\pi \mathrm{i} k l / n} \,,$$

which introduces an *aliasing error*. Furthermore, we replace φ by its truncated version ψ, which introduces a *truncation error*. Obviously, this results in a "transposed" Algorithm 1.1, which reads for $d = 1$ in matrix–vector notation

$$A_N^{\mathrm{T}} \approx D^{\mathrm{T}} F_n B^{\mathrm{T}} .$$

For $l \in I_n$ we introduce the index set $J_{m,n}(l) := \{j \in I_N : l - m \le nw_j \le l + m\}$.

Algorithm 1.2. (Fast computation of NDFT (1.3))

Input: $N \in \mathbb{N}$, $\alpha > 1$, $n := \alpha N$, $w_j \in \Pi^d$, $f_k \in \mathbb{C}$ $(j, k \in I_N)$.

0. *Precompute* $c_k(\varphi)$ $(k \in I_N)$, $\psi(w_j - \frac{l}{n})$ $(l \in I_n, j \in J_{n,m}(l))$.

1. *Set*

$$\tilde{g}_l := \sum_{j \in J_{n,m}(l)} f_j \, \psi(w_j - \frac{l}{n}) \quad (l \in I_n) .$$

2. *Compute by d–variate* FFT

$$\tilde{c}_k(g) := n^{-d} \sum_{l \in I_n} \tilde{g}_l \, \mathrm{e}^{-2\pi \mathrm{i} kl/n} \quad (k \in I_N).$$

3. *Form* $\tilde{h}(k) := \tilde{c}_k(g)/c_k(\varphi)$ $(k \in I_N)$.

Output: $\tilde{h}(k)$ *approximate value of* $h(k)$ $(k \in I_N)$.

Both algorithms introduce the same approximation errors. By (1.13), the error splits as follows:

$$E(w_j) := |f(w_j) - s(w_j)| \le E_a(w_j) + E_t(w_j)$$

with $E_a(w_j) := |f(w_j) - s_1(w_j)|$ and $E_t(w_j) := |s_1(w_j) - s(w_j)|$. Note that $E_a(w_j)$ and $E_t(w_j)$ are the *aliasing error* and the *truncation error* introduced by Algorithm 1.2, respectively. Let

$$E_\infty := \max_{j \in I_N} E(w_j) , \quad ||f||_1 := \sum_{k \in I_N} |f_k|.$$

Then we obtain immediately by (1.8), (1.10) – (1.12) that

$$\begin{aligned} E_a(w_j) &\le \sum_{k \in I_N} |f_k| \sum_{r \in \mathbb{Z}^d \setminus \{0\}} \left| \frac{c_{k+nr}(\varphi)}{c_k(\varphi)} \right| \\ &\le ||f||_1 \max_{k \in I_N} \sum_{r \in \mathbb{Z}^d \setminus \{0\}} \left| \frac{c_{k+nr}(\varphi)}{c_k(\varphi)} \right| \end{aligned} \tag{1.16}$$

and by (1.9), (1.11) – (1.13) that

$$E_t(w_j) \le ||f||_1 \, n^{-d} \, (\max_{k \in I_N} |c_k(\varphi)|^{-1}) \sum_{l \in I_n} |\varphi(w_j - \frac{l}{n}) - \psi(w_j - \frac{l}{n})|.$$

To fill these error estimates with life, we consider special functions φ in the case $d = 1$. In [25], we have proved

Theorem 1.1. *Let* (1.2) *be computed by Algorithm* 1.1 *with the dilated, periodized Gaussian bell*

$$\varphi(v) := (\pi b)^{-1/2} \sum_{r \in \mathbb{Z}} \mathrm{e}^{-(n(v+r))^2/b}$$

and with the truncated version of φ,

$$\psi(v) := (\pi b)^{-1/2} \sum_{r \in \mathbb{Z}} \mathrm{e}^{-(n(v+r))^2/b} \chi_{[-m,m]}(n(v+r)) .$$

Here $\chi_{[-m,m]}$ *denotes the characteristic function of* $[-m, m]$. *Let* $\alpha > 1$ *and* $1 \le b \le \frac{2\alpha m}{(2\alpha-1)\pi}$. *Then* $c_k(\varphi) = \frac{1}{n} \mathrm{e}^{-(\frac{\pi k}{n})^2 b}$ *and*

$$\begin{aligned}
E_a(w_j) &\le ||f||_1 \, \mathrm{e}^{-b\pi^2(1-\frac{1}{\alpha})} \left(1 + \frac{\alpha}{(2\alpha-1)b\pi^2} + \mathrm{e}^{-2b\pi^2/\alpha} \left(1 + \frac{\alpha}{(2\alpha+1)b\pi^2}\right)\right) , \\
E_t(w_j) &\le ||f||_1 \, \frac{2}{\sqrt{\pi b}} \, (1 + \frac{b}{2m}) \, \mathrm{e}^{-b\pi^2((\frac{m}{b\pi})^2 - (\frac{1}{2\alpha})^2)} , \\
E_\infty &\le 4 \, ||f||_1 \, \mathrm{e}^{-b\pi^2(1-\frac{1}{\alpha})} .
\end{aligned}$$

The approximation error decreases with increasing b. Therefore

$$b = \frac{2\alpha m}{(2\alpha - 1)\pi} \tag{1.17}$$

provides a good choice for b as function of α and m. For the above parameter b, the approximation error and the truncation error are balanced. Since $2/\sqrt{b\pi}$ contributes also to the decay of $E_t(w_j)$, we expect that the optimal parameter b lies slightly above the value in (1.17). A. Dutt and V. Rokhlin [8] does not give a criteria for the choice of the parameter b. They determine b by trial computations. Moreover, our error estimate is sharper than the estimate

$$E_\infty \le (b + 9) \, ||f||_1 \, \mathrm{e}^{-b\pi^2(1-\frac{1}{\alpha^2})/4}$$

in [8].

For oversampling factor $\alpha = 2$, the new paper [19] of J. Pelt contains extensive numerical computations to determine the optimal parameter b as function of m.

More recently, A. J. W. Duijndam and M. A. Schoneville [7] have evaluated the RMS-error of Algorithm 1.2 with the Gaussian pulse φ. They found the same optimal parameter b as in (1.17), which confirms our result.

