Fast Fourier Transform at Nonequispaced Nodes and Applications

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Diplom-Mathematiker
Markus Fenn
aus Mannheim

Mannheim, Dezember 2005

Dekan: Professor Dr. Matthias Krause, Universität Mannheim
Referent: Professor Dr. Gabriele Steidl, Universität Mannheim
Korreferent: Professor Dr. Daniel Potts, Technische Universität Chemnitz

Tag der mündlichen Prüfung: 8. Dezember 2005

Acknowledgments

First of all, I would like to thank my supervisor Gabriele Steidl for her excellent support throughout the years, and for having so many inspiring ideas.

I am also very grateful to Daniel Potts, Stefan Kunis and Jianwei Ma for the fruitful collaboration and for the NFFT software package of the former two.

Sincere thanks are given to my colleagues at university for a very familiar atmosphere, especially to Julia who eased the beginning for me and to Robert who always had to solve my computer problems, to my friends who constantly forgave me for my little spare time and to every one else who helped me finish this thesis.

Above all, I want to express my most cordial gratitude to my family for their love and support over the years. Thank you for always believing in me.

Summary

The direct computation of sums

$$f(x_j) = \sum_{k=-N/2}^{N/2-1} \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} k x_j}$$

at arbitrary nodes $x_j \in [-\frac{1}{2}, \frac{1}{2})$ ($j = 1, \ldots, M$) requires $\mathcal{O}(NM)$ arithmetical operations, too much for practical purposes. For equally spaced nodes $x_j = \frac{j}{N}$ ($j = -\frac{N}{2}$, $\ldots$, $\frac{N}{2} - 1$) the computation can be done by the well known *fast Fourier transform* (FFT) in only $\mathcal{O}(N \log N)$ arithmetical operations. Recently, the *fast Fourier transform for nonequispaced nodes* (NFFT) was developed for the fast approximative computation of the above sums in only $\mathcal{O}\big(N \log N + M \log \frac{1}{\varepsilon}\big)$, where ε denotes the required accuracy.

The principal topics of this thesis are generalizations and applications of the NFFT. This includes the following subjects:

- Algorithms for the fast approximative computation of the discrete cosine and sine transform at nonequispaced nodes are developed by applying fast trigonometric transforms instead of FFTs.
- An algorithm for the fast Fourier transform on hyperbolic cross points with nonequispaced spatial nodes in 2 and 3 dimensions based on the NFFT and an appropriate partitioning of the hyperbolic cross is proposed.
- A unified linear algebraic approach to recent methods for the fast computation of matrix–vector–products with special dense matrices, namely the fast multipole method, fast mosaic-skeleton approximation and $\mathcal{H}$-matrix arithmetic, is given. Moreover, the NFFT-based summation algorithm by Potts and Steidl is further developed and simplified by using algebraic polynomials instead of trigonometric polynomials and the error estimates are improved.
- A new algorithm for the characterization of engineering surface topographies with line singularities is proposed. It is based on hard thresholding complex ridgelet coefficients combined with total variation minimization. The discrete ridgelet transform is designed by first using a discrete Radon transform based on the NFFT and then applying a dual-tree complex wavelet transform.
- A new robust local scattered data approximation method is introduced. It is an advancement of the moving least squares approximation (MLS) and generalizes an approach of van den Boomgard and van de Weijer to scattered data. In particular, the new method is space and data adaptive.

Zusammenfassung

Die direkte Berechnung der Summen

$$f(x_j) = \sum_{k=-N/2}^{N/2-1} \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} k x_j}$$

an beliebigen Knoten $x_j \in [-\frac{1}{2}, \frac{1}{2})$ $(j = 1, \ldots, M)$ erfordert $\mathcal{O}(NM)$ arithmetische Operationen. Für äquidistante Knoten $x_j = \frac{j}{N}$ $(j = -\frac{N}{2}, \ldots, \frac{N}{2} - 1)$ kann die Berechnung mittels *schneller Fouriertransformation* (FFT) in nur $\mathcal{O}(N \log N)$ arithmetischen Operationen durchgeführt werden. Auf unstrukturierten Knotenmengen hat sich inzwischen ein Algorithmus (NFFT) zur effizienten näherungsweisen Berechnung obiger Summen in nur $\mathcal{O}\big(N \log N + M \log \frac{1}{\varepsilon}\big)$ etabliert, wobei ε die gewünschte Genauigkeit bezeichnet.

Das Hauptaugenmerk dieser Dissertation liegt auf Verallgemeinerungen und Anwendungen der NFFT. Im Einzelnen beinhaltet dies die folgenden Ergebnisse:

- Es werden schnelle Algorithmen für die näherungsweise Berechnung der diskreten Kosinus- und Sinustransformation auf unstrukturierten Knotenmengen entwickelt, die statt der FFT schnelle trigonometrische Transformationen verwenden.
- Ein Algorithmus für die schnelle Fouriertransformation auf hyperbolischen Kreuzen in 2 und 3 Dimensionen mit unstrukturierten Knotenmengen im Zeitbereich wird aufbauend auf der NFFT und einer geeigneten Unterteilung der hyperbolischen Kreuze konstruiert.
- Ein einheitlicher algebraischer Zugang zu aktuellen Methoden für die schnelle Berechnung von Matrix–Vektor–Produkten mit speziellen vollbesetzten Matrizen, und zwar die „fast multipole method“ (FMM), die „fast mosaic-skeleton“-Approximation und die $\mathcal{H}$-Matrix-Arithmetik, wird gegeben. Desweiteren wird der NFFT-basierte Summationsalgorithmus von Potts und Steidl durch die Verwendung algebraischer an Stelle von trigonometrischen Polynomen vereinfacht, und die Fehlerabschätzungen werden verbessert.
- Ein neuer Algorithmus zur Beschreibung der Beschaffenheit technischer Oberflächen mit Liniensingularitäten wird vorgestellt. Er basiert auf einer Kopplung von „hard thresholding“ von Ridgeletkoeffizienten mit Verfahren zur Minimierung der totalen Variation. Die diskrete Ridgelettransformation erhält man durch Verwendung einer auf der NFFT basierenden diskreten Radontransformation und anschließender Anwendung einer speziellen komplexen Wavelettransformation, der „dual-tree complex wavelet transform“.

- Eine neue robuste lokale Approximationsmethode für gestreute Daten wird eingeführt. Sie stellt eine Weiterentwicklung der sogenannten „moving least squares“ Approximation (MLS) dar und verallgemeinert einen Ansatz von van den Boomgaard und van de Weijer auf nichtäquidistante Knoten. Insbesondere ist der neue Zugang raum- und datenadaptiv.

Contents

Notation

$\mathbb{N}_0$	$:= \mathbb{N} \cup \{0\}$ set of nonnegative integers
$\mathbb{T}^d$	$:= \mathbb{R}^d/\mathbb{Z}^d$ the d–dimensional torus,
	represented by $[-\frac{1}{2}, \frac{1}{2}]^d$ with opposing faces identified
i	$= \sqrt{-1} \in \mathbb{C}$ imaginary unit
e	≈ 2.718 base of the natural logarithm $\log$
$\delta_{j,k}$	$:= 1$ for $j = k$ and zero else (Kronecker-Delta)
$\boldsymbol{k}$	$:= (k_1, \ldots, k_d)^T \in \mathbb{N}_0^d$ d-dimensional multi-index
$\lvert\boldsymbol{k}\rvert$	$:= k_1 + \cdots + k_d$
$\boldsymbol{x}$	$:= (x_1, \ldots, x_d)^T$ d-dimensional column vector
$\lVert\boldsymbol{x}\rVert$	$:= \left(\sum_{t=1}^d \lvert x_t\rvert^2\right)^{1/2}$ Euclidean norm
$\boldsymbol{A}$	$:= \left(A_{jk}\right)_{j,k=1}^{m,n}$ matrix with m rows and n columns
$\boldsymbol{A}^T$	$:= \left(A_{kj}\right)_{k,j=1}^{n,m}$ transpose of the matrix $\boldsymbol{A}$
$\boldsymbol{A}^H$	$:= \left(\bar{A}_{kj}\right)_{k,j=1}^{n,m}$ conjugate transpose of the matrix $\boldsymbol{A}$
I_N	$:= \left\{-\frac{N}{2}, \ldots, \frac{N}{2} - 1\right\} \subseteq \mathbb{Z}$ index set, 6
$I_{\boldsymbol{N}}^d$	$:= I_{N_1} \times \cdots \times I_{N_d} \subseteq \mathbb{Z}^d$ tensor product index set, 5
H_J^d	$\subseteq I^d_{(2^J,\ldots,2^J)^T}$ index set of hyperbolic cross points in dimension d, 23
Π_s^d	the space of d-variate polynomials of absolute degree $\leq s$
N_p	normalized cardinal B-spline of degree p, 53
B_k^p	dilated and translated versions of N_p, 54
$S_p(\Delta)$	spline space of degree p with sampling nodes $\Delta := \{t_k\}_{k=-p}^{2p}$, 54
$S(\mathbb{R}^d)$	the Schwartz space
$D(\mathbb{T}^d)$	the counterpart to the Schwartz space for periodic functions, 20
$D'(\mathbb{T}^d)$	dual space of $D(\mathbb{T}^d)$, tempered distributions, 20
$H^a(\mathbb{T}^d)$	L^2-Sobolev space of order $a \in \mathbb{R}$, 21
$E^a(\mathbb{T}^d)$	Korobov space of order $a \in \mathbb{R}$, 21
$\mathcal{R}_{\boldsymbol{\theta}} f(s)$	continuous Radon transform, 73
$R_{\boldsymbol{\theta}} f(s)$	discrete Radon transform, 74
$\mathfrak{R}_{\boldsymbol{\theta}} f(a, b)$	continuous ridgelet transform, 77
$c_{\boldsymbol{\theta}_t, j_0, k}(f)$,	discrete complex ridgelet coefficients
$d_{\boldsymbol{\theta}_t, j, k}(f)$	(smooth resp. detail components), 78

FFT	fast Fourier transform
NFFT	fast Fourier transform for nonequispaced nodes, 6
NFCT	fast cosine transform for nonequispaced nodes, 9
NFST	fast sine transform for nonequispaced nodes, 10
iNFFT	inverse NFFT, 11
SNFFT	sparse NFFT, NFFT on hyperbolic cross points, 26, 30

I. Introduction

Fourier methods play an important role in various areas of applied mathematics and physics. Originally designed by Fourier (1768-1830) for the solution of differential equations they became one of the fundamental principles in digital signal and image processing. The main premise to make Fourier methods applicable in practice was the development of an algorithm for the fast computation of the discrete Fourier transform (DFT)

$$f_j = \sum_{k=-N/2}^{N/2-1} \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} kj/N} \qquad (j = -\tfrac{N}{2}, \ldots, \tfrac{N}{2} - 1)$$

by Cooley and Tukey [29] in 1965. The well-known *fast Fourier transform* (FFT) needs only $\mathcal{O}(N \log N)$ arithmetical operations instead of $\mathcal{O}(N^2)$ arithmetical operations for the direct computation.

Later, it was discovered that already Gauss found an algorithm for the fast computation of the DFT in connection with his analysis of the orbit of the planetoid Pallas in 1805. However, this one and other algorithms, e.g., by Runge in 1903, were disregarded since there were no appropriate computers.

> Once the [FFT] method was established it became clear that it had a long and interesting prehistory going back as far as Gauss. But until the advent of computing machines it was a solution looking for a problem.
>
> T. W. Körner, Fourier Analysis (1988)

Meanwhile, sophisticated soft- and hardware implementations for the FFT exist. In particular, we refer to the C subroutine library *FFTW* [60], whose performance is typically superior to that of other publicly available FFT software, and is even competitive with vendor-tuned codes.

However, the FFT requires sampling on an equally spaced grid, which poses a significant limitation to many applications. The *fast Fourier transform for nonequispaced nodes* (NFFT) efficiently computes approximations of sums

$$f(x_j) = \sum_{k=-N/2}^{N/2-1} \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} k x_j} \qquad (j = 1, \ldots, M)$$

at arbitrary nodes $x_j \in [-\frac{1}{2}, \frac{1}{2})$. The computation of the NFFT is based on the approximation or interpolation of the trigonometric polynomial

$$f(x) := \sum_{k=-N/2}^{N/2-1} \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} k x}$$

by suitable linear combinations of translates of a window function φ having good localization in the time and frequency domain. The arithmetical complexity of the algorithm is $\mathcal{O}\big(N \log N + M \log \frac{1}{\varepsilon}\big)$, where ε denotes the required accuracy.

About 20 years ago, first NFFT methods were introduced in the field of digital signal and image processing [106, 114], but without profound theoretical knowledge. The first papers deriving the theoretical connection between arithmetical complexity of the algorithm and achieved accuracy were provided by Dutt and Rokhlin (1993) [41] for the Gaussian bell φ and by Beylkin (1995) [12] for B-splines φ. Subsequent work [122, 111, 38] unified the approaches of Dutt, Rokhlin and Beylkin and provided consistent error estimates based on the split of the overall error in an aliasing error and a truncation error. These estimates suggested to look for functions φ with better approximating properties. In particular, Kaiser-Bessel-functions [55, 58] and powers of the sinc-function [90] lead to good results. Further approaches based on 'scaling factors' [104], on minimizing the Frobenius norm of certain error matrices [105] or on min-max-interpolations [55] were proposed, but did not bring forth significant improvements. Moreover, there exist algorithms for the fast summation with nonequispaced nodes in time *and* frequency domain [44].

In the meantime, there are publicly available software implementations for the NFFT. One is the C subroutine library of Kunis and Potts [88], another is the Matlab toolbox by Fessler and Sutton [54].

Contribution

The principal contributions of this thesis are generalizations and applications of the NFFT. This includes the following results:

NDCT, NDST For real input data we develop fast algorithms for the *discrete cosine transform at nonequispaced nodes* (NDCT) and for the *discrete sine transform at nonequispaced nodes* (NDST) by applying fast trigonometric transforms. Our approach is based on the NFFT and is easier than the Chebyshev transform based derivation by Potts [107] and faster than the algorithm by Tian and Liu [125], which still uses FFTs. Instead of FFTs we apply fast algorithms for the *discrete cosine transform* (DCT-I) and for the *discrete sine transform* (DST-I).

SNFFT In order to circumvent the so-called 'curse of dimensionality' in multivariate approximation, i.e., the number of degrees of freedom depends exponentially on the dimension, the interpolation on sparse grids and the related approximation on hyperbolic cross points in Fourier domain have been introduced [136, 120, 17]. Just like the NFFT generalizes the FFT, we generalize the existing algorithms for the computation of the FFT with frequencies from a hyperbolic cross and spatial nodes on a sparse grid [6, 136, 78] to arbitrary spatial nodes.

Fast Summation 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, e.g., in the study of particle summations, in the numerical solution of integral equations and in the approximation by radial basis functions. Various algorithms were designed to speed up the summation process. We describe a unified approach to some of these methods coming from different areas and known under different names. These are the hierarchical and fast multipole method (FMM) by Greengard and Rokhlin [65], the mosaic-skeleton approximation by Tyrtyshnikov [127], the panel clustering algorithm [77] and its more general recent version the $\mathcal{H}$- and $\mathcal{H}^2$-matrix concept by Hackbusch et al. [71, 73].

Recently, a fast summation algorithm based on the NFFT was developed by Potts and Steidl [109] which allows a simple incorporation of different kernels. We further develop these ideas and introduce new regularization techniques. Using algebraic polynomials instead of trigonometric polynomials, we prove more sophisticated error estimates. We particularly focus on special kernels, namely the generalized multiquadrics, which play an important role in the approximation of functions by radial basis functions. Moreover, we modify this approach for the use with real input data by applying fast trigonometric transforms instead of FFTs.

Ridgelets Ridgelets have been designed by Candès and Donoho [18, 20] to deal with line singularities effectively by mapping them into point singularities using the Radon transform. When implementing a discrete ridgelet transform one has to cope with certain technical difficulties. The basic strategy of the ridgelet transform is an application of the wavelet transform on the projections of the Radon transform. The Radon transform seems natural and simple on the continuum but it is a challenging problem for discrete data. Do and Vetterli [34] proposed an orthonormal version of the ridgelet transform based on a discrete Radon transform defined on the finite grid $\mathbb{Z}_p^2$, where p is a prime number. Unfortunately, the $\mathbb{Z}_p^2$ Radon transform integrates over 'lines' which are defined algebraically — due to the arithmetic modulo p — rather than geometrically. This causes a wrap-around effect, i.e., texture-like artifacts in reconstructions. Carré and Andres [22] presented a so-called discrete analytical ridgelet transform (DART) with a flexible redundancy factor based on discrete analytical lines, which only cause a limited wrap-around effect. Donoho et al. [37] have proposed an effective discrete ridgelet transform based on so-called true ridge functions using the Fast Slant Stack (FSS) [4], a pseudopolar FFT based discrete Radon transform. The essential of the FSS-based 'true' ridgelet transform is an interpolation performed using a Dirichlet kernel, which leads to a transform that is geometrically faithful and has no wrap-around effect.

In this work, we develop a discrete Radon transform based on the NFFT. As the FSS, this approach completely avoids linear interpolations. Further, we combine it with a dual-tree complex wavelet transforms (to achieve approximate shift invariance) and total variation (TV) minimization (to reduce the pseudo-Gibbs artifacts) for line-feature extraction.

Robust Approximation A popular approach to scattered data approximation is the *moving least squares method* (MLS) [46] which requires in contrast to standard interpolation methods by radial basis functions only the solution of small linear systems of equations. The size of these systems is governed by the degree of the polynomials which are reproduced by the method. Another way to look at the polynomial reproduction property is the Backus-Gilbert approach, which also offers a possibility for the fast computation of the MLS solution via the NFFT.

Interestingly, similar methods exist in image processing for smoothing noisy data. The Gaussian facet model [129] is basically the same as the MLS method. However, the MLS method and its variants have a big drawback. In their averaging process they smooth edges, since the weights are assigned only data dependent and do not depend on the distribution of the sampling nodes. This led to the development of robust estimation procedures and nonlinear filters that also data-adaptively determine the influence of each data point on the result [118]. Among the rich variety of these methods, we focus on the robust Gaussian facet model [129] developed for image processing.

Having the relation between the linear approaches in image processing and scattered data approximation in mind, we modify this robust nonlinear model in such a way that it can be also applied to scattered data. Moreover, we change the method slightly towards a generalized bilateral filter approach that does not only reproduce constants but also polynomials of higher degree.

II. Fast trigonometric transforms at nonequispaced nodes

In this chapter we mainly develop fast algorithms for the *discrete cosine transform at nonequispaced nodes (NDCT)* and for the *discrete sine transform at nonequispaced nodes (NDST)*. Our approach is based on the *fast Fourier transform at nonequispaced nodes (NFFT)* proposed by Steidl et al. [122, 111].

Let $\boldsymbol{N} := (N_1, \ldots, N_d)^T \in 2\mathbb{N}^d$ and $I_{\boldsymbol{N}}^d := \big\{-\frac{N_1}{2}, \ldots, \frac{N_1}{2}-1\big\} \times \cdots \times \big\{-\frac{N_d}{2}, \ldots, \frac{N_d}{2}-1\big\}$. As usual, let the torus $\mathbb{T}^d$ be represented by the d-dimensional unit cube $\big[-\frac{1}{2}, \frac{1}{2}\big]^d$ with opposing faces identified. For a finite number of given Fourier coefficients $\hat{f}_{\boldsymbol{k}} \in \mathbb{C}$ $(\boldsymbol{k} \in I_{\boldsymbol{N}}^d)$, the d-variate NFFT$(N_1, \ldots, N_d)$ computes approximations $\tilde{f}$ of the trigonometric polynomial

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in I_{\boldsymbol{N}}^d} \hat{f}_{\boldsymbol{k}} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}} \tag{2.1}$$

at arbitrary nodes $\boldsymbol{x}_j \in \mathbb{T}^d$ $(j = 1, \ldots, M)$. In matrix-vector notation this reads as

$$\boldsymbol{f} = \boldsymbol{A} \hat{\boldsymbol{f}}\,, \tag{2.2}$$

where

$$\boldsymbol{f} := \big(f(\boldsymbol{x}_j)\big)_{j=0}^{M}\,, \qquad \boldsymbol{A} := \big(\mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}\big)_{j=0, \boldsymbol{k} \in I_{\boldsymbol{N}}^d}^{M}\,, \qquad \hat{\boldsymbol{f}} := \big(\hat{f}_{\boldsymbol{k}}\big)_{\boldsymbol{k} \in I_{\boldsymbol{N}}^d}\,. \tag{2.3}$$

For equispaced nodes $\boldsymbol{x_j} = \big(\frac{j_1}{N_1}, \ldots, \frac{j_d}{N_d}\big)^T$ $(\boldsymbol{j} \in I_{\boldsymbol{N}}^d)$ the computation of (2.1) can be done by the well known *fast Fourier transform (FFT)* [29] in only $\mathcal{O}(|I_{\boldsymbol{N}}^d| \, \log |I_{\boldsymbol{N}}^d|)$ arithmetic operations. However, the FFT requires sampling on an equally spaced grid which represents a significant limitation in many applications. Unfortunately, for arbitrary nodes $\boldsymbol{x}_j \in \mathbb{T}^d$ $(j = 1, \ldots, M)$, the direct evaluation of (2.4) takes $\mathcal{O}(|I_{\boldsymbol{N}}^d| \, M)$ arithmetical operations, too much for practical purposes. In the following, we are looking for fast algorithms. The results were previously published in [50].

This chapter is organized as follows. We start with briefly developing the NFFT. Details can be found in [111]. For clarity of presentation the ideas behind the following algorithms will be shown for the univariate case $d = 1$ first. In Section 2, we generalize the NFFT to the multivariate setting. Our new fast algorithms for the NDCT and NDST are presented in Sections 3 and 4. A closer examination of the inverse NFFT will be given in Section 5 with particular consideration of different sampling sets, which will be essential for later applications.

1. The NFFT

Let $\boldsymbol{N} = N \in 2\mathbb{N}$ and therefore $I_N := I^1_{\boldsymbol{N}} = \{-\frac{N}{2}, \ldots, \frac{N}{2} - 1\}$. We are interested in the fast evaluation of the 1-periodic trigonometric polynomial

$$f(x) := \sum_{k \in I_N} \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} k x}. \tag{2.4}$$

Let φ be an even *window function* so that its 1-periodic version $\tilde{\varphi}(x) = \sum_{r \in \mathbb{Z}} \varphi(x+r)$ has an absolute convergent Fourier series

$$\tilde{\varphi}(x) = \sum_{k \in \mathbb{Z}} c_k(\tilde{\varphi}) \, \mathrm{e}^{-2\pi \mathrm{i} k x}$$

with the Fourier coefficients

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

We introduce the *oversampling factor* $\sigma > 1$, $n := \sigma N$, and approximate f by

$$s_1(x) := \sum_{\ell \in I_n} g_\ell \, \tilde{\varphi}\left(x - \frac{\ell}{n}\right),$$

i.e., we want to define g_ℓ such that $s_1 \approx f$. Switching to the frequency domain, one obtains

$$\begin{aligned} s_1(x) &= \sum_{k \in \mathbb{Z}} \hat{g}_k \, c_k(\tilde{\varphi}) \, \mathrm{e}^{-2\pi \mathrm{i} k x} \\ &= \sum_{k \in I_n} \hat{g}_k \, c_k(\tilde{\varphi}) \, \mathrm{e}^{-2\pi \mathrm{i} k x} + \sum_{r \in \mathbb{Z} \setminus \{0\}} \sum_{k=-n}^{n-1} \hat{g}_k \, c_{k+nr}(\tilde{\varphi}) \, \mathrm{e}^{-2\pi \mathrm{i} (k+nr) x} \end{aligned} \tag{2.5}$$

with the discrete Fourier coefficients

$$\hat{g}_k := \sum_{\ell \in I_n} g_\ell \, \mathrm{e}^{2\pi \mathrm{i} k \ell / n}, \qquad g_\ell = \frac{1}{n} \sum_{k \in I_n} \hat{g}_k \, \mathrm{e}^{-2\pi \mathrm{i} k \ell / n} \tag{2.6}$$

Suppose that the Fourier coefficients $c_k(\tilde{\varphi})$ become sufficiently small for $|k| \geq n - \frac{N}{2}$ and that $c_k(\tilde{\varphi}) \neq 0$ for $k \in I_N$. Then, comparing (2.5) with (2.4) suggests to set

$$\hat{g}_k := \hat{g}_{k+nr} = \begin{cases} \hat{f}_k / c_k(\tilde{\varphi}) & k \in I_N, \\ 0 & k \in I_n \setminus I_N, \end{cases} \tag{2.7}$$

for $r \in \mathbb{Z}$. Now the values g_ℓ can be obtained from (2.6) by a (reduced) FFT of size n. This approximation causes an aliasing error.

Assume further that φ is also well-localized in time domain such that it can be approximated by a function

$$\psi(x) = \varphi(x)\chi_{[-\frac{m}{n},\frac{m}{n}]}(x)$$

with $\operatorname{supp}\psi = [-\frac{m}{n},\frac{m}{n}]$ and *cut-off parameter* $m \in \mathbb{N}$ ($m \ll n$). Together with its 1-periodic version $\tilde{\psi}$ and with the help of the index set

$$I_{n,m}(x_j) := \{\ell \in I_n : nx_j - m \le \ell \le nx_j + m\}$$

an approximation to s_1 is defined by

$$f(x_j) \approx s_1(x_j) \approx s(x_j) := \sum_{\ell \in I_{n,m}(x_j)} g_\ell \, \tilde{\psi}\left(x_j - \frac{\ell}{n}\right). \tag{2.8}$$

For fixed $x_j \in \mathbb{T}$, the above sum contains at most $2m+1$ nonzero summands. This approximation causes a truncation error.

In summary, the NFFT approximates $f(x_j)$ by computing $s(x_j)$ via (2.7), (2.6) and (2.8) with $\mathcal{O}(n\log n + mM)$ arithmetic operations. See the next section for a short analysis of the approximation error.

2. Multivariate NFFT

Starting with the original problem of evaluating the multivariate trigonometric polynomial (2.1) we have to do a few generalizations of the ideas given in the previous section. Using the tensor product approach, the window function is now given by

$$\varphi(\boldsymbol{x}) := \varphi_1(x_1)\cdots\varphi_d(x_d),$$

where the φ_t ($t = 1,\dots,d$) are univariate window functions. Thus, a simple consequence is

$$c_{\boldsymbol{k}}(\tilde{\varphi}) = c_{k_1}(\tilde{\varphi}_1)\cdots c_{k_d}(\tilde{\varphi}_d).$$

The ansatz is generalized to

$$s_1(\boldsymbol{x}) := \sum_{\boldsymbol{\ell}\in I_{\boldsymbol{n}}^d} g_{\boldsymbol{\ell}} \tilde{\varphi}\left(\boldsymbol{x} - \left(\frac{\ell_1}{n_1},\dots,\frac{\ell_d}{n_d}\right)^T\right)$$

with $\boldsymbol{n} := \sigma\boldsymbol{N}$. Along the lines of (2.7) one defines

$$\hat{g}_{\boldsymbol{k}} := \begin{cases} \frac{\hat{f}_{\boldsymbol{k}}}{c_{\boldsymbol{k}}(\tilde{\varphi})} & \text{for } \boldsymbol{k} \in I_{\boldsymbol{N}}^d \\ 0 & \text{for } \boldsymbol{k} \in I_{\boldsymbol{n}}^d \setminus I_{\boldsymbol{N}}^d. \end{cases}$$

The values $g_{\boldsymbol{\ell}}$ can be obtained by a (multivariate) FFT of size $n_1 \times \cdots \times n_d$ as

$$g_{\boldsymbol{\ell}} = \frac{1}{n_1\cdots n_d} \sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}^d} \hat{g}_{\boldsymbol{k}} \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}(\ell_1/n_1,\dots,\ell_d/n_d)^T} \qquad (\boldsymbol{\ell} \in I_{\boldsymbol{n}}^d).$$

Assume now that φ is well localized in time domain and can be approximated by the function $\psi(\boldsymbol{x}) = \varphi(\boldsymbol{x})\chi_D(\boldsymbol{x})$ with compact support $D := [-\frac{m}{n_1}, \frac{m}{n_1}] \times \cdots \times [-\frac{m}{n_d}, \frac{m}{n_d}]$. Let $\tilde{\psi}$ again denote the 1-periodic version of ψ. One then obtains

$$s(\boldsymbol{x}_j) := \sum_{\boldsymbol{\ell} \in I^d_{\boldsymbol{n},m}(\boldsymbol{x}_j)} g_{\boldsymbol{\ell}} \tilde{\psi}\left(\boldsymbol{x} - \left(\frac{\ell_1}{n_1}, \ldots, \frac{\ell_d}{n_d}\right)^T\right),$$

where the multi-index set is given by

$$I^d_{\boldsymbol{n},m}(\boldsymbol{x}_j) := \left\{\boldsymbol{\ell} \in I^d_{\boldsymbol{n}} : n_t\big(\boldsymbol{x}_j\big)_t - m \leq \ell_t \leq n_t\big(\boldsymbol{x}_j\big)_t + m \quad \forall t\right\}.$$

The d-variate $\mathrm{NFFT}(N_1, \ldots, N_d)$ needs

$$\mathcal{O}(\sigma^d |I^d_{\boldsymbol{N}}| \log |I^d_{\boldsymbol{N}}| + m^d M)$$

arithmetic operations and its approximation error can be split as

$$|f(\boldsymbol{x}_j) - s(\boldsymbol{x}_j)| \leq |f(\boldsymbol{x}_j) - s_1(\boldsymbol{x}_j)| + |f(\boldsymbol{x}_j) - s_1(\boldsymbol{x}_j)|$$

into an aliasing error and a truncation error. To keep the error small, several window functions φ with good localization in time and frequency domain were proposed, e.g., the *Gaussian* [41, 122, 38], *cardinal central B-splines* [12, 122], *sinc functions* [88] or *Kaiser-Bessel functions* [81, 57]. A detailed analysis of the approximation errors can be found in the corresponding papers. In general the approximation error decays exponentially in m, where the basis of the exponent depends on σ. In our numerical experiments, we will focus on the Gaussian window function. Then, by [44], the error can be estimated by

$$\frac{|f(\boldsymbol{x}_\ell) - \tilde{f}(\boldsymbol{x}_\ell)|}{\sum_{\boldsymbol{k} \in I^d_{\boldsymbol{N}}} |\hat{f}_{\boldsymbol{k}}|} \leq d\, 2^{d+1}\, \mathrm{e}^{-m\pi(1-1/(2\alpha-1))}. \tag{2.9}$$

See also [88] for a numerical comparison of the NFFT with different window functions and different choices for the parameters σ and m.

In matrix-vector notation the NFFT can be described by

$$\boldsymbol{A}\hat{\boldsymbol{f}} \approx \boldsymbol{B}\boldsymbol{F}\boldsymbol{D}\hat{\boldsymbol{f}}$$

with the sparse matrix $\boldsymbol{B} := \left(\tilde{\psi}\left(\boldsymbol{x} - \left(\frac{\ell_1}{n_1}, \ldots, \frac{\ell_d}{n_d}\right)^T\right)\right)_{j=1,\boldsymbol{\ell} \in I^d_{\boldsymbol{n}}}^{M}$, the Fourier matrix $\boldsymbol{F} := \left(\mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}(j_1/n_1, \ldots, j_d/n_d)^T}\right)_{\boldsymbol{j},\boldsymbol{k} \in I^d_{\boldsymbol{n}}}$ and the 'diagonal' matrix

$$\boldsymbol{D} := \bigotimes_{t=1}^{d} \left(\mathbf{0}_t \,\middle|\, \mathrm{diag}\left(\frac{1}{c_{k_t}(\tilde{\varphi}_t)}\right)_{k_t \in I_{N_t}} \,\middle|\, \mathbf{0}_t \right)^T$$

with zero matrices $\mathbf{0}_t$ of appropriate size.

For the sake of completeness, we also sketch the *transposed version* of the transform (NFFT^T). Instead of the sum in (2.1), we are now interested in the fast computation of

$$h(\boldsymbol{k}) = \sum_{j=1}^{M} f_j \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \qquad (\boldsymbol{k} \in I_{\boldsymbol{N}}^d) \tag{2.10}$$

with $f_j \in \mathbb{C}$ and arbitrary $\boldsymbol{x}_j \in \mathbb{T}^d$ ($j = 1, \ldots, M$). In matrix-vector notation this reads as

$$\boldsymbol{h} = \boldsymbol{A}^T \boldsymbol{f}$$

with $\boldsymbol{h} := \big(h(\boldsymbol{k})\big)_{\boldsymbol{k} \in I_N}$, $\boldsymbol{f} := \big(f_j\big)_{j=1}^{M}$ and $\boldsymbol{A}$ as defined in (2.3). Therefore, computing the transposed matrix-vector product

$$\boldsymbol{h} \approx \boldsymbol{D}^T \boldsymbol{F} \boldsymbol{B}^T \boldsymbol{f} \, .$$

leads to an algorithm for the fast approximative computation of (2.10) with the same arithmetical complexity and approximation error as for the NFFT.

3. The NFCT

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

$$f^{\mathrm{C}}(x_j) = f_j^{\mathrm{C}} := \sum_{k=0}^{\nu-1} \hat{f}_k^{\mathrm{C}} \cos(2\pi k x_j) \, . \tag{2.11}$$

Choosing $N = 2\nu$ and $\hat{f}_k \in \mathbb{R}$ ($k = 0, \ldots, \nu - 1$) with $\hat{f}_k = \hat{f}_{-k}$ and $\hat{f}_{-\nu} = 0$ in equation (2.4) we obtain

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

where $\varepsilon_{\nu,0} = \varepsilon_{\nu,\nu} := 1/2$ and $\varepsilon_{\nu,k} := 1$ ($k = 1, \ldots, \nu - 1$). Consequently, we have for $\hat{f}_k^{\mathrm{C}} = 2\varepsilon_{\nu,k} \hat{f}_k$ that $f^{\mathrm{C}}(x) = f(x)$. Since $\tilde{\varphi}$ is even, we verify that $c_k(\tilde{\varphi}) = c_{-k}(\tilde{\varphi})$ and further by (2.7) that $\hat{g}_k = \hat{g}_{-k}$ ($k = 1, \ldots, \sigma\nu - 1$). Using this symmetry and (2.6), we get for $\ell = 0, \ldots, \sigma\nu$ that

$$g_\ell = \frac{1}{2\sigma\nu} \sum_{k=-\sigma\nu}^{\sigma\nu-1} \hat{g}_k \, \mathrm{e}^{-\pi \mathrm{i} k \ell / (2\sigma\nu)} = \frac{1}{\sigma\nu} \sum_{k=0}^{\sigma\nu} \varepsilon_{\sigma\nu,k} \, \hat{g}_k \cos\left(\frac{\pi k \ell}{\sigma\nu}\right) . \tag{2.12}$$

Note that $g_{2\sigma\nu r - \ell} = g_\ell$ holds for all $r \in \mathbb{Z}$. Finally, we compute as in (2.8) the sums

$$f^{\mathrm{C}}(x_j) \approx s(x_j) = \sum_{\ell \in I_{2\sigma\nu,m}(x_j)} g_\ell \, \tilde{\varphi}\left(x_j - \frac{\ell}{2\sigma\nu}\right) . \tag{2.13}$$

In summary, we obtain the following algorithm for the fast computation of (2.11) with arithmetic complexity $\mathcal{O}(\sigma\nu \log(\sigma\nu) + mM)$:

Algorithm 2.1 (NFCT).
Input: $\nu, M \in \mathbb{N}$, $\sigma > 1$, $\hat{f}_k^{\mathrm{C}} \in \mathbb{R}$ $(k = 0, \ldots, \nu - 1)$, $x_j \in [0, 1/2]$ $(j = 1, \ldots, M)$.
Precomputation: $c_k(\tilde{\varphi})$ $(k = 0, \ldots, \nu - 1)$, $\tilde{\varphi}\left(x_j - \frac{\ell}{2\sigma\nu}\right)$ $(j = 1, \ldots, M; \ell \in I_{2\sigma\nu,m}(x_j))$

1. *For* $k = 0, \ldots, \nu - 1$ *compute* $\hat{g}_k := \dfrac{\hat{f}_k^{\mathrm{C}}}{2\varepsilon_{\nu,k} c_k(\tilde{\varphi})}$ *and for* $k = \nu, \ldots, \sigma\nu$ *set* $\hat{g}_k := 0$.
2. *For* $\ell = 0, \ldots, \sigma\nu$ *compute* g_ℓ *according to* (2.12) *by a fast DCT-I of length* σn.
3. *For* $j = 1, \ldots, M$ *compute* $s(x_j)$ *by* (2.13).

Output: $s(x_j)$ *approximate values for* $f^{\mathrm{C}}(x_j)$.

4. The NFST

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

$$f^{\mathrm{S}}(x_j) = f_j^{\mathrm{S}} := \sum_{k=1}^{\nu-1} \hat{f}_k^{\mathrm{S}} \sin(2\pi k x_j)\,. \tag{2.14}$$

We consider again equation (2.4) with $N = 2\nu$ and assume that $\hat{f}_k \in \mathbb{R}$ with $\hat{f}_{-k} = -\hat{f}_k$ $(k = 1, \ldots, \nu - 1)$ and that $\hat{f}_0 = \hat{f}_{-\nu} = 0$. Then

$$f(x) = \sum_{k=-\nu}^{\nu-1} \hat{f}_k \ \mathrm{e}^{-2\pi \mathrm{i} k x} = -\mathrm{i} \sum_{k=1}^{\nu-1} 2\hat{f}_k \sin(2\pi k x).$$

Consequently, we have for $\hat{f}_k^{\mathrm{S}} = 2\hat{f}_k$ that $f^{\mathrm{S}}(x) = \mathrm{i} f(x)$. This time, equation (2.7) yields $\hat{g}_k = -\hat{g}_{-k}$ $(k = 1, \ldots, \sigma\nu - 1)$. Thus, for $\ell = 0, \ldots, \sigma\nu$ equation (2.12) becomes

$$\mathrm{i} g_\ell = \frac{\mathrm{i}}{2\sigma\nu} \sum_{k=-\sigma\nu}^{\sigma\nu-1} \hat{g}_k \ \mathrm{e}^{-\pi \mathrm{i} k \ell/(\sigma\nu)} = \frac{1}{\sigma\nu} \sum_{k=1}^{\sigma\nu-1} \hat{g}_k \sin\left(\frac{\pi k \ell}{\sigma\nu}\right). \tag{2.15}$$

Note that $g_{2\sigma\nu r - \ell} = -g_\ell$ $(r \in \mathbb{Z})$. Finally, we compute as in (2.8) the sums

$$f^{\mathrm{S}}(x_j) = \mathrm{i} f(x_j) \approx \mathrm{i} s(x_j) = \sum_{\ell \in I_{2\sigma\nu,m}(x_j)} \mathrm{i} g_\ell \, \tilde{\varphi}\left(x_j - \frac{\ell}{2\sigma\nu}\right). \tag{2.16}$$

In summary, we obtain the following algorithm for the fast computation of (2.14) with arithmetic complexity $\mathcal{O}(\sigma\nu \log(\sigma\nu) + mM)$:

Algorithm 2.2 (NFST).
Input: $n, M \in \mathbb{N}$, $\sigma > 1$, $\hat{f}_k^{\mathrm{S}} \in \mathbb{R}$ $(k = 1, \ldots, \nu - 1)$, $x_j \in [0, 1/2]$ $(j = 1, \ldots, M)$.
Precomputation: $c_k(\tilde{\varphi})$ $(k = 1, \ldots, \nu - 1)$, $\tilde{\varphi}\left(x_j - \frac{\ell}{2\sigma\nu}\right)$ $(j = 1, \ldots, M; \ell = \in I_{2\sigma\nu,m}(x_j))$

1. *For* $k = 1, \dots, \nu - 1$ *compute* $\hat{g}_k := \dfrac{\hat{f}_k^{\mathrm{S}}}{2c_k(\tilde{\varphi})}$ *and for* $k = 0, \nu, \dots, \sigma\nu$ *set* $\hat{g}_k := 0$.

2. *For* $\ell = 0, \dots, \sigma\nu$ *compute* g_ℓ *according to* (2.15) *by a fast DST-I of length* $\sigma\nu$.

3. *For* $j = 1, \dots, M$ *compute* $\mathrm{is}(x_j)$ *by* (2.16).

Output: $\mathrm{is}(x_j)$ *approximate values for* $f^{\mathrm{S}}(x_j)$.

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 $\boldsymbol{C_x} := (\cos 2\pi k x_j)_{jk}$ and $\boldsymbol{S_x} := (\sin 2\pi k x_j)_{jk}$.
In a similar way as for the NFFT [111] we can develop fast algorithms for the computation of

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

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

5. Inverse NFFT

We consider the following reconstruction or recovery problem. Given the values $y_j \in \mathbb{C}$ $(j = 1, \dots, M)$ of a trigonometric polynomial (2.1) at nonequispaced nodes $\boldsymbol{x}_j$, the aim of the *inverse NFFT (iNFFT)* is to reconstruct its Fourier coefficients $\hat{f}_{\boldsymbol{k}}$ $(\boldsymbol{k} \in I_{\boldsymbol{N}}^d)$, i.e., to solve the linear system of equations

$$y_j = \sum_{\boldsymbol{k} \in I_{\boldsymbol{N}}^d} \hat{f}_{\boldsymbol{k}} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \qquad (j = 1, \dots, M). \tag{2.17}$$

With the notation of (2.3) and $\boldsymbol{y} := \big(y_j\big)_{j=1}^{M}$ this reads in matrix-vector-notation as

$$\boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{y}\,. \tag{2.18}$$

Of course, for equally spaced nodes $\boldsymbol{x_j} = \big(\frac{j_1}{N_1}, \dots, \frac{j_d}{N_d}\big)^T$ $(\boldsymbol{j} \in I_{\boldsymbol{N}}^d)$ the iNFFT becomes an ordinary inverse FFT (iFFT) which can be easily computed.

In our applications, the number of nodes will be larger than the dimension of the space of trigonometric polynomials, i.e., $M \geq |I_{\boldsymbol{N}}^d|$, so that the linear system

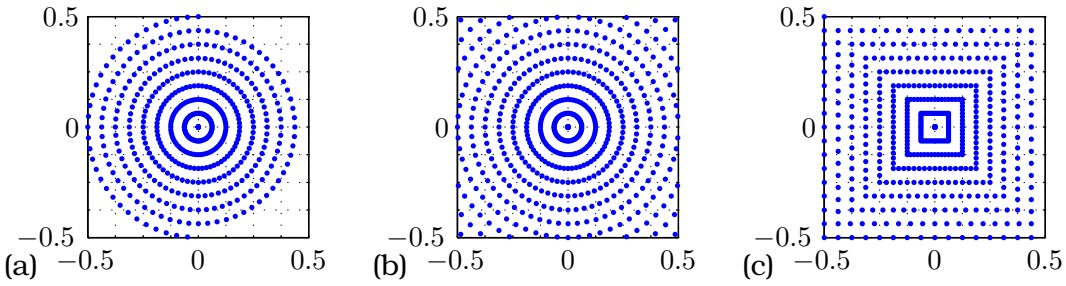


Figure 2.1.: (a) Polar grid, (b) modified polar grid, (c) linogram grid ($T = 32$, $R = 16$)

(2.17) is overdetermined. A standard method is to use the least squares approximation with sampling density compensating weights $w_j > 0$, i.e., to solve the unconstrained minimization problem

$$\left\| \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{f}} \right\|_{\boldsymbol{W}}^2 = \sum_{j=1}^{M} w_j \left| y_j - f(\boldsymbol{x}_j) \right|^2 \xrightarrow{\hat{\boldsymbol{f}}} \min \, . \tag{2.19}$$

This problem is equivalent to the weighted normal equation of the first kind

$$\boldsymbol{A}^H \boldsymbol{W} \boldsymbol{A} \hat{\boldsymbol{f}} = \boldsymbol{A}^H \boldsymbol{W} \boldsymbol{y} \, , \tag{2.20}$$

where $\boldsymbol{W} := \operatorname{diag}(w_j)_{j=1}^M$. Assuming $\boldsymbol{A}^H \boldsymbol{W} \boldsymbol{A} \approx \boldsymbol{I}$ (as it would be for equispaced nodes and $\boldsymbol{W} = \operatorname{diag}(\frac{1}{N_1 \dots N_d})_{j=1}^M$), multiplication with the weight-matrix and the adjoint matrix can serve as first approximation $\hat{\boldsymbol{f}} \approx \boldsymbol{A}^H \boldsymbol{W} \boldsymbol{y}$.

A theoretical consideration for underdetermined systems (2.17) can be found in [89].

For the numerical solution of (2.20), the NFFT software package [88] provides a factorized variant of the conjugated gradients method (CGNR, N for 'Normal equation', R for 'Residual minimization'). This fast iterative CG-type algorithm involves the NFFT for the fast matrix-vector multiplications in the CG steps.

The crucial point for the fast convergence of this iterative method is the distribution of the nodes $\boldsymbol{x}_j$. In view of the applications we have in mind, i.e., the discrete Radon transform in connection with the discrete Ridgelet transform in Chapter VI, we investigate special grid structures of nodes $\boldsymbol{x}_j$ in 2D and restrict ourselves to the case $\boldsymbol{N} = (N, N)^T$ with $N \in 2\mathbb{N}$.

