Schnelle Fourier-Transformationen für nichtäquidistante Daten und Anwendungen

Habilitationsschrift

zur Erlangung der Venia legendi
für das Fach Mathematik

vorgelegt der
Technisch-Naturwissenschaftlichen Fakultät
der Universität zu Lübeck

von
Dr. rer. nat. Daniel Potts
geboren am 11. Juli 1969
in Malchin

Lübeck, im März 2003

Dekan:	Prof. Dr. R. Reischuk
Gutachter:	Prof. Dr. W. Hackbusch, (Max-Planck-Inst. Leipzig)
Gutachter:	Prof. Dr. J. Prestin (Universität zu Lübeck)
Gutachter:	Prof. Dr. R. Schaback (Georg-August-Universität Göttingen)
13.01.2004	Tag der Antrittsvorlesung

Inhaltsverzeichnis

Einführung

Ein wesentliches Anliegen der angewandten Mathematik besteht in der Entwicklung schneller Algorithmen für häufig wiederkehrende Grundaufgaben. Zu den bekanntesten effizienten Algorithmen gehört die schnelle Fourier-Transformation (Fast Fourier Transform, kurz: FFT).

Bereits L. Euler nutzte 1753 die Formeln der reellen, diskreten Fourier-Transformation, um Planetenbahnen zu berechnen. Arbeiten zur Geschichte der Mathematik belegen, dass C.F. Gauß schon 1805 die Rechenvorteile der FFT zu Hilfe nahm, um die reelle diskrete Fourier-Transformation effizient auszuwerten [64]. Zahlreiche Anwendungen beweisen, dass die FFT nicht an Aktualität verloren hat.

Es ist kaum möglich, die Bedeutung der FFT in dieser kurzen Einleitung hinreichend zu würdigen, da sie ganze Arbeitsgebiete, wie z.B. die Signalverarbeitung, revolutionierte und in alle Bereiche der Naturwissenschaften Einzug gehalten hat.

Ein Grund für die Popularität der FFT ist sicher in der Fourier-Analysis zu finden. Nach der Diskretisierung von Fourier-Integralen sind trigonometrische Polynome vom Grad N an M verschiedenen Stützstellen auszuwerten. Falls die Stützstellen äquidistant liegen, ist diese Transformation als diskrete Fourier-Transformation (Discrete Fourier Transform, kurz: DFT) bekannt. Durch die Anwendung der FFT verringert sich die arithmetische Komplexität einer d-variaten DFT der Länge N mit $M = N^d$ von $\mathcal{O}(N^{2d})$ auf $\mathcal{O}(N^d \log N)$. Unter der arithmetischen Komplexität eines Algorithmus verstehen wir die Anzahl der zu dessen Realisierung benötigten Multiplikationen und Additionen. Oft ist bei der arithmetischen Komplexität eines Algorithmus nur dessen Größenordnung, in Abhängigkeit von der Anzahl der Eingabewerte, von Interesse.

Einen weiteren bedeutenden Schub bekamen die Fourier-Techniken durch den Einsatz von Computern und durch die Wiederentdeckung der FFT, die 1965 von J.W. Cooley und J.W. Tukey [25] publiziert wurde.

Die Beschränkung der FFT auf äquidistante Gitter ist in einer Vielzahl von weiteren Anwendungen ein nicht zu übersehender Nachteil.

In dieser Arbeit wird daher die FFT zur schnellen Fourier-Transformation für nichtäquidistante Daten (Nonequispaced Fast Fourier Transform, kurz: NFFT) weiterentwickelt. Wir leiten schnelle Verfahren her, um ein d-variates trigonometrisches Polynom vom Grad N an M verschiedenen Stellen auszuwerten. Die arithmetische Komplexität dieser Algorithmen beträgt $\mathcal{O}(N^d \log N + M)$. Durch die Verallgemeinerung der FFT werden sich sehr viele neue Anwendungsmöglichkeiten eröffnen. Drei verschiedene Einsatzgebiete sollen hier beschrieben werden:

- Algorithmen zur schnellen Summation,
- Verfahren in der Computertomographie,
- schnelle Fourier-Algorithmen auf der Sphäre.

Diese Arbeit ist in vier Kapitel gegliedert, wobei Kapitel 1 die NFFT erklärt und damit die Grundlagen für unsere Betrachtungen in den anderen Kapiteln legt. Alle Kapitel beginnen mit einer Einordnung und Beschreibung der Probleme. Die Präsentation numerischer Ergebnisse der neuen Verfahren findet jeweils im letzten Abschnitt statt.

In *Kapitel* 1 werden schnelle Fourier-Algorithmen für nichtäquidistante Daten und deren Eigenschaften untersucht. Wir entwickeln approximative Verfahren zur Realisierung der diskreten Fourier-Transformation für nichtäquidistante Daten (Nonequispaced Discrete Fourier Transform, kurz: NDFT). Die Grundidee der schnellen Algorithmen für die NDFT besteht in einer Annäherung von trigonometrischen Polynomen im Zeit- und Frequenzbereich durch Translate einer Ansatzfunktion φ. Dadurch tritt ein Approximationsfehler auf, den wir in einen Überlappungsfehler und einen Abschneidefehler aufteilen. Für jeweils verschiedene Ansatzfunktionen φ ergeben sich die NFFT-Algorithmen von A. Dutt und V. Rokhlin [31], G. Beylkin [8] oder K. Fourmont [45]. Um den Wechsel vom Frequenzbereich in den Zeitbereich schnell zu realisieren, verwenden wir die FFT. Diese Darstellung der NFFT-Algorithmen wurde im univariaten Fall erstmals von G. Steidl in [116] und im multivariaten Fall unter anderen vom Autor in [104] präsentiert.

Aus Sicht der linearen Algebra kann die DFT als Matrix-Vektor-Multiplikation mit der Fourier-Matrix dargestellt werden, so dass die FFT als Faktorisierung der Transformationsmatrix in ein Produkt von schwach besetzten Matrizen interpretiert werden kann. In analoger Weise schreiben wir die NDFT als Matrix-Vektor-Multiplikation mit der nichtäquidistanten Fourier-Matrix und leiten die Faktorisierung dieser Matrix in ein Produkt von schwach besetzten Matrizen ab. Dies hat weitreichende Konsequenzen, denn sofort steht auch ein schnelles Verfahren zur Multiplikation mit der transponierten Matrix zur Verfügung. Es muss nur das Matrix-Produkt transponiert und als Algorithmus realisiert werden. Eine genaue Untersuchung des transponierten Problems lässt den Zusammenhang zu sogenannten Gridding-Algorithmen erkennen [91, 114, 67, 111]. Diese Verfahren waren schon vor der NFFT in der Literatur bekannt, jedoch fehlte der exakte Zusammenhang zwischen Komplexität und Genauigkeit. Weitere Hinweise auf frühere Arbeiten befinden sich in [128].

Neben der DFT haben die diskreten trigonometrischen Transformationen eine herausragende Bedeutung, da sie die Grundlage für viele Anwendungen bilden. Nennen lassen sich hier beispielsweise die numerische Analysis (z.B. Lösen von speziellen Integralgleichungen) oder die digitale Signalverarbeitung (z.B. Bilddatenkompression). Wir nehmen wieder die Methoden der Approximationstheorie zu Hilfe, um effiziente Algorithmen für diskrete Kosinus- und Sinus-Transformationen an nichtäquidistanten Daten zu entwickeln.

Da die nichtäquidistanten Transformationen keine orthogonalen Transformationen mehr sind, untersuchen wir am Beispiel der NFFT die „inversen Transformationen“. Die nichtäquidistanten Fourier-Matrizen sind im allgemeinen keine quadratischen Matrizen. Deshalb benutzen wir iterative Gleichungssystemlöser (CGNR, CGNE) zur Berechnung der Kleinste-Quadrate-Lösungen. Wir erläutern den Zusammenhang zu den ACT-Methoden (Adaptive weight, Conjugate gradient acceleration, Toeplitz matrices), die in [39] entwickelt wurden und verdeutlichen die Beziehung zur Approximation von beliebig verteilten Daten mittels trigonometrischer Polynome.

Eine Verallgemeinerung der NFFT ergibt sich, falls nicht nur die Knoten im Zeitbereich, sondern auch die Frequenzen nichtäquidistant gewählt sind.

Schließlich wird gezeigt, dass die NFFT ähnlich gute Eigenschaften bezüglich Rundungsfehlern wie die FFT hat.

Da der breite Einsatz der FFT-Algorithmen auch vom Vorhandensein sehr effizienter Programmpakete abhängt, haben wir die NFFT implementiert und verweisen auf die dazugehörige NFFT-Homepage [76]. Unser allgemeiner Zugang erlaubt insbesondere eine Umsetzung, die den vielfältigen Anwendungen in einfacher Weise angepasst werden kann. So sind mit diesen C-Programmen die NFFT-Algorithmen bezüglich der verschiedenen Ansatzfunktionen φ sehr gut vergleichbar.

