

# Fast Algorithms for Discrete Polynomial Transforms on arbitrary Grids

Daniel Potts \*

## Abstract

Consider the Vandermonde-like matrix  $\mathbf{P} := (P_k(x_{M,l}))_{l,k=0}^{M,N}$ , where the polynomials  $P_k$  satisfy a three-term recurrence relation and  $x_{M,l} \in [-1, 1]$  are arbitrary nodes. If  $P_k$  are the Chebyshev polynomials  $T_k$ , then  $\mathbf{P}$  coincides with  $\mathbf{A} := (T_k(x_{M,l}))_{l=0,k=0}^{M,N}$ . This paper presents a fast algorithm for the computation of the matrix-vector product  $\mathbf{P}a$  in  $\mathcal{O}(N \log^2 N)$  arithmetical operations. The algorithm divides into a fast transform which replaces  $\mathbf{P}a$  with  $\mathbf{A}\tilde{a}$  and a fast cosine transform on *arbitrary* nodes (NDCT). Since the first part of the algorithm was considered in [20], we focus on approximative algorithms for the NDCT. Our considerations are completed by numerical tests.

1991 *Mathematics Subject Classification*. Primary 65T50, 41A15 41A30 42A16

*Key words and phrases*. Discrete polynomial transform, Vandermonde-like matrix, fast cosine transform, fast Fourier transform, fast polynomial transform, Chebyshev knots, nonequispaced grids, *B*-splines, Gaussian bells

## 1 Introduction

Let  $\omega$  be a non-negative, integrable weight function with

$$\int_{-1}^1 \omega(x) dx > 0 \tag{1.1}$$

and let  $L_\omega^2([-1, 1])$  be the weighted Hilbert space of all measurable functions  $f : [-1, 1] \rightarrow \mathbb{R}$  with the inner product and norm defined by

$$\langle f, g \rangle := \frac{2}{\pi} \int_{-1}^1 \omega(x) f(x) g(x) dx, \quad \|f\|_2 := \langle f, f \rangle^{1/2}.$$

Let  $\{P_n\}_{n \in \mathbb{N}_0}$  be a sequence of orthogonal polynomials  $P_n \in \Pi_n$  with respect to  $\langle \cdot, \cdot \rangle$ . Here  $\Pi_n$  denotes the set of polynomials of degree  $\leq n$ . Then every  $P \in \Pi_N$  can be represented as

$$P = \sum_{k=0}^N \frac{\langle P, P_k \rangle}{\|P_k\|_2^2} P_k, \tag{1.2}$$

---

\*Medical University of Lübeck, Institute of Mathematics, Wallstr. 40, D-23560 Lübeck. E-mail: potts@math.mu-luebeck.de. Research supported in part by the EU Research Training Network MINGLE, RTN1-1999-00212.

where  $\langle P, P_k \rangle$  can be computed by a convenient quadrature rule.

Let  $M, N \in \mathbb{N}$  with  $M \geq N$  be given. We are interested in an efficient solution of the following two problems:

1. Given  $f_k \in \mathbb{R}$  ( $k = 0, \dots, N$ ) compute the *discrete polynomial transform*  $\text{DPT}(N+1, M+1)$ :  $\mathbb{R}^{N+1} \rightarrow \mathbb{R}^{M+1}$  on arbitrary nodes  $x_{M,j} \in [-1, 1]$  ( $j = 0, \dots, M$ ) defined by

$$\hat{f}_j := \sum_{k=0}^N f_k P_k(x_{M,j}) \quad (j = 0, \dots, M). \quad (1.3)$$

The transform matrix  $\mathbf{P} := (P_k(x_{M,j}))_{j,k=0}^{M,N}$  is called *Vandermonde-like matrix*.

2. Given  $f_j \in \mathbb{R}$  ( $j = 0, \dots, M$ ) compute the *transposed discrete polynomial transform*  $\text{DPT}^T(M+1, N+1)$ :  $\mathbb{R}^{M+1} \rightarrow \mathbb{R}^{N+1}$  defined by

$$h(k) := \sum_{j=0}^M f_j P_k(x_{M,j}) \quad (k = 0, \dots, N). \quad (1.4)$$

The first problem addresses the evaluation of polynomials  $P \in \Pi_N$  given in the form (1.2) at the nodes  $x_{M,j}$ . The second problem is concerned with the approximation of the Fourier coefficients of  $P \in \Pi_N$  by a convenient quadrature rule. A fast algorithm for (1.3) implies the factorization of the transform matrix  $\mathbf{P}$  into a product of sparse matrices. Consequently, once a fast algorithm for (1.3) is known, a fast algorithm for the “transposed” problem (1.4) with the transform matrix  $\mathbf{P}^T$  is also available by transposing the sparse matrix product. Therefore, we restrict our attention to the fast computation of (1.3).

A straightforward idea for the fast solution of (1.3) with arbitrary orthogonal polynomials  $P_n$  is to realize a basis exchange from  $\{P_n\}_{n=0}^N$  to  $\{T_n\}_{n=0}^N$  followed by a fast cosine transform on arbitrary nodes. A fast algorithm with  $\mathcal{O}(N \log^2 N)$  arithmetical operations for the basis exchange was described in detail in [20].

The main objective of this paper is to derive an efficient approximative algorithm for the NDCT similar to the fast Fourier transforms on nonequidistant data [22] and to combine this algorithm with the basis exchange in [20]. To our knowledge this is the first algorithm for the fast computation of (1.3) and (1.4).

This paper is organized as follows: In Section 2 we derive three different algorithms for the fast realization of the NDCT. In §2.1 we present a fast algorithm for the NDCT based on the fast multipole method. In §2.2 we transfer the general efficient method [22] for the fast evaluation of trigonometric polynomials at nonequispaced nodes to (algebraic) polynomials. In Section 3 we develop the algorithm for the DPT. Numerical results are presented in Section 4.