Estimates for the multivariate setting were given in [11, 7].

Figure 1.2 illustrates our theoretical results by numerical tests. The tests were implemented in MATLAB. The C–programs for Algorithm 1.1 were taken from [6] and were included by cmex. The programs were tested on a Sun SPARCstation 20 in double precision.

Let $N = 2^t$. As example we consider the computation of

$$f(w_j) = \sum_{k=0}^{2^t-1} \mathrm{e}^{-2\pi \mathrm{i} k w_j} \quad (j \in I_{2^t}) \tag{1.18}$$

with uniformly distributed random nodes $w_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$. The exact vector $\hat{f} := (f(w_j))_{j=-2^{t-1}}^{2^{t-1}-1}$ is given by

$$f(w_j) := \frac{\mathrm{e}^{-2\pi \mathrm{i} w_j \left(2^t-1\right)} - \mathrm{e}^{2\pi \mathrm{i} w_j}}{1 - \mathrm{e}^{2\pi \mathrm{i} w_j}} .$$

By $\tilde{f}$ we denote the result of Algorithm 1.1 with φ, ψ as in Theorem 1.1, $m := 15, \alpha := 2$ and b as in (1.17). Let

$$E_{\mathrm{NDFT}}(t) := \log_{10}(\|\tilde{f} - \hat{f}\|_2)/\|\hat{f}\|_2) .$$

Figure 1.2 (left) shows the error $E_{\mathrm{NDFT}}(10)$ for 10 numerical tests and for various $m = 6, \ldots, 20$. Figure 1.2 (right) presents the computation time (in seconds) for the cascade summation of (1.2) and for Algorithm 1.1. Note that for $t = 13$ the fast Algorithm 1.1 requires for less than one second while the direct method based on cascade summation takes more than 3 hours.

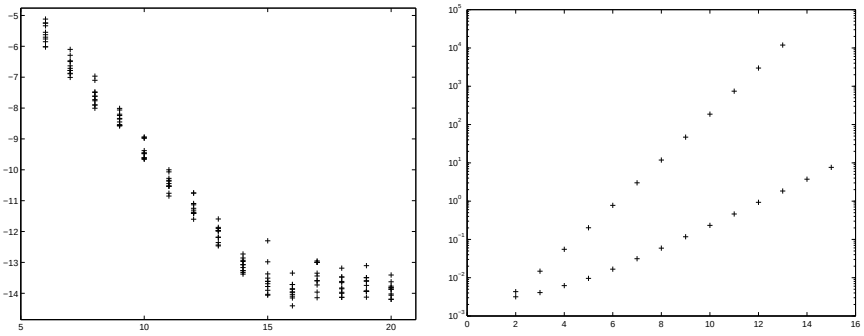

FIGURE 1.2. Left: $(m, E_{\mathrm{NDFT}}(10))$ **for** $m = 6, \ldots, 20$**. Right: Computation time in seconds for** $t = 2, \ldots, 15$**.**

Next, we consider centered cardinal B-splines as window functions, which are in contrast to the Gaussian kernel of compact support such that we have

no truncation error. Algorithm 1.2 with centered cardinal B-splines and with a modified third step was originally introduced in [4]. The difference in the third step was motivated by the tight frequency localization of Lemarié–Battle scaling functions which arise in wavelet theory. See also [25]. With M_{2m} we denote the centered cardinal B-spline of order $2m$.

Theorem 1.2. *Let* (1.2) *be computed by Algorithm* 1.1 *with the dilated, periodized, centered cardinal* B*-spline*

$$\varphi(v) := \sum_{r \in \mathbb{Z}} M_{2m}(n(v+r)) \quad (m \geq 1)$$

of order $2m$ *and let* $\psi = \varphi$. *Then*

$$c_k(\varphi) = \begin{cases} \dfrac{1}{n} & k = 0\,, \\ \dfrac{1}{n} \left(\dfrac{\sin(k\pi/n)}{k\pi/n} \right)^{2m} & otherwise \end{cases}$$

and for $\alpha > 1$ *and* $0 \leq \eta \leq 4m/3$,

$$E_\infty \leq \frac{4m}{2m-1} \left(\frac{N}{2} \right)^{-\eta} (2\alpha - 1)^{-2m} \, |f|_{\eta,1} \,.$$

Here $|f|_{\eta,1}$ *denotes the Sobolev-like seminorm* $|f|_{\eta,1} := \sum\limits_{k=-N/2}^{N/2-1} |f_k| \, |k|^\eta$.

Proof: 1. For $0 < u < 1$ and $m \in \mathbb{N}$, it holds that

$$\begin{aligned} \sum_{r \in \mathbb{Z} \setminus \{0\}} \left(\frac{u}{u+r} \right)^{2m} &\leq 2 \left(\frac{u}{u-1} \right)^{2m} + 2 \sum_{r=2}^{\infty} \left(\frac{u}{u-r} \right)^{2m} \\ &\leq 2 \left(\frac{u}{u-1} \right)^{2m} + 2 \int_1^\infty \left(\frac{u}{u-x} \right)^{2m} \mathrm{d}x \\ &\leq \frac{4m}{2m-1} \left(\frac{u}{u-1} \right)^{2m} . \end{aligned}$$

2. Since $c_k(\varphi) \neq 0$ and

$$\begin{aligned} n\, c_{k+rn}(\varphi) &= \left(\frac{\sin(k\pi/n)}{k\pi/n + r\pi} \right)^{2m} = \left(\frac{\sin(k\pi/n)}{k\pi/n} \right)^{2m} \left(\frac{k\pi/n}{k\pi/n + r\pi} \right)^{2m} \\ &= n\, c_k(\varphi) \left(\frac{k/n}{k/n + r} \right)^{2m} , \end{aligned}$$

we obtain by (1.16) that

$$E_\infty \leq \frac{4m}{2m-1} \sum_{\substack{k=-N/2 \\ (k \neq 0)}}^{N/2-1} |f_k| \, |k|^\eta \, n^{-\eta} \, \frac{|k/n|^{2m-\eta}}{(k/n - \operatorname{sgn}(k))^{2m}} \,.$$

The functions $h_+(u) := u^{2m-\eta}/(u-1)^{2m}$ and $h_-(u) := u^{2m-\eta}/(u+1)^{2m}$ are monotonically increasing for $u \in [0, \frac{1}{2}]$ and $\eta \leq 4m/3$. Thus, since $|k| \leq N/2$, we obtain the assertion. □

By Theorem 1.2, we have for $\eta = 0$ that

$$E_\infty \leq \frac{4m}{2m-1} (2\alpha - 1)^{-2m} ||f||_1 .$$

If the values f_k are Fourier coefficients of a smooth function such that $|f|_{\eta,1}$ does not increase with $N \to \infty$, then the estimates with larger values of η may be useful. For example, we obtain for $\alpha = \eta = m = 2$ by Theorem 1.2 that

$$E_\infty \leq \left(\frac{2}{3}\right)^5 N^{-2} |f|_{2,1} .$$

Multivariate estimates were given in [11, 4].