In *Kapitel* 2 beschreiben wir effiziente Summationsalgorithmen für radiale Funktionen, d.h., wir entwickeln Verfahren zur schnellen Realisierung von speziellen Matrix-Vektor-Multiplikationen. Eine Hauptaufgabe der numerischen Mathematik besteht in der Approximation möglichst großer Klassen von speziellen Matrizen durch Summen und Produkte von strukturierten Matrizen, die eine schnelle Matrix-Vektor-Multiplikation erlauben. Das Interesse ergibt sich aus vielen praktischen Anwendungen, wie z.B. die Approximation mit radialen Basisfunktionen oder die Diskretisierung von Integralgleichungen. In diesen Fällen entstehen lineare Gleichungssysteme mit Matrizen, die im allgemeinen voll besetzt sind, aber eine gewisse Struktur besitzen. Die zu behandelnden Problemgrößen gestatten in der Regel kein explizites Speichern dieser Matrizen, was zur iterativen Berechnung der Lösung aber auch nicht notwendig ist.

Bekannte Verfahren, die auch schnelle Summationen realisieren, sind z.B. die Multipol-Methoden, die Paneel-Methoden, Wavelet-Methoden, $\mathcal{H}$-Matrix-Methoden oder Mosaik-Skeleton-Approximationen.

Dieses Kapitel beginnt mit der Beschreibung des Summationsalgorithmus für äquidistante Knoten. In Matrix-Vektor-Schreibweise bedeutet die Summation, das Matrix-Vektor-Produkt mit einer Toeplitz-Matrix zu realisieren. Der Standard-Algorithmus zur effizienten Berechnung beruht auf der Einbettung der Toeplitz-Matrix in eine zirkulante Matrix, um diese dann mit der Fourier-Matrix zu diagonalisieren. Unser neuer Summationsalgorithmus benutzt an Stelle der Fourier-Matrix die nichtäquidistante Fourier-Matrix. In einigen Fällen ist zusätzlich ein sogenanntes Nahfeld zu berechnen.

Wir fassen die Vorteile unseres Zugangs wie folgt zusammen:

- Der Algorithmus ergibt sich in natürlicher Weise durch die Verallgemeinerung vom äquidistanten Fall zum nichtäquidistanten Fall.
- Der modulare Aufbau (FFT, NFFT, schnelle Summation) gestattet ein einfaches Verstehen der Algorithmen.
- Für Anwendungen ist der Implementierungsaufwand ein nicht zu unterschätzender Aspekt. Durch die Beschreibung mittels NDFT bestechen unsere Algorithmen mit ihrer einfachen Struktur.

- Ein Anpassen an verschiedene Problemstellungen ist sehr einfach möglich.

Im 3. *Kapitel* wenden wir uns einer Anwendung in der Computertomographie, der Inversion der Radon-Transformation, zu. Der Projektionssatz (auch Fourier-Slice-Theorem genannt) liefert sofort einen einfachen Ansatz für einen effizienten Algorithmus zur tomographischen Rekonstruktion von Radon-transformierten Signalen.

Falls nur die Standard-FFT benutzt wird, muss eine Interpolation der Daten im Fourier-Bereich vom Polar-Gitter zum kartesischen Gitter erfolgen. So eine Approximationsmethode führt leider nicht zu akzeptablen Rekonstruktionsergebnissen. Deshalb wird in heutigen Computertomographen mit der Methode der gefilterten Rückprojektion gearbeitet. Der Rechenaufwand ist in diesem Fall deutlich höher als bei den Fourier-Methoden, er beträgt $\mathcal{O}(N^3)$ gegenüber $\mathcal{O}(N^2 \log N)$ arithmetischen Operationen zur Rekonstruktion von Bildern mit N Zeilen und N Spalten. Wir stellen zwei verschiedene Fourier-Rekonstruktions-Algorithmen vor, die auf der NFFT beruhen und vollständig ohne Interpolation auskommen. In den numerischen Beispielen verdeutlichen wir, dass unsere Algorithmen zu ähnlich guten Rekonstruktionsergebnissen wie die gefilterte Rückprojektion führen. Dazu testen wir die Algorithmen nicht nur an Phantom-Bildern, sondern nutzen auch reale Daten von Tomographiegeräten. Der entscheidende Vorteil unserer Methode ist die geringere arithmetische Komplexität, die wieder in der Effizienz der NFFT begründet ist.

Im 4. *Kapitel* entwickeln wir schnelle Fourier-Algorithmen für nichtäquidistante Daten auf der Einheitskugel $\mathbb{S}^2 \subset \mathbb{R}^3$. Diskrete sphärische Fourier-Transformationen haben z.B. in der Meteorologie oder in den Geowissenschaften eine besondere Bedeutung. Sie werden zur Berechnung der Wettervorhersage oder in Klimamodellen benutzt. Auf Grund der herausragenden Eigenschaft, dass durch sphärische Fourier-Summen alle Gebiete der Kugelfläche gleich gut aufgelöst werden, finden diese Summen Anwendung bei der Lösung von partiellen Differentialgleichungen in sphärischen Geometrien. Die diskrete sphärische Fourier-Transformation realisiert den Wechsel vom Orts- in den Frequenzbereich und umgekehrt. Sie benötigt $\mathcal{O}(N^4)$ Operationen für $\mathcal{O}(N^2)$ Daten. Da ein Großteil der Rechenzeit für diese Transformation verwendet wird, ist eine Reduktion der arithmetischen Komplexität eine Herausforderung. Die diskrete sphärische Fourier-Transformation benötigt auf speziellen Gittern unter Ausnutzung der Tensorproduktstruktur nur $\mathcal{O}(N^3)$ Operationen. J.R. Driscoll und D. Healy entwickelten in [27] schnelle sphärische Fourier-Transformationen auf speziellen Gittern mit einer arithmetischen Komplexität von $\mathcal{O}(N^2 \log^2 N)$. Diese Arbeiten lösten eine Vielzahl von weiteren Untersuchungen und Verbesserungen aus [103, 83, 16, 119, 63]). Wir beschreiben erstmals schnelle approximative Algorithmen, die mit beliebig verteilten Daten auf der Sphäre arbeiten, d.h., wir reduzieren mit Hilfe der NFFT die arithmetische Komplexität von $\mathcal{O}(N^4)$ auf $\mathcal{O}(N^2 \log^2 N)$. Damit können z.B. die sphärischen Fourier-Koeffizienten mit beliebigen Quadraturformeln schnell berechnet werden.

An dieser Stelle danke ich Prof. Dr. J. Prestin für die ausgezeichnete Unterstützung beim Anfertigen dieser Arbeit. Die zahlreichen fachlichen Diskussionen waren mir außerordentlich wertvoll. Meinen Kolleginnen und Kollegen vom Institut für Mathematik danke ich für die angenehme Arbeitsatmosphäre. Mein ganz besonderer Dank gilt Frau Prof. Dr. G. Steidl. Von ihrem Wissen, ihrem Arbeitseifer, ihrem Interesse und ihrer Motivation habe ich in besonderer Weise profitiert. Diese Zusammenarbeit ist mir stets eine Freude und war eine ganz entscheidende Voraussetzung für das Entstehen der Arbeit. Meinem Doktorvater Prof. Dr. M. Tasche gilt mein herzlicher Dank, sein reges Interesse am Fortgang meiner Untersuchungen war ständiger Ansporn für mich.

Ohne die große Unterstützung und unschätzbare Hilfe meiner Frau wäre die Arbeit nie fertig geworden. Deshalb danke ich Ina besonders herzlich.

Bezeichnungen

Folgende Standardbezeichnungen werden unter anderem in dieser Arbeit verwendet. Beispielsweise sind $\mathbb{N}, \mathbb{N}_0, \mathbb{Z}, \mathbb{R}, \mathbb{R}^+$ bzw. $\mathbb{C}$ die Mengen aller natürlichen, nichtnegativen ganzen, ganzen, reellen, positiven reellen bzw. komplexen Zahlen.