## 2 Discrete Cosine transform on arbitrary nodes

In this section we consider the DPT for the special Chebyshev weight  $\omega(x) := (1 - x^2)^{-1/2}$ . By  $T_k := \cos(k \arccos)$ , we denote the Chebyshev polynomials of first kind. Note that  $\arccos : [-1, 1] \rightarrow [0, \pi]$  is the inverse function of  $\cos$  restricted to  $[0, \pi]$ . As known, the Chebyshev

polynomials form a complete orthogonal system in  $L^2_\omega([-1, 1])$ . For  $m, n \in \mathbb{N}_0$  we have

$$\langle T_m, T_n \rangle = \begin{cases} 2 & m = n = 0, \\ 1 & m = n > 0, \\ 0 & m \neq n. \end{cases}$$

The DPT( $N + 1, M + 1$ ) with respect to the Chebyshev polynomials can be written as

$$P(x_{M,j}) = \sum_{k=0}^N f_k T_k(x_{M,j}) \quad (j = 0, \dots, M) \quad (2.1)$$

with  $f_k \in \mathbb{R}$ . We call this transform NDCT( $M + 1, N + 1$ ).

The DPT<sup>T</sup>( $M + 1, N + 1$ ) for the Chebyshev polynomials means, the fast evaluation of

$$h(k) := \sum_{j=0}^M f_j T_k(x_{M,j}) \quad (k = 0, \dots, N) \quad (2.2)$$

with  $f_j \in \mathbb{R}$  at the nodes  $x_{M,j}$  ( $j = 0, \dots, M$ ). We call this transform NDCT<sup>T</sup>( $M + 1, N + 1$ ). For Gauss–Chebyshev nodes  $h_{N,l} := \cos(l\pi/N)$  ( $l = 0, \dots, N$ ), the values  $f(h_{N,l})$  ( $l = 0, \dots, N$ ) can be computed by the well-known discrete cosine transform of type-I (DCT-I) with only  $\mathcal{O}(N \log N)$  flops. We use the corresponding DCT-I matrices

$$\mathbf{C}_{N+1}^I := \left( \varepsilon_{N,k} \cos \frac{jk\pi}{N} \right)_{j,k=0}^N, \quad \varepsilon_{N,k} := \begin{cases} 1/2 & k = 0, N \\ 1 & k = 1, \dots, N-1 \end{cases},$$

which satisfy

$$\mathbf{C}_{N+1}^I \mathbf{C}_{N+1}^I = \frac{N}{2} \mathbf{I}_{N+1}. \quad (2.3)$$

There exist implementations of algorithms for the vector multiplication with the above cosine matrix, for example a C-implementation based on [23, 2].

However, the DCT requires sampling on a special grid, which represents a significant limitation for many applications. Unfortunately, for arbitrary nodes  $x_{M,l} \in [-1, 1]$  ( $l = 0, \dots, M$ ), the direct evaluation of (2.1) takes  $\mathcal{O}(NM)$  arithmetical operations, too much for practical purposes. In this section we derive three different methods for the fast computation of (2.1) and (2.2).

1. We use an approximative method for the fast computation of discrete Fourier transform for nonequispaced data (NDFT) (see [22, 21]), i.e. we simply compute

$$\tilde{f}(x) := \operatorname{Re} \left( \sum_{k=0}^N f_k e^{-ikx} \right) \quad (2.4)$$

at the nodes  $y_{M,l} := \arccos(x_{M,l})$  and obtain  $f(x_{M,l}) = \tilde{f}(y_{M,l})$ .

Efficient NDFT algorithms for the computation of the NDFT were given for example in [12, 3]. In [22], Steidl proposed a unified approach to the efficient computations of NDFT, which includes [12, 3]. Recently several algorithms for the efficient evaluation of trigonometric polynomials at irregularly spaced points were presented and analyzed in [24, 21].

2. In §2.1 we use the celebrated fast multipole method (FMM) in order to compute (2.1). A fast Fourier transform for nonequispaced data based on fast multipole methods was given in [13].

3. The main tools for the fast algorithms for the NDFT are the Fourier transforms and the approximation by translates. By using the Chebyshev transform and the Chebyshev shift we are able to transfer the concept of the NDFT to fast algorithms for the NDCT. This approach is described in §2.2.

## 2.1 NDCT based on the fast multipole method

In this section we describe a method for the fast computation of (2.1) by using the fast multipole method (FMM). The FMM is a tool that has been widely applied in several fields with great success. We use an algorithm for the evaluation of potential fields on the line, which was given in [11]. Recently a method for this problem, which is roughly twice as fast as the used algorithm was given in [25]. The algorithm requires  $\mathcal{O}(N \log(1/\epsilon))$  arithmetical operations, where  $\epsilon$  is the precision of computations and  $N$  is the number of nodes.

First we represent the matrix  $\mathbf{A} = (T_k(x_{M,l}))_{l=0,k=0}^{M,N}$  by a product of diagonal matrices, a Cosine matrix and a Cauchy matrix. The problem of matrix–vector multiplication by a Cauchy matrix with complexity less than  $\mathcal{O}(NM)$  is known as *Trummer's problem* (see also [4]). For special Cauchy matrices, i.e for special points  $x_{M,l}$ , a representation of Cauchy matrices using different sine and cosine transforms was presented in [15]. The FMM provides a fast and approximative algorithm for the matrix–vector multiplication with arbitrary Cauchy matrices.

Let  $N$  be a power of 2. Instead of computing  $f(x_{M,l})$  ( $l = 0, \dots, M$ ) we compute

$$f(h_{N,j}) = \sum_{k=0}^N f_k T_k(h_{N,j}) \quad (j = 0, \dots, N)$$

at the Gauss–Chebyshev nodes  $h_{N,j} = \cos(j\pi/N)$  by DCT–I in  $\mathcal{O}(N \log N)$  flops. Using Lagrange interpolation we rewrite  $f$  in the form

$$f(x) = \sum_{j=0}^N f(h_{N,j}) \frac{w(x)}{(x - h_{N,j})w'(h_{N,j})}$$

with  $w(x) = 2^{N-1} \prod_{k=0}^N (x - h_{N,k})$ . Let  $U_{N-1}$  be the Chebyshev polynomials of second kind given by

$$U_{N-1}(x) = \frac{\sin(N \arccos(x))}{\sqrt{1-x^2}} \quad x \in (-1, 1).$$

Since  $y_{M,k} = \arccos(x_{M,k})$ , we see by  $w(x) = (1-x^2)U_{N-1}(x)$  that

$$\begin{aligned} w(x_{M,k}) &= -\sin(y_{M,k}) \sin(N y_{M,k}), \\ w'(h_{N,j}) &= \frac{(-1)^j}{\varepsilon_{N,j}} N \end{aligned}$$

and finally

$$f(x_{M,k}) = \sin(y_{M,k}) \sin(N y_{M,k}) \sum_{j=0}^N \varepsilon_{N,j} \frac{(-1)^{j+1}}{N} f(h_{N,j}) \frac{1}{x_{M,k} - h_{N,j}} \quad (k = 0, \dots, M).$$

**Algorithm 2.1** (*Fast computation of NDCT using FMM*)

*Input:*  $N \in \mathbb{N}$ ,  $x_{M,l} \in [-1, 1]$  ( $l = 0, \dots, M$ ),  $f_k \in \mathbb{R}$  ( $k = 0, \dots, N$ ).