In various papers, other window functions than the Gaussian pulse or centered cardinal B–splines were considered:

- prolate spheroidal functions and Kaiser-Bessel functions [15],
- Gaussian pulse tapered with a Hanning window [7],
- Gaussian kernels combined with sinc kernels [19],
- special optimized windows [15, 7].

Numerical results demonstrate that the approximation error (as function of the speed of the computation) can be further improved by using these functions.

1.3 NDFT for nonequispaced data in time and frequency domain

The following algorithm for the fast computation of the general NDFT (1.1) (with $M = N$) turns out to be a combination of the gridding approach to (1.3) and of our approach to (1.2). Moreover, in order to apply again an aliasing formula, a periodization procedure becomes necessary. In order to form a-periodic functions, a parameter a comes into the play.

By $\varphi_1 \in L_2(\mathbb{R}^d)$ we denote a sufficiently smooth function with Fourier transform

$$\hat{\varphi}_1(v) := \int_{\mathbb{R}^d} \varphi_1(x) \mathrm{e}^{-2\pi \mathrm{i} v x} \mathrm{d}x \neq 0$$

for all $v \in N\Pi^d$. Then we obtain for

$$G(x) := \sum_{k \in I_N} f_k \, \varphi_1(x - x_k)$$

that

$$\hat{G}(v) = \sum_{k \in I_N} f_k \, \mathrm{e}^{-2\pi \mathrm{i} x_k v} \, \hat{\varphi}_1(v)$$

and consequently

$$f(v_j) = \frac{\hat{G}(v_j)}{\hat{\varphi}_1(v_j)} \qquad (j \in I_N).$$

Thus, given $\hat{\varphi}_1(v_j)$, it remains to compute $\hat{G}(v_j)$.
Let $n_1 := \alpha_1 N$ $(\alpha_1 > 1)$, $m_1 \in \mathbb{N}$ $(2m_1 \ll n_1)$. We introduce the parameter $a := 1 + \frac{2m_1}{n_1}$ and rewrite $\hat{G}(v)$ as

$$\begin{aligned} \hat{G}(v) &= \sum_{k \in I_N} f_k \int_{\mathbb{R}^d} \varphi_1(x - x_k) \, \mathrm{e}^{-2\pi \mathrm{i} x v} \, \mathrm{d}x \\ &= \sum_{k \in I_N} f_k \int_{a\Pi^d} \sum_{r \in \mathbb{Z}^d} \varphi_1(x + ra - x_k) \, \mathrm{e}^{-2\pi \mathrm{i}(x+ra)v} \, \mathrm{d}x. \end{aligned} \tag{1.19}$$

Discretization of the integral by the rectangular rule leads to

$$\hat{G}(v) \approx S_1(v) = \sum_{k \in I_N} f_k \, n_1^{-d} \sum_{j \in I_{an_1}} \sum_{r \in \mathbb{Z}^d} \varphi_1\left(\frac{j}{n_1} + ra - x_k\right) \mathrm{e}^{-2\pi \mathrm{i}\left(\frac{j}{n_1} + ra\right)v}. \tag{1.20}$$

Now we approximate the function φ_1 by a function ψ_1 with $\operatorname{supp} \psi_1 \subseteq \frac{2m_1}{n_1}\Pi^d$. Then the third sum in (1.20) contains only one nonzero summand for $r = 0$. Changing the order of the summation, we obtain

$$S_1(v) \approx S_2(v) := n_1^{-d} \sum_{t \in I_{an_1}} \left(\sum_{k \in I_N} f_k \, \psi_1\left(\frac{t}{n_1} - x_k\right) \right) \mathrm{e}^{-2\pi \mathrm{i} t v / n_1}.$$

After the computation of the inner sum for all $t \in I_{an_1}$, we arrive at computation problem (1.2), which can be solved in a fast way by Algorithm 1.1, where the corresponding window functions and parameters are indicated by the index 2. We summarize:

Algorithm 1.3. (Fast computation of NDFT (1.1))

Input: $N \in \mathbb{N}$, $\alpha_1 > 1$, $\alpha_2 > 1$, $n_1 := \alpha_1 N$, $a := 1 + \frac{2m_1}{n_1}$, $n_2 := \alpha_1 \alpha_2 a N$,
$x_k \in \Pi^d$, $v_j \in N\Pi^d$, $f_k \in \mathbb{C}$ $(j, k \in I_N)$.

0. *Precompute* $\hat{\varphi}_1(v_j)$ $(j \in I_N)$, $\psi_1(\frac{t}{n_1} - x_k)$ $(k \in I_N; t \in I_{an_1, m_1}(x_k))$,
$c_t(\varphi_2)$ $(t \in I_{an_1})$, $\psi_2(\frac{v_j}{n_1} - \frac{l}{n_2})$ $(j \in I_N, l \in I_{n_2, m_2}(\frac{v_j}{n_1}))$.

1. *Compute*
$$F(t) := \sum_{k \in I_N} f_k \, \psi_1(\frac{t}{n_1} - x_k) \quad (t \in I_{an_1}) .$$
2. *Form* $\hat{g}_t := F(t)/c_t(\varphi_2) \quad (t \in I_{an_1})$.
3. *Compute by d–variate* FFT
$$g_l := n_2^{-d} \sum_{t \in I_{an_1}} \hat{g}_t \, \mathrm{e}^{-2\pi \mathrm{i} t l / n_2} \quad (l \in I_{n_2}).$$
4. *Form*
$$s(v_j) := n_1^{-d} \sum_{l \in I_{n_2, m_2}(v_j/n_1)} g_l \, \psi_2(\frac{v_j}{n_1} - \frac{l}{n_2}) \quad (j \in I_N) .$$
5. *Form* $S(v_j) := s(v_j)/\hat{\varphi}_1(v_j) \ (j \in I_N)$.