Polar grid

The points of the *polar grid* lie on concentric circles around the origin. Thus, they are given by a radius $r_j := \frac{j}{R}$ ($j \in I_R$) and an angle $\theta_t := \frac{\pi}{T} t$ ($t \in I_T$) as

$$\boldsymbol{x}_{t,j} := r_j \boldsymbol{\theta}_t \, ,$$

where $\boldsymbol{\theta}_t := (\cos \theta_t, \sin \theta_t)^T$. The total number of points is

$$M = TR \, .$$

As you can see in Figure 2.1(a), the points of the polar grid leave out the corners of the unit square. This is the reason why the reconstruction properties for this grid are limited and visible artifacts are left (see our numerical examples at the end of this section). Therefore, we modify this grid as follows.

Modified polar grid

In order to fill the corners, we add more concentric circles and throw away those points not located in the unit square, i.e., $\tilde{R} := \lceil \sqrt{2}R \rceil$ and

$$\boldsymbol{x}_{t,j} := r_j \boldsymbol{\theta}_t$$

with r_j and $\boldsymbol{\theta}_t$ as before, but now let $j \in I_{\tilde{R}}$, cp. Figure 2.1(b).

The number of points for the modified polar grid can be estimated as follows. Since the concentration of points decreases with increasing distance from the origin, the number of points does not increase proportional to the area. But the points are equally distributed on the rays. So we can instead 'measure' the rays. Each ray of the polar grid is of length $\frac{1}{2}$. So we have a total length of

$$\int_0^{2\pi} \frac{1}{2} \, \mathrm{d}\theta = \pi \,.$$

Let $\theta \in [0, \pi/4]$. Then the ray of the modified polar grid at angle θ is of length $\frac{1}{2\cos\theta}$. Because of symmetry, the total length of the rays is

$$8 \int_0^{\pi/4} \frac{1}{2\cos\theta} \, \mathrm{d}\theta = 4\log(1+\sqrt{2}) \,.$$

So the ratio of these two length is $\frac{4}{\pi}\log(1+\sqrt{2}) \approx 1.122$ and therefore the number of points for the modified polar grid is about

$$M \approx \frac{4}{\pi}\log(1+\sqrt{2})TR \,.$$

But the number of points for the modified polar grid is not optimal with respect to the rate of convergence of the iNFFT (see the convergence rates of our numerical examples at the end of this section). In discrete settings the following grid has turned out to be more suitable.

Linogram grid

Instead of concentric circles, the points of the *linogram* or *pseudopolar grid* lie on concentric squares around the origin. Thus, they are given by a slope and an

intercept. Depending on the angle, we distinguish two sets of points. These are

$$\boldsymbol{x}^h_{t,j} := \Big(\frac{j}{R}, \frac{4t}{T}\frac{j}{R}\Big)^T$$
$$\boldsymbol{x}^v_{t,j} := \Big(-\frac{4t}{T}\frac{j}{R}, \frac{j}{R}\Big)^T$$

where $j \in I_R$ and $t \in I_{T/2}$, cp. Figure 2.1(c). Together, the number of points for the linogram grid is

$$M = TR\,.$$

Choice of the parameters T and R

For equally spaced nodes, the Fourier matrix $\boldsymbol{A}$ in (2.18) is orthogonal (except for normalisation) and therefore has condition number 1. In order to achieve a small condition number and therefore a good reconstruction with our special non equally spaced grids, too, we will oversample the domain. Thus, we choose the parameters T and R such that the sampling density is at least that of the equally spaced grid, i.e.,

$$\Delta x_1 \leq \frac{1}{N}\,, \qquad \Delta x_2 \leq \frac{1}{N}\,. \tag{2.21}$$

Our numerical examples at the end of this section show that this is a reasonable choice. For theoretical considerations of the condition number of $\boldsymbol{A}$ in the non equally spaced case see [5].

As already mentioned above, the polar grid is not suitable for the iNFFT. For the modified polar grid, we have with $x_1 = r\cos\theta$ that

$$\Delta x_1 = \left|\frac{\partial x_1}{\partial r}\right| \Delta r = |\cos\theta|\Delta r \leq \Delta r$$

and

$$\Delta x_1 = \left|\frac{\partial x_1}{\partial \theta}\right| \Delta\theta = |r|\ |\sin\theta|\Delta\theta \leq \max|r|\ \Delta\theta$$

(analog for Δx_2). With the choice $r_j = \frac{j}{R}$ ($j \in I_{\tilde{R}}$) and $\theta_t = \frac{\pi}{T}t$ ($t \in I_T$), it holds that $\Delta r = \frac{1}{R}$, $\max|r| = \frac{\sqrt{2}}{2}$ and $\Delta\theta = \frac{\pi}{T}$. Therefore, in order to satisfy the conditions in (2.21), we have to choose

$$R \geq N \qquad \text{and} \qquad T \geq \frac{\sqrt{2}}{2}\pi N\,.$$

For (the first set of points of) the linogram grid with $x_1 = \frac{j}{R}$ ($j \in I_R$), we have

$$\Delta x_1 = \frac{1}{R}$$

and with $x_2 = \frac{4t}{T}x_1$ ($t \in I_T$) it holds with $\max|x_1| = \frac{1}{2}$ that

$$\Delta x_2 = \left|\frac{\partial x_2}{\partial t}\right| \Delta t = \frac{4}{T}\ |x_1| \leq \frac{2}{T}$$

(analog for the second set of points). Thus, we have to choose in this case

$$R \geq N \qquad \text{and} \qquad T \geq 2N\,.$$

Choice of the weights

Weights are introduced in equation (2.19) to compensate sampling density. For every point in the sampling set, we therefore associate a small surrounding area.

In case of the (modified) polar grid, we have small ring segments. The area of such a ring segment around $\boldsymbol{x}_{t,j}$ ($j \neq 0$) is

$$\begin{aligned} w_{t,j} :=& \frac{\pi}{2T}\left(\left(|r_j| + \frac{\Delta r}{2}\right)^2 - \left(|r_j| - \frac{\Delta r}{2}\right)^2\right) \\ =& \frac{\pi}{T}\Delta r\,|r_j| \\ =& \frac{\pi}{TR^2}\,|j|\ . \end{aligned}$$

The area of the small circle of radius $\frac{1}{2R}$ around the origin is $\frac{\pi}{4R^2}$. Divided by the multiplicity of the origin in the sampling set, we get

$$w_{t,0} := \frac{\pi}{4TR^2}\,.$$

For a point $\boldsymbol{x}^h_{t,j}$ ($j \neq 0$) of the linogram grid we use small surrounding trapezoids. The area is

$$\begin{aligned} w^h_{t,j} :=& \frac{2\Delta x_1}{TR}\left(\left(|t| + \frac{1}{2}\right)\left(|j| - \frac{1}{2}\right) - \left(|t| - \frac{1}{2}\right)\left(|j| - \frac{1}{2}\right)\right. \\ &\left. + \left(|t| + \frac{1}{2}\right)\left(|j| + \frac{1}{2}\right) - \left(|t| - \frac{1}{2}\right)\left(|j| + \frac{1}{2}\right)\right) \\ =& \frac{4}{TR^2}\,|j| \end{aligned}$$

(analog for a point $\boldsymbol{x}^v_{t,j}$). Around the origin we have a small square of side length $\frac{1}{R}$. Divided by the multiplicity of the origin in the sampling set, the area is

$$w^h_{t,0} := \frac{1}{TR^2}\,.$$

Remark 5.1. Another possible choice for the weights associated to the points of the grids are Voronoi weights [87]. However, our numerical tests showed that better results can be achieved with our analytical weights.

Numerical examples

The following numerical examples were computed with the NFFT C-subroutine library [88], where we chose Kaiser-Bessel window functions with $m = 4$ and oversampling factor $\sigma = 2$.

Figure 2.2(a) shows the Shepp-Logan phantom of size 256×256 with values in $[0, 1]$ (Matlab-function `phantom(256)`). We now interpret the gray values of the image as Fourier coefficients $\hat{f}_{\boldsymbol{k}}$ given on the grid $I^2_N = [-128, 127]^2$, set $\boldsymbol{y} := \boldsymbol{A}\hat{\boldsymbol{f}}$ and then solve the system $\boldsymbol{A}\tilde{\boldsymbol{f}} = \boldsymbol{y}$ to obtain $\tilde{\boldsymbol{f}}$.

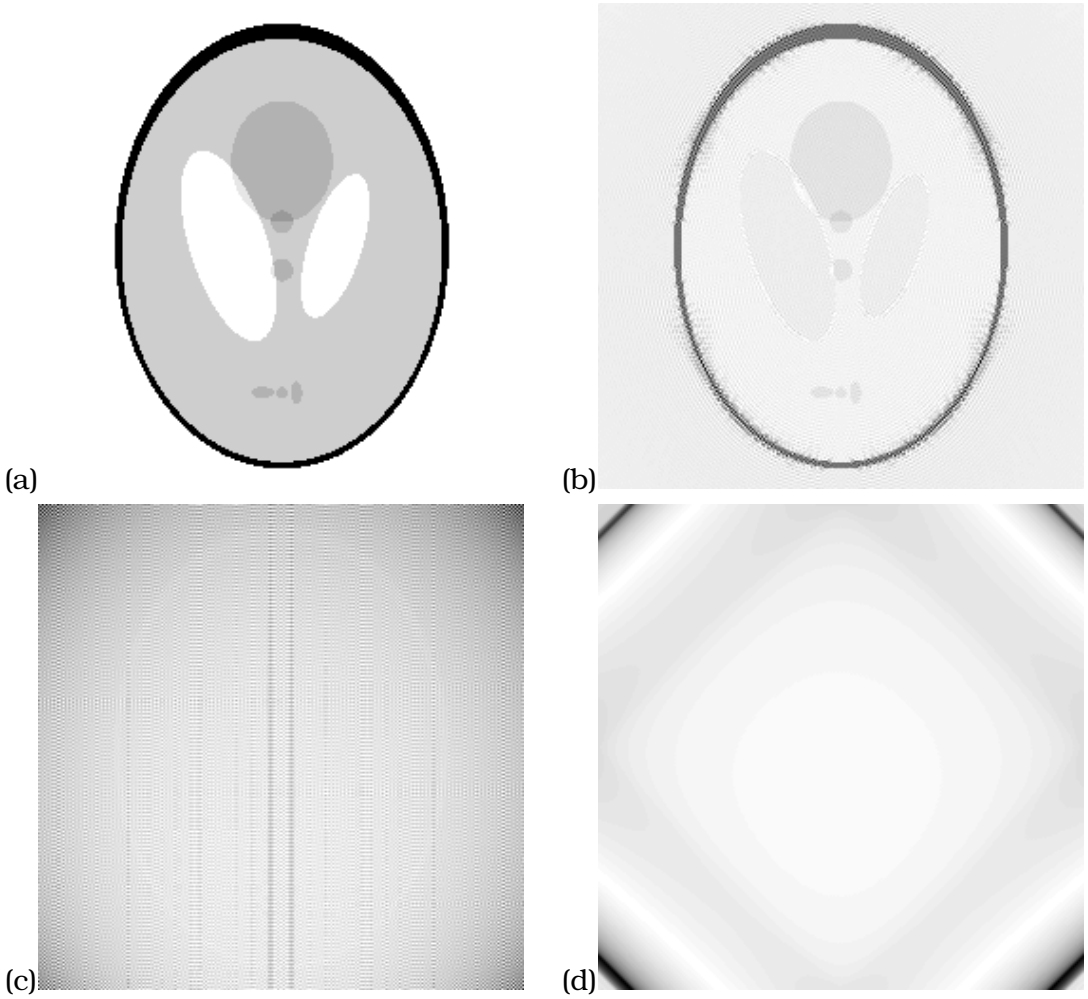


Figure 2.2.: (a) Shepp-Logan-Phantom ($N = 256$); absolute error of reconstruction by adjoint transform with $T = 2.5N$, $R = 1.5N$ for (b) polar grid ($E_\infty = 0.4437$), (c) modified polar grid ($E_\infty = 0.0104$), (d) linogram grid ($E_\infty = 0.0739$).

	M	iterations	E_∞
polar grid	245760	0	$4.4374 \cdot 10^{-01}$
		500	$2.2890 \cdot 10^{-01}$
		1000	$2.2670 \cdot 10^{-01}$
modified polar grid	275810	0	$1.0401 \cdot 10^{-02}$
		80	$1.1626 \cdot 10^{-06}$
		145	$1.1906 \cdot 10^{-12}$
linogram grid	245760	0	$7.3870 \cdot 10^{-02}$
		5	$1.1285 \cdot 10^{-06}$
		10	$1.1804 \cdot 10^{-12}$

Table 2.1.: Comparison of iterative reconstruction of the Shepp-Logan phantom with different grids ($N = 256$, $T = 2.5N$, $R = 1.5N$).

In our first test, we compute a straightforward approximation by multiplication with the weight-matrix and the adjoint. This can be done with the iNFFT algorithm by only doing the precomputation with $\hat{\boldsymbol{f}}_0 = \mathbf{0}$ and no iteration step. As can be seen in Figure 2.2(b), with the nodes chosen from the polar grid, there is still much of the detail left in the absolute error. For the modified polar grid, Figure 2.2(c), and the linogram grid, Figure 2.2(d), no detail is left in the error of the reconstruction, but structural errors can be found.

Table 2.1 compares the results of the conjugated gradients method for the different grids. As a measure for convergence, we use the maximal absolute error

$$E_\infty(\tilde{\boldsymbol{f}}) := \max_{\boldsymbol{k} \in I_N^2} |\hat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}| \,.$$

No convergence is achieved when using the polar grid. The convergence with the modified polar grid is very slow compared to the linogram grid, even though the number of nodes is about 12% larger.

Remark 5.2. The rate of convergence can be improved considerably, especially in case of the modified polar grid, by taking advantage of the fact that the Shepp-Logan phantom only consits of values inside a circle of radius $\frac{N}{2}$, i.e. $\hat{f}_{\boldsymbol{k}} = 0$ for $\|\boldsymbol{k}\| > \frac{N}{2}$. By setting all values outside this circle to zero in every iteration step, we get $E_\infty \approx 10^{-6}$ after only 7 steps and $E_\infty \approx 10^{-12}$ after 17 steps.

III. Fast Fourier transform at nonequispaced nodes on hyperbolic cross points

In multivariate approximation one has to deal with the so called 'curse of dimensionality', i.e., the number of degrees of freedom for representing an approximation of a function with a prescribed accuracy depends exponentially on the dimensionality of the considered problem. This obstacle can be circumvented to some extend by the interpolation on sparse grids and the related approximation on hyperbolic cross points in the Fourier domain, see, e.g., [136, 120, 17]. The basic idea is as follows. Instead of approximating the function

$$g(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} \hat{g}_{\boldsymbol{k}} \ \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}}$$

on the standard tensor product grid $\left\{\boldsymbol{k} = (k_1, \ldots, k_d)^T \in \mathbb{Z}^d : |k_1|, \ldots, |k_d| < N\right\}$ with $\mathcal{O}(N^d)$ degrees of freedom, it can be approximated with only $\mathcal{O}(N \log^{d-1} N)$ degrees of freedom from the set of hyperbolic cross points $\left\{\boldsymbol{k} = (k_1, \ldots, k_d)^T \in \mathbb{Z}^d : (1 + |k_1|) \cdots (1 + |k_d|) < N\right\}$. Under certain conditions, the corresponding approximation error deteriorates only by a factor of $\log^{d-1} N$, see Theorems 1.1 and 1.2.

The fast evaluation of trigonometric polynomials with equispaced nodes in space and frequency domain can be computed by the FFT in only $\mathcal{O}(N^d \log N)$ arithmetic operations [29]. If the frequencies are chosen from a hyperbolic cross and the spatial nodes lie on a sparse grid there exist fast algorithms of arithmetical complexity $\mathcal{O}(N \log^d N)$ [6, 78]. Recently, the FFT has been generalised by the NFFT which requires $\mathcal{O}(N^d \log N + M)$ arithmetic operations for the evaluation of a trigonometric polynomial at M arbitrary nodes, see Chapter II. In this chapter, we present an algorithm for the fast evaluation of trigonometric polynomials with frequencies from the hyperbolic cross, where in contrast to [6, 78], the spatial nodes can be chosen arbitrarily. We will call this algorithm sparse NFFT (SNFFT). The results are previously published in [48].

The outline of this chapter is as follows. The first section is devoted to the basic notation and delivers short insight to the results in the field of hyperbolic Fourier approximation. In Section 2, we show how the NFFT can be coupled with hyperbolic crosses. The main idea consists in an appropriate partitioning of the index set and the application of the NFFT to the resulting blocks. In Section 3, we define the two dimensional hyperbolic cross and its block partition for the SNFFT. Fast algorithms for the sparse discrete cosine and sine transforms at nonequispaced spatial nodes in two dimensions are given in Section 4. In Section 5, we introduce a modified three dimensional hyperbolic cross which easily can be partitioned into

blocks again. Finally, Section 6 presents numerical examples and a discussion of the results.

1. Underlying notation and results

The following notation and results are essentially taken from [119], see also [32].

The counterpart to the Schwartz space $S(\mathbb{R}^d)$ for periodic functions is defined by

$$D(\mathbb{T}^d) := \Big\{ f \in C^\infty(\mathbb{T}^d) : \|f\|_{\boldsymbol{\ell}} := \max_{\boldsymbol{x}\in\mathbb{T}^d} \Big| \frac{\partial^{|\boldsymbol{\ell}|}}{\partial x_1^{\ell_1} \cdots \partial x_d^{\ell_d}} f(\boldsymbol{x}) \Big| < \infty \text{ for all } \boldsymbol{\ell} \in \mathbb{N}_0^d \Big\},$$

where $C^\infty(\mathbb{T}^d)$ denotes the space of functions that are differentiable for all degrees of differentiation. Its dual will be denoted by $D'(\mathbb{T}^d)$ and its elements are often called *tempered periodic distributions*.

We denote the Hilbert space of square integrable complex valued functions on $\mathbb{T}^d$ by

$$L^2(\mathbb{T}^d) := \Big\{ f\colon \mathbb{T}^d \to \mathbb{C} : \|f\|_{L^2(\mathbb{T}^d)} := \Big(\int_{\mathbb{T}^d} |f(\boldsymbol{x})|^2 \,\mathrm{d}\boldsymbol{x} \Big)^{1/2} < \infty \Big\}.$$

Its inner product is given by

$$\langle f, g \rangle_{L^2(\mathbb{T}^d)} := \int_{\mathbb{T}^d} f(\boldsymbol{x}) \overline{g(\boldsymbol{x})} \,\mathrm{d}\boldsymbol{x}. \tag{3.1}$$

Functions $g \in L^2(\mathbb{T}^d)$ can be interpreted by

$$g(f) := \int_{\mathbb{T}^d} g(\boldsymbol{x}) f(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \qquad (f \in D(\mathbb{T}^d)) \tag{3.2}$$

as elements of $D'(\mathbb{T}^d)$ and therefore

$$D(\mathbb{T}^d) \subset L^2(\mathbb{T}^d) \subset D'(\mathbb{T}^d).$$

The *Fourier coefficients* of a distribution $g \in D'(\mathbb{T}^d)$ are definded for all $\boldsymbol{k} \in \mathbb{Z}^d$ by

$$c_{\boldsymbol{k}}(g) := g\big(\mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \cdot} \big).$$

By equations (3.1) and (3.2) we can write for functions $f \in L^2(\mathbb{T}^d)$

$$c_{\boldsymbol{k}}(f) = \langle f, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \cdot} \rangle_{L^2(\mathbb{T}^d)} = \int_{\mathbb{T}^d} f(\boldsymbol{x}) \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}} \,\mathrm{d}\boldsymbol{x} \qquad (\boldsymbol{k} \in \mathbb{Z}^d) \tag{3.3}$$

and the set of functions $\{ \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \cdot} \}_{\boldsymbol{k}\in\mathbb{Z}^d}$ forms an orthonormal basis of $L^2(\mathbb{T}^d)$.

The Hilbert space of complex valued sequences over $\mathbb{Z}^d$ will be denoted by

$$\ell_2(\mathbb{Z}^d) := \Big\{ (c_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathbb{Z}^d} : \big\|(c_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathbb{Z}^d}\big\|_{\ell_2} := \Big(\sum_{\boldsymbol{k}\in\mathbb{Z}^d} |c_{\boldsymbol{k}}|^2 \Big)^{1/2} < \infty \Big\}.$$

By *Parseval's equality* it holds

$$\|f\|_{L^2(\mathbb{T}^d)} = \big\|\big(c_{\boldsymbol{k}}(f)\big)_{\boldsymbol{k}\in\mathbb{Z}^d}\big\|_{\ell_2}.$$

For $a \in \mathbb{R}$, we define the L^2-*Sobolev space* $H^a(\mathbb{T}^d)$ by

$$H^a(\mathbb{T}^d) := \Big\{ f \in D'(\mathbb{T}^d) : \|f\|_{H^a(\mathbb{T}^d)} := \Big\| \sum_{\boldsymbol{k}\in\mathbb{Z}^d} \big(1+\|\boldsymbol{k}\|_2^2\big)^{a/2} c_{\boldsymbol{k}}(f)\, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\cdot} \Big\|_{L^2(\mathbb{T}^d)} \Big\}.$$

With Parseval's equality it follows

$$\|f\|_{H^a(\mathbb{T}^d)} = \big\|\big((1+\|\boldsymbol{k}\|_2^2)^{a/2} c_{\boldsymbol{k}}(f)\big)_{\boldsymbol{k}\in\mathbb{Z}^d}\big\|_{\ell_2}.$$

The *Korobov space* of order $a \in \mathbb{R}$ is defined as

$$E^a(\mathbb{T}^d) := \Big\{ f \in D'(\mathbb{T}^d) : \big|c_{\boldsymbol{k}}(f)\big| = \mathcal{O}\Big(\big((1+|k_1|)\dots(1+|k_d|)\big)^{-a}\Big), \|\boldsymbol{k}\|_2 \to \infty \Big\}.$$

It holds for $m \in \mathbb{N}$ that

$$C^{m+1}([0,1]^d) \cap C^{m-1}(\mathbb{T}^d) \subseteq E^{m+1}(\mathbb{T}^d).$$

Furthermore, we have for $a \geq 0$ that

$$H^{2a}(\mathbb{T}^d) \subset E^a(\mathbb{T}^d)$$

and for $b < a - \frac{1}{2}$ that

$$E^a(\mathbb{T}^d) \subset H^b(\mathbb{T}^d).$$

Univariate Fourier approximation

Let the *discrete Fourier coefficients* be defined by

$$c_k^N(f) := \frac{1}{N} \sum_{\ell\in I_N} f\Big(\frac{\ell}{N}\Big)\, \mathrm{e}^{-2\pi\mathrm{i}k\ell/N} \qquad (k \in I_N).$$

The connection between Fourier coefficients and discrete Fourier coefficients can be expressed by the *aliasing formula*

$$c_k^N(f) = c_k(f) + \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} c_{k+rN}(f). \tag{3.4}$$

The univariate Fourier approximation operator is given as

$$\begin{aligned} L_N f(x) &:= \sum_{k\in I_N} c_k^N(f)\, \mathrm{e}^{2\pi\mathrm{i}kx} \\ &= \sum_{\ell\in I_N} f\Big(\frac{\ell}{N}\Big) \underbrace{\frac{1}{N} \sum_{k\in I_N} \mathrm{e}^{-2\pi\mathrm{i}k(x-\ell/N)}}_{=:\Lambda_N(x-\ell/N)} \end{aligned}$$

with the fundamental interpolant Λ_N. It involves the frequencies from I_N and interpolates at the spatial points from

$$\mathcal{T}_N := \Big\{ \frac{\ell}{N} : \ell \in I_N \Big\}.$$

Interpolation on tensor product grids

Let $\boldsymbol{N} := (N, \dots, N)^T \in 2\mathbb{N}^d$. Following the tensor product approach, the *discrete Fourier coefficients* are now defined by

$$c_{\boldsymbol{k}}^N(f) := \frac{1}{N^d} \sum_{\boldsymbol{\ell} \in I_{\boldsymbol{N}}^d} f\Big(\frac{\boldsymbol{\ell}}{N}\Big)\, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{\ell}/N} \qquad (\boldsymbol{k} \in I_{\boldsymbol{N}}^d)$$

and the *multivariate aliasing formula* is given by

$$c_{\boldsymbol{k}}^N(f) = c_{\boldsymbol{k}}(f) + \sum_{\substack{\boldsymbol{r} \in \mathbb{Z}^n \\ \boldsymbol{r} \neq \boldsymbol{0}}} c_{\boldsymbol{k}+\boldsymbol{r}N}(f)\,. \tag{3.5}$$

For the tensor product interpolation operator $L_{\boldsymbol{N}} := L_N^{x_1} \cdots L_N^{x_d}$, where $L_N^{x_t}$ denotes the univariate Fourier approximation along the x_tth coordinate, we get

$$\begin{aligned} L_{\boldsymbol{N}} f(\boldsymbol{x}) &:= \sum_{\boldsymbol{k} \in I_{\boldsymbol{N}}^d} c_{\boldsymbol{k}}^N(f)\, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} \\ &= \sum_{\boldsymbol{\ell} \in I_{\boldsymbol{N}}^d} f\Big(\frac{\boldsymbol{\ell}}{N}\Big) \underbrace{\frac{1}{N^d} \sum_{\boldsymbol{k} \in I_{\boldsymbol{N}}^d} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}(\boldsymbol{x} - \boldsymbol{\ell}/N)}}_{=: \Lambda_{\boldsymbol{N}}(\boldsymbol{x} - \boldsymbol{\ell}/N)} \end{aligned}$$

with the fundamental interpolant $\Lambda_{\boldsymbol{N}}$. It involves N^d frequencies from the tensor product grid $I_{\boldsymbol{N}}^d$ and interpolates at the N^d spatial points of the tensor product grid

$$\mathcal{T}_{\boldsymbol{N}}^d := \mathcal{T}_N \times \cdots \times \mathcal{T}_N\,.$$

The approximation error can be estimated by the following special case of Theorem 2.23 in [119, p. 55].

Theorem 1.1. *Let $f \in H^a(\mathbb{T}^d)$, $a \geq 0$. Then*

$$\|f - L_N f\|_{L^2(\mathbb{T}^d)} \leq C N^{-a} \|f\|_{H^a(\mathbb{T}^d)}$$

with a constant C independent of N and a.

Interpolation on sparse grids

Let $J \in \mathbb{N}_0$, $N = 2^J$ and $N_r := 2^r$, $r = 0, \dots, J$. The sparse interpolation operator B_J is defined as the *Boolean sum* of univariate interpolation operators

$$B_J := \bigoplus_{\substack{r_1, \dots, r_d \in \mathbb{N}_0^d, \\ r_1 + \cdots + r_d = J}} L_{N_{r_1}}^{x_1} \cdots L_{N_{r_d}}^{x_d}\,.$$

It can also be written as difference of usual sums, e.g., in the bivariate case we can write

$$B_J = \sum_{r=0}^{J} L_{N_r}^{x_1} L_{N_{J-r}}^{x_2} - \sum_{r=0}^{J-1} L_{N_r}^{x_1} L_{N_{J-r-1}}^{x_2}\,.$$

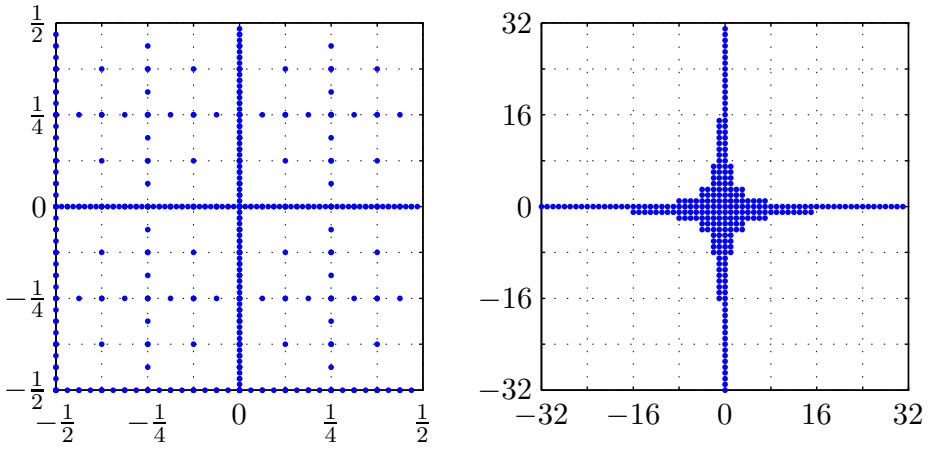


Figure 3.1.: Sparse grid and hyperbolic cross in 2D for $N = 2^6$.

Therefore, it interpolates on the $\mathcal{O}(N \log^{d-1} N)$ spatial points of the *sparse grid*

$$\mathcal{S}_J^d := \bigcup_{\substack{r_1,\dots,r_d \in \mathbb{N}_0^d, \\ r_1+\cdots+r_d=J}} \mathcal{T}_{N_{r_1}} \times \cdots \times \mathcal{T}_{N_{r_d}}$$

with $\mathcal{O}(N \log^{d-1} N)$ frequencies from the *hyperbolic cross*

$$H_J^d := \bigcup_{\substack{r_1,\dots,r_d \in \mathbb{N}_0^d, \\ r_1+\cdots+r_d=J}} I_{N_{r_1}} \times \cdots \times I_{N_{r_d}} .$$

The sparse grid and the hyperbolic cross are depicted in Figure 3.1 for $d = 2$ and $N = 2^6$.

An estimate for the approximation error is given by the following special case of Theorem 2.24 [119, p. 57].

Theorem 1.2. *Let $f \in E^a(\mathbb{T}^d)$, $a > 1$. Then*

$$\|f - B_J f\|_{L^2(\mathbb{T}^d)} \leq C N^{-a} \log^{d-1} N \|f\|_{E^a(\mathbb{T}^d)}$$

with a constant C independent of N and a.

Comparing this result with Theorem 1.1, we see, that the interpolation error deteriorates only by a logarithmic factor, whereas the number of used frequencies is only $\mathcal{O}(N \log^{d-1} N)$ instead of N^d in the case of the full tensor product grid.

Remark 1.3. In [119] a more general error analysis is done for functions from so called spaces of dominating mixed smoothness and for a wider class of fundamental interpolants satisfying some Strang-Fix-conditions [119, Def. 2.1]. These conditions are trivially satisfied by our fundamental interpolants Λ_N being trigonometric polynomials.

2. Coupling NFFT with hyperbolic crosses

Our new fast algorithm for the evaluation of trigonometric polynomials from hyperbolic crosses at arbitrary nodes is based on the application of the NFFT to an appropriate partitioning of the hyperbolic cross.

Our aim is the fast approximate evaluation of the trigonometric polynomial

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in H_J^d} \hat{f}_{\boldsymbol{k}} \; \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}} \tag{3.6}$$

at arbitrary nodes $\boldsymbol{x}_j \in \mathbb{T}^d$ ($j = 1, \ldots, M$) and for a hyperbolic cross H_J^d which can be partitioned into blocks of indices $H_J^d = \bigcup_r (I^d_{\boldsymbol{N}_r} + \boldsymbol{\rho}_r)$ with frequency shifts $\boldsymbol{\rho}_r \in \mathbb{Z}^d$. Then, the sum in (3.6) can be split up according to the blocks as

$$f(\boldsymbol{x}_j) = \sum_r \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\rho}_r \boldsymbol{x}_j} \sum_{\boldsymbol{k} \in I^d_{\boldsymbol{N}_r}} \hat{f}_{\boldsymbol{k}+\boldsymbol{\rho}_r} \; \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \tag{3.7}$$

with the 'nonuniform twiddle factors' $\mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\rho}_r \boldsymbol{x}_j}$. Now we apply the NFFT of size $|I^d_{\boldsymbol{N}_r}|$ on every block. Due to the triangle inequality, the overall error remains bounded by (2.9). The number of arithmetic operations on every block is $\mathcal{O}(\sigma^d |I^d_{\boldsymbol{N}_r}| \log |I^d_{\boldsymbol{N}_r}| + m^d M)$. So our main task consists in the construction of an adequate partition of the hyperbolic cross with only few blocks which leads to a fast overall algorithm.

Remark 2.1. We would like to emphasise a technical detail concerning NFFTs of short size. Obviously, NFFTs with small $N_1, \ldots, N_d$, should be computed directly. The case where only few N_i are small needs more care. We exemplify our solution for $d = 2$ and $N_1 \leq m < N_2$. Splitting up the sum in equation (2.1) into both dimensions yields

$$f(\boldsymbol{x}_j) = f\big((\boldsymbol{x}_j)_1, (\boldsymbol{x}_j)_2\big) = \sum_{k_1 \in I^1_{N_1}} \Big(\sum_{k_2 \in I^1_{N_2}} \hat{f}_{k_1,k_2} \; \mathrm{e}^{-2\pi \mathrm{i} k_2 (\boldsymbol{x}_j)_2} \Big) \, \mathrm{e}^{-2\pi \mathrm{i} k_1 (\boldsymbol{x}_j)_1} \, . \tag{3.8}$$

Now the computation can be done in a total of $\mathcal{O}\big(N_1(\sigma N_2 \log N_2 + mM)\big)$ arithmetic operations by a one dimensional NFFT for the inner bracket, followed by a direct computation of the outer sum.

3. NFFT on hyperbolic cross points – the bivariate case

Let $J \in \mathbb{N}$ and $N = 2^J$. For the sake of simplicity, let $J \geq 2$. Anyway, the consideration of the NFFT on the hyperbolic crosses H_0^d and H_1^d is unnecessary.

We define the following index sets as building blocks for our partitioning of the hyperbolic crosses in two and three dimensions. For $r \in \mathbb{N}_0$, let

$$\begin{array}{lcl} H_r^- & := & \{ \quad -2^{r+1}, \ldots, -2^r - 1 \quad \}, \\ H_r^0 & := & \{ \; -\lfloor 2^{r-1} \rfloor, \ldots, \lceil 2^{r-1} \rceil - 1 \; \}, \\ H_r^+ & := & \{ \quad 2^r, \ldots, 2^{r+1} - 1 \quad \}, \end{array}$$

where $\lfloor x \rfloor := \max\{k \in \mathbb{Z} : k \leq x\}$ and $\lceil x \rceil := \min\{k \in \mathbb{Z} : k \geq x\}$. Obviously, the sets H_r^- and H_r^+ are just shifted versions of H_r^0, i.e.

$$H_r^- = -\left\lceil \tfrac{3}{2}\, 2^r \right\rceil + H_r^0 \qquad \text{and} \qquad H_r^+ = \left\lfloor \tfrac{3}{2}\, 2^r \right\rfloor + H_r^0 \,. \tag{3.9}$$

Furthermore, we have that

$$\left|H_r^-\right| = \left|H_r^0\right| = \left|H_r^+\right| = 2^r \qquad \text{and} \qquad H_{r_1}^0 \times \cdots \times H_{r_d}^0 = I^d_{(N_{r_1},\ldots,N_{r_d})^T} \tag{3.10}$$

for $N_{r_t} := 2^{r_t}$ and $r_1, \ldots, r_d \in \mathbb{N}_0$.

Let us now define the blocks of the hyperbolic cross in 2D. For the levels $r = 0, \ldots, \lceil \frac{J}{2} \rceil - 1$, let

$$\begin{aligned}
H_{J,r}^{\text{right}} &:= H_r^0 \times H_{J-r-2}^+, \\
H_{J,r}^{\text{top}} &:= H_{J-r-2}^+ \times H_r^0, \\
H_{J,r}^{\text{left}} &:= H_r^0 \times H_{J-r-2}^-, \\
H_{J,r}^{\text{bottom}} &:= H_{J-r-2}^- \times H_r^0, \\
H_{J,r} &:= H_{J,r}^{\text{right}} \cup H_{J,r}^{\text{top}} \cup H_{J,r}^{\text{left}} \cup H_{J,r}^{\text{bottom}}, \\
H_J^{\text{centre}} &:= H^0_{\lfloor \frac{J}{2} \rfloor} \times H^0_{\lfloor \frac{J}{2} \rfloor}.
\end{aligned}$$

Now the partition of the two dimensional hyperbolic cross is given by the disjoint union

$$H_J^2 = H_J^{\text{centre}} \cup \bigcup_{r=0}^{\lceil \frac{J}{2} \rceil - 1} H_{J,r} \,.$$

In Figure 3.2, the blocks are depicted for $J = 2$, 3, 4 and 5. Since the cardinalities for these index sets are

$$\left|H_{J,r}^{\text{right}}\right| = \left|H_{J,r}^{\text{top}}\right| = \left|H_{J,r}^{\text{left}}\right| = \left|H_{J,r}^{\text{bottom}}\right| = 2^{J-2}\,, \qquad \left|H_{J,r}^{\text{centre}}\right| = 2^{2\lfloor \frac{J}{2} \rfloor}\,,$$

the total number of hyperbolic cross points is $|H_J^2| = (J+2)2^{J-1}$, compared to $|I^2_{(N,N)^T}| = N^2 = 4^J$ indices for the full tensor product grid.

Following equation (3.7), we are interested in the computation of

$$f(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in H_J^{\text{centre}}} \hat{f}_{\boldsymbol{k}}\ \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} + \sum_{r=0}^{\lceil \frac{J}{2} \rceil - 1} \sum_{\boldsymbol{k} \in H_{J,r}} \hat{f}_{\boldsymbol{k}}\ \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \tag{3.11}$$

at arbitrary nodes $\boldsymbol{x}_j \in \mathbb{T}^2$ ($j = 1, \ldots, M$). We start by computing the centre block, i.e., the first sum in (3.11) by a bivariate NFFT$(2^{\lfloor \frac{J}{2} \rfloor}, 2^{\lfloor \frac{J}{2} \rfloor})$ with arithmetic complexity $\mathcal{O}(J2^J + M)$. Next we consider the following sums

$$f_{J,r}^{\text{label}}(\boldsymbol{x}_j) := \sum_{\boldsymbol{k} \in H_{J,r}^{\text{label}}} \hat{f}_{\boldsymbol{k}}\ \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}\,,$$

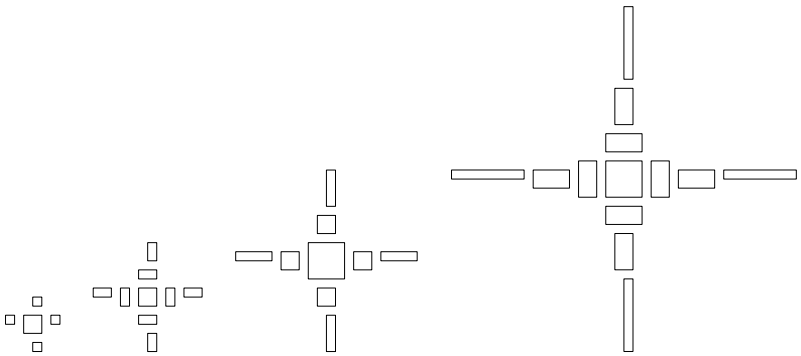


Figure 3.2.: Partition of the hyperbolic cross in 2D for $J = 2, \ldots, 5$.

where $\mathrm{label} \in \{\mathrm{right}, \mathrm{top}, \mathrm{left}, \mathrm{bottom}\}$. We explain the computation of a left block. By using equation (3.9), we obtain

$$\begin{aligned}\sum_{\boldsymbol{k}\in H^{\mathrm{left}}_{J,r}} \hat{f}_{\boldsymbol{k}}\ \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j} &= \sum_{k_1\in H^0_r}\sum_{k_2\in H^-_{J-r-2}} \hat{f}_{k_1,k_2}\,\mathrm{e}^{-2\pi\mathrm{i}\left(k_1(\boldsymbol{x}_j)_1+k_2(\boldsymbol{x}_j)_2\right)}\\ &= \mathrm{e}^{2\pi\mathrm{i}\lceil\frac{3}{2}2^{J-r-2}\rceil(\boldsymbol{x}_j)_2}\sum_{\boldsymbol{k}\in H^0_r\times H^0_{J-r-2}} \hat{f}_{k_1,k_2-\lceil\frac{3}{2}2^{J-r-2}\rceil}\ \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\,.\end{aligned}$$

Due to (3.10) each of these blocks can be computed by a bivariate NFFT$(2^r, 2^{J-r-2})$ with arithmetic complexity $\mathcal{O}(J2^J + M)$, see also Remark 2.1.

Since the number of blocks is $\mathcal{O}(J)$, the overall complexity for computing $f(\boldsymbol{x}_j)$ for $j = 1, \ldots, M$ is $\mathcal{O}(J^2 2^J + JM)$. We refer to the following algorithm on hyperbolic cross points as sparse NFFT (SNFFT).

Algorithm 3.1 (SNFFT 2D)**.**
Input: $J \in \mathbb{N}_0$, $\hat{f}_{\boldsymbol{k}} \in \mathbb{C}$ *for* $\boldsymbol{k} \in H^2_J$,
$M \in \mathbb{N}$, $\boldsymbol{x}_j \in \mathbb{T}^2$ *for* $j = 1, \ldots, M$.

1. *Compute the values*

$$\tilde{f}(\boldsymbol{x}_j) = \sum_{\boldsymbol{k}\in H^{\mathrm{centre}}_J} \hat{f}_{k_1,k_2}\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}$$

by a bivariate NFFT$(2^{\lfloor\frac{J}{2}\rfloor}, 2^{\lfloor\frac{J}{2}\rfloor})$.

2. For $r = 0, \ldots, \lceil \frac{J}{2} \rceil - 1$ *compute*

$$\begin{aligned}
\tilde{f}(\boldsymbol{x}_j) &= \tilde{f}(\boldsymbol{x}_j) \\
&+ \mathrm{e}^{-2\pi \mathrm{i} \lfloor \frac{3}{2} 2^{J-r-2} \rfloor (\boldsymbol{x}_j)_2} \sum_{\boldsymbol{k} \in H_r^0 \times H_{J-r-2}^0} \hat{f}_{k_1, k_2 + \lfloor \frac{3}{2} 2^{J-r-2} \rfloor} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \\
&+ \mathrm{e}^{-2\pi \mathrm{i} \lfloor \frac{3}{2} 2^{J-r-2} \rfloor (\boldsymbol{x}_j)_1} \sum_{\boldsymbol{k} \in H_{J-r-2}^0 \times H_r^0} \hat{f}_{k_1 + \lfloor \frac{3}{2} 2^{J-r-2} \rfloor, k_2} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \\
&+ \mathrm{e}^{2\pi \mathrm{i} \lceil \frac{3}{2} 2^{J-r-2} \rceil (\boldsymbol{x}_j)_2} \sum_{\boldsymbol{k} \in H_r^0 \times H_{J-r-2}^0} \hat{f}_{k_1, k_2 - \lceil \frac{3}{2} 2^{J-r-2} \rceil} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \\
&+ \mathrm{e}^{2\pi \mathrm{i} \lceil \frac{3}{2} 2^{J-r-2} \rceil (\boldsymbol{x}_j)_1} \sum_{\boldsymbol{k} \in H_{J-r-2}^0 \times H_r^0} \hat{f}_{k_1 - \lceil \frac{3}{2} 2^{J-r-2} \rceil, k_2} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}
\end{aligned}$$

by four bivariate NFFT$(2^r, 2^{J-r-2})$*.*

Output: $\tilde{f}(\boldsymbol{x}_j)$ *approximate value of* $f(\boldsymbol{x}_j)$ $(j = 1, \ldots, M)$*.*

Algorithm 3.1 reads in matrix-vector notation as

$$\boldsymbol{A}_J \hat{\boldsymbol{f}} = \left[\boldsymbol{A}_{J,0} \mid \boldsymbol{A}_{J-1,1} \mid \ldots \mid \boldsymbol{A}_{J, \lceil \frac{J}{2} \rceil - 1} \mid \boldsymbol{A}_J^{\mathrm{centre}} \right] \left[\begin{array}{c} \hat{\boldsymbol{f}}_{J,0} \\ \hline \vdots \\ \hline \hat{\boldsymbol{f}}_J^{\mathrm{centre}} \end{array} \right],$$

where the sub-matrices are given by

$$\boldsymbol{A}_{J,r} := \left[\boldsymbol{A}_{J,r}^{\mathrm{right}} \mid \boldsymbol{A}_{J,r}^{\mathrm{top}} \mid \boldsymbol{A}_{J,r}^{\mathrm{left}} \mid \boldsymbol{A}_{J,r}^{\mathrm{bottom}} \right], \qquad \boldsymbol{A}_{J,r}^{\mathrm{label}} := \left(\mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \right)_{j=1,\ldots,M; \boldsymbol{k} \in H_{J,r}^{\mathrm{label}}}$$

for label $\in \{\mathrm{right}, \mathrm{top}, \mathrm{left}, \mathrm{bottom}\}$.

4. NDCT and NDST on hyperbolic cross points in 2D

Similar algorithms can be constructed for the discrete cosine transform and sine transform, based on the fast discrete cosine and sine transforms for nonequispaced nodes, developed in Chapter II. The bivariate fast cosine transform at nonequispaced nodes NFCT(N_1, N_2) computes approximations of

$$f(\boldsymbol{x}_j) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \hat{f}_{k_1,k_2} \, \cos\left(2\pi k_1 (\boldsymbol{x}_j)_1\right) \cos\left(2\pi k_2 (\boldsymbol{x}_j)_2\right)$$

and the bivariate fast sine transform at nonequispaced nodes NFST(N_1, N_2) computes approximations of

$$f(\boldsymbol{x}_j) = \sum_{k_1=1}^{N_1-1} \sum_{k_2=1}^{N_2-1} \hat{f}_{k_1,k_2} \, \sin\left(2\pi k_1 (\boldsymbol{x}_j)_1\right) \sin\left(2\pi k_2 (\boldsymbol{x}_j)_2\right)$$

at arbitrary nodes $\boldsymbol{x}_j \in [0, \frac{1}{2}]^2$ ($j = 1, \ldots, M$).