1. Compute by DCT-I ( $N + 1$ ) in  $\mathcal{O}(N \log N)$  arithmetic operations

$$f(h_{N,j}) = \sum_{k=0}^N f_k T_k(h_{N,j}) \quad (j = 0, \dots, N).$$

2. Form by FMM (see Algorithm 3.1 of [11]) in  $\mathcal{O}(N \log(1/\epsilon))$ .

$$g_k := \sum_{j=0}^N \varepsilon_{N,j} \frac{(-1)^{j+1}}{N} f(h_{N,j}) \frac{1}{x_{M,k} - h_{N,j}} \quad (k = 0, \dots, M).$$

3. Set  $s(x_{M,k}) := \sin(y_{M,k}) \sin(Ny_{M,k}) g_k$  ( $k = 0, \dots, M$ ).

*Output:*  $s(x_{M,k})$  approximate value of  $f(x_{M,k})$  ( $k = 0, \dots, M$ ).

Algorithm 2.1 reads in matrix-vector notation with  $f := (\varepsilon_{N,k}^{-1} f_k)_{k=0}^N$  as

$$\mathbf{A} f = \mathbf{D}_1 \mathbf{B} \mathbf{D}_2 \mathbf{C}_{N+1}^I f, \quad (2.1)$$

where  $\mathbf{D}_1$  and  $\mathbf{D}_2$  denotes the diagonal matrices

$$\begin{aligned} \mathbf{D}_1 &:= \text{diag} \left( \sin(y_{M,k}) \sin(Ny_{M,k}) \right)_{k=0}^M, \\ \mathbf{D}_2 &:= \text{diag} \left( \varepsilon_{N,j} f(h_{N,l}) \frac{(-1)^{j+1}}{N} \right)_{j=0}^N \end{aligned}$$

and  $\mathbf{B}$  is the Cauchy matrix

$$\mathbf{B} := \left( \frac{1}{x_{M,k} - h_{N,j}} \right)_{k=0, j=0}^{M, N}. \quad (2.2)$$

An algorithm for the NDCT<sup>T</sup> is straightforward by transposing the matrix product (2.1).

**Remark 2.2** The FMM-based approach leads to a set of closely related forward and inverse algorithms. Unfortunately the inverse algorithms are only numerical stable for points  $x_{M,j}$  which are distributed similar to the points  $h_{M,j}$ . For the NDFT algorithms a similar behavior was observed in [9].  $\square$

## 2.2 NDCT based on the Chebyshev transform

In this section we introduce the Chebyshev transform and corresponding shifts. Further we use the Chebyshev transform of  $L_\omega^2([-1, 1])$  onto  $l^2$  mapping  $f \in L_\omega^2([-1, 1])$  to  $(a_n[f])_{n=0}^\infty \in l^2$  with the Chebyshev coefficients

$$a_n[f] := \langle f, T_n \rangle \quad (n \in \mathbb{N}_0).$$

For more details on Chebyshev transform see [5]. In order to adapt the concept of shifts to the interval (see [5, 19]) we introduce the Chebyshev shift  $\sigma_{n,j}f$  of  $f$ , i.e.

$$\begin{aligned} (\sigma_{n,j}f)(x) &:= \frac{1}{2}f\left(x \cos \frac{j\pi}{n} - \sqrt{1-x^2} \sin \frac{j\pi}{n}\right) \\ &+ \frac{1}{2}f\left(x \cos \frac{j\pi}{n} + \sqrt{1-x^2} \sin \frac{j\pi}{n}\right) \quad (x \in [-1, 1]). \end{aligned}$$

We have to evaluate the polynomial

$$f(x) = \sum_{k=0}^N f_k T_k(x) \quad (2.3)$$

at the nodes  $x_{M,l} \in [-1, 1]$  ( $l = 0, \dots, M$ ). We introduce the oversampling factor  $\alpha > 1$  and set  $n := \alpha N$ . Let  $\varphi$  be a function on  $[-1, 1]$  with uniformly convergent Chebyshev series, such that

$$\varphi(x) = \sum_{k=0}^{\infty} c_k[\varphi] T_k(x).$$

We approximate  $f$  by

$$s_1(x) = \sum_{j=0}^n g_j \sigma_{n,j} \varphi(x). \quad (2.4)$$

By properties of the Chebyshev shift we obtain

$$(\sigma_{n,j} \varphi)(x) = \sum_{k=0}^{\infty} c_k[\varphi] T_k(x) \cos \frac{kj\pi}{n}.$$

The Chebyshev series of  $s_1$  reads as

$$s_1(x) = \sum_{k=0}^{\infty} c_k[\varphi] \hat{g}_k T_k(x) \quad (2.5)$$

with

$$\hat{g}_k := \sum_{j=0}^n g_j \cos \frac{kj\pi}{n}. \quad (2.6)$$

If the coefficients  $c_k[\varphi]$  become sufficiently small for  $k \geq N$  and if  $c_k[\varphi] \neq 0$  for  $k = 0, \dots, N-1$ , then we suggest by comparing (2.3) with (2.5) to set

$$\hat{g}_k := \begin{cases} \frac{f_k}{c_k[\varphi]} & k = 0, \dots, N-1, \\ 0 & k = N, \dots, n. \end{cases} \quad (2.7)$$

Now the values  $g_j$  can be obtained from (2.6) by the reduced DCT-I of size  $n$ . If the functions  $\sigma_{n,j} \varphi$  ( $j = 0, \dots, n$ ) are also well localized in time domain such that they can be approximated by functions  $\psi_{n,j}$  with small support, then

$$f(x_{M,l}) \approx s_1(x_{M,l}) \approx s(x_{M,l}) := \sum_{j \in I_{n,m}(x_{M,l})} g_j \psi_{n,j}(x_{M,l}) \quad (2.8)$$

with  $I_{n,m}(x_{M,l}) := \left\{ j = -m, \dots, n+m : \cos \frac{(j+m)\pi}{n} \leq x_{M,l} \leq \cos \frac{(j-m)\pi}{n} \right\}$ .