Output: $S(v_j)$ *approximate value of* $f(v_j) \ (j \in I_N)$.

The Algorithm 1.3 requires $\mathcal{O}((\alpha_1 \alpha_2 a N)^d \log(\alpha_1 \alpha_2 a N))$ arithmetical operations. The approximation error is given by

$$E(v_j) \quad \leq \quad E_1(v_j) + E_2(v_j) + E_3(v_j)$$

with $E_1(v_j) := |f(v_j) - \frac{S_1(v_j)}{\hat{\varphi}_1(v_j)}|$, $E_2(v_j) := |\frac{S_1(v_j) - S_2(v_j)}{\hat{\varphi}_1(v_j)}|$, and $E_3(v_j) := |\frac{S_2(v_j) - s(v_j)}{\hat{\varphi}_1(v_j)}|$. The error $E_3(v_j)$ is the product of the error of Algorithm 1.1 and $|\hat{\varphi}_1(v_j)|^{-1}$. The cut-off error $E_2(v_j)$ behaves like the truncation error in Algorithm 1.1. The error $E_1(v_j)$ arising from the discretization of the integral (1.19) can be estimated by an aliasing argument [11].

The following Table 1.1 (see also [10, 11]) contains the relative approximation error
$$\max_{j \in I_N} |f(v_j) - S(v_j)| / \max_{j \in I_N} |f(v_j)| \ .$$
introduced by Algorithm 1.3 for tensor products of Gaussian bells φ_1 and φ_2, for $N := 128$, $m = m_1 = m_2$, $a := \frac{N}{N-m}$, $\alpha_1 := \frac{2}{a}$, $\alpha_2 := 2$ and for randomly distributed nodes x_j and v_k/N in Π^2. By the choice of α_1, α_2 and a, the main effort of the algorithm consists in the bivariate FFT of size $4N = 1024$. The third column of Table 1.1 contains the error of Algorithm 1.3, if we simply set $a = 1$ and $\alpha_1 = \alpha_2 = 2$. This change of parameters influences only the first step of the algorithm. (A similar error occurs, if we consider ψ_1 as 1-periodic function.) Table 1.1 demonstrates the significance of the parameter a.

1.4 Roundoff errors

Beyond the approximation error, the numerical computation of Algorithm 1.1 involves roundoff errors. In this section, we will see that similar to the

m	$a = \frac{N}{N-m}$	$a = 1$
5	5.96608e-06	0.0180850
7	5.44728e-08	0.0318376
9	1.07677e-09	0.0541445
11	3.31061e-11	0.0906439
13	1.26030e-12	0.1507300
15	2.16694e-13	0.2500920

TABLE 1.1. Approximation error of Algorithm 1.3 **for** $N := 128$, $a := \frac{N}{N-m}$, $\alpha_1 := \frac{2}{a}$, $\alpha_2 := 2$ **and for** $a := 1$, $\alpha_1 = \alpha_2 := 2$, **respectively.**

classical FFT [21, 14], our algorithm is robust with respect to roundoff errors. In the following, we use the standard model of real floating point arithmetic (see [14], p. 44): For arbitrary $\xi, \eta \in \mathbb{R}$ and any operation $\circ \in \{+, -, \times, /\}$ the exact value $\xi \circ \eta$ and the computed value $\mathrm{fl}(\xi \circ \eta)$ are related by

$$\mathrm{fl}(\xi \circ \eta) = (\xi \circ \eta)(1+\delta) \quad (|\delta| \le u),$$

where u denotes the *unit roundoff* (or machine precision). In the case of single precision (24 bits for the mantissa (with 1 sign bit), 8 bits for the exponent), we have

$$u = 2^{-24} \approx 5.96 \times 10^{-8}$$

and for double precision (53 bits for the mantissa (with 1 sign bit), 11 bits for the exponent)

$$u = 2^{-53} \approx 1.11 \times 10^{-16}.$$

Since complex arithmetic is implemented using real arithmetic, the complex floating point arithmetic is a consequence of the corresponding real arithmetic (see [14], pp. 78 – 80): For arbitrary $\xi, \eta \in \mathbb{C}$, we have

$$\mathrm{fl}(\xi + \eta) = (\xi + \eta)(1+\delta) \quad (|\delta| \le u), \tag{1.21}$$

$$\mathrm{fl}(\xi\,\eta) = \xi\,\eta\,(1+\delta) \quad (|\delta| \le \frac{2\sqrt{2}\,u}{1-2u}). \tag{1.22}$$

In particular, if $\xi \in \mathbb{R} \cup \mathrm{i}\mathbb{R}$ and $\eta \in \mathbb{C}$, then

$$\mathrm{fl}(\xi\eta) = \xi\eta\,(1+\delta) \quad (|\delta| \le u). \tag{1.23}$$

To simplify the following error analysis, we assume that all entries of A_N, B, F_n and D were precomputed exactly. Moreover, we restrict our attention to real input vectors $f \in \mathbb{R}^N$. If we form $\hat{f} := A_N f$ by conventional multiplication and cascade summation (see [14], p. 70), then we obtain by (1.21) and (1.23) the componentwise estimate

$$|\mathrm{fl}(\hat{f})_j - \hat{f}_j| \le \frac{(\lceil \log_2 N \rceil + 1)\,u}{1 - (\lceil \log_2 N \rceil + 1)u}\,||f||_1$$

and by taking the Euclidean norm

$$\|\mathrm{fl}(\hat{f}) - \hat{f}\|_2 \leq \left(u\, N\, (\lceil \log_2 N \rceil + 1) + \mathcal{O}(u^2) \right) \|f\|_2 .$$

In particular, we have for $\hat{f} := F_N f$ that

$$\|\mathrm{fl}(F_N f) - F_N f\|_2 \leq \left(u\, N\, (\lceil \log_2 N \rceil + 1) + \mathcal{O}(u^2) \right) \|f\|_2 .$$

If we compute $\hat{f} = F_N f$ ($f \in \mathbb{R}^N$, N power of 2) by the radix–2 Cooley–Tukey FFT, then, following the lines of the proof in [26] and using (1.21) – (1.22), the roundoff error estimate can be improved by the factor $\sqrt{N}$, more precisely

$$\|\mathrm{fl}(F_N f) - F_N f\|_2 \leq \left(u\, (4 + \sqrt{2}) \sqrt{N} \log_2 N + \mathcal{O}(u^2) \right) \|f\|_2 . \quad (1.24)$$

The following theorem states that the roundoff error introduced by Algorithm 1.1 can be estimated as the FFT error in (1.24) up to a constant factor, which depends on m and α.