A coupling with hyperbolic crosses can be done as follows. Again let $N = 2^J$. Here we use the index sets depicted in Figure 3.3. For $r = 0, \ldots, J$ define

$$H'_{J,r} := \{0, \ldots, 2^{J-r} - 1\} \times \{\lfloor 2^{r-1} \rfloor, \ldots, 2^r - 1\}, \qquad H'_J := \bigcup_{r=0}^{J} H'_{J,r}.$$

The cardinalities for these index sets are

$$\left|H'_{J,0}\right| = 2^J, \quad \left|H'_{J,r}\right| = 2^{J-1}, \quad \left|H'_J\right| = (J+2)\, 2^{J-1}.$$

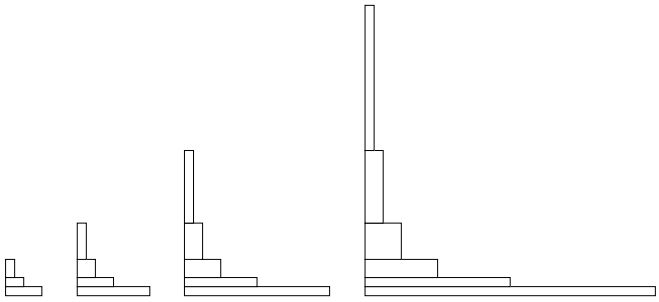


Figure 3.3.: Hyperbolic cross points for the NFCT in 2D for $J = 2, \ldots, 5$.

Then the sparse NFCT can be computed by

$$\begin{aligned}
f(\boldsymbol{x}_j) &= \sum_{(k_1,k_2)^T \in H'_J} \hat{f}_{k_1,k_2} \cos(2\pi k_1 (\boldsymbol{x}_j)_1) \cos(2\pi k_2 (\boldsymbol{x}_j)_2) \\
&= \sum_{r=0}^{J} \sum_{(k_1,k_2)^T \in H'_{J,r}} \hat{f}_{k_1,k_2} \cos(2\pi k_1 (\boldsymbol{x}_j)_1) \cos(2\pi k_2 (\boldsymbol{x}_j)_2) \\
&= \sum_{r=0}^{J} \cos(\lfloor 2^{r-1} \rfloor 2\pi (\boldsymbol{x}_j)_2) \sum_{k_1=0}^{2^{J-r}-1} \sum_{k_2=0}^{2^{r-1}-1} \hat{f}_{k_1,k_2+2^{r-1}} \cos(2\pi k_1 (\boldsymbol{x}_j)_1) \cos(2\pi k_2 (\boldsymbol{x}_j)_2) \\
&+ \sum_{r=1}^{J} \sin(\lfloor 2^{r-1} \rfloor 2\pi (\boldsymbol{x}_j)_2) \sum_{k_1=0}^{2^{J-r}-1} \sum_{k_2=0}^{2^{r-1}-1} \hat{f}_{k_1,k_2+2^{r-1}} \cos(2\pi k_1 (\boldsymbol{x}_j)_1) \sin(2\pi k_2 (\boldsymbol{x}_j)_2).
\end{aligned}$$

Using the fast algorithms from Chapter II we obtain an overall arithmetic complexity of $\mathcal{O}(J^2 2^J + JM)$.

5. NFFT on hyperbolic cross points – the trivariate case

Let us consider the hyperbolic cross in three dimensions. For $r = 1, \ldots, J-1$, we define the index sets

$$\begin{aligned}
H^{\text{front}}_{J,0} &:= H^2_J \times \{0\}, & H^{\text{front}}_{J,r} &:= H^2_{J-r-1} \times H^+_{r-1}, \\
H^{\text{rear}}_{J,0} &:= H^2_{J-1} \times \{-1\}, & H^{\text{rear}}_{J,r} &:= H^2_{J-r-1} \times H^-_{r-1}.
\end{aligned}$$

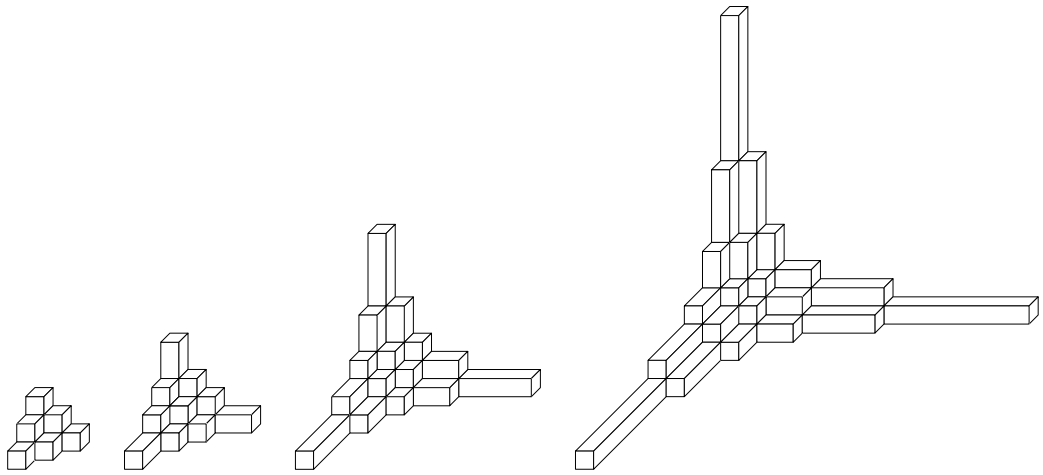


Figure 3.4.: Hyperbolic cross points in 3D for $J = 2, \ldots, 5$, only the part k_1, k_2, $k_3 \leq 0$ is shown.

A patition of the three dimensional hyperbolic cross is now given by the disjoint union

$$H_J^3 = \bigcup_{r=0}^{J-1} H_{J,r}^{\text{front}} \cup \bigcup_{r=0}^{J-1} H_{J,r}^{\text{rear}},$$

cf. Figure 3.4. The total number of hyperbolic cross points is

$$\left|H_J^3\right| = \left|H_J^2\right| + \left|H_{J-1}^2\right| + \sum_{r=1}^{J-1} \left|H_{J-r-1}^2\right| \Big(\left|H_{r-1}^+\right| + \left|H_{r-1}^-\right| \Big) = 2^{J-3}\big(J^2 + 7J + 8\big).$$

The arithmetic complexity of the resulting algorithm is $\mathcal{O}(J^3 2^J + J^2 M)$, since for $r = 1, \ldots, J-1$ we have to compute $\mathcal{O}(J - r)$ trivariate NFFTs with complexity $\mathcal{O}(J2^J + M)$ each. Unfortunately, this algorithm has drawbacks: we have to compute NFFTs for $\mathcal{O}(J^2)$ blocks of our partition. Thus, the (asymptotic) arithmetic complexity for an equal number of nodes and Fourier coefficients $M = |H_J^3|$ is $\mathcal{O}(J^4 2^J)$, i.e., not optimal. Furthermore, the second part of the NFFTs for the blocks is the most time consuming part for interesting problem sizes J.

Therefore, we use a simplification $\tilde{H}_J^3$ of the hyperbolic cross with $H_J^3 \subset \tilde{H}_J^3 \subset I^3_{(N,N,N)^T}$, which can easily be partitioned into only $\mathcal{O}(J)$ blocks but has a total

number of $\mathcal{O}(2^{\frac{3}{2}J})$ points. For $r = 0, \ldots, \lceil \frac{J}{2} \rceil - 1$, we define the following index sets

$$\begin{aligned}
\tilde{H}^{\text{top}}_{J,r} &:= H^{+}_{J-r-2} \times H^{0}_{r} \times H^{0}_{r}\,, \\
\tilde{H}^{\text{left}}_{J,r} &:= H^{0}_{r} \times H^{+}_{J-r-2} \times H^{0}_{r}\,, \\
\tilde{H}^{\text{front}}_{J,r} &:= H^{0}_{r} \times H^{0}_{r} \times H^{+}_{J-r-2}\,, \\
\tilde{H}^{\text{bottom}}_{J,r} &:= H^{-}_{J-r-2} \times H^{0}_{r} \times H^{0}_{r}\,, \\
\tilde{H}^{\text{right}}_{J,r} &:= H^{0}_{r} \times H^{-}_{J-r-2} \times H^{0}_{r}\,, \\
\tilde{H}^{\text{rear}}_{J,r} &:= H^{0}_{r} \times H^{0}_{r} \times H^{-}_{J-r-2}\,, \\
\tilde{H}_{J,r} &:= \tilde{H}^{\text{left}}_{J,r} \cup \tilde{H}^{\text{right}}_{J,r} \cup \tilde{H}^{\text{top}}_{J,r} \cup \tilde{H}^{\text{bottom}}_{J,r} \cup \tilde{H}^{\text{front}}_{J,r} \cup \tilde{H}^{\text{rear}}_{J,r}\,.
\end{aligned}$$

The centre block is given by

$$\tilde{H}^{\text{centre}}_{J} := H^{0}_{\lfloor \frac{J}{2} \rfloor} \times H^{0}_{\lfloor \frac{J}{2} \rfloor} \times H^{0}_{\lfloor \frac{J}{2} \rfloor}\,.$$

The cardinalities for these index sets are

$$\left|\tilde{H}^{\text{left}}_{J,r}\right| = \left|\tilde{H}^{\text{right}}_{J,r}\right| = \left|\tilde{H}^{\text{top}}_{J,r}\right| = \left|\tilde{H}^{\text{bottom}}_{J,r}\right| = \left|\tilde{H}^{\text{front}}_{J,r}\right| = \left|\tilde{H}^{\text{rear}}_{J,r}\right| = 2^{J+r-2}\,, \qquad \left|\tilde{H}^{\text{centre}}_{J}\right| = 2^{3\lfloor \frac{J}{2} \rfloor}\,.$$

The *modified three dimensional hyperbolic cross* $\tilde{H}^3_J$ is given by

$$\tilde{H}^3_J := \tilde{H}^{\text{centre}}_{J} \cup \bigcup_{r=0}^{\lceil \frac{J}{2} \rceil - 1} \tilde{H}_{J,r}\,,$$

cf. Figure 3.5. Thus, the total number of hyperbolic cross points is $|\tilde{H}^3_J| = 2^{J-2} 6 (2^{\lceil \frac{J}{2} \rceil} - 1) + 2^{3\lfloor \frac{J}{2} \rfloor}$, compared to $|I^3_{(N,N,N)^T}| = N^3 = 8^{J+2}$ indices for the full tensor product grid.

Similar to equation (3.11), we are now interested in the computation of

$$f(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in \tilde{H}^{\text{centre}}_J} \hat{f}_{\boldsymbol{k}}\ \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} + \sum_{r=0}^{\lceil \frac{J}{2} \rceil - 1} \sum_{\boldsymbol{k} \in \tilde{H}_{J,r}} \hat{f}_{\boldsymbol{k}}\ \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}$$

at arbitrary nodes $\boldsymbol{x}_j \in \mathbb{T}^3$ ($j = 1, \ldots, M$). We compute each of the blocks as in the two dimensional case and end up with the following algorithm of arithmetic complexity $\mathcal{O}(J 2^{J + \lceil \frac{J}{2} \rceil} + JM)$.

Algorithm 3.2 (SNFFT 3D)**.**
Input: $J \in \mathbb{N}_0$, $\hat{f}_{\boldsymbol{k}} \in \mathbb{C}$ *for* $\boldsymbol{k} \in \tilde{H}^3_J$,
$M \in \mathbb{N}$, $\boldsymbol{x}_j \in \mathbb{T}^3$ *for* $j = 1, \ldots, M$.

1. *Compute the values*

$$\tilde{f}(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in H^0_{\lfloor \frac{J}{2} \rfloor} \times H^0_{\lfloor \frac{J}{2} \rfloor} \times H^0_{\lfloor \frac{J}{2} \rfloor}} \hat{f}_{k_1,k_2,k_3}\, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}$$

by a trivariate NFFT$(2^{\lfloor \frac{J}{2} \rfloor}, 2^{\lfloor \frac{J}{2} \rfloor}, 2^{\lfloor \frac{J}{2} \rfloor})$.

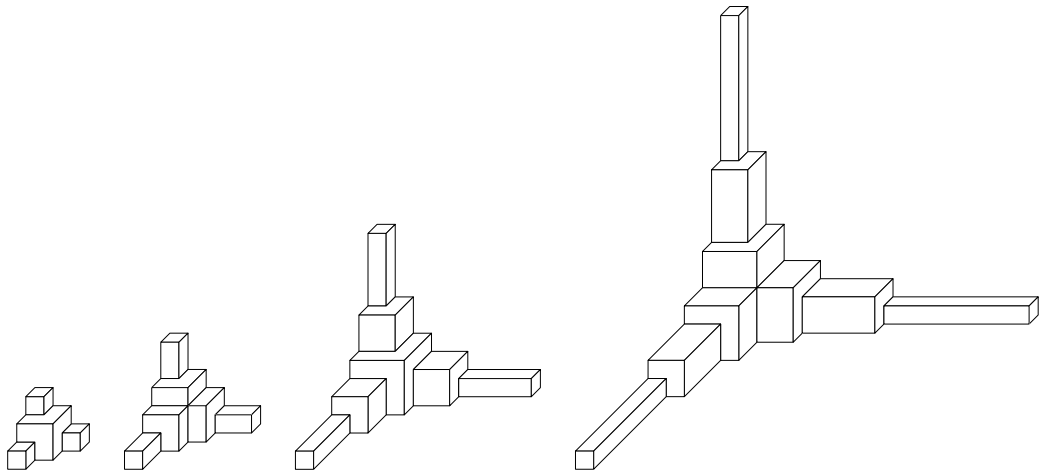


Figure 3.5.: Modified hyperbolic cross in 3D for $J = 2, \ldots, 5$, only the part k_1, k_2, $k_3 \leq 0$ is shown.

2. *For* $r = 0, \ldots, \lceil \frac{J}{2} \rceil - 1$ *compute*

$$
\begin{aligned}
\tilde{f}(\boldsymbol{x}_j) \;=\;& \tilde{f}(\boldsymbol{x}_j) \\
&+ \mathrm{e}^{2\pi \mathrm{i} \lceil \frac{3}{2} 2^{J-r-2} \rceil (\boldsymbol{x}_j)_1} \sum_{\boldsymbol{k} \in H^0_{J-r-2} \times H^0_r \times H^0_r} \hat{f}_{k_1 - \lceil \frac{3}{2} 2^{J-r-2} \rceil, k_2, k_3} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \\
&+ \mathrm{e}^{2\pi \mathrm{i} \lceil \frac{3}{2} 2^{J-r-2} \rceil (\boldsymbol{x}_j)_2} \sum_{\boldsymbol{k} \in H^0_r \times H^0_{J-r-2} \times H^0_r} \hat{f}_{k_1, k_2 - \lceil \frac{3}{2} 2^{J-r-2} \rceil, k_3} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \\
&+ \mathrm{e}^{2\pi \mathrm{i} \lceil \frac{3}{2} 2^{J-r-2} \rceil (\boldsymbol{x}_j)_3} \sum_{\boldsymbol{k} \in H^0_r \times H^0_r \times H^0_{J-r-2}} \hat{f}_{k_1, k_2, k_3 - \lceil \frac{3}{2} 2^{J-r-2} \rceil} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \\
&+ \mathrm{e}^{-2\pi \mathrm{i} \lfloor \frac{3}{2} 2^{J-r-2} \rfloor (\boldsymbol{x}_j)_1} \sum_{\boldsymbol{k} \in H^0_{J-r-2} \times H^0_r \times H^0_r} \hat{f}_{k_1 + \lfloor \frac{3}{2} 2^{J-r-2} \rfloor, k_2, k_3} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \\
&+ \mathrm{e}^{-2\pi \mathrm{i} \lfloor \frac{3}{2} 2^{J-r-2} \rfloor (\boldsymbol{x}_j)_2} \sum_{\boldsymbol{k} \in H^0_r \times H^0_{J-r-2} \times H^0_r} \hat{f}_{k_1, k_2 + \lfloor \frac{3}{2} 2^{J-r-2} \rfloor, k_3} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j} \\
&+ \mathrm{e}^{-2\pi \mathrm{i} \lfloor \frac{3}{2} 2^{J-r-2} \rfloor (\boldsymbol{x}_j)_3} \sum_{\boldsymbol{k} \in H^0_r \times H^0_r \times H^0_{J-r-2}} \hat{f}_{k_1, k_2, k_3 + \lfloor \frac{3}{2} 2^{J-r-2} \rfloor} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}
\end{aligned}
$$

by six trivariate NFFT$(2^r, 2^r, 2^{J-r-2})$.

Output: $\tilde{f}(\boldsymbol{x}_j)$ *approximate value of* $f(\boldsymbol{x}_j)$ $(j = 1, \ldots, M)$.

6. Numerical results

Our algorithms were implemented in C and tested on an AMD Athlon™XP 2700+ with 2GB main memory, SuSe-Linux (kernel 2.4.20-4GB-athlon, gcc 3.3) using

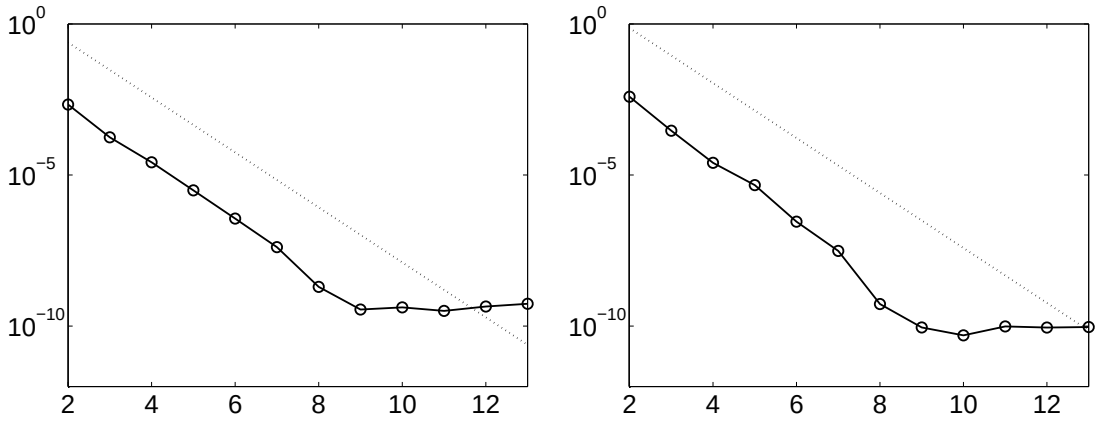


Figure 3.6.: The error E'_∞ (solid) and the error estimate given in (2.9) (dashed). Left: Algorithm 3.1 with $J = 9$ and $m = 2, \ldots, 13$. Right: Algorithm 3.2 with $J = 7$ and $m = 2, \ldots, 13$.

double precision arithmetic. Further, we have used the libraries FFTW 3.0.1 [60] and NFFT 2.0.1 [88]. In the following, we compare Algorithm 3.1 and Algorithm 3.2 with the straightforward summation (3.6), denoted by SNDFT (sparse nonequispaced discrete Fourier transform) and with the 'ordinary' NFFT where $N = 2^J$ and all Fourier coefficients with an index not in the sets H_J^2 and $\tilde{H}_J^3$, respectively, are set to zero. We have chosen random nodes $\boldsymbol{x}_j \in [-\frac{1}{2}, \frac{1}{2}]^d$ and random Fourier coefficients $\hat{f}_{\boldsymbol{k}} \in \{a + b\mathrm{i} : a, b \in [0, 1]\}$.

All tests use an oversampling factor of $\sigma = 2$ and the Gaussian window function. We precomputed the Gaussian at 10^5 equispaced evaluations points. Then, a linear interpolation scheme is used during the NFFT to compute an actual value of the window function at a certain node $\boldsymbol{x}_j$.

First, we examine the error caused by the various approximations within the SNFFT in Algorithms 3.1 and 3.2. The relative error

$$E'_\infty := \frac{|f(\boldsymbol{x}_j) - \tilde{f}(\boldsymbol{x}_j)|}{\sum_{\boldsymbol{k} \in H_{\boldsymbol{N}}^d} |\hat{f}_{\boldsymbol{k}}|} \tag{3.12}$$

is shown in Figure 3.6. According to the estimate in (2.9), the error decays exponentially as m increases. Due to our approximation of the Gaussian window function, it saturates at a level of 10^{-10}.

Next, we are interested in computation times and memory requirements. Here, we choose the number of evaluation nodes equal to the number of Fourier coefficients on the hyperbolic cross, i.e., $M = (J + 2)2^{J-1}$ for $d = 2$ and $M = 2^{J-2}6(2^{\lceil \frac{J}{2} \rceil} - 1) + 2^{3\lfloor \frac{J}{2} \rfloor}$ for $d = 3$. We compare the computation time and the memory requirements of the SNFFT, of the straightforward summation SNDFT, and of the 'ordinary' NFFT. Table 3.1 shows the theoretical CPU-time and the memory requirements. The actually required CPU times of all three algorithms are shown in Figure 3.7. As expected, the SNFFT outperforms the other algorithms. So we

	$d=2$		$d=3$	
algorithm	time	memory	time	memory
NFFT	$J2^{2J}$	2^{2J}	$J2^{3J}$	2^{3J}
SNDFT	$J^2 2^{2J}$	$J2^J$	2^{3J}	$2^{\frac{3}{2}J}$
SNFFT	$J^2 2^J$	$J2^J$	$J2^{\frac{3}{2}J}$	$2^{\frac{3}{2}J}$

Table 3.1.: Theoretical order of magnitude for CPU-time and memory requirements.

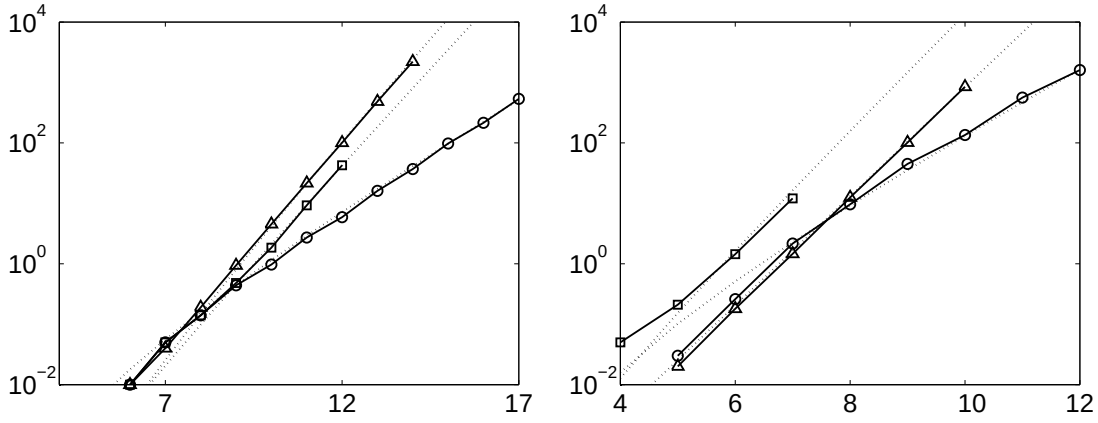


Figure 3.7.: Elapsed CPU-time in seconds (solid) and theoretical orders of magnitude given in Table 3.1 (dashed) for the SNFFT (circle), SNDFT (triangle), and NFFT (square). Left: Algorithm 3.1 with $J = 4, \ldots, 17$, $m = 4$. Right: Algorithm 3.2 with $J = 4, \ldots, 12$, $m = 4$.

obtain, e.g., for $d = 2$, $J = 14$ and $M = |H_{14}^2| = 131072$, a CPU-time of 37 seconds for the SNFFT compared to 37 minutes for the SNDFT.

In the second test, we face the memory requirements of all three algorithms as shown in Figure 3.8. Here, the SNFFT needs only a constant amount of 2 MByte more for precomputations than the SNDFT.

The numerical results show the superiority of the proposed algorithms with respect to computing time, whereas the memory requirements and the approximation error remain bounded.

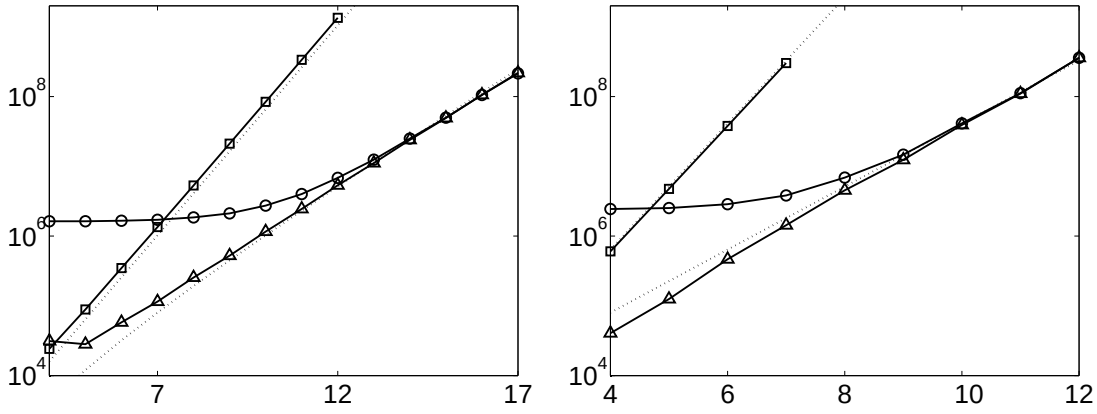


Figure 3.8.: Memory requirements in bytes (solid) and theoretical orders of magnitude given in Table 3.1 (dashed) for the SNFFT (circle), SNDFT (triangle), and NFFT (square). Left: Algorithm 3.1 with $J = 4, \ldots, 17$, $m = 4$. Right: Algorithm 3.2 with $J = 4, \ldots, 12$, $m = 4$.

IV. FMM and $\mathcal{H}$-matrices: a unified approach to the basic idea

This chapter gives a short introduction to a fundamental algorithm for the fast multiplication of a vector by a fully populated matrix $\boldsymbol{M} = (m_{jk})_{j,k=1}^{N}$ which of course must have some special properties. Otherwise the straightforward matrix-vector multiplication requires $\mathcal{O}(N^2)$ arithmetic operations. In literature the considered algorithm appears under three names, namely *fast multipole method* (FMM), *fast mosaic-skeleton matrix multiplication* and *fast $\mathcal{H}$-matrix multiplication*. Each of these approaches shows some special features mainly due to the applications the authors had in mind, but the basic ideas coincide.

The *FMM* with arithmetic complexity $\mathcal{O}(N)$ and its slower variant, the hierarchical multipole method with arithmetic complexity $\mathcal{O}(N \log N)$ were designed by Greengard and Rokhlin [63, 65] for the particle simulation in $\mathbb{R}^d$. Here

$$m_{j,k} = K(\boldsymbol{x}_j - \boldsymbol{x}_k),$$

where K is the radial function (isotropic kernel) $K(\boldsymbol{x} - \boldsymbol{y}) = \log \|\boldsymbol{x} - \boldsymbol{y}\|$ if $d = 2$ and $K(\boldsymbol{x} - \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|^{-1}$ if $d = 3$. Greengard and other authors have also used the method for the fast Gauss transform, where K is the Gaussian [68, 69] and for many other large-scale matrix computations [3, 26, 66, 23, 28, 64, 67, 116]. Further the FMM was adapted to other radial basis functions arising in the approximation of curves and surfaces by Beatson, Light and co-workers [8, 7, 9, 27].

Tyrtyshnikov et al. [127, 128, 62] have designed algorithms for fast $\mathcal{O}(N \log N)$ matrix-vector multiplications from a linear algebraic point of view. Tyrtyshnikov calls the idea behind the algorithm 'mosaic-skeleton approximation' of $\boldsymbol{M}$ and refers to [130] for an early appearance of the idea. Here the matrix coefficients are $m_{j,k} = K(\boldsymbol{x}_j, \boldsymbol{x}_k)$ where the kernel has to be a modified asymptotically smooth function [16].

Hackbusch et al. [71, 75, 74, 73, 76, 72] have created the concept of $\mathcal{H}$-matrices, where $\mathcal{H}$ abbreviates 'hierarchical'. It includes the concept of *panel clustering* earlier developed by Hackbusch and co-workers in order to solve boundary integral equations in an efficient numerical way [77, 70]. The matrix entries arise from a collocation or Galerkin approach and have, e.g., the form

$$m_{j,k} = \int_{\Omega_k} \int_{\Omega_j} K(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y} \,,$$

where K is the same kernel as in the particle simulation. The original algorithm is of arithmetic complexity $\mathcal{O}(N \log N)$. In case of $\mathcal{H}^2$-matrices one can develop

an $\mathcal{O}(N)$ algorithm if in addition a so-called 'consistency condition' is fulfilled. The idea coincides with those of the FMM. We mention that the whole $\mathcal{H}$-matrix concept is not restricted to fast matrix-vector multiplications but includes also fast $\mathcal{H}$-matrix inversions via Schur complement methods.

Although we restrict our attention to FMM-like algorithms we like to mention the existence of other algorithms for the fast matrix-vector multiplication which don't fit into the FMM/$\mathcal{H}$-matrix/mosaic-skeleton-matrix concept:

- *Wavelet methods* [2, 13, 80] are based on an approximation of $\boldsymbol{M}$ by

  $$\boldsymbol{M} \approx \tilde{\boldsymbol{W}} \boldsymbol{S} \boldsymbol{W} ,$$

  where the vector multiplications with the wavelet transform matrices $\tilde{\boldsymbol{W}}, \boldsymbol{W}$ require only $\mathcal{O}(N)$ arithmetic operations and where $\boldsymbol{S}$ is a sparse matrix containing only $\mathcal{O}(N \log N)$ nonzero elements.

  Note that the wavelet method works without the explicit knowledge of K. For a completely discrete approach see [85].

- In Chapter V, we will present a method based on the NFFT. Here, one can find an approximation of $\boldsymbol{M}$ by

  $$\boldsymbol{M} \approx \boldsymbol{B}_y \boldsymbol{T} \boldsymbol{B}_x , \tag{4.1}$$

  where the vector multiplications with the sparse matrices $\boldsymbol{B}_y$ and $\boldsymbol{B}_x$ require only $\mathcal{O}(N)$ arithmetic operations and where $\boldsymbol{T}$ is a Toeplitz matrix which can be multiplied by a vector with $\mathcal{O}(N \log N)$ arithmetic operations [109, 110].

- Beylkin et al. [14] have suggested an algorithm based on two-scale relations of scaling functions arising in wavelet theory or subdivision schemes. This algorithm is closely related to the NFFT based algorithm, in particular it can be written in the form (4.1), see [109].

In the following we want to describe the basic idea of both the $\mathcal{O}(N \log N)$ algorithm and the $\mathcal{O}(N)$ algorithm in a simple way. The ideas of this chapter were previously published in [51] and we mainly profit from [124]. For the sake of simplicity we restrict our attention to the fast computation of

$$\boldsymbol{f} = \boldsymbol{M} \boldsymbol{\alpha} , \tag{4.2}$$

where

$$\boldsymbol{M} = \big(K(x_k, y_j)\big)_{j=1,k=1}^{M,N} ,$$

and where $x_k, y_j, \in [0, 1)$ are one-dimensional nodes. We assume that, except for some singular points, the kernel K is sufficiently smooth and satisfies one of the following conditions

$$|\partial_x^p K(x, y)| \leq Cp! |x - y|^{-p} \qquad (p \in \mathbb{N}), \tag{4.3}$$

$$|\partial_x^\beta \partial_y^\gamma K(x, y)| \leq Cp! |x - y|^{-p} \qquad (\beta + \gamma = p;\ p \in \mathbb{N}). \tag{4.4}$$

As typical example we consider the kernel $K(x,y) = \log|x-y|$ which satisfies (4.3) and (4.4) with $C = 1/p \leq 1$. In literature a couple of different conditions on the kernel was considered, see e.g. [16, 127, 13, 85].

Further, we assume for sake of simplicity that both the source nodes x_k and the target nodes y_j are uniformly distributed and ordered so that $x_1 < \ldots < x_N$ and $y_1 < \ldots < y_M$. Indeed it is sufficient that either source or target points are uniformly distributed. If this is not the case additional adaptation techniques are required [23, 86].

The algorithm is based on

- a hierarchical splitting of $\boldsymbol{M}$ into admissible blocks and
- a low rank approximation of each admissible block.

This chapter is organized as follows. In the first section, we define the hierarchical splitting of $\boldsymbol{M}$ into admissible blocks. Two different low rank approximations of these blocks are explained in Section 2. The hierarchical algorithm with arithmetical complexity of $\mathcal{O}(N \log N)$ is introduced in Section 3 and finally the fast algorithm with $\mathcal{O}(N)$ arithmetic operations is described in detail in Section 4.

1. Hierarchical splitting into admissible blocks

The following notation is mainly adapted from W. Hackbusch and co-workers. Although its strength becomes more clear in the multi-dimensional setting we find it also useful in one dimension.

Let $I = \{1, \ldots, N\}$ and $J = \{1, \ldots, M\}$ be index sets and let $X = \{x_i : i \in I\}$ and $Y = \{y_j : j \in J\}$. Let $\mathcal{P}(I)$ be a partition of I, i.e.,

$$I = \dot{\bigcup_{\sigma \in \mathcal{P}(I)}} \sigma.$$

For $\sigma \in \mathcal{P}(I)$ and $\tau \in \mathcal{P}(J)$, let

$$X(\sigma) = \{x_i \in X : i \in \sigma\}, \quad Y(\tau) = \{y_j \in Y : j \in \tau\}.$$

According to any block of indices $b = \tau \times \sigma$, $\tau \in \mathcal{P}(J)$, $\sigma \in \mathcal{P}(I)$, we can consider the matrix block

$$\boldsymbol{M}^b = \big(m_{ji}\big)_{j\in\tau, i\in\sigma}.$$

We are mainly interested in so-called admissible blocks. These will be the blocks which can be approximated by low rank matrices. Let r_σ and r_τ denote the diameters and c_σ and c_τ be the centers of $X(\sigma)$ and $Y(\tau)$, respectively, i.e.,

$$|x_i - c_\sigma| \leq r_\sigma \quad (i \in \sigma), \quad |y_j - c_\tau| \leq r_\tau \quad (j \in \tau)$$

and let

$$\operatorname{dist}(\tau, \sigma) = \min_{j\in\tau, i\in\sigma} |y_j - x_i|$$

be the distance of two clusters τ and σ. Then a block $b = \tau \times \sigma$ is called *admissible*, if there exists $\eta \in (0, 1]$ so that

$$\eta \operatorname{dist}(\tau, \sigma) \geq r_\tau + r_\sigma. \tag{4.5}$$

In order to split our matrix into admissible blocks we use a hierarchical splitting of the index sets I and J. The tree which corresponds to this hierarchical index splitting is called *$\mathcal{H}$-tree* by W. Hackbusch. In one dimension we can simply use the following binary splitting to obtain a binary tree:

Let $\mathcal{T}_I(0) = I$. At level ℓ, the vertices of our tree are given by the index sets

$$\sigma = \sigma(\ell, m) = \left\{k \in I : x_k \in [m/2^\ell, (m+1)/2^\ell)\right\} \qquad (m = 0, \ldots, 2^\ell - 1).$$

By $\mathcal{T}_I(\ell)$ we denote the corresponding partition of I. We obtain a similar tree $\mathcal{T}_J$ for J. Since our nodes x_k and y_j are uniformly distributed, each $\sigma \in \mathcal{T}_I(\ell)$ has approximately the same number $[N/2^\ell]$ of indices. Here $[a]$ denotes the integer part of a. Note that $r_\sigma \approx 1/2^{\ell+1}$ and $c_\sigma \approx (m+1/2)/2^\ell$, where both values are smaller than the right-hand sides. We stop our binary partitioning if each index set contains only a small number, say $\leq \nu$, of indices. Let $n = [\log_2(N/\nu)]$ be the number of levels.

By $\mathcal{T}_{J\times I}(\ell) = \mathcal{T}_J(\ell) \times \mathcal{T}_I(\ell)$ we denote the tensor block partition of $J \times I$.

Now we can produce a hierarchical splitting of our coefficient matrix $\boldsymbol{M}$ into admissible blocks. We start at level 2. We split $\boldsymbol{M}$ with respect to the blocks $b = \tau \times \sigma \in \mathcal{T}_{J\times I}(2)$ and sort admissible and nonadmissible blocks:

$$\boldsymbol{M} = \boldsymbol{M}_2 + \boldsymbol{N}_2,$$

where $\boldsymbol{M}_2$ consists of the admissible blocks of $\mathcal{T}_{J\times I}(2)$ and $\boldsymbol{N}_2$ of the other ones. We proceed with $\boldsymbol{N}_2$, i.e.

$$\boldsymbol{N}_2 = \boldsymbol{M}_3 + \boldsymbol{N}_3,$$

where $\boldsymbol{M}_3$ consists of the admissible blocks of $\mathcal{T}_{J\times I}(3)$ contained in $\boldsymbol{N}_2$ and $\boldsymbol{N}_3$ of the other ones. Repeating this procedure up to level n we obtain the final additive splitting

$$\boldsymbol{M} = \sum_{\ell=2}^{n} \boldsymbol{M}_\ell + \boldsymbol{N}_n \tag{4.6}$$

of $\boldsymbol{M}$ into admissible blocks contained in the matrices $\boldsymbol{M}_\ell$ and into a 'near-field matrix' $\boldsymbol{N}_n$.

It is easy to check that there is only a small number $\leq \gamma$ of non-zero blocks in each row/column of $\boldsymbol{M}_\ell$. In particular, if $\eta = 2^{-r}$ ($r \in \mathbb{N}$ small) then $\gamma = [2/\eta] + 1$. Therefore, $\boldsymbol{M}_\ell$ consists of no more than $2^\ell\gamma$ non-zero blocks. The same holds for $\boldsymbol{N}_\ell$. Figure 4.1 shows the non-zero blocks of $\boldsymbol{M}_\ell$ (thick lines) for $\ell = 2, 3, 4$ in the cases $\eta = 1$ and $\eta = 1/2$. Indeed for the upper figure $\eta < 1$ can be chosen.

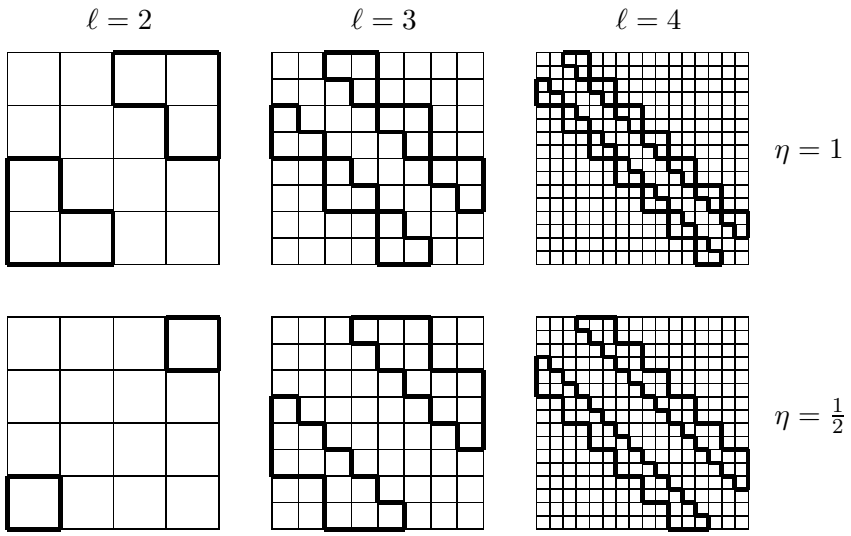


Figure 4.1.: Non-zero blocks of M_ℓ

2. Low rank approximation of admissible blocks

Next we will see how admissible blocks can be approximated by low rank matrices. Of course, supposed that a 'good' low rank approximation exists, it is easy to find, if the singular value decomposition (SVD) of the admissible blocks is accessible. But the SVD is computationally very expensive, so that approximations based on the SVD cannot lead to fast algorithms. In this context E. Tyrtyshnikov et al. have proposed a ***CGR*** decomposition of admissible blocks [127, 128], M. Bebendorf an iterative approximation scheme [10] and W. Hackbusch et al. Taylor expansion [71, 75, 73] and polynomial interpolation [72].

In this thesis, we consider only the simplest case that K is known and satisfies one of the properties (4.3) or (4.4).

Approximation by Taylor expansion

Let $b = \tau \times \sigma$ be an admissible block and let $x \in X(\sigma)$ and $y \in Y(\tau)$. If K satisfies (4.3), then we obtain by Taylor expansion at c_σ with respect to x

$$\begin{aligned} K(x,y) &= \sum_{\ell=0}^{p-1} \frac{1}{\ell!} (x - c_\sigma)^\ell \, \partial_x^\ell K(c_\sigma, y) \; + \; R_p(x,y) \\ &= \sum_{\ell=0}^{p-1} \frac{1}{\ell!} \varphi_\ell^\sigma(x) \psi_\ell^{\tau,\sigma}(y) \; + \; R_p(x,y), \end{aligned}$$

where

$$\varphi_\ell^\sigma(x) = (x - c_\sigma)^\ell \quad \text{and} \quad \psi_\ell^{\tau,\sigma}(y) = \partial_x^\ell K(c_\sigma, y).$$

For the approximation error we have by (4.3) that $|\tilde{x} - y| \geq (r_\tau + r_\sigma)/\eta$ and consequently

$$|R_p(x,y)| = \frac{1}{p!}|x - c_\sigma|^p \, |\partial_x^p K(\tilde{x}, y)| \leq C \, \frac{|x - c_\sigma|^p}{|\tilde{x} - y|^p},$$

where $\tilde{x} = c_\sigma + \theta(x - c_\sigma)$ ($\theta \in (0,1)$), and by the admissibility condition (4.5) that

$$|R_p(x,y)| \leq C\eta^p \left(\frac{r_\sigma}{r_\tau + r_\sigma} \right)^p.$$

Thus, if $\eta \leq 1$, then $\boldsymbol{M}^b = \big(K(x_k, y_j)\big)_{j\in\tau, k\in\sigma}$ can be approximated with a small error by

$$\boldsymbol{M}^b \approx \tilde{\boldsymbol{M}}^b = (\boldsymbol{\Psi}^{\tau,\sigma})^T \, \boldsymbol{D} \, \boldsymbol{\Phi}^\sigma \tag{4.7}$$

where $\boldsymbol{D} = \operatorname{diag}(1/\ell!)_{\ell\in P}$ with index set $P = \{0, \ldots, p-1\}$ and

$$\boldsymbol{\Phi}^\sigma = \big(\varphi_\ell^\sigma(x_k)\big)_{\ell\in P, k\in\sigma} \in \mathbb{R}^{p,|\sigma|}, \quad \boldsymbol{\Psi}^{\tau,\sigma} = \big(\psi_\ell^{\tau,\sigma}(y_j)\big)_{\ell\in P, j\in\tau} \in \mathbb{R}^{p,|\tau|}.$$

The error decays exponentially with increasing p. Since $\tilde{\boldsymbol{M}}^{\boldsymbol{b}}$ is a matrix of rank $\leq p$ its multiplication with a vector requires only $\mathcal{O}(p(|\sigma| + |\tau|))$ arithmetic operations. Note that E. Tyrtyshnikov calls the rank-1 matrices $\big(\psi_\ell^{\tau,\sigma}(y_j)\big)_{j\in\tau}\big(\varphi_\ell^\sigma(x_k)\big)_{k\in\sigma}^T$ *skeletons*.