For fixed  $x_{M,l} \in [-1, 1]$  the above sum contains at most  $(2m+2)$  nonzero summands.

In summary, we obtain the following algorithm for the fast computation of (2.3) with  $\mathcal{O}(\alpha N \log(\alpha N)) + \mathcal{O}(mM)$  arithmetical operations, where  $\alpha$  and  $m$  are constants.

**Algorithm 2.3** (*Fast computation of NDCT using the Chebyshev-transform*).

*Input:*  $N, m \in \mathbb{N}$ ,  $\alpha > 1$ ,  $x_{M,l} \in [-1, 1]$  ( $l = 0, \dots, M$ ),  $f_k \in \mathbb{R}$  ( $k = 0, \dots, N$ ).

0. Precompute  $n := \alpha N$ ,  $c_k[\varphi]$  ( $k = 0, \dots, N$ ).

1. Form  $\hat{g}_k := f_k/c_k[\varphi]$  ( $k = 0, \dots, N$ ).

2. Compute by reduced DCT-I ( $n+1$ )

$$g_j = \frac{1}{n} \sum_{k=0}^N \frac{1}{\varepsilon_{n,k}} \hat{g}_k \cos \frac{kj\pi}{n} \quad (j = 0, \dots, n). \quad (2.9)$$

3. Set

$$s(x_{M,l}) := \sum_{j \in I_{n,m}(x_{M,l})} g_j \psi_{n,j}(x_{M,l}) \quad (l = 0, \dots, M).$$

*Output:*  $s(x_{M,l})$  approximate value of  $f(x_{M,l})$  ( $l = 0, \dots, M$ ).

The Algorithm 2.3 reads in matrix-vector notation as

$$\mathbf{A}f \approx \mathbf{B}\mathbf{C}_{n+1}^I \mathbf{D}f$$

where  $\mathbf{B}$  denotes the sparse matrix

$$\mathbf{B} := (\psi_{n,j}(x_{M,l}))_{l=0, j=0}^{M, n}$$

and where

$$\mathbf{D} := \left( \mathbf{0} \mid \text{diag} \left( 1/(nc_k[\varphi]) \right)_{k=0}^N \mid \mathbf{0} \right)^T$$

with

$$(N, (n-N)/2) - \text{zero matrices } \mathbf{0}.$$

In case of trigonometric polynomials a different method for computing the entries of the sparse matrix  $\mathbf{B}$  was given in [17].

Immediately we obtain an algorithm for the problem (2.2) by transposing the matrix-vector notation

$$\mathbf{A}^T \approx \mathbf{D}^T (\mathbf{C}_{n+1}^I)^T \mathbf{B}^T.$$

For  $j = m, \dots, n+m$ , we introduce the index set  $J_{n,m}(j) := \{l = 0, \dots, M : j - m \leq \frac{n \arccos(x_{M,l})}{\pi} \leq j + m\}$ .

**Algorithm 2.4** (*Fast computation of NDCT<sup>T</sup> using the Chebyshev–transform*).

*Input:*  $N, m \in \mathbb{N}$ ,  $\alpha > 1$ ,  $x_{M,l} \in [-1, 1]$  ( $l = 0, \dots, M$ ),  $f_k \in \mathbb{R}$  ( $k = 0, \dots, N$ ).

0. Precompute  $n := \alpha N$ ,  $c_k[\varphi]$  ( $k = 0, \dots, N$ ).

1. Set

$$\tilde{g}_j := \sum_{l \in J_{n,m}(j)} f_l \psi_{n,j}(x_{M,l}) \quad (j = 0, \dots, n).$$

2. Compute by reduced DCT-I( $n + 1$ )

$$\tilde{c}_k(g) := \frac{1}{n \varepsilon_{n,k}} \sum_{j=0}^n \tilde{g}_j \cos \frac{jk\pi}{n} \quad (k = 0, \dots, N).$$

3. Form  $\tilde{h}(k) := \tilde{c}_k(g)/c_k[\varphi]$  ( $k = 0, \dots, N$ ).

*Output:*  $\tilde{h}(k)$  approximate value of  $h(k)$  ( $k = 0, \dots, N$ ).

Both algorithm introduce the same approximation errors. By (2.8), the error splits as follows:

$$E(x_{M,l}) := |f(x_{M,l}) - s(x_{M,l})| \leq E_a(x_{M,l}) + E_t(x_{M,l})$$

with  $E_a(x_{M,l}) := |f(x_{M,l}) - s_1(x_{M,l})|$  and  $E_t(x_{M,l}) := |s_1(x_{M,l}) - s(x_{M,l})|$ .

Note that  $E_a(x_{M,l})$  and  $E_t(x_{M,l})$  are the aliasing error and the truncation error introduced by Algorithm 2.3, respectively. Let  $E_\infty := \max_{l=1, \dots, M} E(x_{M,l})$  and  $\|f\|_1 := \sum_{k=0}^N \varepsilon_{n,k}^{-1} |f_k|$ .