Theorem 1.3. *Let $m, N \in \mathbb{N}$ and let $n := \alpha N$ ($\alpha > 1$) be a power of 2 with $2m \ll n$. Let h be a nonnegative real–valued even function, which decreases monotonically for $x \geq 0$, and let*

$$\begin{aligned} \varphi(x) &:= \sum_{r \in \mathbb{Z}} h(n(x+r)) , \\ \psi(x) &:= \sum_{r \in \mathbb{Z}} (\chi_{[-m,m]} h)(n(x+r)) . \end{aligned}$$

Suppose that φ has a uniform convergent Fourier expansion with monotone decreasing absolute values of Fourier coefficients

$$c_k(\varphi) = \frac{1}{n}\, \hat{h}\left(\frac{2\pi k}{n}\right) \quad (k \in \mathbb{Z}).$$

Let the nodes $w_j := \frac{v_j}{N} \in [-\frac{1}{2}, \frac{1}{2})$, $w_j \pm 1$ ($j \in I_N$) be distributed such that each "window" $\left[-\frac{m}{n} + \frac{l}{n}, \frac{m}{n} + \frac{l}{n}\right)$ ($l \in I_n$) contains at most γ/α nodes. If (1.2) is computed by Algorithm 1.1 with the above functions φ, ψ, i.e.,

$$\tilde{f} := B\, F_n\, D\, f \quad (f \in \mathbb{R}^N) ,$$

where $D \in \mathbb{C}^{n,N}$ and $B \in \mathbb{R}^{N,n}$ are determined by (1.14) – (1.15), then the roundoff error of Algorithm 1.1 can be estimated by

$$\|\mathrm{fl}(\hat{f}) - \tilde{f}\|_2 \leq \beta \sqrt{\gamma} \left(u(4 + \sqrt{2}) \sqrt{N} \left(\log_2 N + \log_2 \alpha + \frac{2m+1}{4+\sqrt{2}}\right) + \mathcal{O}(u^2) \right) \|f\|_2$$

with

$$\beta := \frac{(h^2(0) + \|h\|_{L_2}^2)^{1/2}}{|\hat{h}(\pi/\alpha)|} .$$

Let us call an algorithm for the computation of the NDFT (1.2) *robust*, if for all $f \in \mathbb{R}^N$ there exists a positive constant k_N with $k_N\, u \ll 1$ such that

$$||\mathrm{fl}(\tilde{f}) - \tilde{f}||_2 \leq (k_N u + \mathcal{O}(u^2))\, ||f||_2 .$$

Then, by Theorem 1.3, Algorithm 1.1 is robust.

Proof: 1. First, we estimate the spectral norms of D and B ([14], p. 120). By assumption and by (1.15), we see immediately that

$$\|D\|_2 \;=\; \max_{k \in I_N} |n\, c_k(\varphi)|^{-1} \;=\; (n\, |c_{N/2}(\varphi)|)^{-1} \;=\; |\hat{h}(\pi/\alpha)|^{-1} . \tag{1.25}$$

Since ψ is even and monotonically decreasing for $x \geq 0$, it is easy to check the integral estimate

$$\frac{1}{n} \sum_{l \in I_n} \psi^2(w_j - \frac{l}{n}) \leq \frac{1}{n}\, \psi^2(0) \;+\; \int_{-1/2}^{1/2} \psi^2(x)\, \mathrm{d}x .$$

Then it follows by definition of ψ that

$$\begin{aligned} \sum_{l \in I_n} \psi^2(w_j - \frac{l}{n}) \;&\leq\; h^2(0) + n \int_{-m/n}^{m/n} h^2(nx)\, \mathrm{d}x \\ &\leq\; h^2(0) + ||h||_{L_2}^2 . \end{aligned} \tag{1.26}$$

By definition (1.14) of the sparse matrix

$$B = (b_{j,k})_{j=-N/2,k=-n/2}^{N/2-1,n/2-1} , \quad b_{j,k} := \psi(w_j - \frac{k}{n}) ,$$

we get for the j–th component $(By)_j$ of By $(y = (y_k)_{k=-n/2}^{n/2-1} \in \mathbb{C}^n)$ that

$$\begin{aligned} |(By)_j|^2 \;&\leq\; \left(\sum_{r=1}^{2m} |b_{j,k_r}|\, |y_{k_r}| \right)^2 \qquad (b_{j,k_r} > 0,\; k_r \in \{-n/2, \ldots, n/2-1\}) \\ &\leq\; \left(\sum_{r=1}^{2m} b_{j,k_r}^2 \right) \left(\sum_{r=1}^{2m} y_{k_r}^2 \right) . \end{aligned}$$

By (1.26), we have

$$\sum_{r=1}^{2m} b_{j,k_r}^2 \;\leq\; \sum_{k \in I_n} \psi(w_j - \frac{k}{n})^2 \;\leq\; h^2(0) + ||h||_{L_2}^2$$

such that

$$|(By)_j|^2 \leq (h^2(0) + ||h||_{L_2}^2) \sum_{r=1}^{2m} y_{k_r}^2 . \tag{1.27}$$

By assumption, each "window" $[-\frac{m}{n}+\frac{l}{n}, \frac{m}{n}+\frac{l}{n})$ $(l \in I_n)$ contains at most γ/α nodes $w_j, w_j \pm 1$ $(j \in I_N)$. Therefore each column of B contains at most γ/α nonzero entries such that by (1.27)

$$\|By\|_2^2 = \sum_{j=-N/2}^{N/2-1} |(By)_j|^2 \le \frac{\gamma}{\alpha}\,(h^2(0) + \|h\|_{L_2}^2)\,\|y\|_2^2$$

and consequently

$$\|B\|_2 \le \sqrt{\frac{\gamma}{\alpha}}\,(h^2(0) + \|h\|_{L_2}^2)^{1/2} =: \tilde{\beta}\,. \tag{1.28}$$