If K satisfies (4.4), then we obtain by bivariate Taylor expansion at (c_σ, c_τ)

$$\begin{aligned}
K(x,y) &= \sum_{\ell=0}^{p-1} \frac{1}{\ell!} \big((x - c_\sigma)\partial_x + (y - c_\tau)\partial_y\big)^\ell K(c_\sigma, c_\tau) \;+\; R_p(x,y) \\
&= \sum_{0\leq \ell+m\leq p-1} \frac{1}{\ell! m!} \partial_x^\ell \partial_y^m K(c_\sigma, c_\tau) \, (x - c_\sigma)^\ell (y - c_\tau)^m \;+\; R_p(x,y) \\
&= \sum_{0\leq \ell+m\leq p-1} \frac{1}{\ell! m!} \partial_x^\ell \partial_y^m K(c_\sigma, c_\tau) \, \varphi_\ell^\sigma(x) \psi_m^\tau(y) \;+\; R_p(x,y),
\end{aligned}$$

where

$$\varphi_\ell^\sigma(x) = (x - c_\sigma)^\ell \quad \text{and} \quad \psi_m^\tau(y) = (y - c_\tau)^m.$$

For the approximation error we have by (4.4) that

$$\begin{aligned}
|R_p(x,y)| &= \frac{1}{p!}\big|(x - c_\sigma)\partial_x + (y - c_\tau)\partial_y\big|^p \, K(\tilde{x}, \tilde{y}) \\
&\leq C \, \frac{\big(|x - c_\sigma| + |y - c_\tau|\big)^p}{|\tilde{x} - \tilde{y}|^p}
\end{aligned}$$

where $\tilde{x} = c_\sigma + \theta(x - c_\sigma)$, $\tilde{y} = c_\tau + \theta(y - c_\tau)$ ($\theta \in (0,1)$), and by the admissibility condition (4.5) that

$$|R_p(x,y)| \leq C\eta^p.$$

Thus, if $\eta < 1$, then $\boldsymbol{M}^b = \big(K(x_k, y_j)\big)_{j\in\tau, k\in\sigma}$ can be approximated with small error by

$$\boldsymbol{M}^b \approx \tilde{\boldsymbol{M}}^b = (\boldsymbol{\Psi}^\tau)^T \, \boldsymbol{A}^{\tau,\sigma} \, \boldsymbol{\Phi}^\sigma \tag{4.8}$$

where

$$\mathbf{\Phi}^\sigma = \big(\varphi_\ell^\sigma(x_k)\big)_{\ell\in P, k\in\sigma} \in \mathbb{R}^{p,|\sigma|}, \quad \mathbf{\Psi}^\tau = \big(\psi_m^\tau(y_j)\big)_{m\in P, j\in\tau} \in \mathbb{R}^{p,|\tau|}$$

and $\boldsymbol{A}^{\tau,\sigma} = \big(a_{m,\ell}^{\tau,\sigma}\big)_{m,\ell\in P} \in \mathbb{R}^{p,p}$ with

$$a_{m,\ell}^{\tau,\sigma} = \frac{1}{\ell!m!}\partial_x^\ell \partial_y^m K(c_\sigma, c_\tau) \quad \text{if} \quad 0 \le \ell + m \le p-1$$

and $a_{m,\ell}^{\tau,\sigma} = 0$ otherwise. Again the error decreases exponentially with increasing p. Since $\tilde{\boldsymbol{M}}^b$ is a matrix of rank $\le p(p+1)/2$ its multiplication with a vector requires only $\mathcal{O}(p(|\sigma| + (p+1)/2 + |\tau|))$ arithmetic operations. Of course we can also use a Taylor expansion of K such that $\boldsymbol{A}^{\tau,\sigma}$ is a fully populated $p \times p$ matrix.

Example 2.1. Let $K(x,y) = \log|x-y|$. Then

$$a_{m,\ell}^{\tau,\sigma} = \begin{cases} \log|c_\tau - c_\sigma| & \text{for } \ell = m = 0, \\ -\frac{(-1)^\ell}{\ell+m}(c_\tau - c_\sigma)^{-\ell-m}\binom{\ell+m}{\ell} & \text{for } \ell + m \le p-1, \\ 0 & \text{otherwise.} \end{cases}$$

Approximation by polynomial interpolation

We need some notation first. Let L_m denote the Lagrange polynomials

$$L_m(x) := \prod_{\substack{\ell=0 \\ \ell\neq m}}^{p-1} \frac{x - c_\ell}{c_m - c_\ell} \qquad (m = 0, \ldots, p-1),$$

where $c_\ell := \cos\big(\frac{2\ell+1}{2p}\pi\big) \in [-1,1]$ $(\ell = 0, \ldots, p-1)$ are the zeros of the Chebyshev polynomial of the first kind T_p. With the linear transformation

$$\lambda_{[a,b]} \colon [-1,1] \to [a,b], \quad x \mapsto \frac{b+a}{2} + \frac{b-a}{2}\,x\,,$$

we can define the basis polynomials for arbitrary intervals $[a,b]$ by

$$L_m^{[a,b]} := L_m \circ \lambda_{[a,b]}^{-1} \qquad (m = 0, \ldots, p-1).$$

With $c_\ell^{[a,b]} := \lambda_{[a,b]}(c_\ell)$ $(\ell = 0, \ldots, p-1)$ it holds

$$L_m^{[a,b]}\big(c_\ell^{[a,b]}\big) = \big(L_m \circ \lambda_{[a,b]}^{-1}\big)\big(\lambda_{[a,b]}(c_\ell)\big) = L_m(c_\ell) = \delta_{ml}$$

and therefore

$$L_m^{[a,b]}(x) = \prod_{\substack{\ell=0 \\ \ell\neq m}}^{p-1} \frac{x - c_\ell^{[a,b]}}{c_m^{[a,b]} - c_\ell^{[a,b]}}\,.$$

Now, let $b = \tau \times \sigma$ be an admissible block and let $[a, b] \supseteq X(\sigma)$ resp. $[c, d] \supseteq Y(\tau)$ be the convex hull, which we denote again with σ resp. τ. If K satisfies (4.3), then we obtain by Lagrange interpolation on σ with respect to x

$$\begin{aligned} K(x,y) &= \sum_{\ell=0}^{p-1} K(c_\ell^\sigma, y) L_\ell^\sigma(x) + R_p(x,y) \\ &= \sum_{\ell=0}^{p-1} \varphi_\ell^\sigma(x) \psi_\ell^{\tau,\sigma}(y) + R_p(x,y)\,, \end{aligned}$$

where

$$\varphi_\ell^\sigma(x) := L_\ell^\sigma(x) \quad \text{and} \quad \psi_\ell^{\tau,\sigma}(y) := K(c_\ell^\sigma, y)\,.$$

For the approximation error we have by (4.3) that

$$|R_p(x,y)| \le \frac{r_\sigma}{2^{2p-1} p!} \left|\partial_x^p K(\tilde{x}, y)\right| \le \frac{C}{2^{2p-1}} \frac{|x - c_\sigma|^p}{|\tilde{x} - y|^p},$$

where $\tilde{x} = c_\sigma + \theta(x - c_\sigma)$ ($\theta \in (0,1)$), and by the admissibility condition (4.5) that

$$|R_p(x,y)| \le \frac{C}{2^{2p-1}} \eta^p \left(\frac{r_\sigma}{r_\tau + r_\sigma}\right)^p.$$

Thus, if $\eta \le 1$, then $\boldsymbol{M}^b$ can again be approximated with small error by equation (4.7), where $\boldsymbol{D}$ is the identity matrix.

If K satisfies (4.4), then we obtain by tensor product polynomial interpolation on $\sigma \times \tau$

$$\begin{aligned} K(x,y) &= \sum_{\ell=0}^{p-1} \sum_{m=0}^{p-1} K(c_\ell^\sigma, c_m^\tau) L_\ell^\sigma(x) L_m^\tau(y) + R_p(x,y) \\ &= \sum_{\ell=0}^{p-1} \sum_{m=0}^{p-1} a_{m,\ell}^{\tau,\sigma} \varphi_\ell^\sigma(x) \psi_m^\tau(y) + R_p(x,y) \end{aligned}$$

where

$$a_{m,\ell}^{\tau,\sigma} := K(c_\ell^\sigma, c_m^\tau)\,, \quad \varphi_\ell^\sigma(x) = L_\ell^\sigma(x) \quad \text{and} \quad \psi_m^\tau(y) = L_m^\tau(y)\,.$$

For the approximation error we have by (4.4) that

$$\begin{aligned} |R_p(x,y)| &\le \frac{p}{2^{2p-1} p!} \left|(x - c_\sigma)\partial_x + (y - c_\tau)\partial_y\right|^p K(\tilde{x}, \tilde{y}) \\ &\le \frac{C}{2^{2p-1}} \frac{\left(|x - c_\sigma| + |y - c_\tau|\right)^p}{|\tilde{x} - \tilde{y}|^p} \end{aligned}$$

where $\tilde{x} = c_\sigma + \theta(x - c_\sigma)$, $\tilde{y} = c_\tau + \theta(y - c_\tau)$ ($\theta \in (0,1)$), and by the admissibility condition (4.5) that

$$|R_p(x,y)| \le \frac{Cp}{2^{2p-1}} \eta^p.$$

Again, we have an approximation of the form (4.8) with small error.

One advantage when using polynomial interpolation is that only the kernel function $K(x,y)$ has to be known, but no derivates. Moreover, the interpolation polynomials can easily be interpolated, which is advantageous particularly with regard to Galerkin methods.

Example 2.2. Let again $K(x,y) = \log|x-y|$. Then

$$a_{m,\ell}^{\tau,\sigma} = \log|c_\ell^\sigma - c_m^\tau|\,.$$

In the following, we assume that each admissible block $\boldsymbol{M}^b$ can be approximated with only small error by a matrix $\tilde{\boldsymbol{M}}^b$ of one of the following forms

$$\tilde{\boldsymbol{M}}^b = (\boldsymbol{\Psi}^{\tau,\sigma})^T\,\boldsymbol{D}^{\tau,\sigma}\,\boldsymbol{\Phi}^{\tau,\sigma}, \tag{4.9}$$

$$\tilde{\boldsymbol{M}}^b = (\boldsymbol{\Psi}^{\tau})^T\,\boldsymbol{A}^{\tau,\sigma}\,\boldsymbol{\Phi}^{\sigma}, \tag{4.10}$$

where

$$\boldsymbol{\Phi}^\bullet \in \mathbb{R}^{p\times|\sigma|},\ \boldsymbol{\Psi}^\bullet \in \mathbb{R}^{p\times|\tau|},\ \boldsymbol{A}^\bullet \in \mathbb{R}^{p\times p}$$

and $\boldsymbol{D}^\bullet \in \mathbb{R}^{p\times p}$ is a diagonal matrix. The first representation (4.9) may be simply obtained from an SVD, while (4.10) is of the form (4.8). The approximation (4.7) corresponds to a mixture of both forms and a fast matrix-vector multiplication algorithm follows straightforward if we have algorithms for (4.9) and (4.10).

Note that one can use level-dependent approximations of admissible blocks $\boldsymbol{M}^b$ where the rank of the approximating matrix $\tilde{\boldsymbol{M}}^b$ depends on the decomposition level ℓ of the $\mathcal{H}$-tree, see [73].

Now (4.6) can be approximated by

$$\boldsymbol{M} \approx \sum_{\ell=2}^{n} \tilde{\boldsymbol{M}}_\ell \;+\; \boldsymbol{N}_n, \tag{4.11}$$

where the blocks in $\tilde{\boldsymbol{M}}_\ell$ are low rank approximations of the form (4.9) or (4.10) of the admissible blocks in $\boldsymbol{M}_\ell$. W. Hackbusch calls the matrix on the right-hand side of (4.11) an $\mathcal{H}$-matrix (in case of (4.10) an *uniform* $\mathcal{H}$-matrix) and E. Tyrtyshnikov a *mosaic-skeleton approximation* of $\boldsymbol{M}$.

If we have an approximation of type (4.10) then $\tilde{\boldsymbol{M}}_\ell$ can be further rewritten as

$$\tilde{\boldsymbol{M}}_\ell = \operatorname{blockdiag}(\boldsymbol{\Psi}^\tau)_{\tau\in\mathcal{T}_J(\ell)}^T\;\boldsymbol{A}_\ell\;\operatorname{blockdiag}(\boldsymbol{\Phi}^\sigma)_{\sigma\in\mathcal{T}_I(\ell)}, \tag{4.12}$$

where $\boldsymbol{A}_\ell \in \mathbb{R}^{p2^\ell,p2^\ell}$ has the non-zero blocks $\boldsymbol{A}^{\tau,\sigma} \in \mathbb{R}^{p,p}$ at the 'position' of the non-zero blocks of $\boldsymbol{M}_\ell$.

3. The hierarchical $\mathcal{O}((N+M)\log N)$-Algorithm

Assume that the non-zero blocks of $\boldsymbol{M}_\ell$ are of the form (4.9). Using (4.11) the matrix-vector multiplication (4.2) can be computed approximately by

$$\boldsymbol{f} = \boldsymbol{M}\boldsymbol{\alpha} \approx \underbrace{\sum_{\ell=2}^{n} \tilde{\boldsymbol{M}}_\ell\boldsymbol{\alpha}}_{\text{far-field}} \;+\; \underbrace{\boldsymbol{N}_n\boldsymbol{\alpha}}_{\text{near-field}} \;=\; f_F + f_N\,.$$

We call the computation of the first $n-1$ matrix-vector products on the right-hand side 'far-field computation' and the last matrix-vector multiplication 'near-field computation'.

Since multiplication with a block $\tilde{\boldsymbol{M}}^b$ requires $\mathcal{O}(p(|\sigma|+|\tau|))$ arithmetic operations, where $|\tau| \leq M/2^\ell$, $|\sigma| \leq N/2^\ell$, and there are no more than $2^\ell\gamma$ such blocks in $\tilde{\boldsymbol{M}}_\ell$, the computation of $\tilde{\boldsymbol{M}}_\ell\boldsymbol{\alpha}$ requires $\mathcal{O}(p(M+N))$ arithmetic operations. Adding this up over all levels, we get an arithmetic complexity of $\mathcal{O}(p(N+M)\log N)$ for the far-field computation. Note that the approximation error becomes smaller with increasing p.

Since $\boldsymbol{N}_n$ has at most γ non-zero blocks per row each with $\leq \nu$ columns, the near-field correction requires $\mathcal{O}(M\nu\gamma)$ arithmetic operations.

Since ν, γ and p are constants, the whole algorithm requires $\mathcal{O}((N+M)\log N)$ arithmetic operations.

4. The fast $\mathcal{O}(N+M)$-Algorithm

In this section we introduce a fast algorithm of arithmetic complexity $\mathcal{O}(N+M)$.

The algorithm is only practicable if the admissible blocks of the matrix can be approximated by an expression of the form (4.10). In addition, the matrices $\boldsymbol{\Phi}^\sigma$ and $\boldsymbol{\Psi}^\tau$ have to be '*nested*', i.e., fulfill the following *consistency conditions*: let $\sigma', \sigma'' \in \mathcal{T}_I(\ell+1)$ be the sons of $\sigma \in \mathcal{T}_I(\ell)$ and let $\tau', \tau'' \in \mathcal{T}_J(\ell+1)$ be the sons of $\tau \in \mathcal{T}_J(\ell)$. Then they have to fulfill

$$\boldsymbol{\Phi}^\sigma = [\boldsymbol{C}^{\sigma,\sigma'}\,\boldsymbol{C}^{\sigma,\sigma''}] \begin{pmatrix} \boldsymbol{\Phi}^{\sigma'} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Phi}^{\sigma''} \end{pmatrix} = \left[\boldsymbol{C}^{\sigma,\sigma'}\boldsymbol{\Phi}^{\sigma'}\,,\, \boldsymbol{C}^{\sigma,\sigma''}\boldsymbol{\Phi}^{\sigma''}\right], \tag{4.13}$$

$$\boldsymbol{\Psi}^\tau = [\boldsymbol{C}^{\tau,\tau'}\,\boldsymbol{C}^{\tau,\tau''}] \begin{pmatrix} \boldsymbol{\Psi}^{\tau'} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Psi}^{\tau''} \end{pmatrix} = \left[\boldsymbol{C}^{\tau,\tau'}\boldsymbol{\Psi}^{\tau'}\,,\, \boldsymbol{C}^{\tau,\tau''}\boldsymbol{\Psi}^{\tau''}\right]. \tag{4.14}$$

Then the matrices in (4.10) are called $\mathcal{H}^2$-*matrices* and the corresponding algorithm either FMM or fast $\mathcal{H}^2$-matrix multiplication.

For $\boldsymbol{\Phi}^\sigma \in \mathbb{R}^{p\times|\sigma|}$ and $\boldsymbol{\Psi}^\tau \in \mathbb{R}^{p\times|\tau|}$ arising from Taylor expansions as in (4.8) the consistency conditions are clearly fulfilled: since

$$\begin{aligned}(x-c_\sigma)^\ell &= \big((x-c_{\sigma'})-(c_\sigma-c_{\sigma'})\big)^\ell \\ &= \sum_{m=0}^{\ell}\binom{\ell}{m}(c_{\sigma'}-c_\sigma)^{\ell-m}(x-c_{\sigma'})^m\end{aligned}$$

for all $\ell = 0, \ldots, p-1$, we obtain

$$\begin{aligned}\Big((x_k-c_\sigma)^\ell\Big)^T_{k\in\sigma'} &= \sum_{m=0}^{\ell} C^{\sigma,\sigma'}_{\ell,m}\,((x_k-c_{\sigma'})^m)^T_{k\in\sigma'}\,, \\ \Big((x_k-c_\sigma)^\ell\Big)^T_{k\in\sigma''} &= \sum_{m=0}^{\ell} C^{\sigma,\sigma''}_{\ell,m}\,((x_k-c_{\sigma''})^m)^T_{k\in\sigma''}\,.\end{aligned}$$

Thus (4.13) is fulfilled with the lower triangular matrix $\boldsymbol{C}^{\sigma,\sigma'} = \left(C_{\ell,m}^{\sigma,\sigma'}\right)_{\ell,m\in P}$, where

$$C_{\ell,m}^{\sigma,\sigma'} = \begin{cases} 0 & \text{for } \ell < m, \\ \binom{\ell}{m}(c_\sigma - c_{\sigma'})^{\ell-m} & \text{for } \ell \geq m. \end{cases}$$

Let us now consider polynomial interpolation. It holds

$$\operatorname{span}\{L_\ell^\sigma : \ell = 0,\dots,p-1\} = \operatorname{span}\{L_m^{\sigma'} : m = 0,\dots,p-1\} = \Pi_{p-1}\,.$$

Thus, there exists coefficients $C_{\ell,m}^{\sigma,\sigma'}$ with

$$L_\ell^\sigma(x) = \sum_{m=0}^{p-1} C_{\ell,m}^{\sigma,\sigma'} L_m^{\sigma'}(x)\,.$$

Since we use Lagrange polynomials, one can easily see that

$$C_{\ell,m}^{\sigma,\sigma'} = L_\ell^\sigma(c_m^{\sigma'})$$

and therefore the consistency conditions hold.

Note that it is often also sufficient if the consistency conditions (4.13) and (4.14) are satisfied only approximately, i.e., up to a small error, see [42, 40].

For $\ell = 2, \dots, n-1$, let

$$\boldsymbol{D}_{\ell,\ell+1}^{\Phi} = \operatorname{blockdiag}\left([\boldsymbol{C}^{\sigma,\sigma'}\boldsymbol{C}^{\sigma,\sigma''}]\right)_{\sigma\in\mathcal{T}_I(\ell)} \in \mathbb{R}^{p2^\ell,2p2^\ell}$$

denote the transform matrices arising from the consistency conditions for all $\sigma \in \mathcal{T}_I(\ell)$. Then the consistency condition at level ℓ reads as

$$\operatorname{blockdiag}\left(\boldsymbol{\Phi}^\sigma\right)_{\sigma\in\mathcal{T}_I(\ell)} = \boldsymbol{D}_{\ell,\ell+1}^{\Phi}\,\operatorname{blockdiag}\left(\boldsymbol{\Phi}^\sigma\right)_{\sigma\in\mathcal{T}_I(\ell+1)}\,.$$

Now successive application of the consistency condition leads to

$$\begin{aligned} \tilde{\boldsymbol{M}}_\ell &= \operatorname{blockdiag}\left(\boldsymbol{\Psi}^\tau\right)_{\tau\in\mathcal{T}_J(\ell+1)}^T \left(\boldsymbol{D}_{\ell,\ell+1}^{\Psi}\right)^T\ \boldsymbol{A}_\ell\ \boldsymbol{D}_{\ell,\ell+1}^{\Phi}\,\operatorname{blockdiag}\left(\boldsymbol{\Phi}^\sigma\right)_{\sigma\in\mathcal{T}_I(\ell+1)} \\ &= \dots \\ &= \operatorname{blockdiag}(\boldsymbol{\Psi}^\tau)_{\tau\in\mathcal{T}_J(n)}^T(\boldsymbol{D}_{n-1,n}^{\Psi})^T\cdots(\boldsymbol{D}_{\ell,\ell+1}^{\Psi})^T\ \boldsymbol{A}_\ell\ \times \\ &\quad \times \boldsymbol{D}_{\ell,\ell+1}^{\Phi}\cdots\boldsymbol{D}_{n-1,n}^{\Phi}\operatorname{blockdiag}(\boldsymbol{\Phi}^\sigma)_{\sigma\in\mathcal{T}_I(n)}. \end{aligned} \tag{4.15}$$

The important observation is that the factors $\operatorname{blockdiag}(\boldsymbol{\Phi}^\sigma)_{\sigma\in\mathcal{T}_I(n)}$ and $\operatorname{blockdiag}(\boldsymbol{\Psi}^\tau)_{\sigma\in\mathcal{T}_I(n)}$ appear in all matrices $\tilde{\boldsymbol{M}}_\ell$ ($\ell = 2, \dots, n$) and that the factors $\boldsymbol{D}_{i,i+1}^{\Phi}$ and $\boldsymbol{D}_{i,i+1}^{\Psi}$ appear in all matrices $\tilde{\boldsymbol{M}}_\ell$ with $\ell \leq i$.

Using (4.15) and (4.11) we can formulate the whole algorithm now. (For readers familiar with the FMM we have written the FMM notation of the algorithm in brackets, where FFE stands for far-field extension and LFE for near-field extension.)

Algorithm 4.1.

1. ***Forward Transform (FFE → FFE)***
   Initialization:
   $$\boldsymbol{x}_n = \operatorname{blockdiag}\big(\boldsymbol{\Phi}^{\sigma}\big)_{\sigma\in\mathcal{T}_I(n)}\,\boldsymbol{\alpha} \quad \in \mathbb{R}^{p2^n}$$
   Arithmetic complexity: $\mathcal{O}(pN)$

   For $\ell = n-1, \ldots, 2$ compute
   $$\boldsymbol{x}_\ell = \boldsymbol{D}^{\Phi}_{\ell,\ell+1}\boldsymbol{x}_{\ell+1}.$$
   Arithmetic complexity: *Since $\boldsymbol{D}^{\Phi}_{\ell,\ell+1}$ consists of 2^ℓ non-zero blocks of the form $[\boldsymbol{C}^{\sigma,\sigma'}\,\boldsymbol{C}^{\sigma,\sigma''}] \in \mathbb{R}^{p,2p}$ we have an amount of $\leq 2p^2\,2^\ell$ arithmetic operations in step ℓ. This adds up over all levels to $\mathcal{O}(2p^2\frac{N}{\nu})$ arithmetic operations.*

2. ***Multiplication Phase (FFE → LFE)***
   For $\ell = 2, \ldots, n$ compute
   $$\boldsymbol{y}_\ell = \boldsymbol{A}_\ell\,\boldsymbol{x}_\ell.$$
   Arithmetic complexity: *There are at most $2^\ell\gamma$ non-zero blocks on level ℓ and each block in A_ℓ is of size $p\times p$. Thus the computation of $\boldsymbol{A}_\ell\boldsymbol{x}_\ell$ requires $\mathcal{O}(p^2 2^\ell)$ arithmetic operations which adds up to $\mathcal{O}(p^2N/\nu)$ arithmetic operations over all levels.*

3. ***Backward Transform (LFE → LFE)***
   In the far-field it remains to compute
   $$\boldsymbol{f}_F = \sum_{\ell=2}^{n-1}\operatorname{blockdiag}\big(\boldsymbol{\Psi}^{\tau}\big)^T_{\tau\in\mathcal{T}_J(n)}\,(\boldsymbol{D}^{\Psi}_{n-1,n})^T\cdots(\boldsymbol{D}^{\Psi}_{\ell,\ell+1})^T\boldsymbol{y}_\ell.$$
   We apply Horner's rule. Set
   $$\boldsymbol{z}_2 = \boldsymbol{y}_2$$
   and compute for $\ell = 3, \ldots, n$ the vectors
   $$\boldsymbol{z}_\ell = (\boldsymbol{D}^{\Psi}_{\ell-1,\ell})^T\boldsymbol{z}_{\ell-1} + \boldsymbol{y}_\ell.$$
   Arithmetic complexity: *Multiplication with $(\boldsymbol{D}^{\Psi}_{\ell,\ell+1})^T$ requires as in Step 1 only $\mathcal{O}(2p^2 2^\ell)$ operations such that we have a total of $\mathcal{O}(p^2N/\nu)$ arithmetic operations.*

   Final multiplication:
   $$\boldsymbol{f}_F = \operatorname{blockdiag}\big(\boldsymbol{\Psi}^{\tau}\big)^T_{\tau\in\mathcal{T}_J(n)}\,\boldsymbol{z}_n.$$
   Arithmetic complexity: $\mathcal{O}(pM)$

4. ***Near-Field Correction**:*
   Compute $\boldsymbol{f}_N = \boldsymbol{N}_n\,\boldsymbol{\alpha}$ directly and add $\boldsymbol{f}_F$.
   Arithmetic complexity: $\mathcal{O}(M\nu\gamma)$ *as in the hierarchical algorithm.*

Choosing $\nu = p$ the arithmetic complexity of the whole algorithm is

$$\mathcal{O}(p(N+M)) = \mathcal{O}(N+M),$$

where p is a constant which regulates the approximation error.

V. Fast NFFT based summation of radial functions

In Chapter IV we gave a short introduction to the basic ideas of the FMM/$\mathcal{H}$-matrix/mosaic-skeleton-matrix concept for the fast multiplication of a vector by a fully populated matrix $\boldsymbol{M} = \big(m_{jk}\big)_{j,k=1}^{M,N}$. The computation of sums of the form

$$\sum_{k=1}^{N} \alpha_k K(\boldsymbol{y}_j - \boldsymbol{x}_k) \qquad (\boldsymbol{x}_k,\, \boldsymbol{y}_j \in \mathbb{R}^d)$$

for $j = 1, \ldots, M$ is a special case with $m_{jk} = K(\boldsymbol{y}_j - \boldsymbol{x}_k)$, e.g., for the kernel $K(\boldsymbol{x}) = \log\|\boldsymbol{x}\|$ in $\mathbb{R}^2$. Recently, Potts and Steidl [109, 110] have proposed a fast summation algorithm based on the fast Fourier transform for nonequispaced nodes (NFFT) which requires $\mathcal{O}(N \log N)$ arithmetic operations and has the following advantages:

- it resembles the well-known algorithm for the fast multiplication of vectors with Toeplitz matrices based on the FFT,
- the incorporation of new kernels is very simple,
- it has a simple structure consisting of the blocks FFT – NFFT – fast summation.

In this chapter, we further develop the ideas from [109]. We introduce new regularization techniques with B-splines and algebraic polynomials. Based on the approach with algebraic polynomials we prove error estimates for our approximative summation algorithm. These error estimates are more sophisticated than those for the regularization with trigonometric polynomials in [109]. The later still involve numerical computations and consequently are only valid for a bounded number of parameters. In [109] only kernels of the form

$$K_0(\boldsymbol{x}) = \log\|\boldsymbol{x}\|, \qquad K_\beta(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{x}\|^\beta} \qquad (\beta \in \mathbb{N}) \tag{5.1}$$

were considered. Here, we add estimates for the parameter-dependent generalized multiquadrics

$$K_{-1}(\boldsymbol{x}; c) = \big(\|\boldsymbol{x}\|^2 + c^2\big)^{\frac{1}{2}}, \qquad K_\beta(\boldsymbol{x}; c) = \big(\|\boldsymbol{x}\|^2 + c^2\big)^{-\frac{\beta}{2}} \qquad (\beta \in \mathbb{N} \text{ odd}) \tag{5.2}$$

which play an important role in the approximation of functions by linear combinations of radial basis functions (RBF) [59].

The results of this chapter were previously published in [52]. A modification for real input data is also possible together with the NFCT and NFST, see Remark 1.1 and [50].

This chapter is organized as follows. The next section describes our summation algorithm in 1D. One essential step of this algorithm consists in an appropriate kernel regularization which we consider in detail in Section 2. Error estimates for our algorithm with regularization by algebraic polynomials and the consequences for the choice of the parameters of the algorithm are derived in Section 3. Section 4 briefly sketches the generalization of the algorithm to the multivariate setting. Finally, Section 5 contains numerical results, mainly in 2D.

1. Fast Summation at one-dimensional nodes

In this section, we recall the idea of the fast summation algorithm in 1D introduced in [109]. Our aim consists in the fast evaluation of sums

$$f(x) := \sum_{k=1}^{N} \alpha_k K(x - x_k) \qquad (x_k \in \mathbb{R}), \tag{5.3}$$

at M nodes $y_j \in \mathbb{R}$ ($j = 1, \ldots, M$) for even kernels $K(x) = K(|x|)$. The kernel function K is in general a non-periodic function, while the use of Fourier methods requires to replace K by a periodic version. Without loss of generality we may assume that the nodes are scaled, such that $|x_k|, |y_j| < \frac{1}{4} - \frac{\varepsilon_B}{2}$ and consequently $|y_j - x_k| < \frac{1}{2} - \varepsilon_B$. The parameter $\varepsilon_B > 0$, which we specify later, guarantees that K has to be evaluated only at points in the interval $[-\frac{1}{2} + \varepsilon_B, \frac{1}{2} - \varepsilon_B]$. This simplifies the later consideration of a 1-periodic version of K. Beyond a special treatment of K near the boundary $\pm\frac{1}{2}$, we have to take care about properties of K in the neighborhood of the origin. The kernels (5.1) considered in [109] are C^∞ except of the origin, where they have a singularity. The parameter-dependent kernels $K = K_\beta(x; c)$ in (5.2), or its derivatives in case $\beta = -1$, have a singularity at zero if $c \to 0$.

To deduce a fast summation algorithm for (5.3) we replace the kernel K by a 1-periodic smooth kernel $\tilde{K}$ by modifying K near the boundary and near the origin:

$$\tilde{K}(x) := \begin{cases} K_I(x) & \text{for } x \in [-\varepsilon_I, \varepsilon_I], \\ K_B(x) & \text{for } x \in [-\frac{1}{2}, -\frac{1}{2} + \varepsilon_B] \cup [\frac{1}{2} - \varepsilon_B, \frac{1}{2}], \\ K(x) & \text{else}, \end{cases} \tag{5.4}$$

where $0 < \varepsilon_I < \frac{1}{2} - \varepsilon_B < \frac{1}{2}$. The functions K_I and K_B will be chosen such that $\tilde{K}$ is in the Sobolev space $H^p(\mathbb{T})$ for an appropriate parameter $p > 0$ which controls the smoothness of $\tilde{K}$. Various regularizations $\tilde{K}$ of K are proposed in Section 2. If p is large enough, then we may assume that $\tilde{K}$ can be approximated with sufficiently small error by the trigonometric polynomial

$$T_n(\tilde{K})(x) := \sum_{\ell \in I_n^1} b_\ell \, \mathrm{e}^{2\pi \mathrm{i} \ell x}, \tag{5.5}$$

where

$$b_\ell := \frac{1}{n} \sum_{j \in I_n^1} \tilde{K}\Big(\frac{j}{n}\Big)\, \mathrm{e}^{-2\pi \mathrm{i} j\ell/n} \qquad (\ell \in I_n^1)\,.$$

Now the original kernel K can be decomposed as

$$K = \big(K - \tilde{K}\big) + \big(\tilde{K} - T_n(\tilde{K})\big) + T_n(\tilde{K}), \tag{5.6}$$

where the summand in the middle becomes small for a sufficiently large parameter $n \in \mathbb{N}$ which we will specify later. We neglect this summand in (5.3) and approximate f by

$$\tilde{f}(x) := \sum_{k=1}^{N} \alpha_k (K - \tilde{K})(x - x_k) + \sum_{k=1}^{N} \alpha_k T_n(\tilde{K})(x - x_k). \tag{5.7}$$

Instead of f we evaluate $\tilde{f}$ at the nodes y_j ($j = 1, \ldots, M$). Indeed this can be done in a fast way by the following two steps:

1) Near field computation (first sum in (5.7))
To achieve the desired complexity of our algorithm we suppose that either the N points x_k or the M points y_j are 'sufficiently uniformly distributed', i.e., we suppose that there exists a small constant $\nu \in \mathbb{N}$ such that each subinterval of $[-\frac{1}{4}, \frac{1}{4}]$ of length $2\varepsilon_I$ contains at most ν of the points x_k or of the points y_j, respectively. This implies that ε_I depends linearly on $1/N$, respectively $1/M$. In the following we restrict our attention to the case

$$\varepsilon_I \approx \frac{\nu}{2N}. \tag{5.8}$$

Then, since $|y_j - x_k| < \frac{1}{2} - \varepsilon_B$ and $\operatorname{supp}(K - \tilde{K}) \cap [-\frac{1}{2} + \varepsilon_B, \frac{1}{2} - \varepsilon_B] = [-\varepsilon_I, \varepsilon_I]$, the evaluation of

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

requires $\leq \nu M$, i.e., $\mathcal{O}(M)$ arithmetic operations.

2) NFFT based summation (second sum in (5.7))
By (5.5), the evaluation of the second sum in (5.7) can be rewritten as

$$\begin{aligned} \sum_{k=1}^{N} \alpha_k T_n(\tilde{K})(y_j - x_k) &= \sum_{k=1}^{N} \alpha_k \sum_{\ell \in I_n^1} b_\ell\, \mathrm{e}^{2\pi \mathrm{i} \ell (y_j - x_k)} \\ &= \sum_{\ell \in I_n^1} b_\ell \left(\sum_{k=1}^{N} \alpha_k\, \mathrm{e}^{-2\pi \mathrm{i} \ell x_k} \right) \mathrm{e}^{2\pi \mathrm{i} \ell y_j}\,. \end{aligned}$$

This expression can be handled based on the NFFT as follows:

1. The sums
$$a_\ell = \sum_{k=1}^{N} \alpha_k \, \mathrm{e}^{-2\pi \mathrm{i} \ell x_k} \qquad (\ell \in I_n^1)$$
can be obtained by an NFFT$^T(n)$.

2. Then we compute the products
$$d_\ell = b_\ell a_\ell \qquad (\ell \in I_n^1).$$

3. Finally we use the NFFT(n) to compute
$$\sum_{\ell \in I_n^1} d_\ell \, \mathrm{e}^{2\pi \mathrm{i} \ell y_j} \qquad (j = 1, \ldots, M).$$

These three steps require $\mathcal{O}(M + N + n \log n)$ arithmetic operations.

In summary, our summation algorithm requires
$$\mathcal{O}(M + N + n \log n)$$
arithmetic operations. The relation between M, N and n determined by the approximation error of the algorithm will be specified in Section 3.

Once the basic idea of the algorithm is clear, it remains to specify the regularization procedure and to give estimates of the approximation error introduced by omitting $\tilde{K} - T_n(\tilde{K})$ in the kernel approximation.

Remark 1.1. A modification for even kernels K is possible [50]. Choose the replacing kernel $\tilde{K}$ in (5.4) even. Instead of approximating $\tilde{K}$ with the trigonometric polynomial in (5.5), we can now approximate $\tilde{K}$ with the cosine polynomial
$$C_n(\tilde{K})(x) := \sum_{\ell=0}^{n-1} b_\ell \cos(2\pi \ell x), \quad \text{where} \quad b_\ell := \frac{2\varepsilon_{n,\ell}}{n} \sum_{j=0}^{n} \varepsilon_{n,j} \tilde{K}\left(\frac{j}{2n}\right) \cos\left(\frac{\pi \ell j}{n}\right)$$
with $\varepsilon_{n,0} := \varepsilon_{n,n} := 1/2$ and $\varepsilon_{n,\ell} := 1$ ($\ell = 1,\ldots,n-1$). Along the lines of step 2, the evaluation of the analog second sum in (5.7) can be rewritten as
$$\begin{aligned} \sum_{k=1}^{N} \alpha_k C_n(\tilde{K})(y_j - x_k) &= \sum_{k=1}^{N} \alpha_k \sum_{\ell=0}^{n-1} b_\ell \cos\bigl(2\pi \ell (y_j - x_k)\bigr) \\ &= \sum_{k=1}^{N} \alpha_k \sum_{\ell=0}^{n-1} b_\ell \bigl(\cos(2\pi \ell y_j)\cos(2\pi \ell x_k) + \sin(2\pi \ell y_j)\sin(2\pi \ell x_k)\bigr). \end{aligned}$$
The expressions in the inner brackets of
$$\sum_{\ell=0}^{n-1} b_\ell \left(\sum_{k=1}^{N} \alpha_k \cos(2\pi \ell x_k)\right) \cos(2\pi \ell y_j) + \sum_{\ell=1}^{n-1} b_\ell \left(\sum_{k=1}^{N} \alpha_k \sin(2\pi \ell x_k)\right) \sin(2\pi \ell y_j)$$
can now be obtained by NFCT^T/NFST^T. This will be followed by $2n$ multiplications with b_ℓ and completed by NFCT/NFST to compute the outer summations.

2. Kernel Regularization

Since K given by (5.2) is differentiable and even, we have that $K^{(j)}(x) = (-1)^j K^{(j)}(-x)$. To ensure that

$$\tilde{K}(x) := \begin{cases} K_I(x) & \text{for } x \in [-\varepsilon_I, \varepsilon_I], \\ K_B(x) & \text{for } x \in [-\frac{1}{2}, -\frac{1}{2} + \varepsilon_B] \cup [\frac{1}{2} - \varepsilon_B, \frac{1}{2}], \\ K(x) & \text{else,} \end{cases}$$

is in $H^p(\mathbb{T})$, we need that the function K_I fulfills the conditions

$$\begin{aligned} K_I^{(j)}(\varepsilon_I) &= K^{(j)}(\varepsilon_I), \\ K_I^{(j)}(-\varepsilon_I) &= K^{(j)}(-\varepsilon_I) = (-1)^j K^{(j)}(\varepsilon_I) \end{aligned} \tag{5.9}$$

and the function K_B the conditions

$$\begin{aligned} K_B^{(j)}\left(\frac{1}{2} - \varepsilon_B\right) &= K^{(j)}\left(\frac{1}{2} - \varepsilon_B\right), \\ K_B^{(j)}\left(\frac{1}{2} + \varepsilon_B\right) &= K^{(j)}\left(-\frac{1}{2} + \varepsilon_B\right) = (-1)^j K^{(j)}\left(\frac{1}{2} - \varepsilon_B\right) \end{aligned} \tag{5.10}$$

for all $j = 0, \ldots, p-1$. Then, the periodicity of $\tilde{K}$ follows by setting

$$K_B\Big(-\frac{1}{2} + x\Big) := K_B\Big(\frac{1}{2} + x\Big) \qquad (x \in [0, \varepsilon_B])\,.$$

As simple regularizing functions K_I and K_B we propose

- algebraic polynomials,
- trigonometric polynomials,
- splines.

The regularization by trigonometric polynomials was considered in [109]. However the error estimates in [109] are not satisfactory since they involve numerical computations which can be done only up to a fixed number $p \in \mathbb{N}$. We briefly sketch the spline approach and consider the regularization by algebraic polynomials in more detail.

2.1. Regularization by spline interpolation

The normalized cardinal B-splines N_p of degree p are recursively defined by

$$N_0(x) := \begin{cases} 1 & \text{for } x \in [0, 1), \\ 0 & \text{otherwise,} \end{cases}$$

and

$$N_p(x) := \frac{x}{k} N_{p-1}(x) + \frac{p+1-x}{k} N_{p-1}(x-1) \qquad (p \in \mathbb{N}).$$

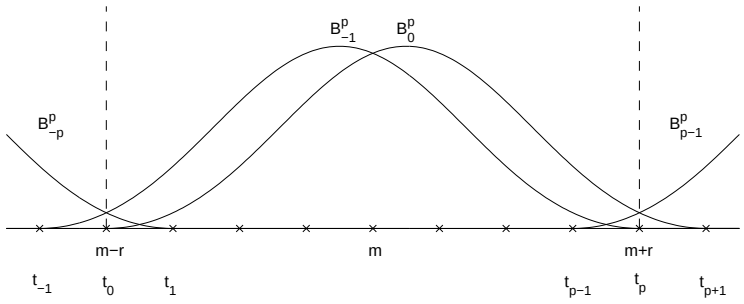


Figure 5.1.: B-splines B_k^p.

Note that $\operatorname{supp} N_p = [0, p+1]$.

In our application we deal with intervals $[m-r, m+r]$ ($r > 0$), more precisely with $[-\varepsilon_I, \varepsilon_I]$ and $[\frac{1}{2} - \varepsilon_B, \frac{1}{2} + \varepsilon_B]$. At the interval $[m-r, m+r]$ we choose the equispaced nodes $\Delta := \{t_k = m - r + \frac{2r}{p}k : k = -p, \ldots, 2p\}$ and introduce the dilated and translated versions of N_p with respect to these spline nodes

$$B_k^p(x) := N_p\left(\frac{p(x-m+r)}{2r} - k\right),$$

see Figure 5.1.

The set of B-splines $\{B_k^p\}_{k=-p}^{p-1}$ forms a basis of the spline space

$$S_p(\Delta) := \{s \in C^{p-1}[m-r, m+r] \,:\, s|_{[t_k, t_{k+1}]} \in \Pi_p,\ k = 0, \ldots, p-1\}.$$

Proposition 2.1 (Spline interpolation)**.** *For given a_j, b_j ($j = 0, \ldots, p-1$) there exists a unique spline $S \in S_p(\Delta)$ which satisfies the interpolation conditions*

$$S^{(j)}(m-r) = a_j, \qquad S^{(j)}(m+r) = b_j \qquad (j = 0, \ldots, p-1)$$

at the endpoints of an interval $[m-r, m+r]$ ($r > 0$). This spline can be written as

$$S(x) = \sum_{k=-p}^{p-1} c_k B_k^p(x)$$

where the coefficients c_k are the solution of the two $p \times p$ linear systems

$$\begin{aligned} &\sum_{k=1}^{p} c_{-k} (B_{-k}^p)^{(j)}(m-r) = a_j, \\ &\sum_{k=1}^{p} c_{k-1} (B_{-k}^p)^{(j)}(m-r) = (-1)^j b_j \end{aligned} \qquad (j = 0, \ldots, p-1)$$

with the same coefficient matrix.

The proposition is a direct consequence of [30, Theorem 1] and the fact that

$$(B^p_{-k})^{(j)}(m-r) = (-1)^j (B^p_{k-1})^{(j)}(m+r)\,.$$

Since our kernels are even, we have by (5.9) and (5.10) for our application that $a_j = (-1)^j b_j$. Hence it remains to solve only one $p \times p$ system to obtain all coefficients c_k. Of course, for large $p \in \mathbb{N}$, this system is ill-conditioned. However, we will only need small values of p in our algorithm, and, for $p \leq 16$, the corresponding systems can be solved without substantial errors.

Finally note that the fast evaluation of the spline $S(x)$ can be realized by the de Boor algorithm [31].

2.2. Regularization by polynomial interpolation

To construct polynomials K_I and K_B of degree $2p-1$ which fulfill the $2p$ Hermite interpolation conditions (5.9) and (5.10), respectively, we use the following two-point Taylor interpolation, see, e.g., [1, Corollary 2.2.6]:

Proposition 2.2 (Two-point Taylor interpolation)**.** *For given a_j, b_j ($j = 0, \ldots, p-1$) there exists a unique polynomial P of degree $2p-1$ which satisfies the interpolation conditions*

$$P^{(j)}(m-r) = a_j, \qquad P^{(j)}(m+r) = b_j \qquad (j = 0, \ldots, p-1) \tag{5.11}$$

at the endpoints of an interval $[m-r, m+r]$ ($r > 0$). This polynomial can be written as

$$\begin{aligned} P(x) = \sum_{j=0}^{p-1} \sum_{k=0}^{p-1-j} & \binom{p-1+k}{k} \\ & \left(\frac{(x-m+r)^j}{j!} \left(\frac{x-m-r}{-2r}\right)^p \left(\frac{x-m+r}{2r}\right)^k a_j \right. \\ & \left. + \frac{(x-m-r)^j}{j!} \left(\frac{x-m+r}{2r}\right)^p \left(\frac{x-m-r}{-2r}\right)^k b_j \right). \end{aligned} \tag{5.12}$$

As in the spline case, the representation (5.12) can be further simplified if we have even kernels and (5.9), (5.10) in mind.

Corollary 2.3. *For given a_j and $b_j = (-1)^j a_j$ ($j = 0, \ldots, p-1$) the unique polynomial P of degree $2p-1$ which satisfies* (5.11) *at the endpoints of an interval $[m-r, m+r]$ ($r > 0$) is given by*

$$P(x) = \frac{1}{2^p} \sum_{j=0}^{p-1} \gamma_j (1-y^2)^j \big((1-y)^{p-j} + (1+y)^{p-j} \big), \tag{5.13}$$

where $y := \frac{x-m}{r}$ and

$$\gamma_j := \sum_{\ell=0}^{j} \binom{p-1+\ell}{\ell} \frac{r^{j-\ell}}{2^\ell (j-\ell)!}\, a_{j-\ell}.$$

Proof. By (5.12) we obtain for our special setting that

$$P(x) = \frac{1}{2^p} \sum_{j=0}^{p-1} \sum_{k=0}^{p-1-j} \binom{p-1+k}{k} \frac{r^j}{2^k} \frac{a_j}{j!} \big((1+y)^{j+k}(1-y)^p + (1-y)^{j+k}(1+y)^p\big).$$

Now the change of the summation order results in the desired formula

$$P(x) = \frac{1}{2^p} \sum_{j=0}^{p-1} \sum_{\ell=0}^{j} \binom{p-1+\ell}{\ell} \frac{r^{j-\ell}}{2^\ell} \frac{a_{j-\ell}}{(j-\ell)!} \big((1+y)^j(1-y)^p + (1-y)^j(1+y)^p\big).$$

□

In the next section we will estimate the approximation error introduced by our fast algorithm. For this purpose we will need an estimate for the pth derivative of K_I and K_B, respectively.