By (2.3) and (2.5) we get for the aliasing error

$$E_a(x_{M,l}) = \left| \sum_{k=N+1}^{\infty} c_k[\varphi] \hat{g}_k T_k(x_{M,l}) \right|.$$

By (2.6) we have  $\hat{g}_k = \hat{g}_{2nr+k}$  ( $k = -n, \dots, n$ ;  $r \in \mathbb{Z}$ ) and with (2.7) we obtain

$$\begin{aligned} E_a(x_{M,l}) &= \left| \sum_{r=1}^{\infty} \sum_{k=-N}^N c_{k+2nr}[\varphi] \hat{g}_k T_{k+2nr}(x_{M,l}) \right| & (2.10) \\ &\leq \sum_{k=-N}^N |f_k| \sum_{r=1}^{\infty} \left| \frac{c_{k+2nr}[\varphi]}{c_{|k|}[\varphi]} \right| \\ &\leq 2 \|f\|_1 \max_{k=-N, \dots, N} \sum_{r=1}^{\infty} \left| \frac{c_{k+2nr}[\varphi]}{c_{|k|}[\varphi]} \right|. \end{aligned}$$

By (2.4) and (2.8) the truncation error can be written as

$$E_t(x_{M,l}) = \left| \sum_{j=0}^n g_j (\sigma_{n,j} \varphi(x_{M,l}) - \psi_{n,j}(x_{M,l})) \right|$$

and further by (2.9) and (2.7) as

$$E_t(x_{M,l}) = \left| \sum_{j=0}^n \frac{1}{n} \sum_{k=0}^N \frac{1}{\varepsilon_{n,k}} \frac{f_k}{c_k[\varphi]} \cos \frac{kj\pi}{n} (\sigma_{n,j}\varphi(x_{M,l}) - \psi_{n,j}(x_{M,l})) \right|.$$

Consequently, we obtain

$$E_t(x_{M,l}) \leq \frac{\|f_1\|}{n} \max_{k=0,\dots,N} \frac{1}{|c_k[\varphi]|} \sum_{j=0}^n |\sigma_{n,j}\varphi(x_{M,l}) - \psi_{n,j}(x_{M,l})|. \quad (2.11)$$

**Remark 2.5** We remark that we can develop similar algorithms for the fast computation of

$$f(x) = \sum_{k=0}^N f_k U_k(x).$$

□

Let us consider special functions  $\varphi$  and  $\psi$ .

### 2.2.1 $\varphi$ with compact support in time domain

The following construction of splines on the interval is based on [19]. Let  $\chi_{[0,1]}$  denote the characteristic function of  $[0, 1)$  and let  $m \in \mathbb{N}$  be a fixed number. The *cardinal B-splines*  $N_m$  ( $m \geq 1$ ) of order  $m$  are defined by

$$N_1 := \chi_{[0,1)}, \quad N_{m+1} := \int_0^1 N_1(t) N_m(\cdot - t) dt$$

and their centered version by

$$M_m := N_m\left(\cdot + \frac{m}{2}\right).$$

Note that  $M_m$  is an even function with  $\text{supp } M_m = [-\frac{m}{2}, \frac{m}{2}]$  and that

$$\int_{-\infty}^{\infty} M_m(t) e^{-ixt} dt = \left(\text{sinc} \frac{x}{2}\right)^m, \quad (2.12)$$

where

$$\text{sinc } x := \begin{cases} \frac{\sin x}{x} & x \neq 0, \\ 1 & x = 0. \end{cases}$$

Assume that  $1 \leq m < \frac{n}{2}$ . Since  $\text{supp } M_{2m} = [-m, m]$ , let  $\tilde{M}_{n,2m}$  be the  $2\pi$ -periodization of  $M_{2m}(n \cdot / (2\pi))$ , i.e.

$$\tilde{M}_{n,2m} := \sum_{r=-\infty}^{\infty} M_{2m}(n \cdot / (2\pi) - nr).$$

Now restricting  $\tilde{M}_{n,2m}$  on  $[0, \pi]$ , we choose as function  $\varphi^B := \tilde{M}_{n,2m}(\arccos)$ . The function  $\varphi^B$  has the Chebyshev coefficients

$$c_k[\varphi^B] = \frac{1}{n} \left(\text{sinc}(k\pi/n)\right)^{2m} \quad (2.13)$$

and

$$\text{supp } \varphi^B = \left( \cos \frac{m\pi}{n}, 1 \right].$$

Note that

$$\text{supp } \sigma_{n,j} \varphi^B = \begin{cases} \left( \cos\left(\frac{(j+m)\pi}{n}\right), 1 \right] & j < m, \\ \left( \cos\left(\frac{(j+m)\pi}{n}\right), \cos\left(\frac{(j-m)\pi}{n}\right) \right) & n-m \geq j \geq m, \\ \left[ -1, \cos\left(\frac{(j-m)\pi}{n}\right) \right) & j > n-m. \end{cases}$$

Consequently we set  $\psi_{n,j}^B := \sigma_{n,j} \varphi^B$ .

By comparing (2.10) and (2.11) with the error estimates for the NDFT in [21] and following the lines of the proof of Theorem 1.2 in [21] we obtain the following theorem.

**Theorem 2.6** *Let the NDCT( $M+1, N+1$ ) be computed by Algorithm 2.3 with the transformed, dilated, periodized, centered cardinal B-spline of order  $2m$  where  $\varphi = \varphi^B$  is given in (2.13) and  $\psi_{n,j}^B := \sigma_{n,j} \varphi^B$ . Then for  $\alpha > 1$  and  $0 \leq \eta \leq 4m/3$ ,*

$$E_\infty \leq \frac{4m}{2m-1} \left( \frac{N}{2} \right)^{-\eta} (2\alpha-1)^{-2m} |f|_{\eta,1}. \quad (2.14)$$

Here  $|f|_{\eta,1}$  denotes the Sobolev-like seminorm  $|f|_{\eta,1} := \sum_{k=0}^N \varepsilon_{n,k}^{-1} |f_k| |k|^\eta$ .

## 2.2.2 Gaussian bell

The use of Gaussian bells for the fast computation of Fourier transforms on arbitrary nodes was first suggested in [12] and a good choice for the parameters was given in [22]. By transferring the concept to the interval we obtain

**Theorem 2.7** *Let the NDCT be computed by Algorithm 2.3 with the transformed, dilated, periodized Gaussian bell*

$$\tilde{\varphi}(v) := (\pi b)^{-1/2} \sum_{r \in \mathbb{Z}} e^{-(n(v+r))^2/b}, \quad \varphi^G := \tilde{\varphi}(\arccos(\pi \cdot))$$

and with the truncated version of  $\varphi^G$ ,

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

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

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

The approximation error decreases with increasing  $b$ . Therefore

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

provides a good choice for  $b$  as function of  $\alpha$  and  $m$  (see [22, 21]).