2. Next, it is easy to check that by (1.23) and (1.25)

$$\|\mathrm{fl}(Df) - Df\|_2 \le u\,|\hat{h}(\pi/\alpha)|^{-1}\,\|f\|_2\,. \tag{1.29}$$

From (1.25) it follows that

$$\|\mathrm{fl}(Df)\|_2 \le \|\mathrm{fl}(Df) - Df\|_2 + \|Df\|_2 \le |\hat{h}(\pi/\alpha)|^{-1}(u+1)\,\|f\|_2\,. \tag{1.30}$$

3. Set $\hat{y} := \mathrm{fl}(F_n(\mathrm{fl}(Df))$ and $y := F_n Df$. Then we can estimate

$$\|\hat{y} - y\|_2 \le \|\hat{y} - F_n(\mathrm{fl}(Df))\|_2 + \|F_n(\mathrm{fl}(Df)) - F_n Df\|_2$$

such that by (1.24), (1.29) and (1.30)

$$\begin{aligned}\|\hat{y} - y\|_2 &\le \left(u(4+\sqrt{2})\sqrt{n}\log_2 n + \mathcal{O}(u^2)\right)\|\mathrm{fl}(Df)\|_2 \\ &\quad + \sqrt{n}\,\|\mathrm{fl}(Df) - Df\|_2 \\ &\le |\hat{h}(\pi/\alpha)|^{-1}\left(u(4+\sqrt{2})\sqrt{n}\log_2 n + \sqrt{n}u + \mathcal{O}(u^2)\right)\|f\|_2\,. \end{aligned}\tag{1.31}$$

4. Finally, we consider the error between $\mathrm{fl}(\tilde{f}) := \mathrm{fl}(B\hat{y})$ and $\tilde{f} := By$. By (1.28) and (1.31), we obtain

$$\begin{aligned}\|\mathrm{fl}(\tilde{f}) - \tilde{f}\|_2 &\le \|\mathrm{fl}(B\hat{y}) - B\hat{y}\|_2 + \|B(\hat{y} - y)\|_2 \\ &\le \|\mathrm{fl}(B\hat{y}) - B\hat{y}\|_2 + \tilde{\beta}\,|\hat{h}(\pi/\alpha)|^{-1} \times \\ &\quad \left(u(4+\sqrt{2})\sqrt{n}\log_2 n + \sqrt{n}u + \mathcal{O}(u^2)\right)\|f\|_2\,. \end{aligned}\tag{1.32}$$

By (1.21), (1.23) and (1.14), it follows from [14], p. 76 that

$$|\mathrm{fl}(B\hat{y}) - B\hat{y}| \le \frac{2mu}{1-2mu}\,B\,|\hat{y}|$$

with $|\hat{y}| := (|\hat{y}_k|)_{k=-n/2}^{n/2-1}$ and consequently by (1.28) that

$$\begin{aligned}\|\mathrm{fl}(B\hat{y}) - B\hat{y}\|_2 &\le \frac{2mu}{1-2mu}\,\|B\|_2\,\|\hat{y}\|_2 \\ &\le \left(2m\tilde{\beta}\,u + \mathcal{O}(u^2)\right)\|\hat{y}\|_2\,. \end{aligned}$$

By (1.30) and (1.25), we obtain

$$\begin{aligned} \|\hat{y}\|_2 &\leq \|\hat{y} - y\|_2 + \|y\|_2 = \|F_n D f\|_2 + \mathcal{O}(u)\, \|f\|_2 \\ &\leq \left(\sqrt{n}\, |\hat{h}(\pi/\alpha)|^{-1} + \mathcal{O}(u)\right) \|f\|_2 . \end{aligned}$$

Thus

$$\|\mathrm{fl}(B\hat{y}) - B\hat{y}\|_2 \leq \left(2m\,\tilde{\beta}\,\sqrt{n}\,|\hat{h}(\pi/\alpha)|^{-1}\,u + \mathcal{O}(u^2)\right) \|f\|_2 . \tag{1.33}$$

Together with (1.32) this yields the assertion. □

Note that $\gamma \approx 2m$ for "uniformly distributed" nodes w_j.

Finally, we confirm our theoretical results by numerical experiments. We use the same setting as in Section 1.2. Further, let $\tilde{f}_{\mathrm{C}} \in \mathbb{C}^{2^t}$ denote the vector, which was evaluated by cascade summation of the right–hand side of (1.18), and let

$$E_{\mathrm{C}}(t) := \log_t(\|\tilde{f}_{\mathrm{C}} - \hat{f}\|_2)/\|\hat{f}\|_2) .$$

Figure 1.3 (left) shows the error $E_{\mathrm{C}}(t)$ for 10 numerical tests with various random nodes w_j as function of the transform length $N = 2^t$. For comparison, Figure 1.3 (right) presents the corresponding error $E_{\mathrm{NDFT}}(t)$ introduced by NDFT–Algorithm 1.1.

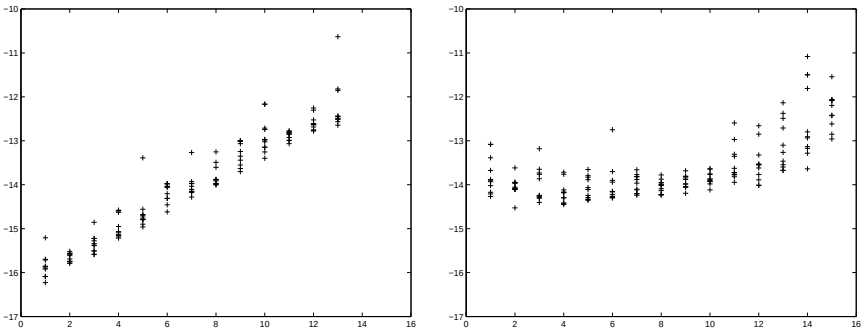

FIGURE 1.3. Left: $(t, E_{\mathrm{C}}(t))$ **for** $t = 1, \ldots, 13$. **Right:** $(t, E_{\mathrm{NDFT}}(t))$ **for** $t = 1, \ldots, 15$.