Theorem 2.4. *For $p \in \mathbb{N}$, the pth derivative of the polynomial P in* (5.13) *can be estimated by*

$$\max_{x\in[m,m+r]} \big|P^{(p)}(x)\big| \le p! \left(\frac{3}{2}\right)^p r^{-p}\, \gamma,$$

where

$$\gamma := \sum_{\ell=0}^{p-2} \binom{p-1+\ell}{\ell} \frac{r^{p-1-\ell}}{2^\ell (p-1-\ell)!}\, |a_{p-1-\ell}|\,.$$

Proof. Since the two-point Taylor interpolation polynomial reproduces polynomials of degree at most $2p-1$, we obtain for the polynomial $\equiv 1$ by Corollary 2.3 that

$$\frac{1}{2^p} \sum_{j=0}^{p-1} \binom{p-1+j}{j} \frac{(1-y^2)^j}{2^j} \big((1-y)^{p-j} + (1+y)^{p-j}\big) = 1. \tag{5.14}$$

On the other hand, if we reorder the sum in (5.13) with respect to the coefficients a_l ($l = 0,\ldots,p-1$), then (5.14) is just the coefficient of a_0. Thus, a_0 does not appear in the pth derivative of any polynomial P of the form (5.13).

Now, since $\frac{\mathrm{d}}{\mathrm{d}x} y = \frac{1}{r}$, the pth derivative of (5.13) can be written as

$$P^{(p)}(x) = \left(\frac{1}{2r}\right)^p \sum_{j=1}^{p-1} \tilde{\gamma}_j \frac{\mathrm{d}^p}{\mathrm{d}y^p} \Big[(1-y^2)^j \big((1-y)^{p-j} + (1+y)^{p-j}\big)\Big], \tag{5.15}$$

where

$$\tilde{\gamma}_j := \sum_{\ell=0}^{j-1} \binom{p-1+\ell}{\ell} \frac{r^{j-\ell}}{2^\ell (j-\ell)!}\, a_{j-\ell}\,.$$

We consider $Q_j(y) := \frac{\mathrm{d}^p}{\mathrm{d}y^p}\left[(1-y^2)^j\, 2R_j(y)\right]$ with

$$\begin{aligned} R_j(y) :=& \tfrac{1}{2}\left((1-y)^{p-j} + (1+y)^{p-j}\right) \\ =& 1 + \binom{p-j}{2} y^2 + \binom{p-j}{4} y^4 + \dots \\ &+ \begin{cases} y^{p-j} & \text{for } p-j \text{ even}, \\ (p-j)y^{p-j-1} & \text{for } p-j \text{ odd}. \end{cases} \end{aligned}$$

Obviously $R_j(y)$ is an even polynomial in y of degree at most $p-j$ with positive coefficients and therefore

$$R_j^{(\ell)}(y) \geq 0 \text{ for } y \geq 0 \quad \text{and} \quad \max_{y\in[0,1]} |R_j^{(\ell)}(y)| = R_j^{(\ell)}(1). \tag{5.16}$$

By applying the Leibniz rule we get

$$\begin{aligned} Q_j(y) &= 2\sum_{k=0}^{p} \binom{p}{k} \frac{\mathrm{d}^k}{\mathrm{d}y^k}\left[(1-y^2)^j\right] \frac{\mathrm{d}^{p-k}}{\mathrm{d}y^{p-k}}\left[R_j(y)\right] \\ &= 2\sum_{k=j}^{p} \binom{p}{k} \frac{\mathrm{d}^{k-j}}{\mathrm{d}y^{k-j}} \frac{\mathrm{d}^j}{\mathrm{d}y^j}\left[(1-y^2)^j\right] \frac{\mathrm{d}^{p-k}}{\mathrm{d}y^{p-k}}\left[R_j(y)\right] \end{aligned}$$

and further by the Rodrigues formula of the Legendre polynomials, i.e., $P_j(x) = (-1)^j \frac{1}{2^j j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j}[(1-x^2)^j]$,

$$Q_j(y) = (-1)^j 2^{j+1} j! \sum_{k=j}^{p} \binom{p}{k} P_j^{(k-j)}(y) R_j^{(p-k)}(y).$$

We know that $\max_{y\in[0,1]} |P_j^{(k-j)}(y)| = P_j^{(k-j)}(1)$ (see, e.g., [100]). Consequently, we obtain together with (5.16) that

$$\max_{y\in[0,1]} \left|Q_j(y)\right| = 2^{j+1} j! \sum_{k=j}^{p} \binom{p}{k} P_j^{(k-j)}(1) R_j^{(p-k)}(1) = \left|Q_j(1)\right|. \tag{5.17}$$

On the other hand we conclude by the Leibniz rule that

$$\begin{aligned} Q_j(y) =& \frac{\mathrm{d}^p}{\mathrm{d}y^p}\Big[(1-y^2)^j\big[(1-y)^{p-j} + (1+y)^{p-j}\big]\Big] \\ =& \frac{\mathrm{d}^p}{\mathrm{d}y^p}\left[(1-y)^p(1+y)^j + (1-y)^j(1+y)^p\right] \\ =& p! \sum_{k=0}^{j} \binom{p}{k}\binom{j}{k} (-1)^k \Big((1-y)^k(1+y)^{j-k}(-1)^p \\ &\qquad + (1+y)^k(1-y)^{j-k}\Big). \end{aligned}$$

Now $|Q_j(1)|$ can be easily estimated by

$$\begin{aligned}
|Q_j(1)| &= p! \left| \sum_{k=0}^{j} \binom{p}{k} \binom{j}{k} (-1)^k \left(\delta_{k,0} 2^{j-k} (-1)^p + 2^k \delta_{k,j} \right) \right| \\
&= p! \left| (-1)^p 2^j + \binom{p}{j} 2^j (-1)^j \right| \\
&= 2^j p! \left| (-1)^j \binom{p}{j} + (-1)^p \right| \\
&\leq 2^j p! \left(\binom{p}{j} + 1 \right).
\end{aligned}$$

Combining this with (5.15) and (5.17), we obtain for $x \in [m, m+r]$ that

$$\begin{aligned}
\left| P^{(p)}(x) \right| &\leq \left(\frac{1}{2r} \right)^p \sum_{j=1}^{p-1} |\tilde{\gamma}_j| \, |Q_j(1)| \\
&\leq p! \left(\frac{1}{2r} \right)^p \left(\sum_{j=1}^{p-1} \binom{p}{j} 2^j + \sum_{j=1}^{p-1} 2^j \right) \max_{j=1,\ldots,p-1} |\tilde{\gamma}_j| \\
&= p! \left(\frac{1}{2r} \right)^p \left((1+2)^p - 2^p + 2^p - 3 \right) \max_{j=1,\ldots,p-1} |\tilde{\gamma}_j| \\
&< p! \left(\frac{3}{2r} \right)^p \max_{j=1,\ldots,p-1} |\tilde{\gamma}_j| \, .
\end{aligned}$$

It remains to estimate $\max |\tilde{\gamma}_j|$. By definition of $\tilde{\gamma}_j$ it follows

$$\begin{aligned}
|\tilde{\gamma}_j| &= \left| \sum_{\ell=0}^{j-1} \binom{p-1+\ell}{\ell} \frac{r^{j-\ell}}{2^\ell (j-\ell)!} \, a_{j-\ell} \right| \\
&\leq \sum_{\ell=0}^{j-1} \binom{p-1+\ell}{\ell} \frac{r^{j-\ell}}{2^\ell (j-\ell)!} \, |a_{j-\ell}| \;=:\; s_j.
\end{aligned}$$

Now one can easily check that $s_j \leq s_{j+1}$ for $1 \leq j \leq p-2$. Thus, $\max_{j=1,\ldots,p-1} |\tilde{\gamma}_j| \leq s_{p-1} = \gamma$ and we are done. □

Now we apply Theorem 2.4 and Corollary 2.3 with respect to our special polynomials K_I and K_B, i.e., we consider the intervals $[-\varepsilon_I, \varepsilon_I]$ and $[\frac{1}{2} - \varepsilon_B, \frac{1}{2} + \varepsilon_B]$ and set $a_j := K^{(j)}(-\varepsilon_I) = (-1)^j K^{(j)}(\varepsilon_I)$ and $a_j := K^{(j)}(\frac{1}{2} - \varepsilon_B)$, respectively. The result can be summarized as follows:

Corollary 2.5. *The polynomials K_I and K_B which satisfy* (5.9) *and* (5.10)*, respectively, are given by* (5.13) *with $y = \frac{x}{\varepsilon_I}$, $y = \frac{x-1/2}{\varepsilon_B}$ and $\gamma_j = \gamma_j^{I/B}$, respectively, where*

$$\begin{aligned}\gamma_j^I &:= \sum_{\ell=0}^{j} \binom{p-1+\ell}{\ell} \frac{(-1)^{j-\ell}\varepsilon_I^{j-\ell}}{2^\ell (j-\ell)!} K^{(j-\ell)}(\varepsilon_I),\\ \gamma_j^B &:= \sum_{\ell=0}^{j} \binom{p-1+\ell}{\ell} \frac{(-1)^{j-\ell}\varepsilon_B^{j-\ell}}{2^\ell (j-\ell)!} K^{(j-\ell)}\left(-\frac{1}{2}+\varepsilon_B\right).\end{aligned}$$

The polynomials fulfill the estimates

$$\max_{x\in[0,\varepsilon_I]} \left|K_I^{(p)}(x)\right| \leq p! \left(\frac{3}{2}\right)^p \varepsilon_I^{-p}\, \gamma^I, \tag{5.18}$$

$$\max_{x\in[\frac{1}{2}-\varepsilon_B,\frac{1}{2}]} \left|K_B^{(p)}(x)\right| \leq p! \left(\frac{3}{2}\right)^p \varepsilon_B^{-p}\, \gamma^B \tag{5.19}$$

with

$$\gamma^I := \sum_{\ell=0}^{p-2} \binom{p-1+\ell}{\ell} \frac{\varepsilon_I^{p-1-\ell}}{2^\ell (p-1-\ell)!} \left|K^{(p-1-\ell)}(\varepsilon_I)\right|, \tag{5.20}$$

$$\gamma^B := \sum_{\ell=0}^{p-2} \binom{p-1+\ell}{\ell} \frac{\varepsilon_B^{p-1-\ell}}{2^\ell (p-1-\ell)!} \left|K^{(p-1-\ell)}\left(\frac{1}{2}-\varepsilon_B\right)\right|. \tag{5.21}$$

3. Error Estimates

Beyond the well-known errors appearing in the NFFT computations which are discussed for example in [109], our algorithm introduces the errors $|f(y_j) - \tilde{f}(y_j)|$ ($j = 1, \ldots, M$). By (5.6), (5.7) and (5.3), we obtain for $|y| \leq \frac{1}{4} - \frac{\varepsilon_B}{2}$ that

$$\begin{aligned}\left|f(y) - \tilde{f}(y)\right| &= \left|\sum_{k=1}^{N} \alpha_k \left(\tilde{K}(y-x_k) - T_n(\tilde{K})(y-x_k)\right)\right|\\ &\leq \sum_{k=1}^{N} |\alpha_k|\, \|K_{\mathrm{err}}\|_\infty,\end{aligned}$$

where

$$\|K_{\mathrm{err}}\|_\infty := \max_{|x|\leq\frac{1}{2}} |K_{\mathrm{err}}(x)|, \quad K_{\mathrm{err}}(x) := \tilde{K}(x) - T_n(\tilde{K})(x). \tag{5.22}$$

Lemma 3.1. *Let K be an even kernel and let $\tilde{K} \in H^p(\mathbb{T})$ be defined by* (5.4)*. Then, for $2 \leq p \ll n$, the following estimate holds true: $\|K_{\mathrm{err}}\|_\infty \leq \frac{C}{(p-1)\pi^p n^{p-1}} \int_0^{\frac{1}{2}} |\tilde{K}^{(p)}(x)|\, \mathrm{d}x$.*

Proof. The proof follows by standard arguments. By Fourier expansion of $\tilde{K}$ and (5.5) we obtain for $x \in [-\frac{1}{2}, \frac{1}{2}]$ that

$$K_{\mathrm{err}}(x) = \sum_{k\in\mathbb{Z}} c_k(\tilde{K})\, \mathrm{e}^{2\pi \mathrm{i} kx} - \sum_{\ell\in I_n^1} b_\ell\, \mathrm{e}^{2\pi\mathrm{i}\ell x},$$

where the Fourier coefficients $c_k(\tilde{K})$ are defined in (3.3). Further, it follows by the aliasing formula (see (3.4)) that

$$K_{\mathrm{err}}(x) = \sum_{k\in I_n^1} \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} c_{k+rn}(\tilde{K})\, \mathrm{e}^{2\pi\mathrm{i}kx}(\, \mathrm{e}^{2\pi\mathrm{i}rnx} - 1).$$

Since $\tilde{K}$ is even, we can estimate

$$\|K_{\mathrm{err}}\|_\infty \leq 4 \sum_{k=\frac{n}{2}}^{\infty} \left|c_k(\tilde{K})\right|.$$

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

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

so that

$$\|K_{\mathrm{err}}\|_\infty \leq 4 \left(\sum_{k=\frac{n}{2}}^{\infty} (2\pi k)^{-p} \right) \int_{-\frac{1}{2}}^{\frac{1}{2}} \left|\tilde{K}^{(p)}(x)\right| \mathrm{d}x.$$

For $p \geq 2$ the above sum can be estimated by an upper integral

$$\|K_{\mathrm{err}}\|_\infty \leq \frac{2\big(1+\frac{p-1}{n}\big)}{(p-1)\pi^p n^{p-1}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \left|\tilde{K}^{(p)}(x)\right| \mathrm{d}x.$$

Since $p \ll n$, this implies the assertion with a constant $C \approx 4$. □

Now we obtain by the definition of $\tilde{K}$ that

$$\int_0^{\frac{1}{2}} \left|\tilde{K}^{(p)}(x)\right| \mathrm{d}x = \int_0^{\varepsilon_I} \left|K_I^{(p)}(x)\right| \mathrm{d}x + \int_{\varepsilon_I}^{\frac{1}{2}-\varepsilon_B} \left|K^{(p)}(x)\right| \mathrm{d}x + \int_{\frac{1}{2}-\varepsilon_B}^{\frac{1}{2}} \left|K_B^{(p)}(x)\right| \mathrm{d}x$$

and for the polynomials K_I and K_B in Corollary 2.5 by (5.18), (5.19)

$$\int_0^{\frac{1}{2}} \left|\tilde{K}^{(p)}(x)\right| \mathrm{d}x \leq p! \left(\frac{3}{2}\right)^p \left(\varepsilon_I^{1-p}\gamma^I + \varepsilon_B^{1-p}\gamma^B\right) + \int_{\varepsilon_I}^{\frac{1}{2}-\varepsilon_B} \left|K^{(p)}(x)\right| \mathrm{d}x. \tag{5.23}$$

It remains to estimate $K^{(p)}$ and the values γ^I, γ^B which depend on $K^{(j)}(\varepsilon_I)$ and $K^{(j)}(\frac{1}{2}-\varepsilon_B)$, respectively. Therefore we have to estimate the derivatives of K.

For the kernels (5.1) and $j \in \mathbb{N}$ we have

$$\left|K_\beta^{(j)}(x)\right| = \frac{(j+\beta-1)!}{(\beta-1)!}\,|x|^{-(j+\beta)} \qquad (x \neq 0;\ \beta \in \mathbb{N}_0), \tag{5.24}$$

where we set $(-1)! := 1$ in case $\beta = 0$.

Theorem 3.2. *For $\beta \in \mathbb{N}_0$, let $K = K_\beta$ be defined by (5.1) and $\tilde{K}$ by (5.4) with K_I and K_B given by Corollary 2.5, where $\varepsilon_I \leq \min\{\varepsilon_B, \frac{1}{2}-\varepsilon_B\}$. Then, for $2 \leq p \ll n$, the error $\|K_{\mathrm{err}}\|_\infty$ in (5.22) can be estimated by*

$$\|K_{\mathrm{err}}\|_\infty \leq C_\beta\,\frac{(p+\beta-2+\delta_{0,\beta})!}{\varepsilon_I^{p+\beta-1}}\,\frac{3^p}{\pi^p n^{p-1}} \tag{5.25}$$

with a constant C_β independent of p, n and ε_I.

Proof. We consider the summands in (5.23). By (5.24) we obtain that

$$\begin{aligned}\int_{\varepsilon_I}^{\frac{1}{2}-\varepsilon_B} \left|K^{(p)}(x)\right| \mathrm{d}x &= \frac{(p+\beta-1)!}{(\beta-1)!}\int_{\varepsilon_I}^{\frac{1}{2}-\varepsilon_B} |x|^{-(p+\beta)}\,\mathrm{d}x \\ &\leq \frac{(p+\beta-2)!}{(\beta-1)!}\,\varepsilon_I^{-(p+\beta-1)}.\end{aligned}$$

Since $\varepsilon_I \leq \min\{\varepsilon_B, \frac{1}{2}-\varepsilon_B\}$ it follows by (5.20), (5.21) and (5.24) that $\gamma^B \varepsilon_B^{1-p} \leq \gamma^I \varepsilon_I^{1-p}$. Thus it remains to estimate $\gamma^I \varepsilon_I^{1-p}$. By (5.20) and (5.24) we get

$$\begin{aligned}\gamma^I \varepsilon_I^{1-p} &\leq \frac{1}{\varepsilon_I^{p-1+\beta}} \sum_{\ell=0}^{p-2} \binom{p-1+\ell}{\ell} \frac{(p-2-\ell+\beta)!\,2^{-\ell}}{(\beta-1)!(p-1-\ell)!} \\ &\leq \frac{1}{\varepsilon_I^{p-1+\beta}} \binom{p-2+\beta}{\beta-1} \sum_{\ell=0}^{p-1} \binom{p-1+\ell}{\ell} 2^{-\ell},\end{aligned}$$

where we set $\binom{n}{-1} := 1$ in case $\beta = 0$. Using $y = 0$ in (5.14) we see that the last sum equals 2^{p-1} so that

$$p!\left(\frac{3}{2}\right)^p \gamma^I \varepsilon_I^{1-p} \leq \frac{p(p+\beta-2+\delta_{0,\beta})!\,3^p}{2(\beta-1)!}\,\varepsilon_I^{-(p+\beta-1)}.$$

Combining these estimates with (5.23) and Lemma 3.1 we obtain the assertion. □

Of course, for small c, the derivatives of the generalized multiquadrics $K_\beta(x;c)$ behave similar to those of $K_\beta(x)$. The following lemma estimates the derivatives of the generalized multiquadrics by taking c into account.

Lemma 3.3. *The derivatives of*

$$K(x) = K_\beta(x;c) := (x^2+c^2)^{-\frac{\beta}{2}} \qquad (\beta \in \mathbb{N} \text{ odd})$$

can be estimated by

$$\left|K_\beta^{(j)}(x;c)\right| \le \frac{\sqrt[4]{\pi\left(1+\frac{2c^2}{x^2}\right)}\,(j+\beta-1)!\,\sqrt{2}^j}{\Gamma\left(\frac{\beta}{2}\right)(x^2+c^2)^{\frac{j+\beta}{2}}}.$$

Proof. We use the well-known formula [112]

$$K_\beta(x;c) = \frac{1}{c^\beta\Gamma\left(\frac{\beta}{2}\right)}\int_0^\infty \mathrm{e}^{-t(x^2/c^2+1)}t^{(\beta-2)/2}\,\mathrm{d}t\,.$$

By differentiation we obtain

$$K_\beta^{(j)}(x;c) = \frac{1}{c^\beta\Gamma\left(\frac{\beta}{2}\right)}\int_0^\infty \frac{\mathrm{d}^j}{\mathrm{d}x^j}\left[\mathrm{e}^{-tx^2/c^2}\right]\mathrm{e}^{-t}t^{(\beta-2)/2}\,\mathrm{d}t\,.$$

Using the Rodrigues formula of Hermite polynomials , i.e., $H_j(x) = (-1)^j e^{x^2}\,\frac{\mathrm{d}^j}{\mathrm{d}x^j}\left[e^{-x^2}\right]$, we can rewrite this as

$$K_\beta^{(j)}(x;c) = \frac{1}{c^\beta\Gamma\left(\frac{\beta}{2}\right)}\int_0^\infty (-1)^j\,\mathrm{e}^{-tx^2/c^2}H_j\left(x\tfrac{\sqrt{t}}{c}\right)\left(\tfrac{\sqrt{t}}{c}\right)^j\mathrm{e}^{-t}t^{(\beta-2)/2}\,\mathrm{d}t\,.$$

Now we substitute $y = x\frac{\sqrt{t}}{c}$ and obtain

$$K_\beta^{(j)}(x;c) = \frac{2(-1)^j}{\Gamma\left(\frac{\beta}{2}\right)x^{j+\beta}}\int_0^\infty \mathrm{e}^{-y^2}H_j(y)\,\mathrm{e}^{-y^2c^2/x^2}y^{j+\beta-1}\,\mathrm{d}y\,.$$

Since the integrand is even, this is equal to

$$K_\beta^{(j)}(x;c) = \frac{(-1)^j}{\Gamma\left(\frac{\beta}{2}\right)x^{j+\beta}}\int_{-\infty}^\infty \mathrm{e}^{-y^2}H_j(y)\,\mathrm{e}^{-y^2c^2/x^2}y^{j+\beta-1}\,\mathrm{d}y\,.$$

By the Cauchy-Schwarz inequality we get

$$\begin{aligned}\left|K_\beta^{(j)}(x;c)\right| \;\le\; & \frac{1}{\Gamma\left(\frac{\beta}{2}\right)|x|^{j+\beta}}\left(\int_{-\infty}^\infty \mathrm{e}^{-y^2}H_j^2(y)\,\mathrm{d}y\right)^{\frac{1}{2}} \\ & \left(\int_{-\infty}^\infty \mathrm{e}^{-y^2(1+2c^2/x^2)}y^{2(j+\beta-1)}\,\mathrm{d}y\right)^{\frac{1}{2}}.\end{aligned}$$

By the normalization of the Hermite polynomials, i.e.,

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2} H_j(x) H_m(x)\,\mathrm{d}y = \begin{cases} 0 & \text{for } j \neq m, \\ 2^j j!\sqrt{\pi} & \text{for } j = m, \end{cases}$$

the first integral is equal to $2^j j!\sqrt{\pi}$. To evaluate the second integral we set $\alpha^2 := 1 + \frac{2c^2}{x^2}$ and use that

$$\int_{-\infty}^{\infty} \mathrm{e}^{-\alpha^2 y^2} y^{2(j+\beta-1)}\,\mathrm{d}y = \frac{1}{\alpha^{2(j+\beta)-1}} \Gamma(j+\beta-\tfrac{1}{2}) \leq \frac{1}{\alpha^{2(j+\beta)-1}} (j+\beta-1)!\,.$$

Combining these estimates we arrive at

$$\left| K_\beta^{(j)}(x;c) \right| \leq \frac{\sqrt[4]{\pi\left(1+\frac{2c^2}{x^2}\right)}\,\left((j+\beta-1)!\,j!\,2^j\right)^{1/2}}{\Gamma\left(\frac{\beta}{2}\right)\left(x^2+2c^2\right)^{\frac{j+\beta}{2}}}\,.$$

□

Theorem 3.4. *For odd $\beta \in \mathbb{N} \cup \{-1\}$, let $K = K_\beta(\,\cdot\,;c)$ be defined by (5.2) and $\tilde{K}$ by (5.4) with K_I and K_B given by Corollary 2.5, where $\varepsilon_I \leq \min\{\varepsilon_B, \frac{1}{2} - \varepsilon_B\}$. Further, let $0 < c \leq \varepsilon_I$. Then the error $\|K_{\mathrm{err}}\|_\infty$ in (5.22) can be estimated by*

$$\|K_{\mathrm{err}}\|_\infty \leq C_\beta \frac{(p+\beta-2+2\delta_{-1,\beta})!}{\left(\varepsilon_I^2+c^2\right)^{\frac{p+\beta-1}{2}}} \frac{(3\sqrt{2})^p}{\pi^p n^{p-1}}$$

with a constant C_β independent of p, n and ε_I.

Proof. The proof follows the same lines as the proof of Theorem 3.2.
First we obtain for $\beta \in \mathbb{N}$ by Lemma 3.3 and since $c^2 \leq \varepsilon_I^2$ that

$$\begin{aligned} \int_{\varepsilon_I}^{\frac{1}{2}-\varepsilon_B} |K^{(p)}(x)|\,\mathrm{d}x &\leq \frac{C\,(p+\beta-1)!\sqrt{2}^{\,p}}{\Gamma\left(\frac{\beta}{2}\right)} \int_{\varepsilon_I}^{\frac{1}{2}-\varepsilon_B} (x^2+c^2)^{-(p+\beta)/2}\,\mathrm{d}x \\ &\leq \frac{C\,(p+\beta-2)!\sqrt{2}^{\,p+1}}{\Gamma\left(\frac{\beta}{2}\right)} \left(\varepsilon_I^2+c^2\right)^{-(p+\beta-1)/2}. \end{aligned}$$

Next we have for $\beta \in \mathbb{N}$ by (5.20) and Lemma 3.3 that

$$\begin{aligned} \gamma^I \varepsilon_I^{1-p} \leq\ & \frac{C\,\sqrt{2}^{p-1}}{\Gamma\left(\frac{\beta}{2}\right)\left(\varepsilon_I^2+c^2\right)^{(p+\beta-1)/2}} \times \\ & \sum_{\ell=0}^{p-2} \binom{p-1+\ell}{\ell} \frac{(p-2-\ell+\beta)!}{(p-1-\ell)!} \left(\frac{\sqrt{\varepsilon_I^2+c^2}}{2\sqrt{2}\varepsilon_I} \right)^{\ell} \end{aligned} \tag{5.26}$$

and since $c^2 \leq \varepsilon_I^2$ further

$$\begin{aligned}\gamma^I \varepsilon_I^{1-p} &\leq C_\beta \frac{(p-2+\beta)!\,\sqrt{2}^{p-1}}{(p-1)!\,(\varepsilon_I^2+c^2)^{(p+\beta-1)/2}} \sum_{\ell=0}^{p-1} \binom{p-1+\ell}{\ell} 2^{-\ell} \\ &\leq C_\beta \frac{(p-2+\beta)!\,(2\sqrt{2})^{p-1}}{(p-1)!\,(\varepsilon_I^2+c^2)^{(p+\beta-1)/2}}\,.\end{aligned}$$

This results in

$$p!\left(\frac{3}{2}\right)^p \gamma^I \varepsilon_I^{1-p} \leq C_\beta \frac{p\,(p+\beta-2)!\,(3\sqrt{2})^p}{2\sqrt{2}}\,(\varepsilon_I^2+c^2)^{-(p+\beta-1)/2}\,.$$

Substituting of these estimates in (5.23) and applying Lemma 3.1 we obtain the assertion for $\beta \in \mathbb{N}$.

The case $\beta = -1$ follows similarly by using the fact that the Hardy multiquadric $K_{-1}(x;c) = (x^2+c^2)^{\frac{1}{2}}$ fulfills

$$K_{-1}^{(j)}(x;c) = c^2\,K_3^{(j-2)}(x;c) \qquad (j = 2,\,3,\,\ldots)\,.$$

□

Note that the right hand side of (5.26) also converges under the weaker condition $c^2 < 7\varepsilon_I^2$ so that one can prove similar estimates with d^p, $d > 3\sqrt{2}$, instead of $(3\sqrt{2})^p$ assuming weaker conditions than $c^2 < \varepsilon_I^2$.

We will use the estimates in the Theorems 3.2 and 3.4 to specify the parameters ε_I, p and n of our algorithm. Since both cases can be handled in the same way, we restrict our attention to Theorem 3.2. Using the Stirling formula $p! \leq 1.1\,\sqrt{2\pi p}\,\left(\frac{p}{\mathrm{e}}\right)^p$ we can rewrite our error estimate as

$$\|K_{\mathrm{err}}\|_\infty \leq \tilde{C}_\beta\,\varepsilon_I^{-\beta}\left(\frac{3}{\mathrm{e}\,\pi}\,\frac{p-1}{\varepsilon_I\,n}\right)^{p-1}\frac{(p+\beta-2+\delta_{0,\beta})!\sqrt{2\pi(p-1)}}{(p-1)!}\,.$$

Thus, choosing ε_I such that $\frac{3(p-1)}{\mathrm{e}\,\pi\,\varepsilon_I n} < 1$, our error decays exponentially in p. In our numerical examples we choose

$$\varepsilon_I = \frac{p}{n}\,. \tag{5.27}$$

While (5.27) steers the error, condition (5.8) on ε_I is necessary to keep the near field computation linear in M. Now (5.27) and (5.8) together imply that

$$n \approx \frac{2Np}{\nu}\,. \tag{5.28}$$

If $M = N$, then the near field computation requires approximately

$$\nu N$$

and the NFFT computations

$$n \log n + \mathcal{O}(N) = \frac{2Np}{\nu} \log \left(\frac{2Np}{\nu} \right) + \mathcal{O}(N)$$

arithmetic operations. One should choose ν such that both operation counts are balanced. It seems that $\nu \approx 2\sqrt{p}$, respectively by (5.28),

$$n \approx \sqrt{p} N$$

is a good choice.

4. Fast Summation at multidimensional nodes

In this section we briefly explain how to extend our one-dimensional scheme to higher dimensions $d \geq 2$ and rotation-invariant kernels $\mathcal{K}(\boldsymbol{x}) = K\big(\|\boldsymbol{x}\|\big)$. We focus on the fast computation of

$$f(\boldsymbol{y}_j) := \sum_{k=1}^{N} \alpha_k \mathcal{K}(\boldsymbol{y}_j - \boldsymbol{x}_k) = \sum_{k=1}^{N} \alpha_k K\big(\|\boldsymbol{y}_j - \boldsymbol{x}_k\|\big) \qquad (\boldsymbol{x}_k,\, \boldsymbol{y}_j \in \mathbb{R}^d) \tag{5.29}$$

for $j = 1, \ldots, M$. Similar as in Section 2 we regularize $\mathcal{K}$ near 0 and near the boundary of $[-\frac{1}{2}, \frac{1}{2})^d$ to obtain a smooth periodic kernel $\tilde{\mathcal{K}}$:

$$\tilde{\mathcal{K}}(\boldsymbol{x}) := \begin{cases} K_I\big(\|\boldsymbol{x}\|\big) & \text{if } \|\boldsymbol{x}\| \leq \varepsilon_I, \\ K_B\big(\|\boldsymbol{x}\|\big) & \text{if } \frac{1}{2} - \varepsilon_B < \|\boldsymbol{x}\| < \frac{1}{2}, \\ K_B\big(\frac{1}{2}\big) & \text{if } \|\boldsymbol{x}\| \geq \frac{1}{2}, \\ K\big(\|\boldsymbol{x}\|\big) & \text{otherwise.} \end{cases}$$

Here we choose K_I as in Corollary 2.3. But instead of (5.10) we require that the polynomial K_B fulfills the conditions

$$\begin{aligned} K_B^{(j)}\left(\frac{1}{2} - \varepsilon_B\right) &= K^{(j)}\left(\frac{1}{2} - \varepsilon_B\right) \quad (j = 0, \ldots, p-1), \\ K_B^{(j)}\left(\frac{1}{2}\right) &= \delta_{0,j}\, K\left(\frac{1}{2}\right) \qquad (j = 0, \ldots, p-1). \end{aligned} \tag{5.30}$$

The unique solution K_B of (5.30) is given by Theorem 2.2, but now it does not have the symmetry of Corollary 2.3.

Then we approximate $\tilde{\mathcal{K}}$ by the Fourier series

$$T_n(\tilde{\mathcal{K}})(\boldsymbol{x}) := \sum_{\boldsymbol{\ell} \in I_n^d} b_{\boldsymbol{\ell}}\, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\ell} \boldsymbol{x}},$$

where

$$b_{\boldsymbol{\ell}} := \frac{1}{n^d} \sum_{\boldsymbol{j} \in I_n^d} \tilde{\mathcal{K}}\Big(\frac{\boldsymbol{j}}{n}\Big)\, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{j} \boldsymbol{\ell} / n} \qquad (\boldsymbol{\ell} \in I_n^d).$$

Now we can decompose the original kernel as

$$\mathcal{K} = (\mathcal{K} - \tilde{\mathcal{K}}) + \big(\tilde{\mathcal{K}} - T_n(\tilde{\mathcal{K}})\big) + T_n(\tilde{\mathcal{K}})$$

and, by neglecting the summand in the middle, we approximate f by

$$\tilde{f}(\boldsymbol{x}) := \sum_{k=1}^{N} \alpha_k (\mathcal{K} - \tilde{\mathcal{K}})(\boldsymbol{x} - \boldsymbol{x}_k) + \sum_{k=1}^{N} \alpha_k T_n(\tilde{\mathcal{K}})(\boldsymbol{x} - \boldsymbol{x}_k). \tag{5.31}$$

Instead of f we evaluate $\tilde{f}$ at the nodes $\boldsymbol{y}_j \in \mathbb{R}^d$ ($j = 1, \ldots, M$) by the following two steps:

1) Near field computation (first sum in (5.31))
To achieve the desired complexity of our algorithm we suppose that either the N points $\boldsymbol{x}_k$ or the M points $\boldsymbol{y}_j$ are 'sufficiently uniformly distributed' in the ball with radius $\frac{1}{2} - \varepsilon_B$, i.e., we suppose that there exists a small constant $\nu \in \mathbb{N}$ such that each ball with radius ε_I contains at most ν of the points $\boldsymbol{x}_k$ or of the points $\boldsymbol{y}_j$, respectively. This implies that ε_I depends linearly on $N^{-1/d}$, respectively $M^{-1/d}$. In the following we restrict our attention to the case

$$\varepsilon_I \approx \frac{1}{2} \left(\frac{\nu}{N}\right)^{1/d}. \tag{5.32}$$

Then, as in one dimension, the computation of the first sum requires only $\leq \nu M$ arithmetic operations.

2) NFFT based summation (second sum in (5.31))
The evaluation of the second sum in (5.31) is done exactly in the same way as in one dimension, but with d-dimensional NFFTs of size n now, which really involve a multidimensional setting. This computation part requires $\mathcal{O}(n^d \log n + N + M)$ arithmetic operations.

To obtain an exponential error decay in p, we have to choose again $\varepsilon_I \approx \frac{p}{n}$; see (5.27). On the other hand, we have to ensure (5.32) for an efficient near field computation. Thus,

$$n \approx 2p \left(\frac{N}{\nu}\right)^{1/d}.$$

To get a balanced arithmetic complexity of both parts of our algorithm one may choose $n \approx \sqrt{p}\, N^{1/d}$ if $N = M$.

5. Numerical results

Our algorithms were implemented in C using double precision arithmetic and tested on an AMD Athlon(tm) XP 1800+, 512MB RAM, SuSe-Linux 8.2.

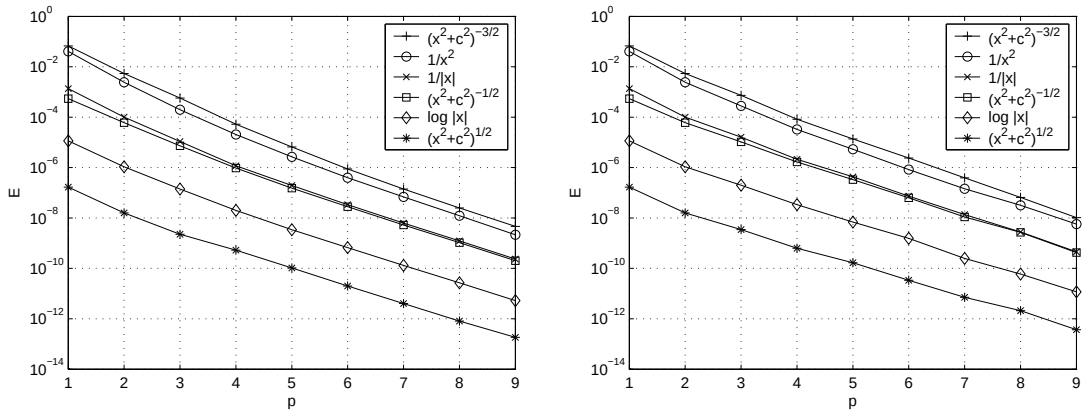


Figure 5.2.: Error E in dependence on p for various kernels in 2D with $N = 512^2$, $n = 512$; regularization by spline interpolation (left) and by polynomial interpolation (right).

Throughout our experiments we apply the NFFT/NFFT$^{\mathrm{T}}$ package [88] with Kaiser-Bessel functions and oversampling factor $\sigma = 2$.

For simplicity we have chosen $M = N$ in our summation algorithm and randomly distributed nodes $\boldsymbol{y}_j = \boldsymbol{x}_j$ ($j = 1, \ldots, N$) in $\{\boldsymbol{x} : \|x\| \leq \frac{7}{32}\}$, i.e., $\varepsilon_B = \frac{1}{16}$. The coefficients α_k were randomly distributed in $[0, 1]$. Moreover, we set $\varepsilon_I = \frac{p}{n}$.

We are interested in the error

$$E := \max_{j=1,\ldots,N} \frac{|f(\boldsymbol{x}_j) - \tilde{f}(\boldsymbol{x}_j)|}{|f(\boldsymbol{x}_j)|}\,. \tag{5.33}$$

Figure 5.2 shows the behavior of E in 2D for various kernels in (5.1) and (5.2) with spline regularization (left) and regularization by algebraic polynomials (right). Here we have chosen $N = 512^2$ points, $n = \sqrt{N}$ and $c = 1/\sqrt{N}$ as parameter of the generalized multiquadrics. Further we use the truncation parameter $m = 8$ in the NFFT computations. First we observe that the error E with spline regularization is slightly better than the error with regularization by algebraic polynomials. Further, the results confirm the exponential error decay with increasing p proved in the Theorems 3.2 and 3.4. In the following we will always use regularization by polynomial interpolation.

Figure 5.3 presents the 1D error E in dependence on p for the Hardy multiquadric (left) and the inverse Hardy multiquadric (right) with various scaling parameters c. Here we took $n = N = 1024$. Further we use the truncation parameter $m = 8$ in the NFFT computations. As expected, for decreasing c, the error increases until $c = \frac{1}{N}$, where it is approximately the same as for $c = 0$ in both cases. For $c = 1$, the error is about the same for both multiquadrics. In this case, we can also apply the algorithm without inner regularization, i. e. without near field computation. The corresponding curve is drawn with symbol $\triangle$. Note that without inner regularization n does not depend on N and the complexity of our algorithm

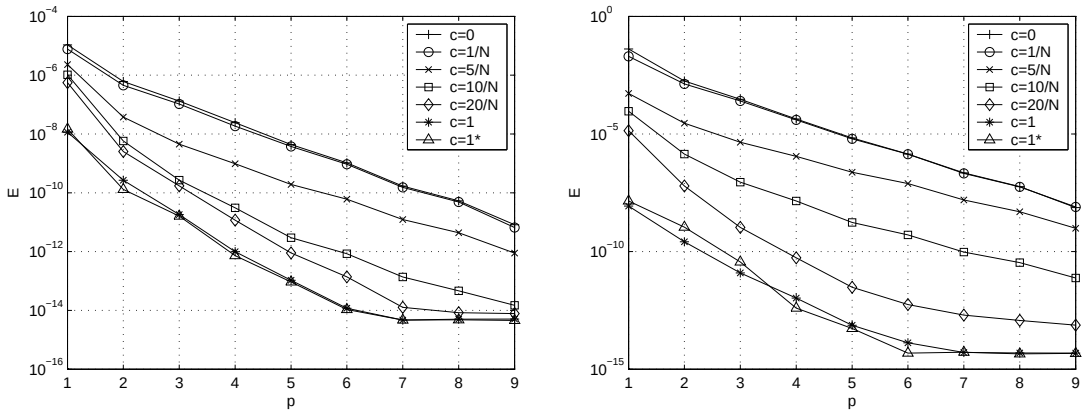


Figure 5.3.: Error E in dependence on p for the Hardy multiquadric (left) and the inverse multiquadric (right) in 1D with various parameters c and $n = N = 1024$. Here $c = 1^*$ denotes the algorithm without near field computation.

becomes linear in N.

Figure 5.4 compares the computational time in dependence on the number N of two-dimensional points for the direct computation of (5.29) and for our algorithm. As kernel function we have used $K(\boldsymbol{x}) = \log\|\boldsymbol{x}\|$. The parameters for our algorithm were $n = 2\sqrt{N}$ and $p = 4$ to achieve an accuracy of $E \leq 10^{-6}$. Further we use the truncation parameter $m = 4$ in the NFFT computations. Note that the computation time for the near field computation includes the time for the search of all points in the near field which requires $\mathcal{O}(\log N)$. The direct computation for $N = 2^{20}$ was only estimated based on the computational time and error for the first 1000 points, since the direct computation would take about 66 hours. Comparing this time with about 1.6 minutes required by our algorithm, the time saving for large problem sizes N becomes clear.

Finally, Table 5.1 compares the computational times required by our algorithm and by the algorithm proposed by Beatson et al. in [27]. In order to achieve an error $E \approx 10^{-6}$ in our algorithm, we have chosen $m = 4$ and $p = 3$. Further we have adapted the length $n \approx \sqrt{pN}$ of our NFFT such that the incorporated FFTs show a good performance. As in [27] the multiquadric parameter was $c = \frac{1}{\sqrt{N}}$ and the coefficients were $\alpha_k = 1$ for all $k = 1, \ldots, N$. The computational times for the Beatson algorithm were taken from Table 9.1 in [27]. Note that a different hardware was used for both algorithms so that the time for the direct computation may serve as a measure for comparison.

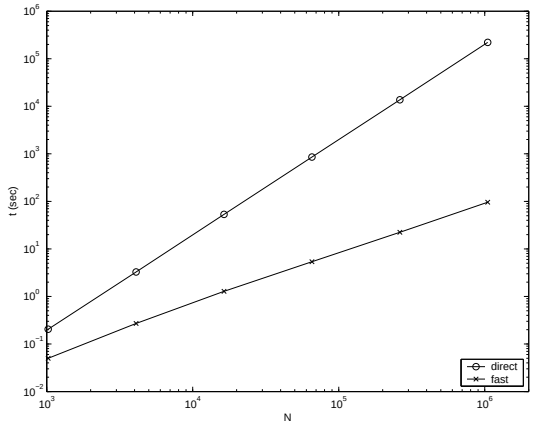


Figure 5.4.: Computational time versus the number N of points in 2D for the direct summation and our algorithm with $n = 2\sqrt{N}$ and $K(x) = \log|x|$.

		our algorithm		Beatson et al.	
N	n	direct	fast	direct	fast
2000	96	$2.70 \cdot 10^{-1}$	$6.0 \cdot 10^{-2}$	$2.97 \cdot 10^{-1}$	$7.8 \cdot 10^{-2}$
4000	144	$1.02 \cdot 10^{+0}$	$1.50 \cdot 10^{-1}$	$1.19 \cdot 10^{+0}$	$2.03 \cdot 10^{-1}$
8000	180	$4.48 \cdot 10^{+0}$	$3.10 \cdot 10^{-1}$	$4.75 \cdot 10^{+0}$	$4.84 \cdot 10^{-1}$
16000	216	$2.32 \cdot 10^{+1}$	$7.20 \cdot 10^{-1}$	$2.50 \cdot 10^{+1}$	$9.84 \cdot 10^{-1}$
32000	288	$9.33 \cdot 10^{+1}$	$1.83 \cdot 10^{+0}$	$1.10 \cdot 10^{+2}$	$2.23 \cdot 10^{+0}$

Table 5.1.: Computational times (in seconds) of the algorithm of Beatson et al. in [27] and of our algorithm for $K(x) = \sqrt{x^2 + c^2}$ in $\mathbb{R}^2$.

VI. NFFT-based Ridgelet Transform

Over the past few years, wavelet methods have been applied for the analysis of functional surfaces including nano-surfaces. Unfortunately, the usual discrete wavelet transform (DWT) suffers from shift-sensitivity, poor directionality and pseudo-Gibbs artifacts. Recently, a dual-tree complex wavelet transform (DT CWT) combined with total variation (TV) minimization has been applied by Ma et al. [94, 93] to solve the above problems. However, wavelet-based techniques show a poor performance at representing line singularities. In this chapter, we pay attention to extract line scratches from engineering surfaces by applying the discrete ridgelet transform.

Ridgelets have been designed by Candès and Donoho [18, 20] to deal with line singularities effectively by mapping them into point singularities using the Radon transform. It should be noted that several other geometric multiresolution structures such as curvelets by Candès and Donoho [21], bandelets by LePennec and Mallat [91] or contourlets by Do and Vetterli [33] have been proposed to restore local image features in a different way. Moreover, these methods have to compete with anisotropic diffusion filtering [132].

When implementing a discrete ridgelet transform one has to cope with certain technical difficulties. The basic strategy of the ridgelet transform is an application of the wavelet transform on the projections of the Radon transform. The Radon transform seems natural and simple on the continuum but it is a challenging problem for discrete data.

Do and Vetterli [34] proposed an orthonormal version of the ridgelet transform based on a discrete Radon transform defined on the finite grid $\mathbb{Z}_p^2$, where p is a prime number. Unfortunately, the $\mathbb{Z}_p^2$ Radon transform integrates over 'lines' which are defined algebraically — due to the arithmetic modulo p — rather than geometrically. This causes a wrap-around effect, i.e., texture-like artifacts in reconstructions.

Carré and Andres [22] presented a so-called discrete analytical ridgelet transform (DART) with a flexible redundancy factor based on discrete analytical lines, which only cause a limited wrap-around effect. By using an arithmetical thickness they choose the best discrete approximation of the Euclidean line for each line direction. The innovative step of this transform is the construction of discrete analytical lines in the Fourier domain, which allows a fast perfect backprojection without interpolation or iteration.