**Remark 2.8** In various papers for the fast computation of the NDFT, other window functions than the Gaussian pulse or centered cardinal B-splines were considered (see e.g. for prolate spheroidal functions and Kaiser-Bessel functions [16], for Gaussian pulse tapered with a Hanning window [10] or for Gaussian kernels combined with sinc kernels [18]).  $\square$

### 3 Fast polynomial transform

Let  $L_\omega^2([-1, 1])$  be the weighted Hilbert space of all measurable functions  $f : [-1, 1] \rightarrow \mathbb{R}$  with the property

$$\int_{-1}^1 \omega(x) f(x)^2 dx < \infty$$

and a weight  $\omega$  fulfilling (1.1).

Let  $\{P_n\}_{n \in \mathbb{N}_0}$  be a sequence of orthogonal polynomials  $P_n \in \Pi_n$  with respect to  $\langle \cdot, \cdot \rangle$  defined by the three-term recurrence relation

$$\begin{aligned} P_{-1}(x) &:= 0, & P_0(x) &:= 1, \\ P_n(x) &= (\alpha_n x + \beta_n) P_{n-1}(x) + \gamma_n P_{n-2}(x) \quad (n = 1, 2, \dots) \end{aligned} \quad (3.1)$$

with  $\alpha_n, \beta_n, \gamma_n \in \mathbb{R}$  and  $\alpha_n > 0, \gamma_n \neq 0$  ( $n \in \mathbb{N}$ ). Consider

$$P := \sum_{k=0}^N f_k P_k \in \Pi_N$$

with known real coefficients  $f_k$ . Our concern is the fast evaluation of  $P(x_{M,j})$  ( $j = 0, \dots, M$ ) with  $\mathcal{O}(N \log^2 N)$  instead of  $\mathcal{O}(MN)$  arithmetical operations. The first part of our algorithm realizes the basis exchange from  $\{P_k\}_{k=0}^N$  to  $\{T_k\}_{k=0}^N$  in  $\Pi_N$  and produces the Chebyshev coefficients  $\tilde{f}_k$  in

$$P = \sum_{k=0}^N \tilde{f}_k T_k. \quad (3.2)$$

This algorithm was developed in [20], see also the approach of Driscoll and Healy for the transposed problem developed in [7, 8]. Our approach computes the basis exchange with  $\mathcal{O}(N \log^2 N)$  arithmetical operations by a divide-and-conquer technique combined with fast polynomial multiplications. The algorithm was designed for arbitrary polynomials  $P_n$  satisfying a three-term recurrence relation. It requires multiplications with precomputed values of associated polynomials of  $P_n$  occupying  $\mathcal{O}(N \log N)$  elements of storage.

Knowing the Chebyshev coefficients  $\tilde{f}_k$ , the values  $P(x_{M,j})$  ( $j = 0, \dots, M$ ) can be computed by Algorithm 2.3 in  $\mathcal{O}(\alpha N \log(\alpha N)) + \mathcal{O}(mM)$  arithmetical operations.

**Algorithm 3.1** (*Fast computation of DPT*).

*Input:*  $N, M \in \mathbb{N}$ ,  $x_{M,l} \in [-1, 1]$  ( $l = 0, \dots, M$ ),  $f_k \in \mathbb{R}$  ( $k = 0, \dots, N$ ).

1. Compute the Chebyshev coefficients  $\tilde{f}_k$  ( $k = 0, \dots, N$ ) by Algorithm 3.2 of [20].
2. Compute the values  $P(x_{M,l})$  ( $l = 0, \dots, M$ ) of (3.2) by Algorithm 2.3.

Output:  $s(x_{M,l})$  approximate value of  $P(x_{M,l})$  ( $l = 0, \dots, M$ ).

## 4 Numerical Examples

In this section we confirm our theoretical results by various numerical examples. The algorithms are implemented in C and tested on a SGI O2 using double precision arithmetic. Let

$$E := \max_{l=0, \dots, M} |f(x_{M,l}) - s(x_{M,l})| / \max_{l=0, \dots, M} |f(x_{M,l})|. \quad (4.1)$$

The values  $f(x_{M,l})$  ( $l = 0, \dots, M$ ) are obtained by a straightforward algorithm based on the three-term recurrence relation (Clenshaw algorithm) in  $\mathcal{O}(MN)$  flops.

Several technical details of our implementation appear to be worth mention here:

1. For the fast matrix-vector multiplication with the Cauchy matrix (2.2) we use the Algorithm 3.2 of [11]. In order to obtain the fixed accuracy  $E_\infty \leq 10^{-13}$  we choose the size of the Chebyshev expansion  $p=16$ . Note that schemes have been constructed to further compress the multipole expansions (see [25]).
2. In Algorithm 2.3 we choose the oversampling factor  $\alpha = 2$ . If we use the Gaussian bell we choose the parameter  $b = \frac{2\alpha m}{(2\alpha-1)\pi}$ . If we use the  $B$ -splines we compute the values  $\psi_{n,j}(x_{M,l})$  recursively by the recurrence relation of  $B$ -splines [6].
3. The implementation of the vector multiplication with the cosine matrix  $\mathbf{C}_{N+1}^I$  based on [23, 2] where taken from [1].

**Example 4.1** The purpose of this example is to confirm the theoretical results of Theorem 2.6. We choose the values  $f_k$  ( $k = 0, \dots, N$ ) randomly distributed in the interval  $[-1, 1]$  and compute (2.1) at the nodes  $x_{N,l} = -1 + 2l/N$ , where  $N = 2^t$  with  $t \in \{8, 9, \dots, 13\}$ . Choosing  $m = 6$  we obtain a relative error  $E \leq 1.2e - 7$  as shown by the dotted line with  $\times$  in Figure 1. Choosing  $m = 9$  we obtain an relative error  $E \leq 1.1e - 10$  as shown by the dotted line with  $*$  in Figure 1. If the values  $f_k$  are the Fourier coefficients of a smooth function than sharper estimates are obtainable (2.14). If we choose  $\eta = 2$  in (2.14), then the values  $(k+1)^2 f_k$  ( $k = 0, \dots, N$ ) are uniformly distributed in the interval  $[-1, 1]$ . The errors for  $m = 2$  are given by (2.14) with