$\delta_{j,k}$	Kronecker–Symbol
$\bar{a}$	konjugiert komplexe Zahl von a
$\boldsymbol{a} = \boldsymbol{a}_N := (a_j)_{j=0}^{N-1} \in \mathbb{C}^N$	Spaltenvektor der Länge N
$\boldsymbol{e}_k := (\delta_{j,k})_{j=0}^{N-1} \in \mathbb{R}^N$	k–ter Einheitsvektor
$\boldsymbol{a}^{\mathrm{T}}$	transponierter Vektor von $\boldsymbol{a}$
$\mathbf{1}_d := (1, \ldots, 1)^{\mathrm{T}} \in \mathbb{Z}^d$	Eins-Vektor der Länge d
$\boldsymbol{ab} := \boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$	Skalarprodukt zweier Vektoren
$\boldsymbol{A} := (a_{j,k})_{j,k=0}^{M-1,N-1} \in \mathbb{C}^{M \times N}$	Matrix vom Format (M, N)
$\boldsymbol{I}_N := (\delta_{j,k})_{j,k=0}^{N-1} \in \mathbb{R}^{N \times N}$	Einheitsmatrix
$\boldsymbol{A}^{\mathrm{T}}$	transponierte Matrix von $\boldsymbol{A}$
$\boldsymbol{A}^{\mathrm{H}} = \bar{\boldsymbol{A}}^{\mathrm{T}}$	transponierte, konjugiert komplexe Matrix
$\operatorname{diag} \boldsymbol{a}_N := (a_j\, \delta_{j,k})_{j,k=0}^{N-1} \in \mathbb{R}^{N \times N}$	Diagonalmatrix mit $\boldsymbol{a}_N = (a_j)_{j=0}^{N-1}$
$\lfloor x \rfloor$	größte ganze Zahl kleiner oder gleich $x \in \mathbb{R}$
$\lceil x \rceil$	größte ganze Zahl kleiner $x + 1$
$\lVert \boldsymbol{\alpha} \rVert_p := \begin{cases} \left(\sum\limits_{k=0}^{N-1} \lvert \alpha_k \rvert^p \right)^{1/p} & \text{für } 1 \le p < \infty, \\ \max\limits_{k=0,\ldots,N-1} \lvert \alpha_k \rvert & \text{für } p = \infty \end{cases}$	Norm des Vektors $\boldsymbol{\alpha} = (\alpha_k)_{k=0}^{N-1}$
$\varepsilon_{N,j} := \begin{cases} 1/2 & \text{für } j = 0, N, \\ 1 & \text{für } j = 1, \ldots, N-1 \end{cases}$	Definition von $\varepsilon_{N,j}$
$\chi_A(x) := \begin{cases} 1 & \text{falls } x \in A, \\ 0 & \text{sonst} \end{cases}$	charakteristische Funktion von A
$I_N^d := \left\{ \boldsymbol{k} \in \mathbb{Z}^d : -\frac{N}{2}\, \mathbf{1}_d \le \boldsymbol{k} < \frac{N}{2}\, \mathbf{1}_d \right\}$	Index-Bereich, die Ungleichung gilt komponentenweise

Indexabbildung:

Die Multiindizes bei Vektoren und Matrizen sind wie folgt zu verstehen.
Es seien $d, N \in \mathbb{N}$, dann definieren wir den Vektor

$$(a_{\boldsymbol{j}})_{\boldsymbol{j} \in I_N^d} := (a_n)_{n=0}^{N^d-1} \text{ mit } a_n := a_{\boldsymbol{j}},$$

falls für die Indizes n und $\boldsymbol{j}$ der Zusammenhang

$$n = \left(j_1 + \frac{N}{2}\right) + N\left(j_2 + \frac{N}{2}\right) + \ldots + N^{d-1}\left(j_d + \frac{N}{2}\right) \in \left\{0, \ldots, N^d - 1\right\}$$

gilt. Analog erklären wir eine Matrix durch

$$(a_{\boldsymbol{j},\boldsymbol{k}})_{\boldsymbol{j} \in I_N^d, \boldsymbol{k} \in I_M^l} := (a_{m,n})_{m=0,n=0}^{N^d-1,M^l-1} \text{ mit } a_{m,n} := a_{\boldsymbol{j},\boldsymbol{k}},$$

falls für die Indizes $m, n, \boldsymbol{j}$ und $\boldsymbol{k}$ der Zusammenhang

$$m = \left(j_1 + \frac{N}{2}\right) + N\left(j_2 + \frac{N}{2}\right) + \ldots + N^{d-1}\left(j_d + \frac{N}{2}\right) \in \left\{0, \ldots, N^d - 1\right\},$$

$$n = \left(k_1 + \frac{M}{2}\right) + M\left(k_2 + \frac{M}{2}\right) + \ldots + M^{l-1}\left(k_l + \frac{M}{2}\right) \in \left\{0, \ldots, M^l - 1\right\}$$

gilt.

Notation: $\mathcal{O}$

Zur Beschreibung der arithmetischen Komplexität von Algorithmen benutzen wir, wie in vielen Arbeiten zu ähnlicher Thematik die Landau-Notation, die nicht im strengen Sinne zu verstehen ist. Wir verwenden für große Werte N und M die Schreibweise $\mathcal{O}(\cdot)$ in der Form

$$F(N, M) = \mathcal{O}(\alpha\, G_1(N) + \beta\, G_2(M)),$$

falls wir ein $c \in \mathbb{R}$, unabhängig von den Parametern α und β, angeben können mit

$$\left|\frac{F(N, M)}{\alpha\, G_1(N) + \beta\, G_2(M)}\right| \leq c < \infty \text{ für } N > 1 \text{ und } M > 1\,.$$

Mit dieser Notation sind wir insbesondere in der Lage, die arithmetische Komplexität eines Algorithmus genau in Abhängigkeit von den Parametern α und β zu beschreiben. Wir erlauben uns damit auch die Schreibweise $\mathcal{O}(\alpha N) = \mathcal{O}(N)$.

1 Fourier-Transformationen für nichtäquidistante Daten

In diesem Kapitel stellen wir schnelle Algorithmen zur Berechnung der Fourier-Transformation für beliebige Daten vor.
Es sei $d \in \mathbb{N}$ die vorgegebene Dimension der Problemstellung. Dann definieren wir den d-dimensionalen Torus $\mathbb{T}^d$ durch

$$\mathbb{T}^d := \{\boldsymbol{x} = (x_1, \dots, x_d)^{\mathrm{T}} \in \mathbb{R}^d; -1/2 \le x_t < 1/2, t = 1, \dots, d\}.$$

Im Folgenden leiten wir einen Algorithmus für die schnelle und stabile Auswertung von d-variaten trigonometrischen Polynomen

$$f_j := f(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in I_N^d} \hat{f}_{\boldsymbol{k}}\, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}_j} \qquad (j \in I_M^1) \tag{1.1}$$

und

$$h(\boldsymbol{k}) := \sum_{j \in I_M^1} f_j\, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}_j} \qquad (\boldsymbol{k} \in I_N^d) \tag{1.2}$$

an beliebigen Knoten $\boldsymbol{x}_j \in \mathbb{T}^d$ $(j \in I_M^1)$ her, wobei $\hat{f}_{\boldsymbol{k}} \in \mathbb{C}$ $(\boldsymbol{k} \in I_N^d)$ und $f_j \in \mathbb{C}$ $(j \in I_M^1)$ gegebene Daten sind.

Die Summen (1.1) und (1.2) können als Matrix-Vektor-Multiplikationen interpretiert werden. Dazu führen wir die Bezeichnungen

$$\hat{\boldsymbol{f}} := (\hat{f}_{\boldsymbol{k}})_{\boldsymbol{k} \in I_N^d} \quad \in \mathbb{C}^{N^d} \tag{1.3}$$

und die nichtäquidistante Fourier-Matrix

$$\boldsymbol{A} := \left(\mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}_j / N} \right)_{j \in I_M^1, \boldsymbol{k} \in I_N^d} \quad \in \mathbb{C}^{M \times N^d} \tag{1.4}$$

ein. Damit ist die Auswertung der Summe in (1.1) für $j = -M/2, \dots, M/2 - 1$ äquivalent zur Berechnung des Matrix-Vektor-Produktes $\boldsymbol{A}\hat{\boldsymbol{f}}$. Ein naiver Algorithmus benötigt dafür $\mathcal{O}(N^d M)$ arithmetische Operationen.

Nur für die speziellen äquidistanten Knoten $\boldsymbol{x}_j := -(j_1, \dots, j_d)/N$ $(j_t \in I_N^1, t = 1, \dots, d)$ sind schnelle, exakte Algorithmen der Komplexität $\mathcal{O}(N^d \log N)$ bekannt. In diesem Spezialfall ist die Matrix $\bar{\boldsymbol{A}}$ die Matrix der d-variaten diskreten Fourier-Transformation

$$\boldsymbol{F}_N^d := \left(\mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{j}/N} \right)_{\boldsymbol{j} \in I_N^d, \boldsymbol{k} \in I_N^d} \quad \in \mathbb{C}^{N^d \times N^d}.$$

Eine lineare Abbildung von $\mathbb{C}^{N^d}$ in sich, die jedem Vektor $\hat{\boldsymbol{f}}$ den Vektor $\boldsymbol{F}_N^d \hat{\boldsymbol{f}}$ zuordnet, wird diskrete Fourier-Transformation genannt. Die schnelle Realisierung des Matrix-Vektor-Produktes wird als schnelle Fourier-Transformation (FFT) bezeichnet. Viele Verfahren sind erst durch die Effizienz der FFT von praktischem Interesse, so z.B. Polynommultiplikation, Matrizenmultiplikation, Matrix-Vektor-Multiplikation, Invertierung großer strukturierter Matrizen oder trigonometrische Interpolation auf feinen Gittern.

In einer Vielzahl von Anwendungen wird die Beschränkung auf äquidistante Gitter als Nachteil der FFT aufgeführt.