Other discrete ridgelet transforms have been also explored by Donoho et al. In [121], they offer a discrete transform with exact reconstruction, stability against perturbations, and low computational complexity. It uses the linogram grid by

means of a simple nearest-neighbor interpolation scheme. In [37], an effective discrete ridgelet transform based on so-called true ridge functions was proposed using the Fast Slant Stack (FSS) in [4], a pseudopolar FFT based discrete Radon transform, followed by fast 1D wavelet transforms. The essential of the FSS-based 'true' ridgelet transform is an interpolation performed using a Dirichlet kernel, which leads to a transform that is geometrically faithful and has no wrap-around effect. But an iterative approximation process is required for the inverse transform.

In this chapter, we develop a discrete complex ridgelet transform based on the NFFT. This ridgelet transform uses the NFFT for the computation of the discrete Radon transform. As the FSS, this approach completely avoids linear interpolations and requires only $\mathcal{O}(n^2 \log n)$ arithmetic operations. Then, 1D DT CWTs are applied to the projections of the Radon transform. The DT CWT was introduced by Kingsbury [82, 83] to cope with the lack of translation and rotation invariance of the (decimated) wavelet transform in an efficient way. For a mathematical treatment of the DT CWT see also [115, 103]. Replacing the DWT by approximate shift invariant DT CWT improves the quality of ridgelet denoising remarkably.

For denoising or feature extraction, a shrinkage function is applied to the ridgelet coefficients. However, this usual shrinkage is far from optimal and leads to blurred edges and pseudo-Gibbs artifacts. Therefore, we combine thresholding with TV minimization in order to reduce these artifacts.

TV minimization was first introduced by Rudin, Osher and Fatemi [113] for denoising, and then has been widely studied in image processing and computer vision (e.g. restoration, inpainting, blind deconvolution). Recently, the TV model has been combined with computational harmonic analysis (e.g. wavelets, wavelet packets, curvelets, etc.) to reduce both the pseudo-Gibbs and staircasing artifacts [96, 25, 39, 93, 123]. The idea of coupled TV minimization used in this chapter is similar to [39, 93], but applied for complex ridgelet coefficients and in two dimensions. The ridgelet transform combined with TV minimization attempts to give a better restoration of ridgelet coefficients, since it does not set the nonsignificant ridgelet coefficients simply to zero, but typically inputs optimal small values to cancel the oscillations (pseudo-Gibbs artifacts) in the vicinity of discontinuities and eliminate the ripples, while perfectly preserving the strong edges and shapes of features.

Although NFFT-based Radon transform, DT CWT and TV minimization are not new, they are combined for the first time. The main ideas of this chapter are also published in [49].

This chapter is organized as follows. First, we introduce the continuous Radon transform in Section 1. In Section 2, we explain our NFFT-based discrete Radon transform. Then, the continuous ridgelet transform is introduced in Section 3. Our new discrete complex ridgelet transform and its inverse are described in Section 4. Thereafter, we explain hard thresholding and combine it with TV minimization in Section 5. Numerical experiments show the good performance of our new method for feature extraction and denoising in Section 6.

1. Continuous Radon Transform

In this section we introduce the continuous Radon transform. As general reference we recommend the books of Natterer [101] and Natterer, Wübbeling [102].

The (two dimensional) *continuous Radon Transform* maps a function on $\mathbb{R}^2$ into the set of its integrals over the straight lines of $\mathbb{R}^2$. More precisely, for an angle $\varphi \in [0, 2\pi)$, $\boldsymbol{\theta} := (\cos\varphi, \sin\varphi)^T \in S^1$, and an offset $s \in \mathbb{R}$, let $\boldsymbol{x}\boldsymbol{\theta} = x_1 \cos\varphi + x_2 \sin\varphi = s$ specify a straight line of $\mathbb{R}^2$. Then, the Radon transform for $f \in L^2(\mathbb{R}^2)$ is defined as

$$\mathcal{R}_{\boldsymbol{\theta}} f(s) := \mathcal{R} f(\varphi, s) := \int\limits_{\boldsymbol{x}\boldsymbol{\theta}=s} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int\limits_{\mathbb{R}^2} f(\boldsymbol{x}) \delta(s - \boldsymbol{x}\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x} \tag{6.1}$$

with the Dirac delta function δ, i.e., $\delta(0) = 1$ and zero else.

With the orthogonal complement $\boldsymbol{\theta}^\perp = (-\sin\varphi, \cos\varphi)^T$, we can also write

$$\mathcal{R}_{\boldsymbol{\theta}} f(s) = \int\limits_{\mathbb{R}} f\left(s\boldsymbol{\theta} + t\boldsymbol{\theta}^\perp\right) \mathrm{d}t \, .$$

Note that—by abuse of notation— $\mathcal{R}_{\boldsymbol{\theta}} f(s)$ denotes the projection along $\boldsymbol{\theta}^\perp$ onto $s\boldsymbol{\theta}$.

The Fourier transform and the Radon transform are connected by the so-called 'projection theorem' or 'Fourier Slice Theorem' [101].

Theorem 1.1. *Let $f \in S(\mathbb{R}^2)$, where $S(\mathbb{R}^2)$ denotes the Schwartz space. For $\varphi \in [0, 2\pi)$, $\boldsymbol{\theta} := (\cos\varphi, \sin\varphi)^T$, and $\sigma \in \mathbb{R}$ it holds*

$$\hat{f}(\sigma\boldsymbol{\theta}) = \widehat{\mathcal{R}_{\boldsymbol{\theta}} f}(\sigma) \, .$$

Proof. We have that

$$\begin{aligned} \widehat{\mathcal{R}_{\boldsymbol{\theta}} f}(\sigma) &= \int\limits_{\mathbb{R}} \mathcal{R}_{\boldsymbol{\theta}} f(s) \, \mathrm{e}^{-2\pi \mathrm{i} \sigma s} \, \mathrm{d}s \\ &= \int\limits_{\mathbb{R}} \int\limits_{\mathbb{R}} f(s\boldsymbol{\theta} + t\boldsymbol{\theta}^\perp) \, \mathrm{e}^{-2\pi \mathrm{i} \sigma s} \, \mathrm{d}s \, \mathrm{d}t \, . \end{aligned}$$

With the substitution $\boldsymbol{x} = s\boldsymbol{\theta} + t\boldsymbol{\theta}^\perp$ for the integration variable, we get with $\mathrm{d}\boldsymbol{x} = \mathrm{d}s \, \mathrm{d}t$ and $s = \boldsymbol{\theta}\boldsymbol{x}$ that

$$\begin{aligned} \widehat{\mathcal{R}_{\boldsymbol{\theta}} f}(\sigma) &= \int\limits_{\mathbb{R}^2} f(\boldsymbol{x}) \, \mathrm{e}^{-2\pi \mathrm{i} \sigma \boldsymbol{\theta}\boldsymbol{x}} \, \mathrm{d}\boldsymbol{x} \\ &= \hat{f}(\sigma\boldsymbol{\theta}) \, . \end{aligned}$$

□

By Theorem 1.1 the continuous Radon transform can be written as

$$\mathcal{R}_{\boldsymbol{\theta}} f(s) = \int\limits_{\mathbb{R}} \hat{f}(\sigma\theta) \, \mathrm{e}^{2\pi \mathrm{i} s \sigma} \, \mathrm{d}\sigma \, . \tag{6.2}$$

2. Discrete Radon Transform

In this section, we are interested in an efficient and high quality discrete Radon transform based on the NFFT, e.g., applied in [108].

Let discrete data $f_{\boldsymbol{k}} = f(\frac{1}{R}\boldsymbol{k})$ ($\boldsymbol{k} \in I_N^2$, $R \geq N$) be given. Since none or only few of these discrete data points would fall on a given straight line $\boldsymbol{x\theta} = s$, we have to modify the approach of equation (6.1). The idea is to smear the straight lines somewhat in the sense that discrete data points close-by the straight line will be included. This can be done by approximating the Dirac delta function in equation (6.1) with a kernel function

$$K_R(x) := \sum_{r \in I_R} w_r \, \mathrm{e}^{2\pi \mathrm{i} r x}. \tag{6.3}$$

Now, we can introduce a semi-discrete Radon transform as the discrete analog of equation (6.1), i.e.,

$$R_{\boldsymbol{\theta}} f(s) := \sum_{\boldsymbol{k} \in I_N^2} f_{\boldsymbol{k}} K_R\Big(s - \frac{1}{R}\boldsymbol{k\theta}\Big). \tag{6.4}$$

The second question is how to discretize the set of straight lines $\boldsymbol{x\theta} = s$. Based on the results of Chapter II, Section 5, we will restrict to the following discrete set of straight lines.

The offset will be equally subdivided, but instead of specifying an angle, we use the slope. Two different types of lines have to be distinguished. These are

$$x_1 + \frac{4t}{T} x_2 = \frac{s}{R} \qquad \text{and} \qquad -\frac{4t}{T} x_1 + x_2 = \frac{s}{R} \qquad (t \in I_{T/2},\ s \in I_R).$$

They differ only by their slopes in x_1 and x_2, respectively. Their directions are given by

$$\boldsymbol{\theta}_t^h := \Big(1, \frac{4t}{T}\Big)^T \qquad \text{and} \qquad \boldsymbol{\theta}_t^v := \Big(-\frac{4t}{T}, 1\Big)^T$$

with $t \in I_{T/2}$. To simplify matters, we introduce the symbol $\boldsymbol{\theta}_t$ ($t \in I_T$) defined by $\boldsymbol{\theta}_t := \boldsymbol{\theta}_{t+T/4}^h$ for $t < 0$ and $\boldsymbol{\theta}_t := \boldsymbol{\theta}_{t-T/4}^v$ for $t \geq 0$.

Using this discrete set of straight lines together with the definition of the kernel function in equation (6.4), we define the *discrete Radon transform* as

$$R_{\boldsymbol{\theta}_t} f\left(\frac{s}{R}\right) = \sum_{r \in I_R} w_r \sum_{\boldsymbol{k} \in I_N^2} f_{\boldsymbol{k}} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \left(\frac{r}{R}\boldsymbol{\theta}_t\right)} \, \mathrm{e}^{2\pi \mathrm{i} r s / R} \qquad (t \in I_T,\ s \in I_R). \tag{6.5}$$

Obviously, a fast algorithm for the computation of the discrete Radon transform is given by computing the inner sum of equation (6.5) by a 2D NFFT at the knots of the linogram grid, followed by a multiplication with the Fourier coefficients of the kernel function and finished by computing the outer sum by 1D iFFTs for every direction $\boldsymbol{\theta}_t$.

Since the last step can be easily inverted by 1D iFFTs and the first step can be inverted (approximately to arbitrary accuracy if $R \geq N$ and $T \geq 2N$) by 2D iNFFT

as discussed in Chapter II, Section 5, we also have a fast algorithm for the inverse discrete Radon transform.

By choosing the Dirichlet kernel

$$D_{R/2-1}(x) := \sum_{r=-R/2+1}^{R/2-1} \mathrm{e}^{2\pi \mathrm{i} r x} = \frac{\sin(R-1)\pi x}{\sin(\pi x)}$$

or the Fejér kernel

$$F_{R/2-1}(x) := \sum_{r=-R/2+1}^{R/2-1} \left(1 - \frac{|r|}{R/2}\right) \mathrm{e}^{2\pi \mathrm{i} r x} = \frac{2}{R} \left(\frac{\sin(R/2)\pi x}{\sin(\pi x)}\right)^2$$

in (6.3), i.e., in particular $w_{-R/2} = 0$ and $w_{-r} = w_r$ ($r \in I_R$), we assure real values for the discrete Radon transform.

In Figure 6.1, we compare the results of the discrete Radon transform with these kernels. In (a) you can see an (256×256) pixels image of an object used in [34]. It is considered in order to compare our later algorithm with other methods conveniently. Figure 6.3 (b) shows the object contaminated with additive zero-mean Gaussian white noise. As can be seen from Figure 6.1, the discrete Radon transform with Fejér kernel (d) results in less noisy projections compared to the ones with Dirichlet kernel (c).

Remark 2.1. Using a (modified) Dirichlet kernel in our definition of the discrete Radon transform (6.5) leads essentially to the same notion of discrete Radon transform as proposed by Averbuch et al. [4]. They call their transform *Fast Slant Stack*. For the computation of the inner sum in (6.5), these authors use the so-called pseudopolar FFT based on the Chirp-Z transform as alternative for the NFFT. Recently, Candès et al. also apply the unequispaced FFT (USFFT) for the efficient computation of the curvelet transform [19].

3. Continuous Ridgelet Transform

The (two dimensional) continuous ridgelet transform can be defined as follows. Let the function $\psi \in L^2(\mathbb{R})$ be a wavelet, i.e., it fulfills the admissibility condition

$$\int_{\mathbb{R}} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} \, \mathrm{d}\omega < \infty \, .$$

A *ridgelet* $\psi_{\boldsymbol{\theta},a,b}$ with orientation parameter $\boldsymbol{\theta} := (\cos\varphi, \sin\varphi)^T$, $\varphi \in [0, 2\pi)$, scale parameter $a > 0$ and location parameter $b \in \mathbb{R}$ is defined by

$$\psi_{\boldsymbol{\theta},a,b}(\boldsymbol{x}) := a^{-1/2} \, \psi\left(\frac{\boldsymbol{x}\boldsymbol{\theta} - b}{a}\right) . \tag{6.6}$$

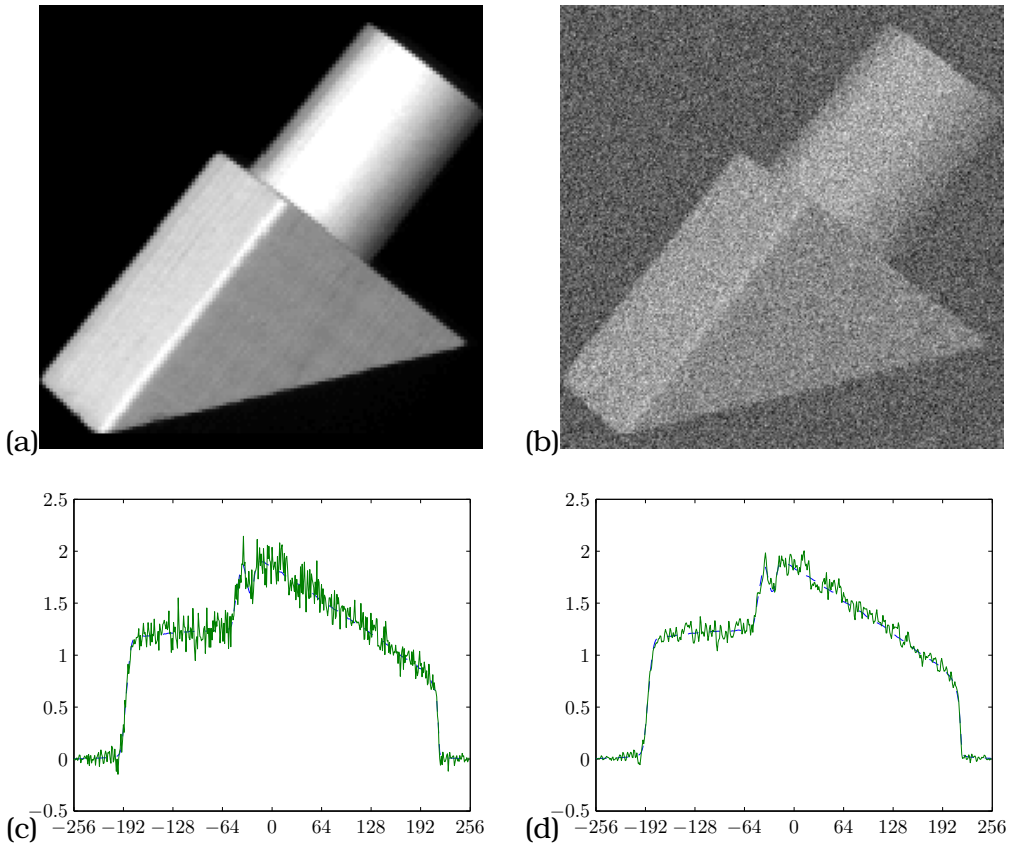


Figure 6.1.: (a) An object ($N = 256$) with (b) some Gaussian white noise added ($\mathrm{SNR} = 0.81$). Projections taken along one direction with the discrete Radon transform ($T = R = 2N$) of the object (dashed line) resp. noisy object (solid line) incorporating (c) the Dirichlet kernel resp. (d) the Fejér kernel.

This function is constant along lines $\boldsymbol{x}\boldsymbol{\theta} = c$. In the orthogonal direction it is a wavelet. Now, the *continuous ridgelet transform* for $f \in L^2(\mathbb{R}^2)$ is defined as

$$\mathfrak{R}_{\boldsymbol{\theta}} f(a,b) := \int_{\mathbb{R}^2} f(\boldsymbol{x}) \psi_{\boldsymbol{\theta},a,b}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \,. \tag{6.7}$$

By a change of the integration variable in (6.7) to $\boldsymbol{x} = s\boldsymbol{\theta} + t\boldsymbol{\theta}^{\perp}$, we can write

$$\begin{aligned} \mathfrak{R}_{\boldsymbol{\theta}} f(a,b) &= a^{-1/2} \int_{\mathbb{R}} \int_{\mathbb{R}} f\big(s\boldsymbol{\theta} + t\boldsymbol{\theta}^{\perp}\big) \psi\left(\frac{s-b}{a}\right) \mathrm{d}t \, \mathrm{d}s \\ &= a^{-1/2} \int_{\mathbb{R}} \mathcal{R}_{\boldsymbol{\theta}} f(s) \psi\left(\frac{s-b}{a}\right) \mathrm{d}s \,. \end{aligned} \tag{6.8}$$

Thus, ridgelet analysis can be seen as a form of wavelet analysis in the Radon domain. Furthermore, by using Plancherel's Theorem and the Fourier Slice Theorem, we get

$$\begin{aligned} \mathfrak{R}_{\boldsymbol{\theta}} f(a,b) &= a^{1/2} \int_{\mathbb{R}} \widehat{\mathcal{R}_{\boldsymbol{\theta}} f}(\sigma) \hat{\psi}(a\sigma) \, \mathrm{e}^{2\pi \mathrm{i} b\sigma} \, \mathrm{d}\sigma \\ &= a^{1/2} \int_{\mathbb{R}} \hat{f}(\sigma\theta) \hat{\psi}(a\sigma) \, \mathrm{e}^{2\pi \mathrm{i} b\sigma} \, \mathrm{d}\sigma \,, \end{aligned}$$

which compares excellently to equation (6.2).

For $f \in S(\mathbb{R}^2)$ there exists a reconstruction formula (see, e.g., [18])

$$f(\boldsymbol{x}) = \int_0^{2\pi} \int_{-\infty}^{\infty} \int_0^{\infty} \mathfrak{R}_{\boldsymbol{\theta}} f(a,b) \psi_{\boldsymbol{\theta},a,b}(\boldsymbol{x}) \, \frac{\mathrm{d}a}{a^3} \, \mathrm{d}b \, \frac{\mathrm{d}\varphi}{4\pi} \,.$$

4. Discrete Ridgelet Transform

Based on the connection of the continuous ridgelet transform and the Radon transform expressed in equation (6.8), we propose to compute the *discrete ridgelet transform* of $f_{\boldsymbol{k}}$ ($\boldsymbol{k} \in I_N^2$) by the following two steps:

1. Compute the discrete Radon transform $R_{\boldsymbol{\theta}_t} f(\frac{s}{R})$ ($t \in I_T$, $s \in I_R$) by (6.5).
2. Compute a 1D discrete wavelet transform of $R_{\boldsymbol{\theta}_t} f(\frac{s}{R})$ for every $\boldsymbol{\theta}_t$.

Let's have a closer look at step 2. Let the projection $R_{\boldsymbol{\theta}} f$ in direction $\boldsymbol{\theta}$ be represented by a wavelet decomposition of the form

$$R_{\boldsymbol{\theta}} f = \sum_{k \in \mathbb{Z}} c_{\boldsymbol{\theta},j_0,k} \phi_{j_0,k} + \sum_{j \le j_0} \sum_{k \in \mathbb{Z}} d_{\boldsymbol{\theta},j,k} \psi_{j,k} \tag{6.9}$$

where $\phi_{j_0,k}$ denotes the scaling function at the coarsest level j_0 associated to the orthogonal wavelet basis $\{\psi_{j,k} := \psi\big((x-b)/a\big) : a = 2^{-j}, b = k2^{-j}, k, j \in \mathbb{Z}\}$. Then, the coefficients $c_{\boldsymbol{\theta},j_0,k} = \langle R_{\boldsymbol{\theta}} f, \phi_{j_0,k} \rangle$ and $d_{\boldsymbol{\theta},j,k} = \langle R_{\boldsymbol{\theta}} f, \psi_{j,k} \rangle$ refer to the smooth components at level j_0 and the detail components at level j in direction $\boldsymbol{\theta}$, respectively.

Discrete wavelet transform (DWT) suffers from shift sensitivity, which is caused by aliasing due to the transform having maximal decimation at each level. Many wavelet techniques, including undecimated wavelet transform and complex wavelet transform, have been explored to solve this problem, but they either require highly redundant computational cost or absent perfect reconstruction and good filter characteristics. Based on the Z-transform theory of linear time invariant systems, Kingsbury [82, 83] proposed the *dual-tree complex wavelet transform* (DT CWT), which adds perfect reconstruction to approximate shift invariance of complex wavelets. In contrast to the $\mathcal{O}(n \log n)$ undecimated DWT, which is $\log n$ times redundant in 1D and $3 \log n$ times redundant in 2D, the DT CWT is only slightly redundant by a factor of 2 in 1D and 4 in 2D. Thus, the computational complexity of DT CWT remains $\mathcal{O}(n)$, the same as for the decimated DWT.

Furthermore, the complex ridgelet transform can provide better phase information; a useful property for many applications. In [95] Ma showed that the ridgelet transform can achieve approximate shift invariance by the use of the DT CWT. Unfortunately, textural artifacts are still unavoidable since the finite $\mathbb{Z}_p^2$ Radon transform is used as a building block in [95]. However, the DT CWT used in this work is well-motivated by this prior attempt.

The DT CWT is a special discrete complex wavelet transform, which is essentially a combination of two different real wavelet transforms by a sophisticated dual tree. More precisely, we have two wavelet decompositions of the form (6.9) with scaling functions $\phi^r_{j_0,k}$, $\phi^i_{j_0,k}$ and wavelets $\psi^r_{j,k}$, $\psi^i_{j,k}$, respectively. We set $\phi_{j_0,k} := \frac{1}{\sqrt{2}}(\phi^r_{j_0,k} - \mathrm{i}\phi^i_{j_0,k})$ and $\psi_{j,k} := \frac{1}{\sqrt{2}}(\psi^r_{j,k} - \mathrm{i}\psi^i_{j,k})$. Then the DT CWT can be seen as the real part of a decomposition of the form (6.9) with the outputs of the dual tree interpreted as the real and imaginary parts of the complex coefficients

$$
\begin{aligned}
c_{\boldsymbol{\theta}_t,j_0,k}(f) &:= \tfrac{1}{\sqrt{2}}\big\langle R_{\boldsymbol{\theta}_t} f, \phi^r_{j_0,k} \big\rangle + \mathrm{i}\tfrac{1}{\sqrt{2}}\big\langle R_{\boldsymbol{\theta}_t} f, \phi^i_{j_0,k} \big\rangle\,, \\
d_{\boldsymbol{\theta}_t,j,k}(f) &:= \tfrac{1}{\sqrt{2}}\big\langle R_{\boldsymbol{\theta}_t} f, \psi^r_{j,k} \big\rangle + \mathrm{i}\tfrac{1}{\sqrt{2}}\big\langle R_{\boldsymbol{\theta}_t} f, \psi^i_{j,k} \big\rangle\,.
\end{aligned}
$$

Therefore, we will call the proposed ridgelets *complex ridgelets*. Note that the filter/wavelet design for the DT CWT is rather sophisticated. In our experiments, we will use the nearly orthogonal (13,19)-tap filters at level 1 and the 18-tap Q-shift filters constructed in [83] for the higher levels.

In summary, we obtain the following algorithm for the fast computation of the discrete complex ridgelet transform:

Algorithm 6.1 (Discrete Complex Ridgelet Transform (DCRT))**.**
Input: discrete function $f := \big(f_{\boldsymbol{k}}\big)_{\boldsymbol{k} \in I_N^2}$.

1. *For $t \in I_T$ and $r \in I_R$ compute*

$$\hat{f}_{\boldsymbol{\theta}_t,r} := w_r \sum_{\boldsymbol{k} \in I_N^2} f_{\boldsymbol{k}}\, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \left(\frac{r}{R}\boldsymbol{\theta}_t\right)}$$

   by 2D NFFT at the points $\frac{r}{R}\boldsymbol{\theta}_t$ of the linogram grid.

2. *Obtain the discrete Radon transform for every direction $\boldsymbol{\theta}_t$ by computing*

$$R_{\boldsymbol{\theta}_t} f\left(\frac{s}{R}\right) = \sum_{r \in I_R} \hat{f}_{\boldsymbol{\theta}_t,r}\, \mathrm{e}^{2\pi \mathrm{i} rs/R}$$

   for every $s \in I_R$ by 1D iFFT of $\hat{f}_{\boldsymbol{\theta}_t,r}$.

3. *Compute the complex ridgelet coefficients*

$$c_{\boldsymbol{\theta}_t,j_0,k}(f) \quad \textit{and} \quad d_{\boldsymbol{\theta}_t,j,k}(f)$$

   by 1D DT CWT of $R_{\boldsymbol{\theta}_t} f$ for every direction $\boldsymbol{\theta}_t$.

Output: discrete complex ridgelet coefficients $c_{\boldsymbol{\theta}_t,j_0,k}(f)$, $d_{\boldsymbol{\theta}_t,j,k}(f)$.

Remark 4.1. The linogram grid together with unequispaced fast Fourier transform (USFFT) was recently also applied by Candès et al. for the efficient computation of the curvelet transform [19]. In the second generation of curvelets, the USFFT is used to evaluate the frequency angular partitioning supported on parallelepiped tilings (sheared grid) at a series of disjoint scales. In some sense, the use of the USFFT in [19] and of the NFFT in this work both analogously view a special grid as a nonequispaced grid in Cartesian coordinate system. However, Candès et al. don't involve the ridgelet transform. Also the USFFT, which is based on zero-padding and Taylor expansion, is itself very different from the NFFT used in this work.

The Inverse Complex Ridgelet Transform

The inverse discrete complex ridgelet transform can be achieved by inverting the steps of Algorithm 6.1 one after the other in reverse order.

Since the second step of Algorithm 6.1 can be easily inverted by 1D FFTs and the first step can be inverted (approximately to arbitrary accuracy) by 2D iNFFT as discussed in Chapter II, Section 5, we also have a fast algorithm for the inverse discrete Radon transform. The algorithm possesses low arithmetic costs and leads to a good reconstruction quality similar to that of filtered backprojection.

Algorithm 6.2 (Inverse Discrete Complex Ridgelet Transform (iDCRT))**.**
Input: discrete complex ridgelet coefficients $c_{\boldsymbol{\theta}_t,j_0,k}(f)$, $d_{\boldsymbol{\theta}_t,j,k}(f)$.

1. *Optain the Radon transform*

$$R_{\boldsymbol{\theta}_t} f\left(\frac{s}{R}\right)$$

   for $s \in I_R$ by computing the inverse 1D DT CWT of the ridgelet coefficients for every direction $\boldsymbol{\theta}_t$ ($t \in I_T$).

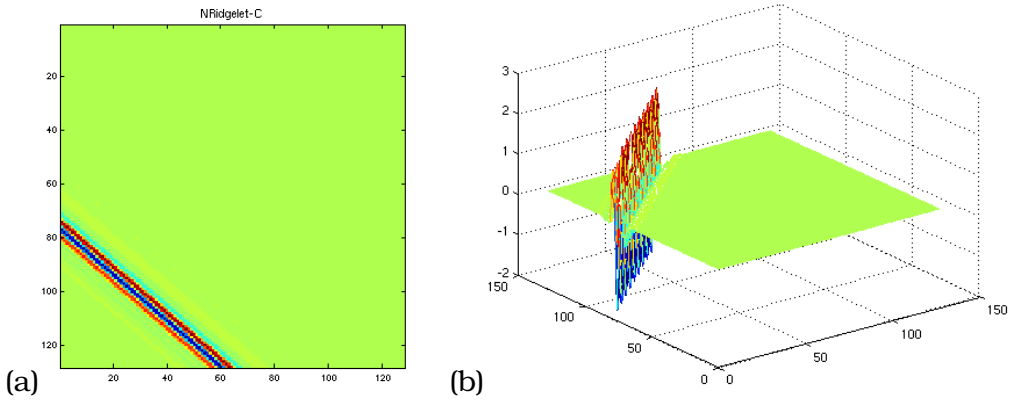


Figure 6.2.: The iDCRT of only one non-zero coefficient shown by 2D (a) and 3D surface (b), respectively.

2. *For* $r \in I_R$ *compute*

$$\hat{f}_{\boldsymbol{\theta}_t,r} = \frac{1}{R} \sum_{s \in I_R} R_{\boldsymbol{\theta}_t} f \left(\frac{s}{R} \right) \mathrm{e}^{-2\pi \mathrm{i} rs/R}$$

by 1*D FFT for every direction* θ_t*.*

3. *For* $\boldsymbol{k} \in I_N^2$ *compute* $f_{\boldsymbol{k}}$ *by solving the linear system*

$$\hat{f}_{\boldsymbol{\theta}_t,r} = w_r \sum_{\boldsymbol{k} \in I_N^2} f_{\boldsymbol{k}} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \left(\frac{r}{R} \boldsymbol{\theta}_t \right)}$$

with the CGNR-based 2*D iNFFT at the points* $\frac{r}{R}\boldsymbol{\theta}_t$ *of the linogram grid.*

Output: discrete function $f := \left(f_{\boldsymbol{k}} \right)_{\boldsymbol{k} \in I_N^2}$.

Figure 6.2 shows a discrete complex ridgelet, i.e., the iDCRT of a point (all ridgelet coefficients are zero but one). In Figure 6.2 (a) and Figure 6.2 (b) the obtained 2D resp. 3D surface is depicted. Unlike FRIT, it can be seen that the DCRT is free from wrap-around artifacts. The similar sample was given using recent DART in [22]. Our method is also better than DART in terms of the influence of the wrap-around artifacts in comparison to the displayed illustrations. However, we note that it is non-smooth along the ridge shown in Figure 6.2 (b).

5. Combination of Hard Thresholding with TV minimization

The combination of ordinary wavelet shrinkage with TV minimization was successfully applied in various papers [25, 39, 93, 96]. In this work we apply the technique with respect to our ridgelet coefficients. We mainly follow the approach in [39].

Many problems such as denoising and feature extraction can be written as the model

$$f = f_{\text{true}} + \varepsilon\,,$$

where f is the observed function, f_{true} is the function (or component) to estimate and ε is white Gaussian noise or, more generally, an unwanted component. Our aim is to restore f_{true} without artifacts, i.e., recover the smooth parts of f_{true} while preserving the discontinuities.

The *hard thresholding function* τ is defined for $x \in \mathbb{C}$ as

$$\tau(x) := \begin{cases} x, & |x| \geq \sigma, \\ 0, & |x| < \sigma. \end{cases}$$

We restrict our attention to hard thresholding although other thresholding functions as soft thresholding or garotte thresholding can be used, too. Applying hard thresholding with some fixed threshold σ to the complex ridgelets coefficients $d_{\boldsymbol{\theta}_t,j,k}(f)$ of our discrete function f and applying the iDCRT (Algorithm 6.2) results in a function $f^{(0)}$. The indices of the ridgelet coefficients retained after hard thresholding are recorded in Λ, i.e.,

$$\Lambda := \left\{(\boldsymbol{\theta}_t, j, k) : |d_{\boldsymbol{\theta}_t,j,k}(f)| \geq \sigma\right\}.$$

Unfortunately, wavelet thresholding produces artificial oscillations near discontinuities, a phenomenon known as 'pseudo-Gibbs phenomenon' or 'side-band effect'. To lower the pseudo-Gibbs artifacts we will use TV minimization. For a function $f : \mathbb{R}^2 \supseteq \Omega \to \mathbb{R}$ with $|\nabla f(x)| \in L^1(\mathbb{R}^2)$ the total variation functional is defined by

$$\mathrm{TV}(f) = \int_\Omega \left|\nabla f(x)\right| \mathrm{d}x\,. \tag{6.10}$$

TV functionals as regularizing terms became very popular in image processing. Meanwhile, there exists a broad literature on this topic. In particular, relations between TV regularization and wavelet shrinkage were examined in [123, 135].

To circumvent computational difficulties arising from the non-differentiability of the modulus at zero, the TV functional is often replaced by

$$\mathcal{J}_\beta(f) = \int_\Omega \sqrt{\left|\nabla f(x)\right|^2 + \beta^2}\,\mathrm{d}x\,, \tag{6.11}$$

with a small parameter β, see [131]. In the following, we restrict our attention to (6.11) and use the following discrete version for $f := \left(f_{\boldsymbol{k}}\right)_{\boldsymbol{k}\in I_N^2}$

$$J_\beta(f) = \sum_{\boldsymbol{k}} \sqrt{\left|(\delta_1 f)_{\boldsymbol{k}}\right|^2 + \left|(\delta_2 f)_{\boldsymbol{k}}\right|^2 + \beta^2}\,, \tag{6.12}$$

where $(\delta_1 f)_{\boldsymbol{k}} = f_{k_1+1,k_2} - f_{k_1,k_2}$, $(\delta_2 f)_{\boldsymbol{k}} = f_{k_1,k_2+1} - f_{k_1,k_2}$. More precisely, for our given f let

$$U := \Big\{ u := \big(u_{\boldsymbol{k}}\big)_{\boldsymbol{k}\in I_N^2} : c_{\boldsymbol{\theta}_t,j_0,k}(u) = c_{\boldsymbol{\theta}_t,j_0,k}(f) \quad \forall(\boldsymbol{\theta}_t,k), \\ d_{\boldsymbol{\theta}_t,j,k}(u) = d_{\boldsymbol{\theta}_t,j,k}(f) \quad \forall(\boldsymbol{\theta}_t,j,k)\in\Lambda \Big\}.$$

Then we are looking for the solution of the constrained minimization problem

$$\min_{u\in U} J_\beta(u)\,. \tag{6.13}$$

Let the linear subspace V of functions on I_N^2 be given by

$$V := \Big\{ v := \big(v_{\boldsymbol{k}}\big)_{\boldsymbol{k}\in I_N^2} : c_{\boldsymbol{\theta}_t,j_0,k}(v) = 0 \quad \forall(\boldsymbol{\theta}_t,k), \\ d_{\boldsymbol{\theta}_t,j,k}(v) = 0 \quad \forall(\boldsymbol{\theta}_t,j,k)\in\Lambda \Big\}.$$

Then it holds that $U = f^{(0)} + V$, i.e., U is an affine and hence convex space. Since the functional (6.12) is convex, too, problem (6.13) has a solution. Given a positive sequence $(t_\ell)_{\ell=0}^\infty$ with $\lim_{\ell\to\infty} t_\ell = 0$ and $\sum_{\ell=0}^\infty t_\ell = \infty$, it was shown in [39] that a solution of (6.13) can be computed by the following *projected gradient descent scheme*

$$f^{(\ell+1)} = f^{(\ell)} - t_\ell P_V\big(\nabla_f J_\beta(f^{(\ell)})\big)$$

with our hard-thresholded function $f^{(0)}$ as initial guess. Here, $P_V(g)$ denotes the projection of g onto V, i.e., the inverse discrete complex ridgelet transform of

$$c_{\boldsymbol{\theta}_t,j_0,k}(v) := 0, \qquad d_{\boldsymbol{\theta}_t,j,k}(v) := \begin{cases} 0 & \text{for } (\boldsymbol{\theta}_t,j,k)\in\Lambda, \\ d_{\boldsymbol{\theta}_t,j,k}(g) & \text{else.} \end{cases}$$

It is applied to the gradient ∇_f of the discrete modified TV functional (6.12) given by

$$\begin{aligned} \nabla_f J_\beta(f) :=\ & (2f_{k_1,k_2} - f_{k_1,k_2+1} - f_{k_1+1,k_2})\big[(f_{k_1,k_2+1} - f_{k_1,k_2})^2 + (f_{k_1+1,k_2} - f_{k_1,k_2})^2 + \beta^2\big]^{-1/2} \\ & + (f_{k_1,k_2} - f_{k_1-1,k_2})\big[(f_{k_1-1,k_2+1} - f_{k_1-1,k_2})^2 + (f_{k_1-1,k_2} - f_{k_1,k_2})^2 + \beta^2\big]^{-1/2} \\ & + (f_{k_1,k_2} - f_{k_1,k_2-1})\big[(f_{k_1+1,k_2-1} - f_{k_1,k_2-1})^2 + (f_{k_1,k_2} - f_{k_1,k_2-1})^2 + \beta^2\big]^{-1/2} \end{aligned} \tag{6.14}$$

for the inner points $(k_1,k_2)\in I_N^2$ and corresponding modifications at the boundary ∂I_N^2.

Algorithm 6.3 (TV Minimization of Complex Ridgelet Coefficients (DCRT with TV))**.** *Input: discrete (noisy) data f, threshold σ, time step size t_ℓ*

1. *Compute the discrete complex ridgelet coefficients $c_{\boldsymbol{\theta}_t,j_0,k}(f)$, $d_{\boldsymbol{\theta}_t,j,k}(v)$ by DCRT (Algorithm 6.1).*

2. *Apply hard thresholding and record indices of retained coefficients in* Λ.

3. *Compute initial guess* $f^{(0)}$ *by iDCRT (Algorithm 6.2) of the retained ridgelet coefficients.*

4. *Minimize the TV norm of* $f^{(\ell)}$ *by doing the following steps for* $\ell = 0, 1, \ldots$:

   a) *Compute the subgradient* $g^{(\ell)}$ *of* $f^{(\ell)}$ *by* (6.14).

   b) *Compute the discrete complex ridgelet coefficients* $c_{\boldsymbol{\theta}_t,j_0,k}(g^{(\ell)})$, $d_{\boldsymbol{\theta}_t,j,k}(g^{(\ell)})$ *by DCRT (Algorithm 6.1).*

   c) *Compute* $P_V(g^{(\ell)})$ *by applying the iDCRT (Algorithm 6.2) to the coefficients*

$$c_{\boldsymbol{\theta}_t,j_0,k}(v) := 0, \qquad d_{\boldsymbol{\theta}_t,j,k}(v) := \begin{cases} 0 & \text{for } (\boldsymbol{\theta}_t, j, k) \in \Lambda, \\ d_{\boldsymbol{\theta}_t,j,k}(g^{(\ell)}) & \text{else.} \end{cases}$$

   d) *Set*

$$f^{(\ell+1)} := f^{(\ell)} + t_\ell P_V(g^{(\ell)}) \,.$$

Output: denoised discrete data $f^{(\ell)}$

6. Numerical results

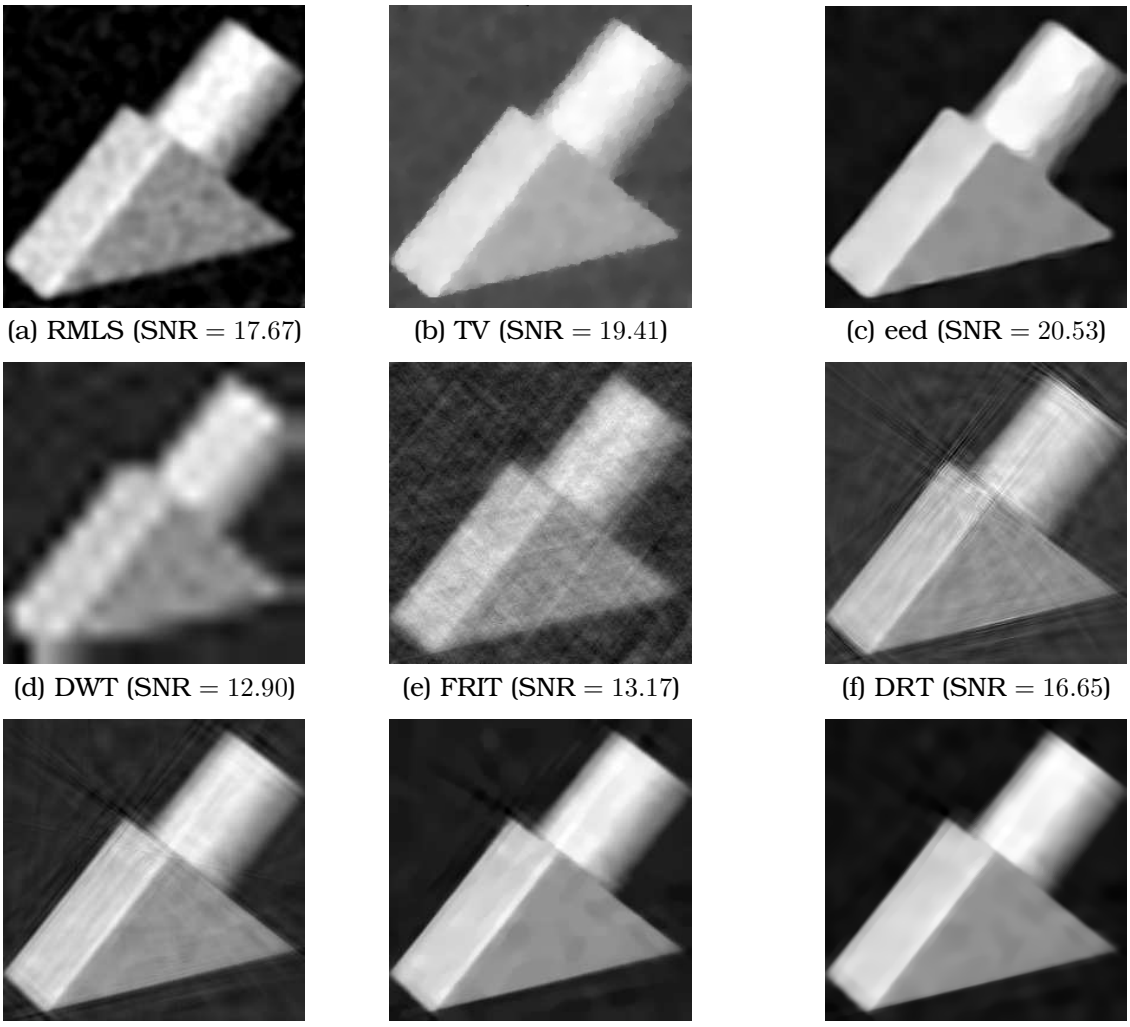
In our numerical experiments, we always use the NFFT [88] with the Gaussian window function, cut-off parameter $m = 4$ and oversampling factor $\sigma = 2$. The DT CWT is due to Kingsbury [84] and applied with the Near-symmetry (13,19)-tap biorthogonal filters at level 1 and 18-tap Q-shift filters in a cascaded way at beyond level 1. We will denote with DRT the discrete *real* ridgelet transform, in which we use DWT (Sym4 wavelets [36]) in the second step instead of the DT CWT; and FRIT stands for Do and Vetterli's finite $\mathbb{Z}_p^2$ ridgelet transform [35].

As a measure of quality we use the *signal-to-noise ratio* (SNR) defined by

$$\mathrm{SNR} = 20 \log_{10} \frac{\|\boldsymbol{z} - \bar{\boldsymbol{z}}\|_2}{\|\boldsymbol{n}\|_2}$$

with $\boldsymbol{z}$ standing for the original signal with mean $\bar{\boldsymbol{z}}$, and $\boldsymbol{n}$ representing noise.

In our first test, we compare different methods for edge-preserving denoising of an image with line singularities. It will show the good performance of the proposed method. We use the noisy object of Figure 6.1 (b). First, Figure 6.3 (a) shows the result obtained using the robust local approximation method (RMLS) of Chapter VII. There is still some textural noise left. By varying the parameters, we can remove this noise at the cost of clear edges. Figure 6.3 (b) shows the result obtained using our own Matlab implementation of the 2D TV minimization algorithm by Chambolle [24]. The edges are frayed and the staircasing effect can be seen. Figure 6.3 (c) shows the result obtained using the edge-enhancing anisotrop diffusion filtering algorithm (eed) of Weickert [133]. A universal hard thresholding was applied to the wavelet/ridgelet coefficients of the next images. Figure 6.3 (d) shows



(a) RMLS (SNR = 17.67) (b) TV (SNR = 19.41) (c) eed (SNR = 20.53)

(d) DWT (SNR = 12.90) (e) FRIT (SNR = 13.17) (f) DRT (SNR = 16.65)

(g) DCRT (SNR = 19.16) (h) DCRT with TV (SNR = 20.40) (i) TV after DCRT (SNR = 20.57)

Figure 6.3.: Denoising using different methods.