1.5 Fast Bessel transform

In this section, we apply the NDFT for a fast Bessel transform. We are interested in a fast algorithm for the computation of

$$\hat{f}(n) := \int_0^\infty f(x)\, J_n(x)\, \mathrm{d}x \quad (n \in \mathbb{N}_0)\,, \tag{1.34}$$

where

$$J_n(x) := \sum_{k=0}^{\infty} (-1)^k \frac{(x/2)^{n+2k}}{k!\,(n+k)!} \quad (x \in \mathbb{R})$$

is the *Bessel function (of first kind) of order* n. Let f be a real–valued, compactly supported function with supp $f \subseteq [0, a]$ $(0 < a < \infty)$. Using the formula (see [1], p. 361)

$$\mathrm{e}^{-\mathrm{i}q\cos t} = 2 \sum_{k=0}^{\infty}{}' (-1)^k\, J_{2k}(q)\, \cos(2kt) + 2\mathrm{i} \sum_{k=0}^{\infty} (-1)^k\, J_{2k+1}(q)\, \cos(2k+1)t$$

with

$$\sum_{k=0}^{\infty}{}' a_k := \frac{1}{2} + \sum_{k=1}^{\infty} a_k\,,$$

we obtain for $x \in [-1, 1]$ and $q \in \mathbb{R}$ that

$$\mathrm{e}^{-\mathrm{i}qx} = 2 \sum_{k=0}^{\infty}{}' (-1)^k\, J_{2k}(q)\, T_{2k}(x) - 2\mathrm{i} \sum_{k=0}^{\infty} (-1)^k\, J_{2k+1}(q)\, T_{2k+1}(x)\,.$$

Now multiplication with $f(q)$ and integration with respect to q yields

$$\begin{aligned} h(x) &:= \int_0^a f(q)\, \mathrm{e}^{-\mathrm{i}qx}\, dq \qquad (1.35)\\ &= 2 \sum_{k=0}^{\infty}{}' (-1)^k\, \hat{f}(2k)\, T_{2k}(x) - 2\mathrm{i} \sum_{k=0}^{\infty} (-1)^k\, \hat{f}(2k+1)\, T_{2k+1}(x)\,. \end{aligned}$$

Note that

$$(\mathrm{Re}\, h)(x) = (\mathrm{Re}\, h)(-x)\,, \quad (\mathrm{Im}\, h)(x) = -(\mathrm{Im}\, h)(-x)\,.$$

Finally, orthogonality of the Chebyshev polynomials $T_k(x)$

$$\int_{-1}^{1} w(x)\, T_k(x)\, T_l(x)\, \mathrm{d}x = \begin{cases} \pi & \text{if } k = l = 0,\\ \pi/2 & \text{if } k = l \neq 0,\\ 0 & \text{if } k \neq l \end{cases}$$

with $w(x) := (1-x^2)^{-1/2}$ implies

$$\hat{f}(2k) = \frac{(-1)^k}{\pi} \int_{-1}^{1} w(x)\,(\operatorname{Re} h)(x)\,T_{2k}(x)\,\mathrm{d}x, \tag{1.36}$$

$$\hat{f}(2k+1) = \frac{(-1)^{k+1}}{\pi} \int_{-1}^{1} w(x)\,(\operatorname{Im} h)(x)\,T_{2k+1}(x)\,\mathrm{d}x. \tag{1.37}$$

We approximate the integrals (1.35) – (1.37) by the following quadrature formulas: First, we compute (1.35) by the Simpson rule on the Chebyshev grid $\{x_l := \cos\frac{(2l+1)\pi}{2N};\ l = 0,\dots,N/2-1\}$, i.e.,

$$h(x_l) \approx h_l := \frac{a}{3N} \sum_{j=0}^{N} \omega_j f(\frac{aj}{N})\,\mathrm{e}^{-\mathrm{i}jax_l/N} \quad (l = 0,\dots,N/2-1)\,. \tag{1.38}$$

with

$$w_j := \begin{cases} 1 & j = 0, N\,, \\ 4 & j = 2k-1 \quad (k = 0,\dots,N/2)\,, \\ 2 & j = 2k \quad\ \ (k = 0,\dots,N/2-1)\,. \end{cases}$$

We realize the NDFT (1.38) by Algorithm 1.1 in $\mathcal{O}(N\log N)$ arithmetical operations. Again, we choose φ, ψ as in Theorem 1.1 with $m := 15, \alpha := 2$ and b in (1.17). Next, we compute (1.36) and (1.37) by the trapezoidal rule, which results for $k = 0,\dots,N/2-1$ in

$$\hat{f}(2k) \approx \tilde{f}_{2k} := \frac{(-1)^k}{N} \sum_{l=0}^{N-1} (\operatorname{Re} h_l)\,\cos\frac{2k(2l+1)\pi}{2N}\,, \tag{1.39}$$

$$\hat{f}(2k+1) \approx \tilde{f}_{2k+1} := \frac{(-1)^{k+1}}{N} \sum_{l=0}^{N-1} (\operatorname{Im} h_l)\,\cos\frac{(2k+1)(2l+1)\pi}{2N}\,. \tag{1.40}$$

For the fast computation of (1.39) and (1.40) in $\mathcal{O}(N\log N)$ arithmetical operations we use the *fast cosine transform of type* II (DCT–II) proposed in [24, 3]. In summary, we obtain the following algorithm for the fast computation of the Bessel transform in $\mathcal{O}(N\log N)$ arithmetical operations:

Algorithm 1.4. (Fast Computation of Bessel transform (1.34))

Input: $N := 2^t$, $f(ja/N) \in \mathbb{R}$ $(j = 0,\dots,N)$.

1. *Compute* h_l *by* (1.38) *with Algorithm* 1.1.
2. *Set* $\operatorname{Re}(h_{l+N/2-1}) := \operatorname{Re}(h_{N/2-l})$, $\operatorname{Im}(h_{l+N/2-1}) := -\operatorname{Im}(h_{N/2-l})$ $(l = 1,\dots,N/2)$.
3. *Compute* (1.39) – (1.40) *by fast* DCT–II.