Ziel dieses Kapitels ist es, diesen Nachteil durch Entwicklung neuer schneller Algorithmen zu beseitigen und die FFT zur NFFT (Nonequispaced FFT) weiterzuentwickeln. Ein wesentlicher Unterschied zur FFT besteht darin, dass die NFFT ein approximativer Algorithmus ist, d.h., der Anwender kann bestimmen, mit welcher Genauigkeit das Ergebnis berechnet werden soll. Die hier vorgeschlagene NFFT zur Berechnung von (1.1) hat eine arithmetische Komplexität von $\mathcal{O}(N^d \log N + m^d M)$, wobei wir mit m die geforderte Exaktheit des Ergebnisses steuern.

Einige Teile dieses Kapitels wurden bereits in [104, 95, 96, 40] veröffentlicht.

Dieses Kapitel ist wie folgt gegliedert:
Im Abschnitt 1.1 entwickeln wir einen einheitlichen Zugang zur NFFT. Wir zeigen, dass die NFFT-Algorithmen durch Approximationen eines trigonometrischen Polynoms durch Translate einer Ansatzfunktion φ hergeleitet werden können. Die Wahl von unterschiedlichen Ansatzfunktionen φ gestattet es, verschiedene Zugänge von A. Dutt und V. Rokhlin [31], G. Beylkin [8] und K. Fourmont [45] zur NFFT einzuordnen und neue effiziente Algorithmen zu entwickeln. Im Abschnitt 1.2 schreiben wir die NFFT-Verfahren in Matrix-Vektor-Form, d.h., wir approximieren die Transformationsmatrix durch ein Produkt von verschiedenen Matrizen, die zu den entsprechenden Schritten der Algorithmen korrespondieren. Im Abschnitt 1.3 stellen wir Fehlerabschätzungen für allgemeine Ansatzfunktionen φ vor, und spezifizieren diese dann in den folgenden Unterabschnitten. Insbesondere verdeutlichen wir durch die spezielle Wahl der Ansatzfunktionen φ die Beziehungen zu den früheren NFFT-Algorithmen. Der genaue Zusammenhang zwischen arithmetischer Komplexität und Approximationsfehler wird im Satz 1.11 zusammengefasst. Eine Verallgemeinerung der NFFT wird in Abschnitt 1.4 behandelt. Dort nehmen wir nicht nur an, dass die Knoten, sondern auch die Frequenzen nichtäquidistant gegeben sind. Neben den Fourier-Algorithmen sind eine Reihe weiterer diskreter trigonometrischer Transformationen von praktischem Interesse. Aus der Herleitung für die NFFT werden wir in Abschnitt 1.5 schnelle, approximative Algorithmen für die diskrete Kosinus-Transformation (Nonequispaced Fast Cosine Transform, NFCT) und für die diskrete Sinus-Transformation (Nonequispaced Fast Sine Transform, NFST) an nichtäquidistanten Knoten folgern. Eine Rundungsfehleranalyse in Abschnitt 1.6 zeigt, dass die NFFT ein robuster Algorithmus bezüglich Rundungsfehlern ist. In Abschnitt 1.7 steht die iterative Rekonstruktion von Funktionen aus nichtäquidistant abgetasteten Daten im Vordergrund. Schließlich enthält Abschnitt 1.8 zahlreiche numerische Testbeispiele, die die theoretischen Aussagen der anderen Abschnitte belegen. Wir beenden dieses Kapitel mit einigen weiteren Bemerkungen sowie Literaturhinweisen und ordnen außerdem weitere Arbeiten zur NFFT ein.

1.1 Schnelle Fourier-Transformationen für nichtäquidistante Daten

Wir beginnen mit der Berechnung der Summe (1.1), d.h., wir suchen eine Methode, das 1-periodische trigonometrische Polynom

$$f(\boldsymbol{x}) := \sum_{\boldsymbol{k}\in I_N^d} \hat{f}_{\boldsymbol{k}}\, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} \qquad (\hat{f}_{\boldsymbol{k}} \in \mathbb{C}) \tag{1.5}$$

an beliebigen Knoten $\boldsymbol{x}_j \in \mathbb{T}^d$ ($j \in I_M^1$) schnell auszuwerten. Dazu soll das Polynom f in (1.5) durch eine Linearkombination von Translaten einer stetigen 1-periodischen Funktion $\tilde{\varphi}$ approximiert werden. Es sei $\varphi \in L^2(\mathbb{R}^d) \cap L^1(\mathbb{R}^d)$ so gegeben, dass die 1-periodisierte Version

$$\tilde{\varphi}(\boldsymbol{x}) := \sum_{\boldsymbol{r}\in\mathbb{Z}^d} \varphi(\boldsymbol{x}+\boldsymbol{r}) \tag{1.6}$$

eine gleichmäßig konvergente Fourier-Reihe hat. Die Funktion $\tilde{\varphi}$ kann damit als Fourier-Reihe

$$\tilde{\varphi}(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^d} c_{\boldsymbol{k}}(\tilde{\varphi})\, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} \tag{1.7}$$

mit den Fourier-Koeffizienten

$$c_{\boldsymbol{k}}(\tilde{\varphi}) := \int_{\mathbb{T}^d} \tilde{\varphi}(\boldsymbol{x})\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}}\, \mathrm{d}\boldsymbol{x} \quad (\boldsymbol{k} \in \mathbb{Z}^d) \tag{1.8}$$

geschrieben werden. Wir erhalten durch Einsetzen der Definition (1.6) der periodisierten Funktion $\tilde{\varphi}$ in die Definition der Fourier-Koeffizienten (1.8) sofort den Zusammenhang mit der (integralen) Fourier-Transformierten $\hat{\varphi}$ der Funktion φ und den Fourier-Koeffizienten $c_{\boldsymbol{k}}(\tilde{\varphi})$ der periodisierten Funktion $\tilde{\varphi}$, d.h., es gilt

$$\hat{\varphi}(\boldsymbol{k}) := \int_{\mathbb{R}^d} \varphi(\boldsymbol{x})\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}}\, \mathrm{d}\boldsymbol{x} = c_{\boldsymbol{k}}(\tilde{\varphi})\,, \tag{1.9}$$

was eingesetzt in (1.6) und (1.7) auch als Poissonsche Summenformel bezeichnet wird. Die Fourier-Koeffizienten der periodisierten Funktion $\tilde{\varphi}$ sind also an der Stelle $\boldsymbol{k}$ abgetastete Funktionswerte der (integralen) Fourier-Transformierten $\hat{\varphi}$. Im Folgenden wird die Summation über eine Index-Menge $I_{\sigma N}^d$ betrachtet. Dazu sei $\sigma > 1$ der Oversamplingfaktor, so dass σN eine natürliche Zahl ist, die später insbesondere die Länge einer DFT beschreibt. Um effiziente Algorithmen zu erhalten, sollte σN z.B. eine Zweierpotenz sein. Dann kann die DFT sehr schnell durch eine FFT realisiert werden. Ziel ist es nun, die Koeffizienten $g_{\boldsymbol{l}} \in \mathbb{C}$ ($\boldsymbol{l} \in I_{\sigma N}^d$) der Linearkombination

$$s_1(\boldsymbol{x}) := \sum_{\boldsymbol{l}\in I_{\sigma N}^d} g_{\boldsymbol{l}}\, \tilde{\varphi}\left(\boldsymbol{x} - \frac{\boldsymbol{l}}{\sigma N}\right) \tag{1.10}$$

so zu bestimmen, dass die Funktion s_1 das trigonometrische Polynom (1.5) annähert. Dazu entwickeln wir die 1-periodische Funktion s_1 in eine Fourier-Reihe und erhalten

$$\begin{aligned} s_1(\boldsymbol{x}) &= \sum_{\boldsymbol{k}\in\mathbb{Z}^d} c_{\boldsymbol{k}}(s_1)\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} \\ &= \sum_{\boldsymbol{k}\in\mathbb{Z}^d} \hat{g}_{\boldsymbol{k}}\, c_{\boldsymbol{k}}(\tilde{\varphi})\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} \\ &= \sum_{\boldsymbol{k}\in I_{\sigma N}^d} \hat{g}_{\boldsymbol{k}}\, c_{\boldsymbol{k}}(\tilde{\varphi})\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} + \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{0\}} \sum_{\boldsymbol{k}\in I_{\sigma N}^d} \hat{g}_{\boldsymbol{k}}\, c_{\boldsymbol{k}+\sigma N\boldsymbol{r}}(\tilde{\varphi})\,\mathrm{e}^{2\pi\mathrm{i}(\boldsymbol{k}+\sigma N\boldsymbol{r})\boldsymbol{x}} \end{aligned} \tag{1.11}$$

mit den diskreten Fourier-Koeffizienten

$$\hat{g}_{\boldsymbol{k}} := \sum_{\boldsymbol{l}\in I_{\sigma N}^d} g_{\boldsymbol{l}}\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{l}/(\sigma N)}\,. \tag{1.12}$$