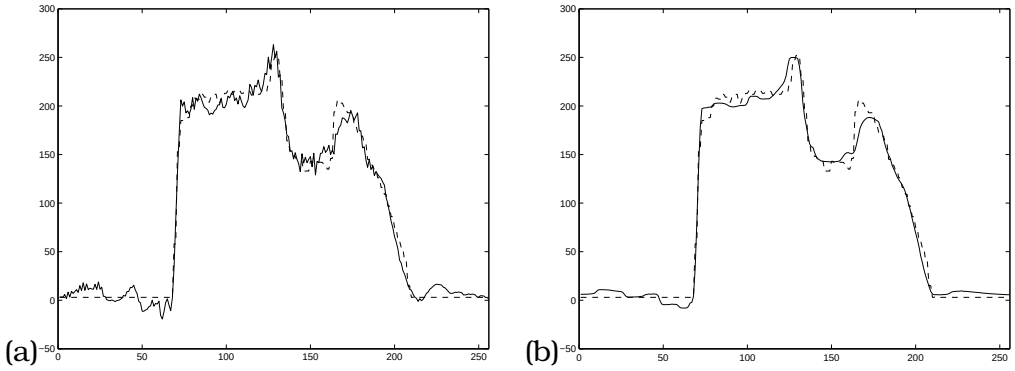
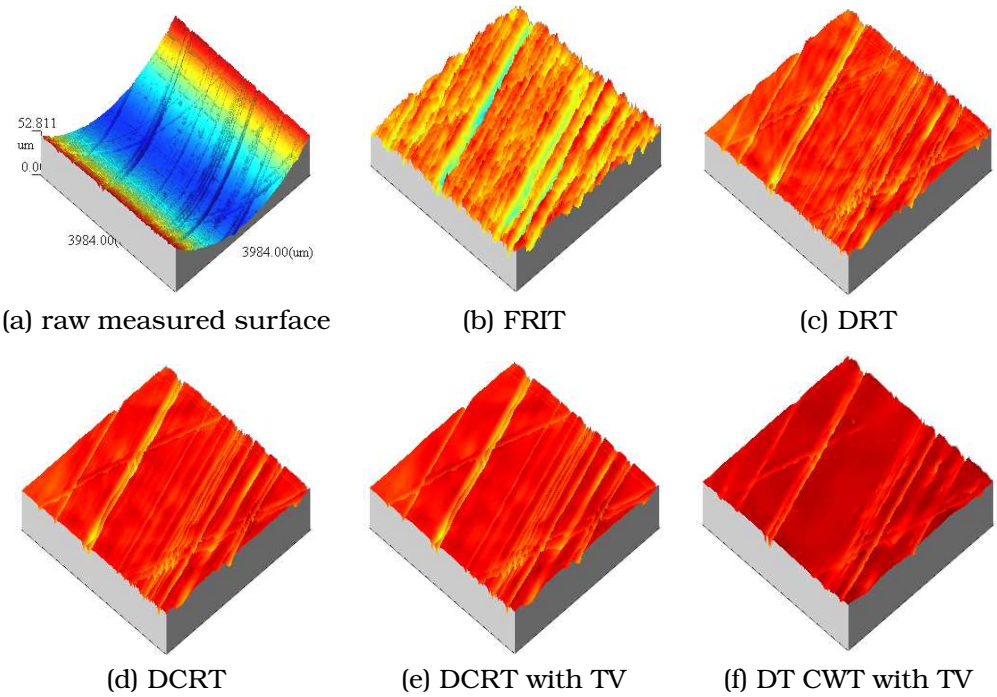


Figure 6.4.: The center profile taken across the original object (shown as dashed line) and the reconstruction by (a) DCRT (shown in Figure 6.3 (g)) and (b) DCRT with TV (shown in Figure 6.3 (h)), respectively.

the result obtained using (9,7)-tap DWT. As we know, it causes non-smoothness along the edges. Figure 6.3 (e) shows the result obtained using FRIT. It suffers from strongly textural artifacts. Figure 6.3 (f) shows the result obtained using the DRT. Figure 6.3 (g) shows the result obtained using the DCRT. The DCRT with TV result in Figure 6.3 (h) shows a better result with less line-like artifacts, as well as in terms of the signal to noise ratio (SNR). It can be seen from Figure 6.3 (g) and (h) that the proposed framework is effective in recovering straight edges, even for so heavy noisy background. Generally speaking, a more oscillating wavelet basis used in the second step produces less artifacts, as mentioned in [34], but costs more computational burden because of its larger support. A similar denoising sample using the Fast Slant Stack based ridgelet transform and DART with a very oscillating wavelet basis (Daubechies20) is depicted in [22, Figure 3]. There seem to be less disturbing artifacts than in the DRT result, but the edges are not very clear. Figure 6.3 (h) in this chapter shows the result obtained using the DCRT with TV where we chose $t_\ell = 0.1$ ($\ell = 1, \ldots, 120$). It is much more effective in reducing undesirable artifacts including the pseudo-Gibbs artifacts while preserving the edges. However, it is somewhat disillusioning, that the result in Figure 6.3 (i), which was obtained from Chambolle's TV algorithm applied to the DCRT result, is quite as good as the presented method even though it is less sophisticated.

Figure 6.4 shows a comparison of the center profile taken across the original object and Figure 6.3 (g) and (h), respectively. The effectivity of DCRT with TV can be seen more clearly from these profiles.

The simple thresholding scheme is effective for the ridgelet methods in denoising the piecewise smooth image with line singularities, because the line singularities are represented by a few large coefficients while noise unlikely generates significant coefficients. From the view of the hybrid approach, the DCRT just combines the multiresolution analysis of DT CWT with the anisotropy of the Radon transform. DCRT with TV offers an better choice of reconstructed coefficients in the



(a) raw measured surface (b) FRIT (c) DRT

(d) DCRT (e) DCRT with TV (f) DT CWT with TV

Figure 6.5.: Extraction of scratches from a honed surface of automotive engine cylinder.

procedure of thresholding.

In our second test, we apply the proposed method to extract straight scratches in the bands of roughness and waviness, in order to study the functional performance of the 3D surface topography of systems according to different applications. The extracted information could be fed back to monitor and manufacturing processes, or to study actual contact stress, loaded area, asperity volume and lubrication regimes occurring during the initial stages of wear of surfaces in service.

Figure 6.5 shows a typical engineering surface: a honed surface from an automotive engine cylinder. This kind of surface includes the form, waviness and roughness components that almost submerge the main features of the deep valleys/scratches. Usually, the most important features that effect the performance of the cylinder are scratches whose distribution and amplitude will considerably influence the flow of gas or air in a pressure balance of an engine.

Figure 6.5 (a) is the raw measured surface including the form, waviness and roughness components. Figure 6.5 (b) is the extracted surface using FRIT followed by removing form error. It is far way from satisfaction because of the strongly texture-like artifacts. Figure 6.5 (c) is the extracted surface using the DRT. As we supposed, the DWT suffers from shift aliasing, which leads to the line-like artifacts in the extracted surface by DRT. Figure 6.5 (d) is the extracted surface using DCRT. The edge-preserving scratches are nicely restored. But the pseudo-Gibbs artifacts resulting from the aberrant ridgelet coefficients are still visible.

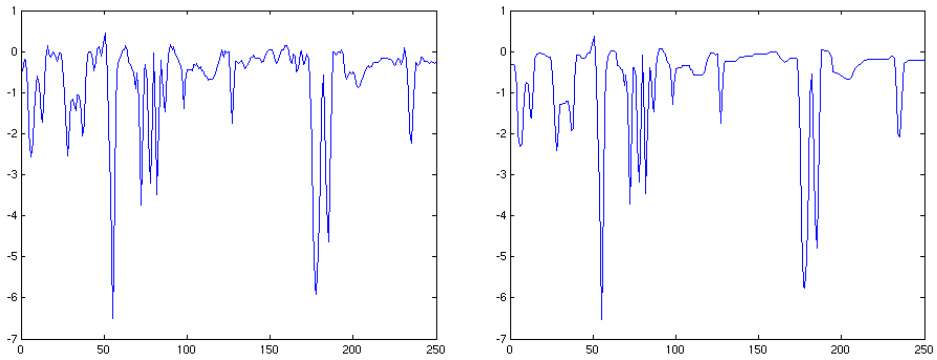


Figure 6.6.: The center profiles taken across (a) the DCRT extracted surface (shown in Figure 6.5 (d)) and (b) the DCRT with TV extracted surface (shown in Figure 6.5 (e)), respectively.

Figure 6.5 (e) is the extracted surface using the DCRT with TV where $t_\ell = 0.001$ ($\ell = 1, \ldots, 100$). The non-smooth artifacts are reduced remarkably. This is due to the fact that DCRT with TV allows the reconstruction of some small ridgelet coefficients, which are canceled by thresholding. Furthermore, the elimination of pseudo-Gibbs artifacts can be seen more clearly from their center profiles shown in Figure 6.6, in which Figure 6.6 (a) and (b) are taken across the extracted surfaces using DCRT and DCRT with TV, as shown in Figure 6.5 (d) and (e), respectively. The experiments show that the method is more efficient in removing the pseudo-Gibbs oscillations when DT CWT and TV are joined. As to the method when one uses TV minimization only, i.e., combining TV with DRT instead of DCRT, one needs much more iterations than when using DCRT with TV, in order to eliminate the strong line-like artifacts caused by shift variance of DWT. In such a case, the intrinsic scratches will be smoothed over to some extent.

Figure 6.5 (f) shows the extracted surface using the TV-based DT CWT method with 500 iterations proposed in [93]. The method has a good performance but in terms of the extraction of scratches, setting a high threshold to kick out the point-like features causes the intrinsic features to be destroyed and some shallow scratches to be missed. This is because wavelets lack of line sensitivity.

In our last test, we examine the decay of the ridgelet coefficients. Our proposed method is an overcomplete system. Although it shows good performance for denoising and feature extraction, the main disappointment is the relatively slow decay of the coefficients in ridgelet domain. Figure 6.7 gives two examples to show the decreasing rearrangement of the ridgelet transform coefficients, in comparison to the orthogonal FRIT. The upper line shows the rearrangement of the Object used in the second test. The Object can not be represented as a summation of a few global linear singularities, thus it is not in the optimal class of the ridgelet transform [34]. The lower line shows those of a typically optimal class: HalfDome, i.e., the mutilated Gaussian $g(x_1, x_2) = 1_{\{x_2>0\}}\, \mathrm{e}^{-x_1^2-x_2^2}$. The delay of coefficients of

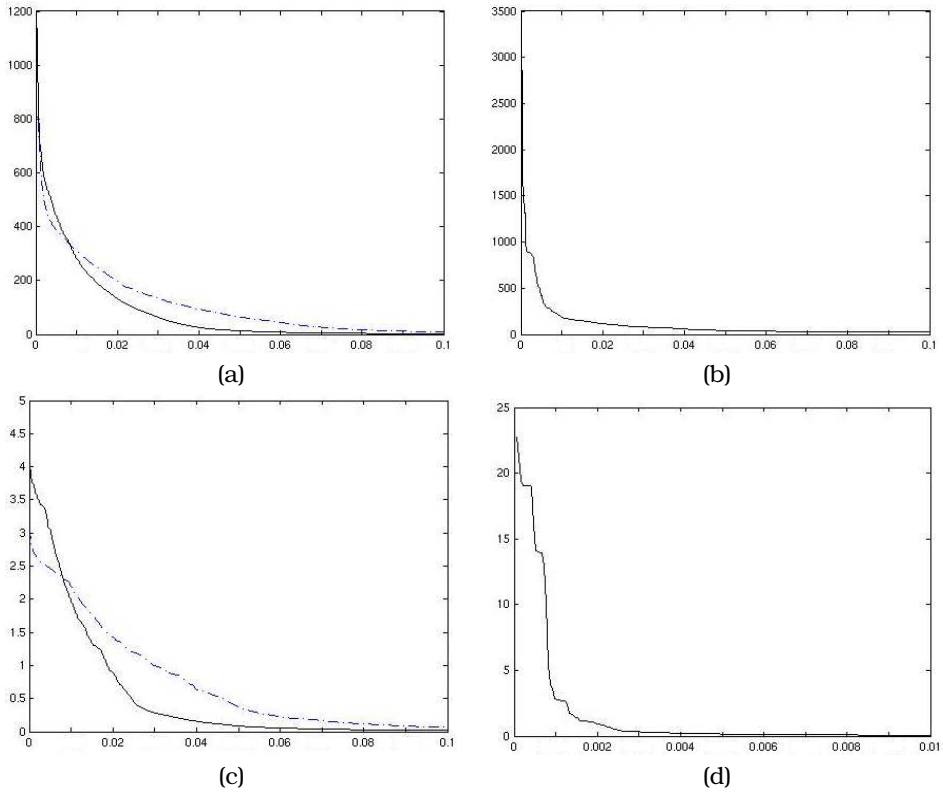


Figure 6.7.: Decreasing rearrangement of ridgelet coefficients of Object (upper) and HalfDome (lower). (a) and (c): real line denotes DRT coefficients and dot-dashed line denotes DCRT coefficients; (b) and (d): FRIT coefficients. Vertical coordinate denotes the absolute value of coefficients, Horizontal coordinate denotes the ratio of rearranged coefficients to all coefficients. Note the difference of the horizontal coordinate in (d).

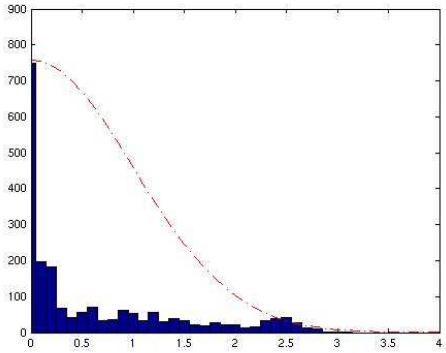


Figure 6.8.: Histogram of the coarsest subband $d_{\theta,j_0,k}$ of HalfDome. The dot-dashed line denotes the sketch of normal distribution.

the proposed ridgelet transform is much slower than that of FRIT, especially for the optimal class.

However, leaving out the captious criticism, the proposed ridgelet transform is very effective to sparsely represent line singularities. Figure 6.8 shows the histogram of the coarsest subband $d_{\theta,j_0,k}$ of HalfDome. The distribution is characterized by a very sharp peak (at zero) with an extended tail. The highly non-Gaussian distribution implies that the transform is very spare. Other experiments of real surfaces with line scratches display similar distributions. It is rare to meet natural images with global linear singularities, but for engineering surfaces, many kinds of those belong to the optimal class of ridgelet transform. So, application of ridgelets to this field is of practical interest.

We remark, that using the Dirichlet kernel instead of the Fejér kernel in (6.5) yields to slightly worse results.

VII. Robust local approximation of scattered data

A popular approach to scattered data approximation is the *moving least squares method* (MLS) which requires in contrast to standard interpolation methods by radial basis functions only the solution of small linear systems of equations. The size of these systems is governed by the degree of the polynomials which are reproduced by the method. The MLS approximation is theoretically well examined, see, e.g., [46] and the references therein. In particular, the Backus-Gilbert approach offers another way to look at the polynomial reproduction property which in turn determines the approximation order of the method.

On the other hand, there exist various local linear methods for smoothing noisy data in image processing. One example is the Gaussian facet model introduced by van den Boomgaard and van de Weijer [129] in the linear scale-space context. Interestingly, this method is basically the same as the MLS technique with a Gaussian weight function. The only difference consists in the fact that in scattered data approximation we know the (noisy) function only at some special, in general nonequispaced nodes and no data are given within these nodes, while in denoising problems in image processing the noisy function is known on the whole grid. This leads to an ansatz with shifted basis functions in the MLS approach in contrast to the Gaussian facet model.

In their averaging process, the MLS method and its variants give similar weights to data within a similar distance from the evaluation point, where neighbors are heavier weighted even if these neighbors are on very different levels of the function. Consequently, edges are smoothed. This led to the development of robust estimation procedures and nonlinear filters that also data-adaptively determine the influence of each data point on the result. To this end we are looking for tonal (data) and spatial adaptive methods. Among the rich variety of these methods, see, e.g., [118] and the references therein, we focus on the robust Gaussian facet model [129]. Having the relation between the linear approaches in image processing and scattered data approximation in mind, we modify this robust model in such a way that it can be also applied to scattered data. Moreover, we change the method slightly towards a generalized bilateral filter approach that does not only reproduce constants but also polynomials of higher degree.

This is our first attempt to incorporate robust estimators in scattered data approximation. A couple of theoretical questions is still open. In particular, the convergence behavior of the algorithm and its dependence on the distribution of the scattered nodes as well as stability properties were not examined up to now. Moreover, it should be possible to further speed up the performance of the algorithm

by using the NFFT, which was recently also applied by Fasshauer and Zhang [47] for scattered data approximation with MLS. The main ideas of this chapter were previously published in [53].

This chapter is organized as follows. First, we consider the linear methods used independently in image processing and scattered data approximation, where we start with the continuous MLS method in Subsection 1.1 and move to the discrete method in Subsection 1.2. In Subsection 2.1, we use these results for introducing our robust scattered data approximation method. Its power is demonstrated by numerical examples in Subsection 2.2.

1. MLS from different points of views

The aim of this section is twofold. Firstly, we want to show the relation between the well examined MLS method in approximation theory and the Gaussian facet model recently introduced in the context of linear scale-space theory by van den Boomgaard and van de Weijer [129]. It is not hard to see that both methods differ only by an ansatz with shifted basis functions such that applied to spaces of polynomials they lead to the same result. However, we find it useful to direct the attention of people from the image processing society to theoretical results from approximation theory and vice versa, to benefit from new ideas in image processing for the approximation of scattered data.

Secondly, the MLS results of this section will serve as the basis for our robust approach in Section 2. In particular, we will use the MLS approximation as initial input for our iterative algorithm.

1.1. Continuous MLS

Let

$$V := \operatorname{span}\{\varphi_j : j = 1, \dots, M\}$$

be an M-dimensional space of real-valued functions defined on $\mathbb{R}^d$. Although some results can be formulated in this general setting, we will restrict ourselves to polynomial spaces. More precisely, let $V := \Pi_s^d$ be the space of d-variate polynomials of absolute degree $\leq s$. Then V has dimension $M = \binom{s+d}{s}$. Our main reason for the restriction to polynomial spaces is that Π_s^d can be also spanned by the translates of φ_j with respect to an arbitrary fixed $\boldsymbol{x} \in \mathbb{R}^d$, i.e.,

$$V = \operatorname{span}\{\varphi_j(\cdot - \boldsymbol{x}) : j = 1, \dots, M\}\,. \tag{7.1}$$

Let w be a non-negative weight function with moments

$$\int_{\mathbb{R}^d} w(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} = 1 \qquad \text{and} \qquad \int_{\mathbb{R}^d} \boldsymbol{t}^{\boldsymbol{\alpha}} w(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} < \infty \quad \text{for all } \boldsymbol{\alpha} \in \mathbb{N}_0^d, |\boldsymbol{\alpha}| \leq 2s\,.$$

Then

$$\langle p, q \rangle_w := \int_{\mathbb{R}^d} p(\boldsymbol{t}) q(\boldsymbol{t}) w(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t}$$

is an inner product on V with norm $\|p\|_w^2 = \int_{\mathbb{R}^d} p^2(\boldsymbol{t}) w(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t}$.

Now the *continuous MLS problem* can be formulated as follows, see, e.g., [11]: for a given function $f \in L^\infty(\mathbb{R}^d)$ and $\boldsymbol{x} \in \mathbb{R}^d$ find the coefficients $c_j = c_j(\boldsymbol{x})$ such that

$$u(\boldsymbol{x},\boldsymbol{t}) := \sum_{j=1}^{M} c_j(\boldsymbol{x})\varphi_j(\boldsymbol{t}) \tag{7.2}$$

minimizes the functional

$$\mathcal{J}(\boldsymbol{x}) := \int\limits_{\mathbb{R}^d} \big(f(\boldsymbol{t}) - u(\boldsymbol{x},\boldsymbol{t})\big)^2 w(\boldsymbol{t}-\boldsymbol{x})\,\mathrm{d}\boldsymbol{t}\,. \tag{7.3}$$

Then

$$u(\boldsymbol{x}) = u(\boldsymbol{x},\boldsymbol{x}) = \sum_{j=1}^{M} c_j(\boldsymbol{x})\varphi_j(\boldsymbol{x}) \tag{7.4}$$

can be taken as an approximation of $f(\boldsymbol{x})$. Obviously, for arbitrary fixed $\boldsymbol{x} \in \mathbb{R}^d$, the function $u(\boldsymbol{x},\cdot)$ is the $w(\cdot-\boldsymbol{x})$-orthogonal projection of f onto V.

On the other hand, we obtain by (7.1) that the polynomial $\tilde{u}(\boldsymbol{x},\cdot)$ of the form

$$\tilde{u}(\boldsymbol{x},\boldsymbol{t}) := \sum_{j=1}^{M} a_j(\boldsymbol{x})\varphi_j(\boldsymbol{t}-\boldsymbol{x}) \tag{7.5}$$

which minimizes (7.3), i.e.,

$$\int\limits_{\mathbb{R}^d} \big(f(\boldsymbol{t}) - \tilde{u}(\boldsymbol{x},\boldsymbol{t})\big)^2 w(\boldsymbol{t}-\boldsymbol{x})\,\mathrm{d}\boldsymbol{t} = \int\limits_{\mathbb{R}^d} \Big(f(\boldsymbol{x}+\boldsymbol{t}) - \sum_{j=1}^{M} a_j(\boldsymbol{x})\varphi_j(\boldsymbol{t})\Big)^2 w(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} \tag{7.6}$$

is also the $w(\cdot-\boldsymbol{x})$-orthogonal projection of f onto V. Consequently, $u(\boldsymbol{x},\boldsymbol{t}) = \tilde{u}(\boldsymbol{x},\boldsymbol{t})$ and

$$u(\boldsymbol{x}) = \tilde{u}(\boldsymbol{x},\boldsymbol{x}) = \sum_{j=1}^{M} a_j(\boldsymbol{x})\varphi_j(\boldsymbol{0})\,. \tag{7.7}$$

The approximation (7.7) of f, where the coefficients $a_j = a_j(\boldsymbol{x})$ are determined by the minimization of (7.6) is exactly the approximation method that van den Boomgaard and van de Weijer have considered [129]. In particular, they have used monomials φ_j, where $\varphi_1 \equiv 1$, as basis functions in (7.7), so that they have only to compute $u(\boldsymbol{x}) = a_1(\boldsymbol{x})$. This simplification of MLS by using shifted monomials was also mentioned by Fasshauer in [45] and examined in detail by Belytschko et al. in [92].

The minimization problem (7.6) can be solved for any fixed $\boldsymbol{x} \in \mathbb{R}^d$ by setting the gradient with respect to $\boldsymbol{a}(\boldsymbol{x}) := \big(a_j(\boldsymbol{x})\big)_{j=1}^M$ to zero. Using the vector notation $\boldsymbol{\varphi}(\boldsymbol{t}) := \big(\varphi_k(\boldsymbol{t})\big)_{k=1}^M$, this leads to

$$\boldsymbol{a}(\boldsymbol{x}) = \boldsymbol{G}^{-1}\Big(\big\langle f(\boldsymbol{x}+\cdot), \varphi_k\big\rangle_w\Big)_{k=1}^M = \Big(\big\langle f(\boldsymbol{x}+\cdot), \big(\boldsymbol{G}^{-1}\boldsymbol{\varphi}(\cdot)\big)_j\big\rangle_w\Big)_{j=1}^M, \tag{7.8}$$

where $\left(\boldsymbol{G}^{-1}\boldsymbol{\varphi}(\cdot)\right)_j$ denotes the jth component of the vector and where the Gramian $\boldsymbol{G}$ is given by

$$\boldsymbol{G} := \left(\langle\varphi_j, \varphi_k\rangle_w\right)_{j,k=1}^{M}.$$

In summary, we obtain by (7.7) and (7.8) that

$$\begin{aligned} u(\boldsymbol{x}) &= \Big\langle f(\boldsymbol{x}+\cdot), \sum_{j=1}^{M} \left(\boldsymbol{G}^{-1}\boldsymbol{\varphi}(\cdot)\right)_j \varphi_j(\boldsymbol{0}) \Big\rangle_w \\ &= \int_{\mathbb{R}^d} f(\boldsymbol{x}+\boldsymbol{t}) q(\boldsymbol{t}) w(\boldsymbol{t}) \,\mathrm{d}\boldsymbol{t} \;=\; \int_{\mathbb{R}^d} f(\boldsymbol{x}+\boldsymbol{t}) \psi(\boldsymbol{t}) \,\mathrm{d}\boldsymbol{t}\,, \end{aligned} \tag{7.9}$$

where

$$q(\boldsymbol{t}) := \sum_{j=1}^{M} \left(\boldsymbol{G}^{-1}\boldsymbol{\varphi}(\boldsymbol{t})\right)_j \varphi_j(\boldsymbol{0}) \qquad \text{and} \qquad \psi(\boldsymbol{t}) := q(\boldsymbol{t}) w(\boldsymbol{t})\,. \tag{7.10}$$

In other words, u is the correlation of f with the function ψ.

Van den Boomgaard and van de Weijer have used the monomials of absolute degree $\leq s$ as basis of Π_s^d. We can orthogonalize this basis with respect to $\langle\cdot,\cdot\rangle_w$ so that the new basis fulfills $\langle\varphi_j, \varphi_k\rangle_w = \|\varphi_j\|_w^2 \delta_{jk}$ ($j,\ k = 1,\ \ldots,\ M$). Then $\boldsymbol{G} = \operatorname{diag}\left(\|\varphi_j\|_w^2\right)_{j=1}^{M}$ is a diagonal matrix and the polynomial q in (7.10) can be represented alternatively as

$$q(\boldsymbol{t}) = \sum_{j=1}^{M} \frac{\varphi_j(\boldsymbol{0})}{\|\varphi_j\|_w^2}\, \varphi_j(\boldsymbol{t})\,. \tag{7.11}$$

The function ψ has various properties.

Proposition 1.1. *The function ψ in (7.10) fulfills the moment condition*

$$\int_{\mathbb{R}^d} \boldsymbol{t}^{\boldsymbol{\alpha}} \psi(\boldsymbol{t}) \,\mathrm{d}\boldsymbol{t} = \delta_{\boldsymbol{0\alpha}} \qquad (|\boldsymbol{\alpha}| \leq s) \tag{7.12}$$

and has for all $p \in \Pi_s^d$ the reproducing property

$$\int_{\mathbb{R}^d} p(\boldsymbol{t}+\boldsymbol{x}) \psi(\boldsymbol{t}) \,\mathrm{d}\boldsymbol{t} = p(\boldsymbol{x})\,. \tag{7.13}$$

Proof. Let $\{\varphi_j : j = 1, \ldots, M\}$ be w-orthogonal. Then it is easy to check that the Christoffel-Darboux kernel

$$K(\boldsymbol{t}, \boldsymbol{x}) = \sum_{j=1}^{M} \frac{1}{\|\varphi_j\|_w^2} \varphi_j(\boldsymbol{x}) \varphi_j(\boldsymbol{t})$$

is a reproducing kernel in Π_s^d with respect to $\langle\cdot,\cdot\rangle_w$, i.e.,

$$\int_{\mathbb{R}^d} p(\boldsymbol{t}) K(\boldsymbol{t}, \boldsymbol{x}) w(\boldsymbol{t}) \,\mathrm{d}\boldsymbol{t} = p(\boldsymbol{x}) \qquad \text{for all } p \in \Pi_s^d\,.$$

In particular, we obtain for the monomials $p(\boldsymbol{t}) = \boldsymbol{t}^{\boldsymbol{\alpha}}$ with $|\boldsymbol{\alpha}| \leq s$ and $\boldsymbol{x} = \boldsymbol{0}$ by (7.11) that

$$\int_{\mathbb{R}^d} \boldsymbol{t}^{\boldsymbol{\alpha}} K(\boldsymbol{t}, \boldsymbol{0}) w(\boldsymbol{t}) \, \mathrm{d}\boldsymbol{t} = \int_{\mathbb{R}^d} \boldsymbol{t}^{\boldsymbol{\alpha}} \psi(\boldsymbol{t}) \, \mathrm{d}\boldsymbol{t} = \delta_{\boldsymbol{0\alpha}} \, .$$

By the binomial formula this implies for any fixed $\boldsymbol{x} \in \mathbb{R}^d$ that

$$\int_{\mathbb{R}^d} (\boldsymbol{t} + \boldsymbol{x})^{\boldsymbol{\alpha}} \psi(\boldsymbol{t}) \, \mathrm{d}\boldsymbol{t} = \boldsymbol{x}^{\boldsymbol{\alpha}} \, .$$

Consequently, (7.13) holds true. □

In the following, we are mainly interested in radial weights w.

Proposition 1.2. *Let $w(\boldsymbol{t}) = \omega\big(\|\boldsymbol{t}\|\big)$ be a radial weight function. Then the function ψ in* (7.10) *is also radial.*

Proof. On the one hand, the polynomial $p(y) := \sum_{k=0}^{s'} \gamma_k \, y^{2k}$ with $s' := \lfloor s/2 \rfloor$ which satisfies

$$\int_{\mathbb{R}^d} \|\boldsymbol{t}\|^{2j} p\big(\|\boldsymbol{t}\|\big) \, \omega\big(\|\boldsymbol{t}\|\big) \, \mathrm{d}\boldsymbol{t} = \delta_{0j} \qquad (j = 0, \, \ldots, \, s')$$

is uniquely determined and $p\big(\|\boldsymbol{t}\|\big) \in \Pi_d^s$. Since on the other hand the polynomial $q \in \Pi_d^s$ in (7.10) is also uniquely determined by the moment condition (7.12), it suffices to show that $p\big(\|\cdot\|\big)$ actually fulfills

$$\int_{\mathbb{R}^d} \boldsymbol{t}^{\boldsymbol{\alpha}} \, p\big(\|\boldsymbol{t}\|\big) \, \omega\big(\|\boldsymbol{t}\|\big) \, \mathrm{d}\boldsymbol{t} = \delta_{\boldsymbol{0\alpha}} \, . \qquad (|\boldsymbol{\alpha}| \leq s) \tag{7.14}$$

Switching to polar coordinates, the left side of (7.14) reads as

$$\int_0^\infty r^{|\alpha|+d-1} \, p(r) \, \omega(r) \, \mathrm{d}r \int_{S^{d-1}} \boldsymbol{t}^{\boldsymbol{\alpha}} \, \mathrm{d}S \, ,$$

where $\mathrm{d}S$ is the element of the $(d-1)$-dimensional measure on the unit sphere S^{d-1} in $\mathbb{R}^d$. If $\boldsymbol{\alpha}$ contains any odd component, then it is easy to check by the orthogonality of $\sin$ and $\cos$ functions, that $\int_{S^{d-1}} \boldsymbol{t}^{\boldsymbol{\alpha}} \, \mathrm{d}S = 0$, cf. [56, p. 80]. Otherwise, we have by definition of p with $|\boldsymbol{\alpha}| = 2j$ that

$$\int_0^\infty r^{|\boldsymbol{\alpha}|+d-1} p(r) \, \omega(r) \, \mathrm{d}r = \int_{\mathbb{R}^d} \|\boldsymbol{t}\|^{2j} \, p\big(\|\boldsymbol{t}\|\big) \, \omega\big(\|\boldsymbol{t}\|\big) \, \mathrm{d}\boldsymbol{t} = \delta_{\boldsymbol{0\alpha}} \, .$$ □

Example 1.3. The most popular weight function is the Gaussian

$$w(\boldsymbol{t}) := \pi^{-d/2} \, \mathrm{e}^{-\|\boldsymbol{t}\|^2} .$$

By the separability of the d-variate Gaussian, orthogonal polynomials with respect to the d-variate Gaussian weight are given by the tensor products of the univariate Hermite-polynomials

$$H_n(y) := (-1)^n \, \mathrm{e}^{y^2} \frac{\mathrm{d}^n}{\mathrm{d}y^n} \, \mathrm{e}^{-y^2}.$$

Using their three-term recurrence relation

$$\begin{aligned} H_0(y) = 1, \qquad H_1(y) = 2y, \\ H_{n+1}(y) = 2yH_n(y) - 2nH_{n-1}(y), \end{aligned}$$

we see that $H_{2n+1}(0) = 0$ and $H_{2n}(0) = (-1)^n \frac{(2n)!}{n!}$. Moreover, it is well known that $\langle H_n, H_k \rangle_w = 2^n \, n! \, \delta_{nk}$, so that

$$\frac{H_{2n}(0)}{\|H_{2n}\|_w^2} = \frac{(-1)^n}{4^n n!}.$$

Consequently, we obtain for even s and $\boldsymbol{t} := (t_1, \ldots, t_d)$ by (7.11) that

$$\begin{aligned} \psi(\boldsymbol{t}) &= \sum_{\substack{|\boldsymbol{\alpha}| \le s, \\ \boldsymbol{\alpha} \text{ even}}} \prod_{j=1}^{d} H_{\alpha_j}(t_j) \frac{(-1)^{\beta_j}}{4^{\beta_j} \beta_j!} w(\boldsymbol{t}) \quad \left(\beta_j := \frac{\alpha_j}{2} \right) \\ &= \sum_{\substack{|\boldsymbol{\alpha}| \le s, \\ \boldsymbol{\alpha} \text{ even}}} \prod_{j=1}^{d} \frac{\mathrm{d}^{\alpha_j}}{\mathrm{d}t^{\alpha_j}} \omega(t_j) \frac{(-1)^{\beta_j}}{2^{\alpha_j} \beta_j!} \\ &= \sum_{\substack{|\boldsymbol{\alpha}| \le s, \\ \boldsymbol{\alpha} \text{ even}}} \frac{\mathrm{d}^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{t}^{\boldsymbol{\alpha}}} w(\boldsymbol{t}) \frac{(-1)^{|\boldsymbol{\alpha}|/2}}{2^{|\boldsymbol{\alpha}|} \beta_1! \cdots \beta_d!} \\ &= \sum_{r=0}^{s/2} \frac{(-1)^r}{2^{2r} r!} \sum_{\substack{|\boldsymbol{\alpha}| = 2r, \\ \boldsymbol{\alpha} \text{ even}}} \frac{r!}{\beta_1! \cdots \beta_d!} \frac{\mathrm{d}^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{t}^{\boldsymbol{\alpha}}} w(\boldsymbol{t}) \\ &= \sum_{r=0}^{s/2} \frac{(-1)^r}{4^r r!} \Delta^r w(\boldsymbol{t}) \, , \end{aligned}$$

where $\Delta w(\boldsymbol{t}) := \sum_{j=1}^{d} \frac{\partial^2}{\partial t_j^2} w(\boldsymbol{t})$ is the Laplacian of w and $\Delta^r w(\boldsymbol{t})$ its rth iterate.

In particular, we have for $d = 2$ that

s	0	2	4
ψ	$w(\boldsymbol{t})$	$w(\boldsymbol{t}) - \frac{1}{4}\Delta w(\boldsymbol{t})$	$w(\boldsymbol{t}) - \frac{1}{4}\Delta w(\boldsymbol{t}) + \frac{1}{32}\Delta^2 w(\boldsymbol{t})$

.

These special functions were also computed by van den Boomgaard and van de Weijer [129]. Fasshauer and Zhang [47] found the corresponding polynomials q in the context of the so-called approximate approximation. For the relation of q to generalized Laguerre polynomials see [97] and the references therein. Since the convolution of a function f with the Laplacian of the Gaussian can be considered

as backward diffusion, the convolution with ψ for $s \geq 2$ leads to a better reproduction of f in particular at edges. This is another way of looking at the improvement of the approximation by a better polynomial reproduction with increasing s. The influence of the additional sharpening terms in ψ is illustrated in [129] and in our examples in Subsection 2.2.

Other weights used in the scattered data literature are the *Wendland functions* [134]. In contrast to the Gaussian these functions have a compact support. For $d = 2$ and $s = 1$ the corresponding functions ψ can be found in [45].

Another popular weight function in image processing is the characteristic function $w(\boldsymbol{x}) := \chi_{\{\boldsymbol{x}:\|\boldsymbol{x}\|_\infty \leq C\}}$, which leads to the so-called *Haralick facet model* [79].

Remark 1.4. The computation of our approximating function u of f in (7.9) requires the discretization of the correlation integral. If we use the rectangular quadrature rule over a grid of mesh size h and equispaced integration nodes $\{x_{\boldsymbol{k}} := h\boldsymbol{k} : \boldsymbol{k} \in \mathbb{Z}^d\}$, we obtain

$$u(\boldsymbol{x}) \approx h^d \sum_{\boldsymbol{k}\in\mathbb{Z}^d} f(\boldsymbol{x_k})\psi(\boldsymbol{x_k} - \boldsymbol{x}).$$

If we replace w by its dilated version $w_\sigma = \frac{1}{\sigma^d} w(\frac{\cdot}{\sigma})$, then ψ with respect to w_σ becomes $\psi_\sigma = \frac{1}{\sigma^d}\psi(\frac{\cdot}{\sigma})$ and the discretized continuous MLS approximation of f with respect to w_σ with $\sigma = \sqrt{D}h$ is

$$u(\boldsymbol{x}) \approx u_{\sqrt{D}h} = D^{-d/2} \sum_{\boldsymbol{k}\in\mathbb{Z}^d} f(\boldsymbol{x_k})\psi\left(\frac{\boldsymbol{x_k} - \boldsymbol{x}}{h\sqrt{D}}\right). \tag{7.15}$$

The right-hand side of (7.15) is known as *approximate approximation* of f. V. Maz'ya and G. Schmidt [98] have proved that for $f \in L^\infty(\mathbb{R}^d) \cap C^{s+1}(\mathbb{R}^d)$ and a function ψ satisfying the moment condition (7.12), the following error estimate holds true

$$\|f - u_{\sqrt{D}h}\|_C = \mathcal{O}\big(h^{s+1} + \varepsilon(\psi, D)\big),$$

where $\varepsilon(\psi, D)$ denotes a saturation error which can be controlled by appropriately choosing the dilation factor σ of the generating function ψ.
Note that [98] contains also error estimates if nonequispaced nodes x_k are used in (7.15).

1.2. Discrete MLS

In scattered data approximation, the function f is in general only known at nonequispaced nodes $\boldsymbol{x}_k \in \mathbb{R}^d$ ($k = 1, \ldots, N$), where $N \geq M$. Instead of using a continuous MLS approach with a discretization of the convolution integral at these nodes, we prefer a discrete MLS approach. Basically, we have the same setting as in Subsection 1.1, (7.2)–(7.4), except that we want to minimize

$$J(\boldsymbol{x}) := \sum_{k=1}^{N} \big(f(\boldsymbol{x}_k) - u(\boldsymbol{x}, \boldsymbol{x}_k)\big)^2 w(\boldsymbol{x}_k - \boldsymbol{x}) \tag{7.16}$$

instead of (7.3). For fixed $\boldsymbol{x} \in \mathbb{R}^d$, this is a weighted least squares problem for the coefficients $c_j = c_j(\boldsymbol{x})$ which has the solution

$$\boldsymbol{c}(\boldsymbol{x}) = \left(\boldsymbol{\Phi}\, \boldsymbol{W}(\boldsymbol{x})\, \boldsymbol{\Phi}^{\mathrm{T}}\right)^{-1} \boldsymbol{\Phi}\, \boldsymbol{W}(\boldsymbol{x})\, \boldsymbol{f}\,, \tag{7.17}$$

where $\boldsymbol{c}(\boldsymbol{x}) := \big(c_j(\boldsymbol{x})\big)_{j=1}^{M}$, $\boldsymbol{f} := \big(f(\boldsymbol{x}_k)\big)_{k=1}^{N}$ and

$$\boldsymbol{\Phi} := \big(\varphi_j(\boldsymbol{x}_k)\big)_{j,k=1}^{M,N}\,, \qquad \boldsymbol{W}(\boldsymbol{x}) := \operatorname{diag}\big(w(\boldsymbol{x}_k - \boldsymbol{x})\big)_{k=1}^{N}\,.$$

Here we have to assume that the points $\boldsymbol{x}_k \in \mathbb{R}^d$ are distributed such that $\boldsymbol{\Phi}$ has full rank, i.e., not all $\boldsymbol{x}_k$ lie on the zero set of a polynomial of degree $\leq s$. Then, by (7.4),

$$u(\boldsymbol{x}) = \boldsymbol{\varphi}(\boldsymbol{x})^{\mathrm{T}}\, \boldsymbol{c}(x) \tag{7.18}$$

is taken as approximation of $f(\boldsymbol{x})$.

Remark 1.5. In the case $s = 0$, i.e., $V = \{1\}$ and $M = 1$, we obtain that $\boldsymbol{\Phi} = (1, \dots, 1)$ and consequently by (7.17) and (7.18) that

$$u(\boldsymbol{x}) = c_1(\boldsymbol{x}) = \frac{\sum\limits_{k=1}^{N} f(\boldsymbol{x}_k) w(\boldsymbol{x}_k - \boldsymbol{x})}{\sum\limits_{k=1}^{N} w(\boldsymbol{x}_k - \boldsymbol{x})}. \tag{7.19}$$

The approximate value $u(\boldsymbol{x})$ of $f(\boldsymbol{x})$ is the weighted average of the values $f(\boldsymbol{x}_k)$, where the weights decrease with an increasing distance of $\boldsymbol{x}_k$ from $\boldsymbol{x}$. This approximation is known as *Shepard's method* [117], which originally looked like

$$u(\boldsymbol{x}) = \sum_{k=1}^{N} f(\boldsymbol{x}_k) w_k(\boldsymbol{x}) \qquad \text{with} \qquad w_k(\boldsymbol{x}) := \frac{\|\boldsymbol{x} - \boldsymbol{x}_k\|^{-\mu_k}}{\sum_{j=1}^{N} \|\boldsymbol{x} - \boldsymbol{x}_j\|^{-\mu_j}} \qquad (\mu_k > 0).$$

The weight functions w_k are normed, continuous and positive. With the convention $w_k(\boldsymbol{x}_j) := \delta_{kj}$ they have the interpolating property $u(\boldsymbol{x}_k) = f(\boldsymbol{x}_k)$. Further, it can be shown by differentiation of the function u that the approximation has a peak at $\boldsymbol{x}_k$ for $0 < \mu_k < 1$, a corner for $\mu_k = 1$ and a horizontal tangent plane for $\mu_k > 1$ [61].

We will have a look at this method again in connection with bilateral filters.

Remark 1.6. From the *Backus-Gilbert approach* [15] it is well-known that, for an appropriate function g, the function ψ_g which solves the constrained minimization problem

$$\frac{1}{2} \sum_{k=1}^{N} \frac{\psi_g^2(\boldsymbol{x}_k, \boldsymbol{x})}{g(\boldsymbol{x}_k, \boldsymbol{x})} \longrightarrow \min$$

subject to the polynomial reproducing property

$$\sum_{k=1}^{N} p(\boldsymbol{x}_k) \psi_g(\boldsymbol{x}_k, \boldsymbol{x}) = p(\boldsymbol{x}) \quad \text{for all } p \in \Pi_s^d \tag{7.20}$$

is given by

$$\big(\psi_g(x_k,x)\big)_{k=1}^N = \boldsymbol{\varphi}(x)^{\mathrm{T}}\ \big(\boldsymbol{\Phi}\,\boldsymbol{D}\,\boldsymbol{\Phi}^{\mathrm{T}}\big)^{-1}\,\boldsymbol{\Phi}\,\boldsymbol{D}\,,$$

where

$$\boldsymbol{D} := \operatorname{diag}\big(g(\boldsymbol{x}_k,\boldsymbol{x})\big)_{k=1}^N\,.$$

Usually, $g(\boldsymbol{x},\boldsymbol{x}_k) := w(\boldsymbol{x}_k-\boldsymbol{x})$ is chosen in the literature. Then, by (7.17), we can rewrite (7.18) in the form

$$u(\boldsymbol{x}) = \sum_{k=1}^N f(\boldsymbol{x}_k)\psi_w(\boldsymbol{x}_k,\boldsymbol{x})\,. \tag{7.21}$$

This approach is also known as *quasi-interpolation* of f. If f is a polynomial of absolute degree $\le s$, then, by the constraint (7.20), it is reproduced exactly, i.e., u coincides with f.

Note that on the other hand, the discrete MLS problem can be considered with the shifted ansatz (7.5), where one has to minimize a discrete functional corresponding to (7.6). This leads directly to the form (7.21) of u.

2. Robust local approximation of scattered data

In [129], R. van den Boomgaard and J. van de Weijer suggested a robust Gaussian facet model for various applications in image processing. Robust estimators classically dealt with statistical outliers, but can be also used to better reconstruct edges. In this section, we want to use the robust facet approach in a slightly more general form for the approximation of (noisy) scattered data. Furthermore, we propose a novel method which seems to be more related to the idea of bilateral filters.

2.1. Generalized bilateral filters

In order to make our approximation more sensible with respect to edges we introduce a differentiable function ρ in J which punishes small differences harder but sees larger differences more gently, i.e., instead of (7.16) we minimize the functional

$$J_\rho(\boldsymbol{x}) := \sum_{k=1}^N \rho\Big(\big(f(\boldsymbol{x}_k)-u(\boldsymbol{x},\boldsymbol{x}_k)\big)^2\Big)w(\boldsymbol{x}_k-\boldsymbol{x})\,.$$

In (7.16) we have simply used $\rho(s^2)=s^2$. In this section, we apply

$$\rho(s^2) := \sqrt{s^2+\varepsilon^2} \qquad (\varepsilon \ll 1) \tag{7.22}$$

which results (approximately) in a weighted ℓ_1-norm of $\big(f(\boldsymbol{x}_k)-u(\boldsymbol{x},\boldsymbol{x}_k)\big)_{k=1}^N$ in J_ρ, and

$$\rho(s^2) = 1-\mathrm{e}^{-s^2/(2m^2)} \tag{7.23}$$

which gives an approximation of a weighted ℓ_0-norm. The function (7.23) was suggested in [129].