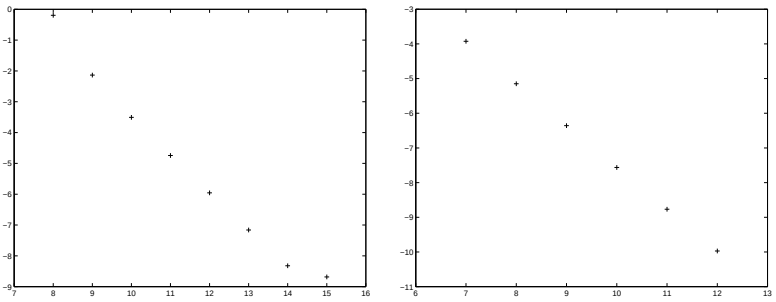

FIGURE 1.4. Left: $(t, E(t))$ **for** $t = 8, \ldots, 15$ **and** f **in (1.41). Right:** $(t, E(t))$ **for** $t = 7, \ldots, 12$ **and** f **in (1.43).**

Output: $\tilde{f}_n$ *approximate value of* $\hat{f}(n)$ $(n = 0, \ldots, N-1)$.

We test Algorithm 1.4 by the following examples:

1. Consider

$$f(x) := \mathrm{e}^{-\lambda x}\, \chi_{[0,a]}(x) \tag{1.41}$$

with $a := 2000$ and $\lambda = 0.01$. By [17], p. 772 we have

$$\int_0^\infty \mathrm{e}^{-\lambda x} J_n(x)\, \mathrm{d}x = (1+\lambda^2)^{-1/2} \left(\sqrt{1+\lambda^2} + \lambda\right)^{-n} ,$$

such that we can approximate $\hat{f}(k)$ by

$$\hat{f}(k) \approx \hat{f}_k := (1+\lambda^2)^{-1/2} \left(\sqrt{1+\lambda^2} + \lambda\right)^{-k} .$$

By $\tilde{f}$, we denote the vector computed by Algorithm 1.4. Figure 1.4 (left) shows the relative error

$$E(t) := \log_{10} \left(\|\tilde{f} - \hat{f}\|_2\right) / \|\hat{f}\|_2 . \tag{1.42}$$

2. Let

$$f(x) := \chi_{[0,a]}(x) \tag{1.43}$$

with $a := 100$. Then we have by [1], p. 480 that

$$\int_0^a J_n(x)\, \mathrm{d}x = 2 \sum_{l=0}^{\infty} J_{2l+n+1}(a) .$$

We choose M such that $J_{2k+n+1}(100) < 10^{-16}$ and set

$$\hat{f}(k) \approx \hat{f}_k = 2 \sum_{l=0}^{M} J_{2l+k+1}(100) .$$

Figure 1.4 (right) shows the corresponding error (1.42).

References

[1] M. Abramowitz and J. A. Stegun, *Handbook of Mathematical Functions*, Dover Publ., New York, 1972.

[2] C. Anderson and M. D. Dahleh, Rapid computation of the discrete Fourier transform, *SIAM J. Sci. Comput.* 17, 913 – 919, 1996.

[3] G. Baszenski and M. Tasche, Fast polynomial multiplication and convolutions related to the discrete cosine transform, *Linear Algebra Appl.* 252, 1 – 25, 1997.

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

[5] A. Brandt, Multilevel computations of integral transforms and particle interactions with oscillatory kernels, *Comput. Phys. Comm.* 65, 24 – 38, 1991.

[6] J. O. Droese, Verfahren zur schnellen Fourier–Transformation mit nichtäquidistanten Knoten, Master Thesis, TH Darmstadt, 1996.

[7] A. J. W. Duijndam and M. A. Schonewille, Nonuniform fast Fourier transform, Preprint, Univ. Delft, 1998.

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

[9] A. Dutt and V. Rokhlin, Fast Fourier transforms for nonequispaced data II, *Appl. Comput. Harmon. Anal.* 2, 85 – 100, 1995.

[10] B. Elbel, Mehrdimensionale Fouriertransformation für nicht äquidistante Daten, Master Thesis, TU Darmstadt, 1998.

[11] B. Elbel and G. Steidl, Fast Fourier transform for nonequispaced data, In: Approximation Theory IX, C. K. Chui and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1998.

[12] H. G. Feichtinger, K. H. Gröchenig and T. Strohmer, Efficient numerical methods for nonuniform sampling theory, *Numer. Math.* 69, 423 – 440, 1995.

[13] L. Greengard and V. Rokhlin, A fast algorithm for particel simulation, *J. Comput. Phys.* 73, 325 – 348, 1987.

[14] N. J. Higham, *Accuracy and Stability of Numerical Algorithms,* SIAM, Philadelphia, 1996.

[15] J. I. Jackson, Selection of a convolution function for Fourier inbersion using gridding, *IEEE Trans. Medical Imaging* 10, 473-478, 1991.

[16] K. Jetter and J. Stöckler, Algorithms for cardinal interpolation using box splines and radial basis functions, *Numer. Math.* 60, 97 – 114, 1991.

[17] M. A. Lawrentjew und B. W. Schabat, *Methoden der komplexen Funktionentheorie,* Deutscher Verlag der Wissenschaften, Berlin, 1967.

[18] G. Newsam, Private communication.

[19] J. Pelt, Fast computation of trigonometric sums with application to frequency analysis of astronomical data, Preprint, 1997.

[20] W. H. Press and G. B. Rybicki, Fast algorithms for spectral analysis of unevenly sampled data, *Astrophys. J.* 338, 227 – 280, 1989.

[21] G. U. Ramos, Roundoff error analysis of the fast Fourier transform, *Math. Comp.* 25, 757 – 768, 1971.

[22] M. Rauth and T. Strohmer, Smooth approximation of potential fields from noisy scattered data, Preprint, Univ. Vienna, 1997.

[23] Sramek, R. A. and F. R. Schwab, Imaging, in *Astronomical Society of the Pacific Conference,* Vol. 6, R. Perley, F. R. Schwab and A. Bridle (eds.), 1988, 117 – 138.

[24] G. Steidl, Fast radix–p discrete cosine transform, *Appl. Algebra Engrg. Comm. Comput.* 3, 39 – 46, 1992.

[25] G. Steidl, A note on fast Fourier transforms for nonequispaced grids, *Adv. Comp. Math.* (in print).

[26] P. Y. Yalamov, Improvements of some bounds on the stability of fast Helmholtz solvers, Preprint, Univ. Rousse, 1998.