Falls die Fourier-Koeffizienten $c_{\boldsymbol{k}}(\tilde{\varphi})$ für $\|\boldsymbol{k}\|_\infty \geq \sigma N - \frac{N}{2}$ klein werden und für alle $\boldsymbol{k} \in I_N^d$ die Werte $c_{\boldsymbol{k}}(\tilde{\varphi})$ ungleich Null sind, legt ein Vergleich zwischen dem trigonometrischen Polynom f in (1.5) und der Fourier-Reihe von s_1 in (1.11) die Berechnung von $\hat{g}_{\boldsymbol{k}}$ durch

$$\hat{g}_{\boldsymbol{k}} = \begin{cases} \hat{f}_{\boldsymbol{k}}/c_{\boldsymbol{k}}(\tilde{\varphi}) & \boldsymbol{k} \in I_N^d, \\ 0 & \boldsymbol{k} \in I_{\sigma N}^d \setminus I_N^d \end{cases} \tag{1.13}$$

nahe. Die Koeffizienten $g_{\boldsymbol{l}}$ in der Linearkombination (1.10) berechnen sich dann auf Grund der Invertierbarkeit der Fourier-Matrix aus (1.12) zu

$$g_{\boldsymbol{l}} = (\sigma N)^{-d} \sum_{\boldsymbol{k}\in I_N^d} \hat{g}_{\boldsymbol{k}}\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{l}/(\sigma N)} \qquad (\boldsymbol{l} \in I_{\sigma N}^d). \tag{1.14}$$

Somit ergeben sich diese Koeffizienten $g_{\boldsymbol{l}}$ ($\boldsymbol{l} \in I_{\sigma N}^d$) mit einer d-dimensionalen inversen diskreten Fourier-Transformation der Länge σN.
Setzen wir voraus, dass φ im Ortsbereich gut lokalisiert ist, so kann sie durch eine Funktion ψ, die außerhalb des Quaders $[-m/(\sigma N), m/(\sigma N)]^d$ ($m \ll \sigma N, m \in \mathbb{N}$) verschwindet, gut angenähert werden. Genauer, wir definieren die abgeschnittene Funktion

$$\psi(\boldsymbol{x}) := \begin{cases} \varphi(\boldsymbol{x}) & \text{falls } \boldsymbol{x} \in [-m/(\sigma N), m/(\sigma N)]^d, \\ 0 & \text{sonst,} \end{cases} \tag{1.15}$$

und bilden wieder die 1-periodisierte Funktion $\tilde{\psi}(\boldsymbol{x}) := \sum_{\boldsymbol{r}\in\mathbb{Z}^d} \psi(\boldsymbol{x}+\boldsymbol{r}) \in L^2(\mathbb{T}^d)$. Damit können wir s_1 durch die Funktion

$$\begin{aligned} s(\boldsymbol{x}) &:= \sum_{\boldsymbol{l}\in I_{\sigma N}^d} g_{\boldsymbol{l}}\,\tilde{\psi}\left(\boldsymbol{x} - \frac{\boldsymbol{l}}{\sigma N}\right) \\ &= \sum_{\boldsymbol{l}\in I_{\sigma N,m}(\boldsymbol{x})} g_{\boldsymbol{l}}\,\tilde{\psi}\left(\boldsymbol{x} - \frac{\boldsymbol{l}}{\sigma N}\right) \end{aligned} \tag{1.16}$$

approximieren. Als Index-Bereich $I_{\sigma N,m}(\boldsymbol{x})$ definieren wir wegen der Beschränktheit des Trägers von ψ einfach

$$I_{\sigma N,m}(\boldsymbol{x}) := \{\boldsymbol{l} \in I_{\sigma N}^d : \sigma N\boldsymbol{x} - m\mathbf{1}_d \le \boldsymbol{l} \le \sigma N\boldsymbol{x} + m\mathbf{1}_d\}\,.$$

Für einen festen Knoten $\boldsymbol{x}_j$ enthält die Summe (1.16) höchstens $(2m+1)^d$ von Null verschiedene Summanden.
Wir erhalten schließlich

$$f(\boldsymbol{x}_j) \approx s_1(\boldsymbol{x}_j) \approx s(\boldsymbol{x}_j)$$

und damit eine Möglichkeit, die Summe (1.5) näherungsweise für alle $\boldsymbol{x}_j$ $(j \in I_M^1)$ mit einer Komplexität von $\mathcal{O}(N^d \log N + m^d M)$ zu berechnen.
Dazugehörige Fehlerabschätzungen werden in Abschnitt 1.3 präsentiert. Doch zuvor fassen wir die obige Herleitung in Algorithmus 1.1 zusammen und bezeichnen ihn als NFFT (Nonequispaced Fast Fourier Transform) bzw. mit NFFT$(N^{\otimes d})$, um die Transformationslänge N in d Richtungen zu kennzeichnen.

Algorithmus 1.1 (NFFT)

Eingabe: $N, M \in \mathbb{N}$, $\sigma > 1$, $m \in \mathbb{N}$, $\boldsymbol{x}_j \in \mathbb{T}^d$ $(j \in I_M^1)$, $\hat{f}_{\boldsymbol{k}} \in \mathbb{C}$ $(\boldsymbol{k} \in I_N^d)$.

Vorberechnung: (i) *Berechne die Fourier-Koeffizienten* $c_{\boldsymbol{k}}(\tilde{\varphi})$ $(\boldsymbol{k} \in I_N^d)$ *der 1-periodischen Funktion* $\tilde{\varphi}$, *wobei* $c_{\boldsymbol{k}}(\tilde{\varphi}) \neq 0$ *vorausgesetzt wird.*
(ii) *Berechne die Funktionswerte* $\tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right)$ *für* $j \in I_M^1$ *und* $\boldsymbol{l} \in I_{\sigma N,m}(\boldsymbol{x}_j)$.

1. *Setze* $\hat{g}_{\boldsymbol{k}} := \hat{f}_{\boldsymbol{k}}/c_{\boldsymbol{k}}(\tilde{\varphi})$ *für* $\boldsymbol{k} \in I_N^d$.
2. *Berechne mit Hilfe der d-dimensionalen FFT*$((\sigma N)^{\otimes d})$ *die Daten*

$$g_{\boldsymbol{l}} := (\sigma N)^{-d} \sum_{\boldsymbol{k} \in I_N^d} \hat{g}_{\boldsymbol{k}}\, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{l}/(\sigma N)} \qquad (\boldsymbol{l} \in I_{\sigma N}^d).$$

3. *Setze*

$$s(\boldsymbol{x}_j) := \sum_{\boldsymbol{l} \in I_{\sigma N,m}(\boldsymbol{x}_j)} g_{\boldsymbol{l}} \tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right) \qquad (j \in I_M^1).$$

Ausgabe: $s(\boldsymbol{x}_j)$ $(j \in I_M^1)$ *Näherungswerte für* $f_j = f(\boldsymbol{x}_j)$ *in* (1.1).

Komplexität: $\mathcal{O}(N^d \log N + m^d M)$.

Bemerkung 1.2 In Abschnitt 1.3 werden wir verschiedene Ansatzfunktionen φ und ψ diskutieren. Darunter sind auch Funktionen $\tilde{\psi}$, die sich sehr schnell berechnen lassen, so dass der Vorberechnungsschritt (ii) entfallen kann und damit viel Speicherplatz eingespart wird. In diesem Zusammenhang verweisen wir auf unser Softwarepaket [76], welches durch Setzen einfacher Flags eine Vorberechnung optional ermöglicht (siehe auch [78]). $\square$

Die Berechnung der Summe $h(\boldsymbol{k})$ $(\boldsymbol{k} \in I_N^d)$ in (1.2) soll nun folgen. Dazu wird zunächst die 1-periodische Funktion

$$\tilde{g}(\boldsymbol{x}) := \sum_{j \in I_M^1} f_j \tilde{\varphi}(\boldsymbol{x} + \boldsymbol{x}_j)$$

definiert. Wir berechnen die Fourier-Koeffizienten von $\tilde{g}$ und erhalten mit der Definition der Fourier-Koeffizienten (1.8) und der Definition (1.2) die Identität

$$\begin{aligned} c_{\boldsymbol{k}}(\tilde{g}) &= \int_{\mathbb{T}^d} \tilde{g}(\boldsymbol{x})\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}}\, \mathrm{d}\boldsymbol{x} \quad (\boldsymbol{k} \in \mathbb{Z}^d) \\ &= \sum_{j \in I_M^1} f_j\, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\ c_{\boldsymbol{k}}(\tilde{\varphi}) \\ &= h(\boldsymbol{k}) c_{\boldsymbol{k}}(\tilde{\varphi}). \end{aligned} \tag{1.17}$$

Damit können die gesuchten Werte $h(\boldsymbol{k})$ $(\boldsymbol{k} \in I_N^d)$ berechnet werden, falls $c_{\boldsymbol{k}}(\tilde{\varphi})$ und $c_{\boldsymbol{k}}(\tilde{g})$ $(\boldsymbol{k} \in I_N^d)$ bekannt sind.
Die Fourier-Koeffizienten (1.17) von $\tilde{g}$ bestimmen wir näherungsweise mit Hilfe der Trapez-Regel

$$c_{\boldsymbol{k}}(\tilde{g}) \approx \frac{1}{(\sigma N)^d} \sum_{\boldsymbol{l} \in I_{\sigma N}^d} \sum_{j \in I_M^1} f_j \tilde{\varphi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right) \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{l}/(\sigma N)}\,.$$

Analog obiger Herleitung sei φ im Ortsereich gut lokalisiert, so dass φ durch ψ ersetzt werden kann, genauer die periodisierte Funktion $\tilde{\varphi}$ wird durch die periodisierte Funktion $\tilde{\psi}$ ersetzt.
Wir fassen diese Herleitung in Algorithmus 1.3 zusammen und bezeichnen ihn als NFFT$^{\mathrm{T}}$ (Nonequispaced Fast Fourier Transform (Transposed)) bzw. mit NFFT$^{\mathrm{T}}(N^{\otimes d})$, um die Transformationslänge N in d Richtungen zu kennzeichnen.