Computing the gradient of $J_\rho(\boldsymbol{x})$ with respect to $c_\ell(\boldsymbol{x})$ ($\ell = 1, \ldots, M$) and setting this gradient to zero, leads to the following nonlinear system of equations

$$\boldsymbol{\Phi}\, \boldsymbol{W}(\boldsymbol{x}) \boldsymbol{B}_\rho(\boldsymbol{x})\, \boldsymbol{\Phi}^{\mathrm{T}}\, \boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{\Phi}\, \boldsymbol{W}(\boldsymbol{x}) \boldsymbol{B}_\rho(\boldsymbol{x})\, \boldsymbol{f}\,, \tag{7.24}$$

where

$$\begin{aligned} \boldsymbol{B}_\rho(\boldsymbol{x}) &:= \operatorname{diag}\Big(\rho'\big((f(\boldsymbol{x}_k) - u(\boldsymbol{x}, \boldsymbol{x}_k))^2\big)\Big)_{k=1}^N \\ &= \operatorname{diag}\Big(\rho'\big((f(\boldsymbol{x}_k) - \sum_{\ell=1}^M c_\ell \varphi_\ell(\boldsymbol{x} - \boldsymbol{x}_k))^2\big)\Big)_{k=1}^N . \end{aligned} \tag{7.25}$$

Note that for ρ defined by (7.22) or (7.23) the function $\rho'(s^2)$ is a monotone decreasing function in s^2. In contrast to the diagonal matrix $\boldsymbol{W}(\boldsymbol{x})$ appearing in (7.17), we incorporate now the diagonal matrix $\boldsymbol{W}(\boldsymbol{x})\boldsymbol{B}_\rho(\boldsymbol{x})$ which does not only depend on the nodes $\boldsymbol{x}_k$, but also on the data $f(\boldsymbol{x}_k)$. Thus, we obtain both a node and data dependent method. We solve (7.24) by a fixed point iteration, i.e., we compute successively

$$\boldsymbol{c}^{(i+1)}(\boldsymbol{x}) = \big(\boldsymbol{\Phi}\, \boldsymbol{W}(\boldsymbol{x}) \boldsymbol{B}_\rho^{(i)}(\boldsymbol{x})\, \boldsymbol{\Phi}^{\mathrm{T}}\big)^{-1} \boldsymbol{\Phi}\, \boldsymbol{W}(\boldsymbol{x}) \boldsymbol{B}_\rho^{(i)}(\boldsymbol{x})\, \boldsymbol{f}\,,$$

where

$$\boldsymbol{B}_\rho^{(i)}(\boldsymbol{x}) := \operatorname{diag}\Big(\rho'\big((f(\boldsymbol{x}_k) - \sum_{\ell=1}^M c_\ell^{(i)}(\boldsymbol{x}) \varphi_\ell(\boldsymbol{x}_k - \boldsymbol{x}))^2\big)\Big)_{k=1}^N$$

and set

$$u^{(i+1)}(\boldsymbol{x}) := \boldsymbol{\varphi}(\boldsymbol{x})^{\mathrm{T}}\, \boldsymbol{c}^{(i+1)}(\boldsymbol{x})\,. \tag{7.26}$$

As initial vector $\boldsymbol{c}^{(0)}(x)$ we use the values obtained from the discrete MLS in Subsection 1.2. The question of convergence of this iterative method is still open.

Remark 2.1. If $s = 0$, then we obtain as in Remark 1.5, that $u^{(i)}(\boldsymbol{x}) = c_1^{(i)}(\boldsymbol{x})$, in particular, after one iteration,

$$u^{(1)}(\boldsymbol{x}) = \frac{\sum_{k=1}^N f(\boldsymbol{x}_k) w(\boldsymbol{x}_k - \boldsymbol{x}) \rho'\big((f(\boldsymbol{x}_k) - u^{(0)}(\boldsymbol{x}))^2\big)}{\sum_{k=1}^N w(\boldsymbol{x}_k - \boldsymbol{x}) \rho'\big((f(\boldsymbol{x}_k) - u^{(0)}(\boldsymbol{x}))^2\big)}\,. \tag{7.27}$$

For $\boldsymbol{x} := \boldsymbol{x}_j$ ($j = 1, \ldots, N$) and input $u^{(0)}(\boldsymbol{x}_j) := f(\boldsymbol{x}_j)$, the approximation (7.27) is known as *bilateral filter* [43, 126]. In contrast to Shepard's method (7.19) do the weights of the values $f(\boldsymbol{x}_k)$ in (7.27) not only decrease with an increasing distance of $\boldsymbol{x}_k$ from $\boldsymbol{x}$, but also with an increasing distance of $f(\boldsymbol{x}_k)$ from $f(\boldsymbol{x})$ (or its approximation $u^{(0)}(\boldsymbol{x})$). Thus the averaging process is reduced at edges.

Based on Remark 2.1 and Remark 1.6 we propose the following new approximation method which can be considered as a generalization of the bilateral filter. Obviously, the division by $\sum_{k=1}^{N} w(\boldsymbol{x}_k - \boldsymbol{x})\rho'\big((f(\boldsymbol{x}_k) - u^{(0)}(\boldsymbol{x}))^2\big)$ in (7.27) ensures at each iteration step i that $u^{(i)}$ reproduces constants $f \equiv C$. By Remark 1.6, the idea of using bilateral filters for scattered data approximation can be generalized such that polynomials of arbitrary absolute degree $\leq s$ are reproduced. We have to compute

$$u^{(i+1)}(\boldsymbol{x}) := \boldsymbol{\varphi}(\boldsymbol{x})^{\mathrm{T}} \left(\boldsymbol{\Phi}\, \boldsymbol{W}(\boldsymbol{x}) \boldsymbol{D}_{\rho}^{(i)}(\boldsymbol{x})\, \boldsymbol{\Phi}^{\mathrm{T}}\right)^{-1} \boldsymbol{\Phi}\, \boldsymbol{W}(\boldsymbol{x}) \boldsymbol{D}_{\rho}^{(i)}(\boldsymbol{x})\, \boldsymbol{f}\,, \tag{7.28}$$

where

$$\boldsymbol{D}_{\rho}^{(i)}(\boldsymbol{x}) := \operatorname{diag}\left(\rho'\big((f(\boldsymbol{x}_k) - u^{(i)}(\boldsymbol{x}))^2\big)\right)_{k=1}^{N}.$$

In contrast to $B_{\rho}^{(i)}$ in (7.25), where we find it difficult to interpret the differences $f(\boldsymbol{x}_k) - u^{(i)}(\boldsymbol{x}, \boldsymbol{x}_k)$, our diagonal matrix $\boldsymbol{D}_{\rho}^{(i)}$ contains the approximated differences $f(\boldsymbol{x}_k) - f(\boldsymbol{x}) \approx f(\boldsymbol{x}_k) - u^{(i)}(\boldsymbol{x})$. The function ρ' may be any appropriate decreasing function. Moreover, as initial data $u^{(0)}$ we can take any reasonable approximation of f. Of course, for $s = 1$, both methods (7.26) and (7.28) coincide.

Remark 2.2. In a very recent work, Mrázek, Weickert and Bruhn [99] deal with a number of widely-used nonlinear methods for digital image processing. Especially, they consider iterated bilateral filters and so-called local M-smoothers, which correspond to our robust method (7.28) for $\boldsymbol{x} := \boldsymbol{x}_j$ ($j = 1, \ldots, N$) and $s = 0$. While local M-smoothers use the initial image $\boldsymbol{f}$ in (7.28), the iterated bilateral filters use the evolving image $u^{(i)}(\boldsymbol{x})$ in step $i + 1$. In this case, one has to stop after a certain number of iterations in order to avoid obtaining a flat image.

2.2. Numerical results

In this section, we present numerical examples with the proposed algorithms in one and two dimensions. The algorithms were implemented in C. As weight function w, we have always used a dilated Gaussian function $w_\sigma(y) = \mathrm{e}^{-y^2/(2\sigma^2)}$ which we have truncated for $|y| > 3\sigma$. For this thesis, we have restricted ourselves to the nonlinear function $\rho(s^2) = 1 - \mathrm{e}^{-s^2/(2m^2)}$ in (7.23). However, we have computed various examples with the function ρ in (7.22) as well. In 2D, these results look very similar to those obtained by applying (7.23). The corresponding images can be found at our web page

`http://kiwi.math.uni-mannheim.de/~mfenn/RMLS.html`

The nonlinear methods were always performed with five iterations, since we observed reasonable convergence in all our experiments within ≤ 5 iteration steps.

Figure 7.1 shows a onedimensional example with the 'ramp'-signal. The first row contains the original 256 pixel data in (a) and 64 scattered data points (uniformly distributed random numbers) with some Gaussian noise added in (b) ($\mathrm{SNR} = 8$ dB). The following rows of Figure 7.1 show the results of the MLS approximation in (c)–(e), of iteration scheme (7.26) with the diagonal matrix B_ρ in (f)–(h), and of our

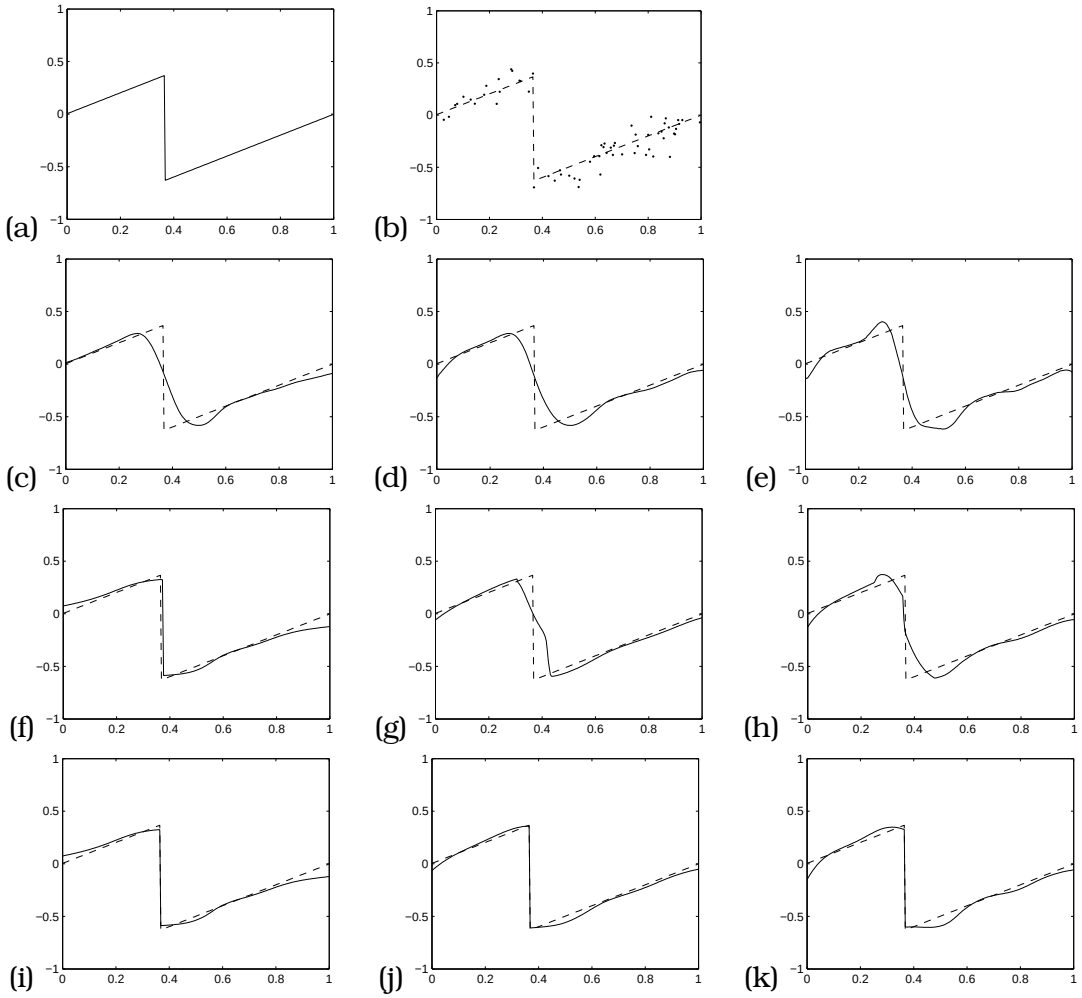


Figure 7.1.: (a) original signal; (b) scattered noisy signal ($1/8$ of the original data, $\mathrm{SNR} = 8$); (c)–(e) MLS approximation; (f)–(h) method (7.26); (i)–(k) our generalized bilateral filter (7.28).

generalized bilateral filter (7.28) with the diagonal matrix D_ρ in (i)–(k), where the polynomial reproduction degree increases from $s = 0$ to $s = 2$ from left to right. The parameters σ for the node-dependent weights and the parameter m for the data-dependent weights were chosen such that the optical impression was the best. In the MLS approximations, we have taken $\sigma = 3/64$, and in the nonlinear schemes (7.26) and (7.28), the parameters $\sigma = 6/64$ and $m = 0.2$. As initial data for the iterative algorithms we have always used the results from the MLS approximation with the same degree of polynomial reproduction. However, it should be noted that our algorithm (7.28) has shown a quite robust behavior with respect to the choice of the initial data. Even very rough initial data approximations, e.g., a simple linear approximation, has led to nearly the same results (i)–(k).

As expected, the MLS approximation smoothes at edges. This effect can be reduced by using the data dependent iteration schemes. However, the nonlinear method (7.26) still introduces some artefacts at edges. The same effect can be observed in 2D.

Since the original signal is piecewise linear, the methods which reproduces quadratic polynomials (right column) do not bring some further improvements.

Figure 7.2 compares scattered data approximation in 2D. We took the 256×256 pixel image 'trui.png' in (a), added some Gaussian noise with $\mathrm{SNR} = 16$ dB in (b). Finally, we chose randomly $1/16$ of the data in (c). The images (d)–(f) in the second row of Figure 7.2 show the results of the MLS approximation for $s = 0,\ 1,\ 2$ from left to right. The parameter $\sigma = 6/256$ was chosen such that the images look best. However, we have also computed the images with respect to that parameter σ which gives the best SNR. The results are reported at our web page. The third and fourth row present the results for the nonlinear methods (7.26) and (7.28), respectively, with an increasing degree of the polynomial reproduction $s = 0,\ 1,\ 2$ from left to right. Here $\sigma = 6/256$, too, and the parameter m was chosen such that we have obtained the best SNR. In general, we had $m \in [0.18, 0.28]$. The SNR of each image can be found in the caption of Figure 7.2. The quality of the images improves with an increasing degree of polynomial reproduction. As expected, the nonlinear methods produce somewhat sharper images. In order to observe this effect more carefully, the reader may have again a look at details of the images at our web page. The best result was obtained with our generalized bilateral filter (7.28) and $s = 2$. Note that one iteration step takes less than two seconds here.

Figure 7.3 is based on a data set frequently used in numerical examples for scattered data approximation: we are given 873 scattered data points representing certain contour lines of a glacier. First, we applied the MLS method with $\sigma = 6/128$ and $s = 2$. The contour plots evaluated at the 128×128 grid are presented in (b). Part (c) of the figure shows the result for our algorithm (7.28) applied with $\sigma = 8/128$, $m = 15$ and $s = 2$. The corresponding 3D plot can be seen in (a).

The contour plots (b), (c) reveal the differences of both methods. Although the MLS approximation (b) is quite good, our nonlinear method (c) better reconstructs smaller structures. For example, the peaks in the middle right part of the images are smoothed by the MLS, but retain by our algorithm.

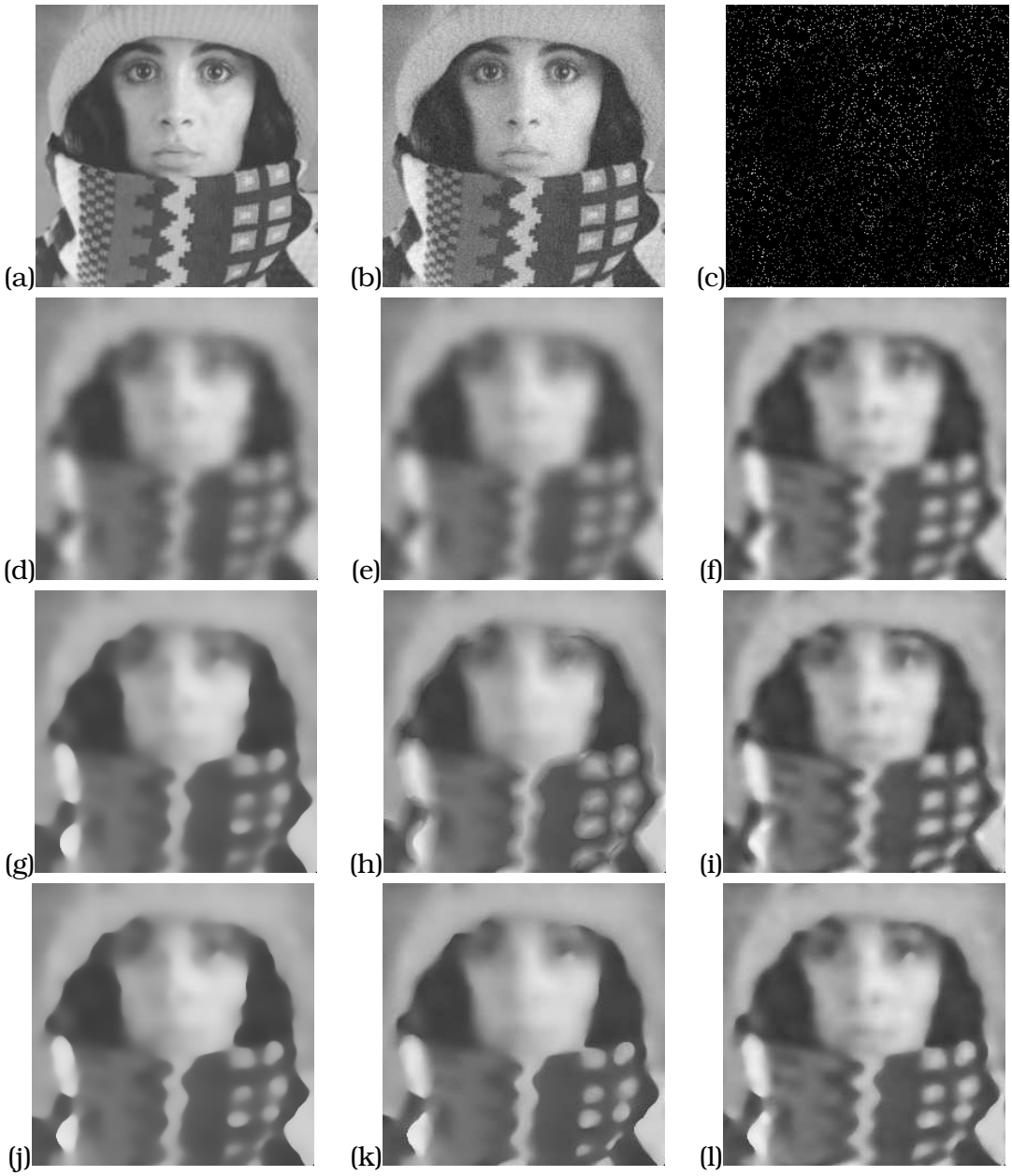
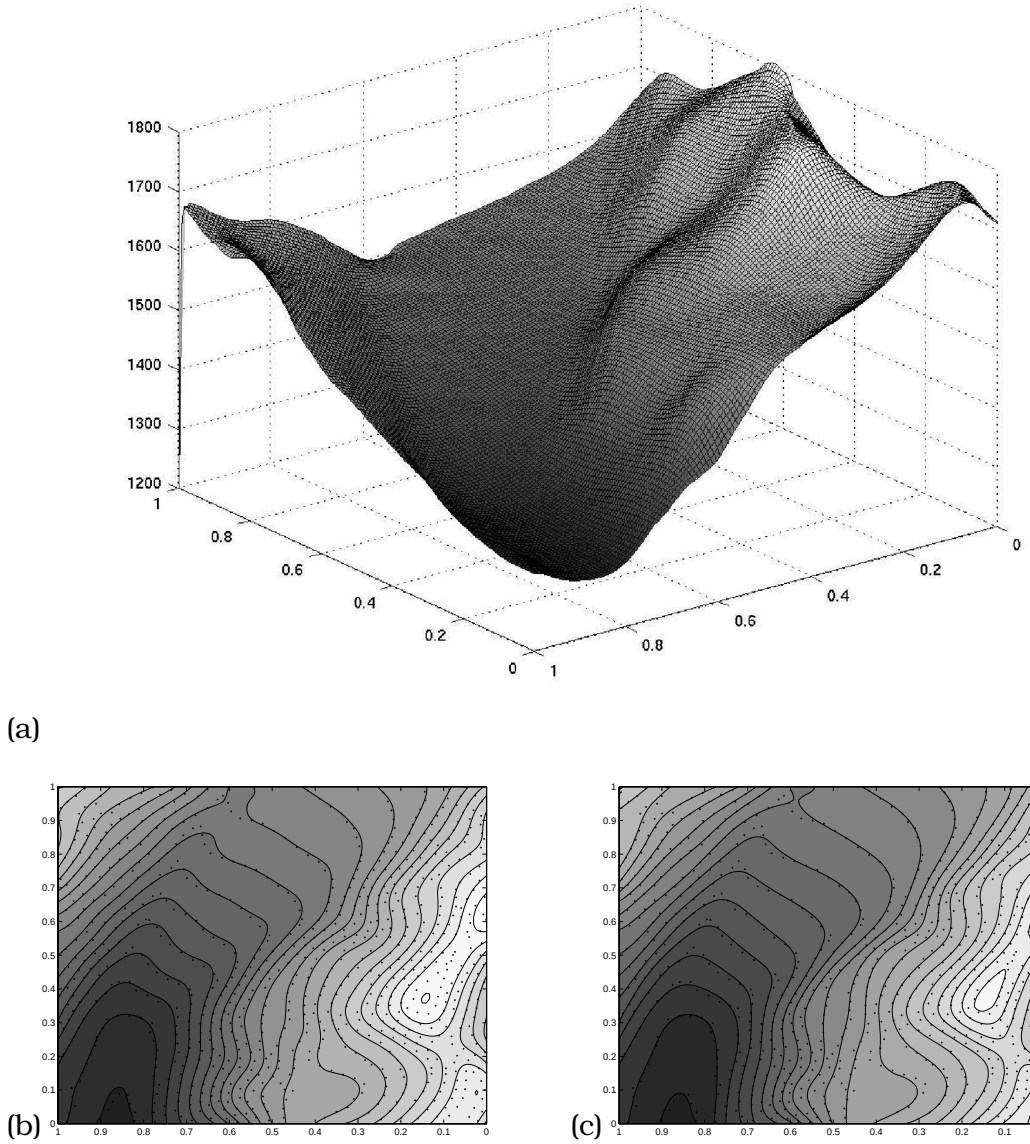


Figure 7.2.: (a) original image; (b) noisy image (SNR $= 16$); (c) scattered noisy image (1/16 of the data); (d)–(f) MLS with $s = 0$ (SNR $= 7.62$), $s = 1$ (SNR $= 7.73$), $s = 2$ (SNR $= 9.79$); (g)–(i) method (7.26) with $s = 0$ (SNR $= 8.70$), $s = 1$ (SNR $= 8.58$), $s = 2$ (SNR $= 10.48$); (j)–(l) our generalized bilateral filter (7.28) with $s = 0$ (SNR $= 8.82$), $s = 1$ (SNR $= 9.41$), $s = 2$ (SNR $= 10.62$).



(a)

(b)

(c)

Figure 7.3.: Approximation of 873 scattered data points from the 'glacier' at the 128×128 grid; (a) 3-D plot of (c); (b) original data (dotted) and contour plot of the MLS approximation with $s = 2$, (c) our generalized bilateral filter (7.28) with $s = 2$.

Bibliography

[1] R. P. Agarwal and P. J. Y. Wong. *Error Inequalities in Polynomial Interpolation and Their applications*, volume 262 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1993.

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

[3] C. R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM J. Sci. Statist. Comput.*, 13(4):923–947, 1992.

[4] A. Averbuch, R. Coifman, D. Donoho, M. Israeli, and Y. Shkolnisky. Fast Slant Stack: A notion of Radon Transform for Data in a Cartesian grid which is Rapidly Computible, Algebraically Exact, Geometrically Faithful and Invertible. SIAM J. Sci. Comput., submitted for publication.

[5] R. F. Bass and K. Gröchenig. Random sampling of multivariate trigonometric polynomials. *SIAM J. Math. Anal.*, 36(3):773–795 (electronic), 2004.

[6] G. Baszenski and F.-J. Delvos. A discrete Fourier transform schema for Boolean sums of trigonometric operators. In C. Chui, W. Schempp, and K. Zeller, editors, *Multivariate Approximation Theory IV*, pages 15–24, Basel, 1989. Birkhäuser.

[7] R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines. *IMA J. Numer. Anal.*, 17(3):343–372, 1997.

[8] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions. I. *Comput. Math. Appl.*, 24(12):7–19, 1992.

[9] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: moment-based methods. *SIAM J. Sci. Comput.*, 19(5):1428–1449, 1998.

[10] M. Bebendorf. Approximation of boundary element matrices. *Numer. Math.*, 86(4):565–589, 2000.

[11] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Comput. Methods Appl. Mech. Eng.*, 139(1-4):3–47, 1996.

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

[13] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. I. *Comm. Pure Appl. Math.*, 44(2):141–183, 1991.

[14] G. Beylkin and R. Cramer. A multiresolution approach to regularization of singular operators and fast summation. *SIAM J. Sci. Comput.*, 24(1):81–117, 2002.

[15] L. P. Bos and K. Šalkauskas. Moving least-squares are Backus-Gilbert optimal. *J. Approx. Theory*, 59(3):267–275, 1989.

[16] A. Brandt. Multilevel computations of integral transforms and particle interactions with oscillatory kernels. *Comput. Phys. Comm.*, 65(1-3):24–38, 1991.

[17] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[18] E. Candès. Harmonic analysis of neural networks. *Appl. Comput. Harmon. Anal.*, 6:197–218, 1999.

[19] E. Candès, L. Demanet, D. Donoho, et al. Fast discrete curvelet transforms. Tech. Report, California Institute of Technology, 2005.

[20] E. Candès and D. Donoho. Ridgelets: a key to higher-dimensional intermittency? *Phil. Trans. R. Soc. Lond. A*, 357:2495–2509, 1999.

[21] E. Candès and D. Donoho. Curvelets and curvilinear integrals. *J. Approx. Theory*, 113:59–90, 2001.

[22] P. Carré and E. Andres. Discrete analytical ridgelet transform. *Signal Process.*, 84(11):2165–2173, 2004.

[23] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Statist. Comput.*, 9(4):669–686, 1988.

[24] A. Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vision*, 20(1-2):89–97, 2004. Special issue on mathematics and image analysis.

[25] T. Chan and H. Zhou. Optimal constructions of wavelet coefficients using total variation regularization in image compression. CAM Technique Report 00-27, Dept. of Math., UCLA, July 2000.

[26] H. Cheng and L. Greengard. A method of images for the evaluation of electrostatic fields in systems of closely spaced conducting cylinders. *SIAM J. Appl. Math.*, 58(1):122–141, 1998.

[27] J. B. Cherrie, R. K. Beatson, and G. N. Newsam. Fast evaluation of radial basis functions: methods for generalized multiquadrics in $\mathbb{R}^n$. *SIAM J. Sci. Comput.*, 23(5):1549–1571, 2002.

[28] W. C. Chew, J. M. Jin, C. C. Lu, E. Michielssen, and J. M. Song. Fast solution methods in electromagnetics. *IEEE Trans. Antennas Propagation*, 45:533–543, 1997.

[29] J. W. Cooley and J. W. Tukey. An algorithm for machine calculation of complex Fourier series. *Math. Comput.*, 19:297–301, 1965.

[30] C. de Boor. Total positivity of the spline collocation matrix. *Indiana Univ. Math. J.*, 25(6):541–551, 1976.

[31] C. de Boor. *Splinefunktionen*. Lectures in Mathematics ETH Zürich. Birkhäuser Verlag, Basel, 1990. With a preface by Manfred R. Trummer.

[32] F.-J. Delvos and W. Schempp. *Boolean Methods in Interpolation and Approximation*, volume 230 of *Pitman Research Notes in Mathematics Series*. Longman Scientific & Technical, Harlow, 1989.

[33] M. Do and M. Vetterli. Contourlets. In J. Stoeckler and G. Welland, editors, *Beyond Wavelets*. Academic Press, New York, 2003.

[34] M. Do and M. Vetterli. The finite ridgelet transform for image representation. *IEEE Trans. Image Procss.*, 12(1):16–28, 2003.

[35] M. N. Do. Finite Ridgelet Transform Matlab Toolbox (FRIT) - Version 1.0. University of Illinois at Urbana-Champaign, November 2003.

[36] D. Donoho et al. Wavelab 802 - a collection of Matlab functions. `http://www-stat.stanford.edu/~wavelab/`, 1999.

[37] D. Donoho and A. Flesia. Digital ridgelet transform based on true ridge functions. In J. Stoeckler and G. Welland, editors, *Beyond Wavelets*. Academic Press, New York, 2003.

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

[39] S. Durand and J. Froment. Reconstruction of wavelet coefficients using total variation minimization. *SIAM J. Sci. Comput.*, 24(5):1754–1767, 2003.

[40] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration, and differentiation. *SIAM J. Numer. Anal.*, 33(5):1689–1711, 1996.

[41] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, 1993.

[42] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. II. *Appl. Comput. Harmon. Anal.*, 2(1):85–100, 1995.

[43] M. Elad. On the origin of the bilateral filter and ways to improve it. *IEEE Trans. Image Process.*, 11(10):1141–1151, 2002.

[44] B. Elbel and G. Steidl. Fast Fourier transforms for nonequispaced data. In C. Chui and L. Schumaker, editors, *Approximation Theory IX, Vol. 2*, pages 39–46. Vanderbilt Univ. Press, Nashville, TN, 1998.

[45] G. E. Fasshauer. Approximate moving least-squares approximation with compactly supported radial weights. In M. Griebel and M. A. Schweitzer, editors, *Meshfree methods for partial differential equations (Bonn, 2001)*, volume 26 of *Lect. Notes Comput. Sci. Eng.*, pages 105–116. Springer, Berlin, 2003.

[46] G. E. Fasshauer. Multivariate Meshfree Approximation. Technical report, Dept. of Mathematics, Illinois Inst. of Technology, Chicago, 2003. `http://amadeus.math.iit.edu/~fass/603_handouts.html`.

[47] G. E. Fasshauer and J. G. Zhang. Recent results for moving least squares approximation. In M. L. Lucian and M. Neamtu, editors, *Geometric Modeling and Computing: Seattle 2003*, pages 163–176, Brentwood, TN, 2004. Nashboro Press.

[48] M. Fenn, S. Kunis, and D. Potts. Fast evaluation of trigonometric polynomials from hyperbolic crosses. *Numerical Algorithms*, submitted.

[49] M. Fenn and J. Ma. Combined complex ridgelet shrinkage and total variation minimization. *SIAM J. Sci. Comput.*, accepted.

[50] M. Fenn and D. Potts. Fast summation based on fast trigonometric transforms at non-equispaced nodes. *Numer. Linear Algebra Appl.*, 12(2-3):161–169, 2005.

[51] M. Fenn and G. Steidl. FMM and $\mathcal{H}$-matrices: a short introduction to the basic idea. Technical report, Univ. Mannheim, 2002.

[52] M. Fenn and G. Steidl. Fast NFFT based summation of radial functions. *Sampl. Theory Signal Image Process.*, 3(1):1–28, 2004.

[53] M. Fenn and G. Steidl. Robust local approximation of scattered data. In R. Klette, R. Kozera, L. Noakes, and J. Weickert, editors, *Geometric Properties from Incomplete Data*, volume 31 of *Computational Imaging and Vision*, pages 317–334, Dordrecht, The Netherlands, 2006. Springer.

[54] J. A. Fessler and B. P. Sutton. NUFFT – nonuniform FFT toolbox for Matlab. `http://www.eecs.umich.edu/~bpsutton/MR/Code/NUFFT/`, 2002.

[55] J. A. Fessler and B. P. Sutton. Nonuniform fast Fourier transforms using min-max interpolation. *IEEE Trans. Signal Process.*, 51(2):560–574, 2003.

[56] G. B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Wiley-Interscience Series of Texts, Monographs, and Tracts. J. Wiley and Sons, Inc., New Nork, 1999.

[57] K. Fourmont. *Schnelle Fourier–Transformation bei nichtäquidistanten Gittern und tomographische Anwendungen*. PhD thesis, University of Münster, 1999.

[58] K. Fourmont. Non-equispaced fast Fourier transforms with applications to tomography. *J. Fourier Anal. Appl.*, 9(5):431–450, 2003.

[59] R. Franke. Scattered data interpolation: tests of some methods. *Math. Comp.*, 38(157):181–200, 1982.

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

[61] W. J. Gordon and J. A. Wixom. Shepard's method of "metric interpolation" to bivariate and multivariate interpolation. *Math. Comp.*, 32(141):253–264, 1978.

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

[63] L. Greengard. *The rapid evaluation of potential fields in particle systems*. MIT Press, Cambridge, MA, 1988.

[64] L. Greengard and J. Helsing. On the numerical evaluation of elastostatic fields in locally isotropic two-dimensional composites. *J. Mech. Phys. Solids*, 46(8):1441–1462, 1998.

[65] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.

[66] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 135(2):280–292, 1997.

[67] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997.

[68] L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. Sci. Statist. Comput.*, 12(1):79–94, 1991.

[69] L. Greengard and X. Sun. A new version of the fast Gauss transform. *Doc. Math.*, pages 575–584, 1998.

[70] W. Hackbusch. The panel clustering algorithm. In J. R. Whiteman, editor, *MAFELAP 1990*, pages 339–348. Academic Press, London, 1991.

[71] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. I. Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[72] W. Hackbusch, L. Grasedyck, and S. Börm. An introduction to hierarchical matrices. *Math. Bohem.*, 127:229–241, 2002.

[73] W. Hackbusch, B. Khoromskij, and S. A. Sauter. On $\mathcal{H}^2$-matrices. In H.-J. Bungartz, R. H. W. Hoppe, and C. Zenger, editors, *Lectures on Applied Mathematics (Munich, 1999)*, pages 9–29, Berlin, 2000. Springer.

[74] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic: general complexity estimates. *J. Comput. Appl. Math.*, 125(1-2):479–501, 2000.

[75] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. II. Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.

[76] W. Hackbusch and B. N. Khoromskij. Towards $\mathcal{H}$-matrix approximation of linear complexity. In J. Elschner, I. Gohberg, and B. Silbermann, editors, *Problems and Methods in Mathematical Physics. The Siegfried Prössdorf Memorial Volume (Chemnitz, 1990)*, pages 194–220, Basel, 2001. Birkhäuser.

[77] W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math.*, 54(4):463–491, 1989.

[78] K. Hallatschek. Fouriertransformation auf dünnen Gittern mit hierarchischen Basen. *Numer. Math.*, 63:83–97, 1992.

[79] R. M. Haralick, T. J. Laffey, and L. Watson. The topographic primal sketch. *The International Journal of Robotics Research*, 2:50–72, 1983.

[80] A. Harten and I. Yad-Shalom. Fast multiresolution algorithms for matrix-vector multiplication. *SIAM J. Numer. Anal.*, 31(4):1191–1218, 1994.

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

[82] N. Kingsbury. Image processing with complex wavelets. *Phil. Tran. R. Soc. Lond. A*, 357:2543–2560, 1999.

[83] N. Kingsbury. Complex wavelets for shift invariant analysis and filtering of signals. *Appl. Comput. Harmon. Anal.*, 10(3):234–253, 2001.

[84] N. Kingsbury. Dual-Tree Complex Wavelet Transform Pack - A toolbox of Matlab code - version 4.1. Cambridge University, 2002.

[85] M. Konik, R. Schneider, and G. Steidl. Matrix compression by discrete wavelet transform. In Shumaker et al., editors, *Approximation Theory VIII, Vol. 2 (College Station, TX, 1995)*, pages 225–234. World Sci. Publishing, River Edge, NJ, 1995.

[86] S. Krishnan and L. V. Kale. A parallel adaptive fast multipole algorithm for n-body problems. In K. Gallivan, editor, *Prooceedings of the International Conference on Parallel Processing*, volume 3, pages 46–50. CRC Press, 1995.

[87] S. Kunis. Iterative Fourier-Rekonstruktion. Master's thesis, University of Lübeck, August 2003.

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

[89] S. Kunis and D. Potts. Stability results for scattered data interpolation by trigonometric polynomials. Technical report, University of Lübeck, 2005.

[90] S. Kunis, D. Potts, and G. Steidl. Fast Fourier transform at nonequispaced knots - a user's guide to a C-library. Preprint, MU-Lübeck B-02-13, September 2002.

[91] E. LePennec and S. Mallat. Sparse geometric image representations with bandelets. IEEE Trans. on Image Process., accepted, 2004.

[92] W.-K. Liu, S. Li, and T. Belytschko. Moving least-square reproducing kernel methods. I. Methodology and convergence. *Comput. Methods Appl. Mech. Engrg.*, 143(1-2):113–154, 1997.

[93] J. Ma. Towards artifact-free characterization of surface topography using complex wavelets and total variation minimization. *Appl. Math. Comput., accepted, 2005*, 170(2):1014–1030, 2005.

[94] J. Ma, X. Jiang, and L. Blunt. Complex wavelet transform for extraction of morphological features on surface topography. In *Euspen's 4th int. conf.*, pages 350–351, Glasgow, UK, May 30 – June 2 2004.

[95] J. Ma, X. Jiang, and P. Scott. Complex ridgelets for shift invariant characterization of surface topography with line singularities. Phys. Lett. A, accepted, 2005.

[96] F. Malgouyres. A framework for image deblurring using wavelet packet bases. *Appl. Comput. Harmon. Anal.*, 12(3):309–331, 2002.

[97] V. Maz'ya and G. Schmidt. On approximate approximation by using Gaussian kernels. *IMA J. Numer. Anal.*, 16:13–29, 1996.

[98] V. Maz'ya and G. Schmidt. On quasi-interpolation with non-uniformly distributed centers on domains and manifolds. *J. Approx. Theory*, 110(2):125–145, 2001.

[99] P. Mrázek, J. Weickert, and A. Bruhn. On robust estimation and smoothing with spatial and tonal kernels. In R. Klette, R. Kozera, L. Noakes, and J. Weickert, editors, *Geometric Properties from Incomplete Data*, volume 31 of *Computational Imaging and Vision*, pages 335–352, Dordrecht, The Netherlands, 2006. Springer.

[100] C. Müller. Über die ganzen Lösungen der Wellengleichung. *Math. Annalen*, 124:235–264, 1952.

[101] F. Natterer. *The Mathematics of Computerized Tomography*. B. G. Teubner, Stuttgart, 1986.

[102] F. Natterer and F. Wübbeling. *Mathematical Methods in Image Reconstruction*. SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.

[103] J. Neumann and G. Steidl. Dual-tree complex wavelet transform in the frequency domain and an application to signal classification. *Int. J. Wavelets Multiresolut. Inf. Process.*, 3(1):43–65, 2005.

[104] N. Nguyen and Q. H. Liu. The regular Fourier matrices and nonuniform fast Fourier transforms. *SIAM J. Sci. Compt.*, 21(1):283–293, 1999.

[105] A. Nieslony and G. Steidl. Approximate factorizations of Fourier matrices with nonequispaced knots. *Linear Algebra Appl.*, 266:337–351, 2003.

[106] J. O'Sullivan. A fast sinc function gridding algorithm for Fourier inversion in computer tomography. *IEEE Trans. Med. Imag.*, 4(4):200–207, December 1985.

[107] D. Potts. Fast algorithms for discrete polynomial transforms on arbitrary grids. *Linear Algebra Appl.*, 366:353–370, 2003. Special issue on structured matrices: analysis, algorithms and applications (Cortona, 2000).

[108] D. Potts and G. Steidl. Fourier reconstruction of functions from their nonstandard sampled Radon transform. *J. Fourier Anal. Appl.*, 8(6):513–533, 2002.

[109] D. Potts and G. Steidl. Fast summation at nonequispaced knots by NFFTs. *SIAM J. Sci. Comput.*, 24(6):2013–2037 (electronic), 2003.

[110] D. Potts, G. Steidl, and A. Nieslony. Fast convolution with radial kernels at nonequispaced knots. *Numer. Math.*, 98(2):329–351, 2004.

[111] D. Potts, G. Steidl, and M. Tasche. Fast Fourier transforms for nonequispaced data: A tutorial. In J. Benedetto and P. Ferreira, editors, *Modern sampling theory: Mathematics and Applications*, Appl. Numer. Harmon. Anal., pages 247–270. Birkhäuser, Boston, MA, 2001.

[112] M. J. D. Powell. The theory of radial basis function approximation in 1990. In *Advances in numerical analysis, Vol. II (Lancaster, 1990)*, Oxford Sci. Publ., pages 105–210. Oxford Univ. Press, New York, 1992.

[113] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithm. *Physica D*, 60:259–268, 1992.

[114] R. Scramek and F. Schwab. Imaging. In R. A. Perley, F. R. Schwab, and A. H. Bridle, editors, *Synthesis imaging in radio astronomy, A collection of Lectures from the Third NRAO Synthesis Imaging Summer School*, volume 6, pages 117–138. Astronomical Society of the Pacific Conference, San Francisco, 1988.

[115] I. W. Selesnick. The design of approximate Hilbert transform pairs of wavelet bases. *IEEE Trans. Signal Process.*, 50(5):1144–1152, 2002.

[116] B. Shanker, S.-K. Han, and W. C. Chew. A fast multipole approach to computing scattering from an inhomogeneous bianisotropic cylindrical object using Beltrami fields. In *IEEE Antennas and Propagation Society International Symposium*, pages 43–51, 1997.

[117] D. Shepard. A two dimensional interpolation function for irregularly spaced data. In *Proc. 23rd Nat. Conf. ACM*, pages 517–523, 1968.

[118] N. Sochen, R. Kimmel, and A. M. Bruckstein. Diffusions and confusions in signal and image processing. *J. Math. Imaging Vision*, 14(3):195–209, 2001. Mathematics and image analysis 2000 (Paris).

[119] F. Sprengel. *Interpolation und Waveletzerlegung multivariater periodischer Funktionen*. PhD thesis, University of Rostock, 1997.

[120] F. Sprengel. A class of periodic function spaces and interpolation on sparse grids. *Numer. Funct. Anal. Optimization*, 21(1-2):273–293, 2000.

[121] J. Starck, E. Candès, and D. Donoho. The curvelet transform for image denoising. *IEEE Trans. Image Process.*, 11(6):670–684, 2002.

[122] G. Steidl. A note on fast Fourier transforms for nonequispaced grids. *Adv. Comput. Math.*, 9(3-4):337–352, 1998.

[123] G. Steidl, J. Weickert, T. Brox, et al. On the equivalence of soft wavelet shrinkage, total variation diffusion, total variation regularization, and SIDEs. *SIAM J. Numer. Anal.*, 42(2):686–713, 2004.

[124] X. Sun and N. P. Pitsianis. A matrix version of the fast multipole method. *SIAM Rev.*, 43(2):289–300, 2001.

[125] B. Tian and Q. Liu. Nonuniform fast cosine transform and Chebyshev PSTD algorithms. *J. Electromagn. Waves Appl.*, 14(6):797–798, 2000.

[126] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. 6th Internat. Conf. on Computer Vision*, pages 839–846, 1998.

[127] E. E. Tyrtyshnikov. Mosaic-skeleton approximations. *Calcolo*, 33(1-2):47–57, 1996.

[128] E. E. Tyrtyshnikov. Mosaic ranks and skeletons. In L. G. Vulkov, J. Wasniewski, and P. Y. Yalamov, editors, *Numerical Analysis and its Applications (Rousse, 1996)*, volume 1196 of *Lecture Notes in Computer Science*, pages 505–516. Springer, Berlin, 1997.

[129] R. van den Boomgaard and J. van de Weijer. Least Squares and Robust Estimation of Local Image Structure. In L. D. Griffin and M. Lillholm, editors, *Scale-Space 2003*, volume 2695 of *Lect. Notes Comput. Sci.*, pages 237–254. Springer, 2003.

[130] V. V. Voevodin. On a method of reducing the matrix order while solving integral equations. In V. V. Voevodin, editor, *Numerical Analysis on FORTRAN*, volume 17, pages 21–26. Moscow University Press, Moscow, 1979.

[131] C. Vogel and M. Oman. Fast, robust total variation-based reconstruction of noisy, blurred images. *IEEE Trans. Image Process.*, 7(6):813–824, 1998.

[132] J. Weickert. *Anisotropic Diffusion in Image Processing*. ECMI Series. Teubner, Stuttgart, 1998.

[133] J. Weickert. C-programs for edge-enhancing anisotropic diffusion filtering. Universität des Saarlandes, 1998.

[134] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4:389–396, 1995.

[135] A. Yip and F. Park. Solution dynamics, causality, and critical behaviour of the regularization parameter in total variation denoising problems. UCLA Report, 2003.

[136] C. Zenger. Sparse grids. In *Parallel algorithms for partial differential equations (Kiel, 1990)*, volume 31 of *Notes Numer. Fluid Mech.*, pages 241–251. Vieweg, Braunschweig, 1991.

Index