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

We show the error  $E$  of Algorithm 2.3 using  $B$ -splines by a solid line with  $\times$  in Figure 1. For  $m = 4$  we obtain

$$E \leq \frac{1}{63} \left(\frac{2}{3}\right)^6 N^{-2} |f|_{4,1}$$

where the error  $E$  of Algorithm 2.3 is shown by a solid line with  $*$  in Figure 1. Note that for  $\eta > 0$  the error  $E$  decreases for increasing  $N$ . For  $\eta = 0$  we have to increase  $m$  in order to get a smaller error  $E$ . This is what we observe in Figure 1.  $\square$

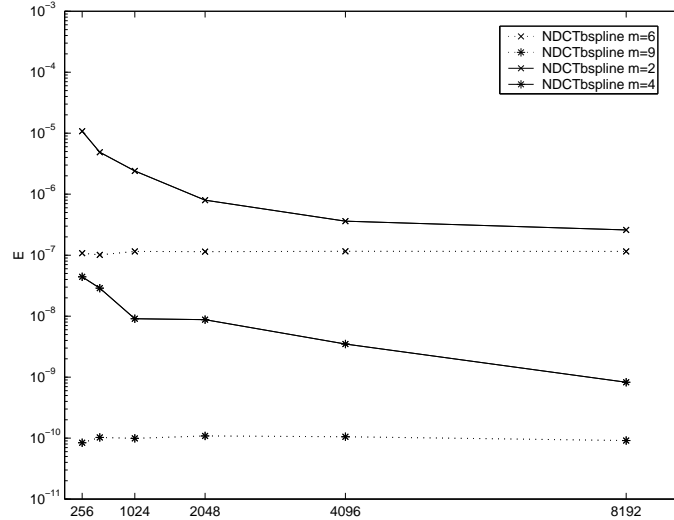


Figure 1: Error  $E$  for Algorithm 2.3 by using  $B$ -splines, dotted line  $\eta = 0$ , solid line  $\eta = 2$  (see (2.14))

**Example 4.2** In this example we compare the computational time of our implementation for the fixed accuracy  $E \leq 10^{-12}$ . Hence we have to choose  $m = 12$  in Algorithm 2.3. The computational time for the straightforward algorithm is marked by a square  $\square$ . As mention in 2.4 we can also use the NDFt algorithms. We mark the computational time with a solid line with  $*$  (NDFtgauss) in case of Gaussian bells and with a solid line with  $\times$  (NDFtbspline) in case of  $B$ -splines (see also [14]). The computational time for Algorithm 2.3 based on Gaussian bells (NDCTgauss) is marked with a dotted line and  $*$  and with a dotted line and  $\times$  in case of  $B$ -splines. The computational time for Algorithm 2.1 (FMM) is marked by a dashed line with  $\diamond$ .

Note that Algorithm 2.3 is approximately 45 times faster for  $N = 2^{13}$  and 145 times faster for  $N = 2^{14}$  (if we use the Gaussian bell) than the straightforward algorithm.  $\square$

Various numerical test with our implementation of the different algorithms shown that Algorithm 2.3 based on Gaussian bells realize the NDCT in the fastest way. That is the reason why we use these Algorithm in step 2 of Algorithm 3.1 in order to compute the DPT.

**Example 4.3** We consider the ultraspherical polynomials  $P_n^\lambda$  ( $\lambda > -1/2$ ) given by

$$\begin{aligned} P_{-1}^\lambda(x) &:= 0, & P_0^\lambda(x) &:= 1, \\ P_{n+1}^\lambda(x) &:= \frac{2(n+\lambda)}{n+1} x P_n^\lambda(x) - \frac{n+2\lambda-1}{n+1} P_{n-1}^\lambda(x) \quad (n = 0, 1, 2, \dots). \end{aligned}$$

These polynomials are orthogonal with respect to the weight function  $\omega(x) = (1-x^2)^{\lambda-1/2}$ . For  $\lambda = \frac{n-2}{2}$ , the ultraspherical polynomials are the zonal spherical polynomials of  $S^{n-1}$  with respect to  $SO(n)/SO(n-1)$ . For the 2-sphere  $S^2$ , i.e. for  $\lambda = 1/2$ , the ultraspherical polynomials are the Legendre polynomials. For given  $f_k \in \mathbb{R}$  ( $k = 0, \dots, N$ ) we compute

$$f(x_{N,j}) = \sum_{k=0}^N f_k P_k^\lambda(x_{N,j}) \quad (j = 0, \dots, N) \quad (4.2)$$

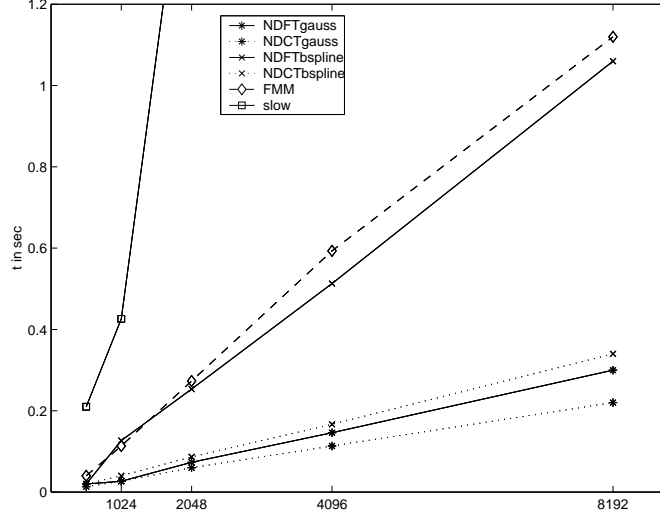


Figure 2: Computational time for different algorithms.

by the Clenshaw algorithm in double precision arithmetic and by Algorithm 3.1. Table 1 compares the results for different transform lengths  $N$  ranging between 128 and 4096 and various parameters  $\lambda$ . We choose the values  $f_k$  ( $k = 0, \dots, N$ ) randomly distributed in the interval  $[-1, 1]$  and compute (2.1) at the zeros of  $P_{N+1}^\lambda$ . The first and the second columns of Table 1 contain the parameters  $N$  and  $\lambda$ , respectively. The last three columns contain the error  $E$  for different values  $m$ . Finally Figure 3 shows the CPU time of Algorithm 3.1.  $\square$