Algorithmus 1.3 (NFFT$^{\mathrm{T}}$)

Eingabe: $N \in \mathbb{N}$, $\sigma > 1$, $m \in \mathbb{N}$, $\boldsymbol{x}_j \in \mathbb{T}^d$ $(j \in I_M^1)$, $\tilde{f}_{\boldsymbol{k}} \in \mathbb{C}$ $(\boldsymbol{k} \in I_N^d)$.

Vorberechnung: *(i) Berechne die Fourier-Koeffizienten* $c_{\boldsymbol{k}}(\tilde{\varphi})$ $(\boldsymbol{k} \in I_N^d)$ *der 1-periodischen Funktion* $\tilde{\varphi}$, *wobei* $c_{\boldsymbol{k}}(\tilde{\varphi}) \neq 0$ *vorausgesetzt wird.*
(ii) Berechne die Funktionswerte $\tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right)$ *für* $\boldsymbol{l} \in I_{\sigma N}^d$ *und* $j \in I_{\sigma N,m}^{\mathrm{T}}(\boldsymbol{l})$, *wobei* $I_{\sigma N,m}^{\mathrm{T}}(\boldsymbol{l}) := \{j \in I_M^1 : \boldsymbol{l} - m\mathbf{1}_d \leq \sigma N \boldsymbol{x}_j \leq \boldsymbol{l} + m\mathbf{1}_d\}$.

1. Setze

$$\hat{g}_{\boldsymbol{l}} := \sum_{j \in I_{\sigma N,m}^{\mathrm{T}}(\boldsymbol{l})} f_j \tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right) \qquad (\boldsymbol{l} \in I_{\sigma N}^d).$$

2. Berechne mit Hilfe der d-dimensionalen FFT die Daten

$$\tilde{c}_{\boldsymbol{k}}(\tilde{g}) := (\sigma N)^{-d} \sum_{\boldsymbol{l} \in I_{\sigma N}^d} \hat{g}_{\boldsymbol{l}}\, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{l}/(\sigma N)} \qquad (\boldsymbol{k} \in I_N^d).$$

3. Setze $\tilde{h}(\boldsymbol{k}) := \tilde{c}_{\boldsymbol{k}}(\tilde{g})/c_{\boldsymbol{k}}(\tilde{\varphi})$ *für* $\boldsymbol{k} \in I_N^d$.

Ausgabe: $\tilde{h}(\boldsymbol{k})$ *Näherungswerte für* $h(\boldsymbol{k})$ $(\boldsymbol{k} \in I_N^d)$.

Komplexität: $\mathcal{O}(N^d \log N + m^d M)$.

1.2 Die Algorithmen in der Matrix-Vektor-Darstellung

In vielen Fällen ist es hilfreich, Algorithmen in der Matrix-Vektor-Notation darzustellen. Es zeigt sich, dass sich die Algorithmen 1.1 und 1.3 nur durch das Transponieren des Matrix-Vektor-Produktes unterscheiden. Mit den Bezeichnungen (1.3) und (1.4) ist die Berechnung von $f(\boldsymbol{x})$ an den Knoten $\boldsymbol{x}_j$ $(j \in I_M^1)$ äquivalent zur Berechnung des Matrix-Vektor-Produktes $\boldsymbol{A}\hat{\boldsymbol{f}}$.
Im Folgenden zeigen wir, dass der Algorithmus 1.1 als näherungsweise Faktorisierung der Matrix $\boldsymbol{A}$ durch das Produkt der strukturierten Matrizen

$$\boldsymbol{B}\boldsymbol{F}_{\sigma N,N}^d\boldsymbol{D} \tag{1.18}$$

interpretiert werden kann. Jede der drei Matrizen korrespondiert zu einem Schritt im Algorithmus 1.1:
1. $\boldsymbol{D} \in \mathbb{C}^{N^d \times N^d}$ ist eine Diagonalmatrix

$$\boldsymbol{D} := \operatorname{diag}\left(\frac{1}{c_{\boldsymbol{k}}(\tilde{\varphi})}\right)_{\boldsymbol{k}\in I_N^d}. \tag{1.19}$$

2. $\boldsymbol{F}_{\sigma N,N}^d \in \mathbb{C}^{(\sigma N)^d \times N^d}$ beschreibt die σN-te d-variate, abgeschnittene Fourier-Matrix

$$\begin{aligned} \boldsymbol{F}_{\sigma N,N}^d \quad &:= \quad \frac{1}{(\sigma N)^d}\left(\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{l}/(\sigma N)}\right)_{\boldsymbol{l}\in I_{\sigma N}^d,\boldsymbol{k}\in I_N^d} \\ &= \quad \underbrace{\boldsymbol{F}_{\sigma N,N}^1 \otimes \ldots \otimes \boldsymbol{F}_{\sigma N,N}^1}_{d\text{-mal}}, \end{aligned} \tag{1.20}$$

die wir als Tensorprodukt von d einfachen, abgeschnittenen Fourier-Matrizen

$$\boldsymbol{F}_{\sigma N,N}^1 = \frac{1}{\sigma N}\left(\mathrm{e}^{2\pi\mathrm{i}kl/(\sigma N)}\right)_{l\in I_{\sigma N}^1,k\in I_N^1}$$

erhalten.
3. $\boldsymbol{B} \in \mathbb{R}^{M\times(\sigma N)^d}$ ist die dünn besetzte, multilevel Bandmatrix

$$\boldsymbol{B} := \left(\tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right)\right)_{j\in I_M^1,\boldsymbol{l}\in I_{\sigma N}^d}. \tag{1.21}$$

Es ist offensichtlich, dass wir die Werte $h(\boldsymbol{k})$ $(\boldsymbol{k} \in I_N^d)$ in (1.2) einfach als Matrix-Vektor-Multiplikation mit der transponierten Matrix von $\boldsymbol{A}$ erhalten können, d.h., es gilt

$$(h(\boldsymbol{k}))_{\boldsymbol{k}\in I_N^d} = \boldsymbol{A}^{\mathrm{T}}(f_j)_{j\in I_M^1}.$$

Somit können die Daten $h(\boldsymbol{k})$ ($\boldsymbol{k} \in I_N^d$) näherungsweise durch Transponieren der Faktorisierung (1.18), also durch

$$(h(\boldsymbol{k}))_{\boldsymbol{k}\in I_N^d} \approx \boldsymbol{D}^{\mathrm{T}}(\boldsymbol{F}_{\sigma N,N}^d)^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}(f_j)_{j\in I_M^1}$$

berechnet werden.
Ein Vergleich mit Algorithmus 1.3 zeigt, dass diese Faktorisierung genau die drei Schritte des Algorithmus beschreibt. Mit den Algorithmen 1.1 und 1.3 haben wir zwei approximative Verfahren hergeleitet, aber noch nichts über die Approximationsfehler ausgesagt. Dies soll im folgenden Abschnitt geschehen.