$N$	$\lambda$	E			
		$m = 6$	$m = 8$	$m = 10$	$m = 12$
128	0.5	1.551e-07	3.328e-09	1.478e-11	4.316e-13
256	0.5	1.567e-07	1.699e-09	1.368e-11	2.985e-12
512	0.5	1.156e-07	3.703e-09	5.118e-11	1.990e-11
1024	0.5	8.850e-08	3.033e-09	7.266e-11	6.533e-11
2048	0.5	9.409e-08	3.374e-09	1.503e-10	1.634e-10
4096	0.5	9.541e-08	3.097e-09	3.416e-09	1.792e-09
128	1.5	9.547e-07	4.779e-09	4.312e-11	5.542e-13
256	1.5	3.238e-07	7.325e-09	3.133e-11	2.246e-11
512	1.5	2.974e-07	2.460e-09	2.800e-11	2.800e-11
1024	1.5	4.530e-07	1.234e-09	6.453e-11	1.860e-11
2048	1.5	4.865e-07	1.518e-09	2.976e-10	5.691e-10

Table 1: Relative error (4.1) of Algorithm 3.1

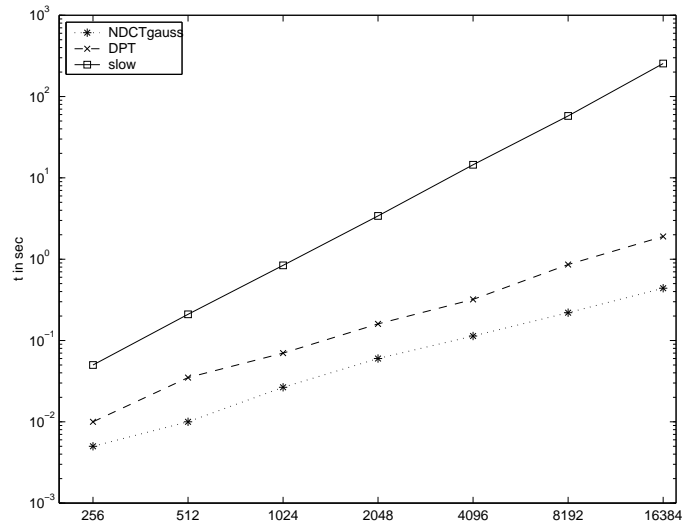


Figure 3: Computational time for Algorithm 3.1, the Clenshaw algorithm and for NDCTgauss ( $m = 8$ ).

## References

- [1] G. Baszenski. Programmpaket zur Berechnung diskreter trigonometrischer Transformationen. 1995. <http://www.iuk.fh-dortmund.de/~baszenski/>.
- [2] G. Baszenski and M. Tasche. Fast polynomial multiplication and convolution related to the discrete cosine transform. *Linear Algebra Appl.*, 252:1 – 25, 1997.
- [3] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmonic. Anal.*, 2:363 – 381, 1995.
- [4] D. Bini and V. Pan. *Polynomial and Matrix Computations*, volume 1. Birkhäuser, Boston, 1994.
- [5] P. Butzer and R. Stens. The operational properties of the Chebyshev transform. I: General properties. *Funct. Approx. Comment. Math.*, 5:129 – 160, 1977.
- [6] C. K. Chui. *An Introduction to Wavelets*. Academic Press, Boston, 1992.
- [7] J. Driscoll and D. M. Healy. Computing Fourier transforms and convolutions on the 2-sphere. *Adv. Appl. Math.*, 15:202 – 240, 1994.
- [8] J. Driscoll, D. M. Healy, and D. N. Rockmore. Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs. *SIAM J. Comput.*, 26:1066 – 1099, 1996.
- [9] J. O. Droese. Verfahren zur schnellen Fourier-Transformation mit nichtäquidistanten Knoten. Diplomarbeit, TH Darmstadt, 1996.
- [10] A. J. W. Duijndam and M. A. Schonewille. Nonuniform fast Fourier transform. *Geophysics*, 64:539 – 551, 1999.

- [11] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration and differentiation. *SIAM J. Numer. Anal.*, 33:1689 – 1711, 1996.
- [12] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Stat. Comput.*, 14:1368 – 1393, 1993.
- [13] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data II. *Appl. Comput. Harmonic. Anal.*, 2:85 – 100, 1995.
- [14] E. Elbel. Mehrdimensionale Fouriertransformation für nichtäquidistanten Daten. Diplomarbeit, TH Darmstadt, 1998.
- [15] G. Heinig and K. Rost. Representations of Cauchy matrices with Chebyshev nodes using trigonometric transformations. In *Structured Matrices: Recent Advances in Theory and Computation*, Nova Press.
- [16] J. I. Jackson. Selection of a convolution function for Fourier inversion using gridding. *IEEE Trans. Medical Imaging*, 10:473 – 478, 1991.
- [17] N. Nguyen and Q. H. Liu. The regular Fourier matrices and nonuniform fast Fourier transforms. *SIAM J. Sci. Comput.*, 21:283 – 293, 1999.
- [18] J. Pelt. Fast computation of trigonometric sums with applications to frequency analysis of astronomical data. In D. Maoz, A. Sternberg, and E. Leibowitz, editors, *Astronomical Time Series*, pages 179 – 182, Kluwer Academic Publishers, 1997.
- [19] G. Plonka, K. Selig, and M. Tasche. On the construction of wavelets on a bounded interval. *Adv. in Comput. Math.*, 4:357 – 358, 1995.
- [20] D. Potts, G. Steidl, and M. Tasche. Fast algorithms for discrete polynomial transforms. *Math. Comp.*, 67:1577 – 1590, 1998.
- [21] 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 249 – 274, Boston, 2001.
- [22] G. Steidl. A note on fast Fourier transforms for nonequispaced grids. *Adv. Comput. Math.*, 9:337 – 353, 1998.
- [23] G. Steidl and M. Tasche. A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms. *Math. Comp.*, 56:281 – 296, 1991.
- [24] A. F. Ware. Fast approximate Fourier transforms for irregularly spaced data. *SIAM Review*, 40:838 – 856, 1998.
- [25] N. Yarvin and V. Rokhlin. An improved fast multipole algorithm for potential fields on the line. *SIAM J. Numer. Anal.*, 36:629 – 666, 1999.