1.3 Fehlerabschätzungen

Im Gegensatz zur FFT ist die NFFT ein approximativer Algorithmus. Deshalb wird der genaue Zusammenhang zwischen Exaktheit des berechneten Ergebnisses und arithmetischer Komplexität des Algorithmus untersucht. In diesem Abschnitt zeigen wir zunächst allgemeine Fehlerabschätzungen, die anschließend für spezielle Funktionen φ präzisiert werden. Wir können uns im Folgenden auf die NFFT konzentrieren, da die NFFT$^{\mathrm{T}}$ den gleichen Approximationsfehler liefert. Dies wird insbesondere durch die Matrixfaktorisierung in Abschnitt 1.2 deutlich.
Den Approximationsfehler des Algorithmus 1.1

$$E(\boldsymbol{x}_j) := |f(\boldsymbol{x}_j) - s(\boldsymbol{x}_j)| \tag{1.22}$$

zerlegen wir in den Überlappungsfehler (aliasing error)

$$E_{\mathrm{a}}(\boldsymbol{x}_j) := |f(\boldsymbol{x}_j) - s_1(\boldsymbol{x}_j)|$$

und den Abschneidefehler (truncation error)

$$E_{\mathrm{t}}(\boldsymbol{x}_j) := |s_1(\boldsymbol{x}_j) - s(\boldsymbol{x}_j)|,$$

so dass $E(\boldsymbol{x}_j) \le E_{\mathrm{a}}(\boldsymbol{x}_j) + E_{\mathrm{t}}(\boldsymbol{x}_j)$ gilt. Der Überlappungsfehler, der durch das Abschneiden im Frequenzbereich entsteht, kann wegen (1.5), (1.9), (1.11), (1.12) und (1.13) durch

$$\begin{aligned}
E_{\mathrm{a}}(\boldsymbol{x}_j) &= \left| \sum_{\boldsymbol{k}\in I_{\sigma N}^d} \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{\boldsymbol{0}\}} \hat{g}_{\boldsymbol{k}} \hat{\varphi}(\boldsymbol{k}+\sigma N\boldsymbol{r})\, \mathrm{e}^{2\pi\mathrm{i}(\boldsymbol{k}+\sigma N\boldsymbol{r})\boldsymbol{x}_j} \right| \\
&\le \sum_{\boldsymbol{k}\in I_N^d} |\hat{f}_{\boldsymbol{k}}| \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{\boldsymbol{0}\}} \left| \frac{\hat{\varphi}(\boldsymbol{k}+\sigma N\boldsymbol{r})}{\hat{\varphi}(\boldsymbol{k})} \right| \\
&\le \|\hat{\boldsymbol{f}}\|_1 \max_{\boldsymbol{k}\in I_N^d} \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{\boldsymbol{0}\}} \left| \frac{\hat{\varphi}(\boldsymbol{k}+\sigma N\boldsymbol{r})}{\hat{\varphi}(\boldsymbol{k})} \right| \qquad (1.23)
\end{aligned}$$

abgeschätzt werden.
Der Abschneidefehler E_{t} entsteht durch das Abschneiden von φ im Ortsbereich und

hängt damit von der Lokalisierung von φ im Ortsbereich ab. Es ergibt sich wegen der Definition von s_1 in (1.10) und der Definition von s in (1.16)

$$E_{\mathrm{t}}(\boldsymbol{x}_j) = \left| \sum_{\boldsymbol{l} \in \boldsymbol{I}_{\sigma N}^d} g_{\boldsymbol{l}} \left(\tilde{\varphi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right) - \tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right)\right)\right|.$$

Beachten wir nun den Zusammenhang (1.9), die Berechnung von $\hat{g}_{\boldsymbol{k}}$ in (1.13) und die Identität

$$g_{\boldsymbol{l}} \quad = \quad \frac{1}{(\sigma N)^d} \sum_{\boldsymbol{k} \in I_N^d} \frac{\hat{f}_{\boldsymbol{k}}}{\hat{\varphi}(\boldsymbol{k})} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{l}/(\sigma N)}, \tag{1.24}$$

so folgt

$$E_{\mathrm{t}}(\boldsymbol{x}_j) \quad \leq \quad \frac{1}{(\sigma N)^d} \left| \sum_{\boldsymbol{l} \in \boldsymbol{I}_{\sigma N}^d} \sum_{\boldsymbol{k} \in I_N^d} \frac{\hat{f}_{\boldsymbol{k}}}{\hat{\varphi}(\boldsymbol{k})} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{l}/(\sigma N)} \left(\tilde{\varphi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right) - \tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right)\right)\right|$$

und weiter

$$\begin{aligned} E_{\mathrm{t}}(\boldsymbol{x}_j) \quad &\leq \quad \frac{1}{(\sigma N)^d} \left| \sum_{\boldsymbol{k} \in I_N^d} \frac{\hat{f}_{\boldsymbol{k}}}{\hat{\varphi}(\boldsymbol{k})} \sum_{\boldsymbol{l} \in \boldsymbol{I}_{\sigma N}^d} \left(\tilde{\varphi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right) - \tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right)\right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{l}/(\sigma N)} \right| \\ &\leq \quad \frac{\|\hat{\boldsymbol{f}}\|_1}{(\sigma N)^d} \max_{\boldsymbol{k} \in \boldsymbol{I}_N^d} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|} \left| \sum_{\boldsymbol{l} \in \boldsymbol{I}_{\sigma N}^d} \left(\tilde{\varphi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right) - \tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right)\right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{l}/(\sigma N)} \right| . \end{aligned} \tag{1.25}$$

Die Summe über $\boldsymbol{l}$ kann weiter vereinfacht werden. Dazu benutzen wir die Definitionen von $\tilde{\varphi}$ in (1.6), ψ in (1.15), die Definition von $\tilde{\psi}$ und erhalten

$$\tilde{\varphi}(\boldsymbol{x}) - \tilde{\psi}(\boldsymbol{x}) = \sum_{\boldsymbol{r} \in \mathbb{Z}^d} \varphi(\boldsymbol{x} + \boldsymbol{r}) - \varphi(\boldsymbol{x} + \boldsymbol{r}) \; \chi_{[-\frac{m}{\sigma N}, \frac{m}{\sigma N}]^d}(\boldsymbol{x} + \boldsymbol{r}).$$

Für die Summe (1.25) ergibt

$$\begin{aligned} &\sum_{\boldsymbol{l} \in \boldsymbol{I}_{\sigma N}^d} \left(\tilde{\varphi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right) - \tilde{\psi}\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N}\right)\right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{l}/(\sigma N)} \\ = &\sum_{\boldsymbol{l} \in \boldsymbol{I}_{\sigma N}^d} \left(\sum_{\boldsymbol{r} \in \mathbb{Z}^d} \varphi\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N} + \boldsymbol{r}\right) \right. \\ &\left. -\varphi\left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N} + \boldsymbol{r}\right) \; \chi_{[-\frac{m}{\sigma N}, \frac{m}{\sigma N}]^d} \left(\boldsymbol{x}_j - \frac{\boldsymbol{l}}{\sigma N} + \boldsymbol{r}\right)\right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{l}/(\sigma N)} \\ \leq &\sum_{\boldsymbol{r} \in \mathbb{Z}^d} \left(\varphi\left(\boldsymbol{x}_j + \frac{\boldsymbol{r}}{\sigma N}\right) - \varphi\left(\boldsymbol{x}_j + \frac{\boldsymbol{r}}{\sigma N}\right) \; \chi_{[-\frac{m}{\sigma N}, \frac{m}{\sigma N}]^d} \left(\boldsymbol{x}_j + \frac{\boldsymbol{r}}{\sigma N}\right)\right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{r}/(\sigma N)} \\ = &\sum_{\left\|\boldsymbol{x}_j + \frac{\boldsymbol{r}}{\sigma N}\right\|_\infty \geq \frac{m}{\sigma N}} \varphi\left(\boldsymbol{x}_j + \frac{\boldsymbol{r}}{\sigma N}\right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{r}/(\sigma N)} \end{aligned}$$

und schließlich für den Abschneidefehler (1.25) die Ungleichung

$$\begin{aligned} E_{\mathrm{t}}(\boldsymbol{x}_j) &\leq \frac{\|\hat{\boldsymbol{f}}\|_1}{(\sigma N)^d} \max_{\boldsymbol{k}\in \boldsymbol{I}_N^d} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|} \left| \sum_{\left\|\boldsymbol{x}_j+\frac{\boldsymbol{r}}{\sigma N}\right\|_\infty \geq \frac{m}{\sigma N}} \varphi\left(\boldsymbol{x}_j + \frac{\boldsymbol{r}}{\sigma N}\right) \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{r}/(\sigma N)} \right| && (1.26) \\ &\leq \frac{\|\hat{\boldsymbol{f}}\|_1}{(\sigma N)^d} \max_{\boldsymbol{k}\in \boldsymbol{I}_N^d} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|} \sum_{\left\|\boldsymbol{x}_j+\frac{\boldsymbol{r}}{\sigma N}\right\|_\infty \geq \frac{m}{\sigma N}} \left|\varphi\left(\boldsymbol{x}_j + \frac{\boldsymbol{r}}{\sigma N}\right)\right| . && (1.27) \end{aligned}$$

Damit haben wir allgemeine Fehlerabschätzungen für die Ansatzfunktion φ angegeben.

In den folgenden Teilabschnitten werden spezielle Ansatzfunktionen φ gewählt und die Fehlerabschätzungen präzisiert.
Um den Approximationsfehler E in (1.22) möglichst klein zu halten, müssen wir den Überlappungsfehler E_{a}, welcher von der Frequenzlokalisation von φ abhängt, und den Abschneidefehler E_{t}, welcher von der Ortslokalisation abhängt, kontrollieren. Wir werden uns im Folgenden deshalb mit Funktionen, die im Ortsbereich optimal lokalisiert sind, d.h. $E_{\mathrm{t}} = 0$, die im Frequenzbereich optimal lokalisiert sind, d.h. $E_{\mathrm{a}} = 0$ und mit Funktionen für die $E_{\mathrm{t}} \approx E_{\mathrm{a}}$ gilt, beschäftigen.

Die Definitionen für φ bzw. ψ im multivariaten Fall erfolgen durch ein Tensorprodukt von univariaten Funktionen, welche wir auch wieder mit φ bzw. ψ bezeichnen.
Wir definieren

$$\varphi : \mathbb{R}^d \to \mathbb{R} \quad \text{durch} \quad \varphi(\boldsymbol{x}) := \prod_{t=1}^{d} \varphi(x_t)\,,$$

wobei wie üblich der Vektor $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^{\mathrm{T}}$ ist. Analog definieren wir $\psi : \mathbb{R}^d \to \mathbb{R}$ als Tensorprodukt von Funktionen $\psi : \mathbb{R} \to \mathbb{R}$. Es ist klar, dass sich die Fourier-Koeffizienten der multivariaten Funktionen φ einfach als Produkt der Fourier-Koeffizienten der univariaten Funktionen φ ergeben. Diese Beziehung folgt aus der Definition (1.9), denn es gilt

$$\hat{\varphi}(\boldsymbol{k}) = \prod_{t=1}^{d} \hat{\varphi}(k_t) \text{ mit } \boldsymbol{k} := (k_1, \ldots, k_d)^{\mathrm{T}}.$$

Wir unterscheiden in der Bezeichnung von φ und ψ nicht bezüglich der Raumdimension d, da dies stets aus dem Zusammenhang deutlich wird.
In den drei folgenden Unterabschnitten werden wir uns auf Fehlerabschätzungen im eindimensionalen Fall beschränken. Für multivariate Fehlerabschätzungen verweisen wir auf [34, 29]. Im Abschnitt 1.8 werden verschiedene numerische Tests durchgeführt. Wir vergleichen dort die verschiedenen Ansatzfunktionen φ bezüglich der Approximationsfehler und der Laufzeiten der Algorithmen (siehe Beispiele 1.1 und 1.2).

1.3.1 Ansatz mit B-Splines (Zeitbeschränkte Funktionen)

In diesem Abschnitt wollen wir Funktionen φ betrachten, für die der Abschneidefehler E_{t} verschwindet. Im Zusammenhang mit der NFFT wurden solche Funktionen erstmals

in [8] betrachtet. Die Abschätzungen in diesem Abschnitt sind im wesentlichen aus [116] übernommen. Es sei $0 < m < N$ eine gegebene natürliche Zahl.
Wir führen die zentrierten kardinalen B-Splines über die Rekursionsbeziehung

$$\begin{aligned} M_1(x) &:= \begin{cases} 1 & \text{für} \quad x \in [-1/2, 1/2), \\ 0 & \text{sonst}, \end{cases} \\ M_{m+1}(x) &:= \int\limits_{-1/2}^{1/2} M_m(x-t)\,\mathrm{d}t \qquad m = 1, 2, \ldots \end{aligned} \tag{1.28}$$

ein. Als Träger der B-Splines ergibt sich $\operatorname{supp} M_{2m} = [-m, m]$. Die Fourier-Transformierte von M_1 kann mit Hilfe der sinc-Funktion

$$\operatorname{sinc} x := \begin{cases} 1 & \text{für} \quad x = 0, \\ \dfrac{\sin x}{x} & \text{sonst} \end{cases}$$

in der Form

$$\hat{M}_1(v) = \int\limits_{-1/2}^{1/2} \mathrm{e}^{-2\pi \mathrm{i} v x}\,\mathrm{d}x = \operatorname{sinc}(\pi v) \tag{1.29}$$

geschrieben werden. Wir benutzen den Faltungssatz, um die Fourier-Transformation der Funktion M_m zu berechnen. Unter Berücksichtigung der Definition (1.28) ist

$$(M_m * M_1)(x) := \int\limits_{\mathbb{R}} M_m(x-t) M_1(t)\,\mathrm{d}t = M_{m+1}(x)$$

und mit der Faltungseigenschaft der Fourier-Transformation ergibt sich

$$\hat{M}_m(k) = (\operatorname{sinc}(\pi k))^m \qquad (m \in \mathbb{N}). \tag{1.30}$$

Mit diesen Vorbemerkungen sind wir in der Lage, Beispiele für Funktionen φ und ψ aus Abschnitt 1.1 mit Hilfe von B-Splines zu definieren.
Dazu sei φ mittels der skalierten, zentrierten, kardinalen B-Splines der Ordnung $2m$ durch die Formel

$$\varphi(x) := M_{2m}(\sigma N x) \qquad (x \in \mathbb{R}) \tag{1.31}$$

gegeben. Damit ist φ eine stetige Funktion mit beschränktem Träger $\operatorname{supp} \varphi \subseteq [-\frac{2m}{\sigma N}, \frac{2m}{\sigma N}]$. Mit der Definition $\psi := \varphi$ ergibt sich sofort, dass in (1.25) der Abschneidefehler E_{t} verschwindet. Um auch den Überlappungsfehler E_{a} abzuschätzen, formulieren wir zunächst folgendes Lemma (vgl. [116, 104]).

Lemma 1.4 *Für $0 < u < 1$ und $m \in \mathbb{N}$ gilt die Ungleichung*

$$\sum_{r\in\mathbb{Z}\setminus\{0\}} \left(\frac{u}{u+r}\right)^{2m} < \frac{4m}{2m-1} \left(\frac{u}{u-1}\right)^{2m} .$$

Beweis: Für $r \geq 0$ gelten wegen der Ungleichung

$$\left(\frac{u}{u+r}\right)^{2m} \leq \left(\frac{u}{u-r}\right)^{2m}$$

die Abschätzungen

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

Dies liefert die Behauptung des Lemmas. ■

Wir formuliern folgenden Satz (vgl. [116]).

Satz 1.5 *Zur Berechnung der Summe* (1.1) *durch den NFFT-Algorithmus 1.1 sei für den univariaten Fall $d = 1$ die Ansatzfunktion $\varphi = \psi$ in* (1.31) *gegeben. Dann kann der maximale Approximationsfehler für alle Vektoren $\hat{\boldsymbol{f}} := (\hat{f}_k)_{k\in I_N^1}$ durch*

$$E_\infty := \max_{j\in I_M^1} E(x_j) \leq \|\hat{\boldsymbol{f}}\|_1 \frac{4m}{2m-1} \left(\frac{1}{2\sigma-1}\right)^{2m}$$

abgeschätzt werden.

Beweis: Da φ einen kompakten Träger hat und wir $\psi = \varphi$ gesetzt haben, verschwindet der Abschneidefehler E_t in (1.25). Um den Überlappungsfehler E_a in (1.23) abzuschätzen, betrachten wir die Fourier-Koeffizienten von φ. Diese ergeben sich nach der Definition (1.9) unter Beachtung von (1.30) und (1.31) zu

$$\begin{aligned}
\hat{\varphi}(k) = c_k(\tilde{\varphi}) &= \int_{\mathbb{R}} \varphi(x)\, \mathrm{e}^{-2\pi\mathrm{i}kx}\, \mathrm{d}x \\
&= \int_{\mathbb{R}} M_{2m}(\sigma N x)\, \mathrm{e}^{2\pi\mathrm{i}kx}\, \mathrm{d}x \\
&= \frac{1}{\sigma N} \left(\operatorname{sinc}\frac{\pi k}{\sigma N}\right)^{2m} .
\end{aligned}$$

Damit folgt dann

$$\begin{aligned}
\sigma N \hat{\varphi}(k + r\sigma N) &= \left(\frac{\sin(k\pi/(\sigma N))}{k\pi/(\sigma N) + r\pi} \right)^{2m} \\
&= \left(\frac{\sin(k\pi/(\sigma N))}{k\pi/(\sigma N)} \right)^{2m} \left(\frac{k\pi/(\sigma N)}{k\pi/(\sigma N) + r\pi} \right)^{2m} \\
&= \sigma N \hat{\varphi}(k) \left(\frac{k/(\sigma N)}{k/(\sigma N) + r} \right)^{2m}
\end{aligned}$$

und wir erhalten mit (1.23) die Ungleichung

$$E_\infty \leq \|\hat{\boldsymbol{f}}\|_1 \, \frac{4m}{2m-1} \sum_{\substack{k=-N/2 \\ k \neq 0}}^{N/2-1} \frac{(k/(\sigma N))^{2m}}{(k/(\sigma N) - 1)^{2m}}.$$

Da die Funktion $u^{2m}/(u-1)^{2m}$ monoton wachsend für $u \in [0, 1/2]$ und $|k| \leq N/2$ ist, folgt die Behauptung. ■

Bemerkung 1.6 In [104] haben wir Fehlerabschätzungen für diskrete Sobolev-Normen hergeleitet. Dazu sind Informationen zum Abfall der Fourier-Koeffizienten $\hat{f}_k$ nötig. Dann können die Fehlerabschätzungen und damit die Laufzeiten der Algorithmen verbessert werden. Am Beispiel von diskreten trigonometrischen Transformationen für nichtäquidistante Daten (siehe Abschnitt 1.5) erfolgte die Untersuchung in [95, Example 4.1]. □

1.3.2 Ansatz mit Gauß-Funktionen

In diesem Abschnitt geben wir Fehlerabschätzungen für den Approximationsfehler (1.22), die sich aus den Algorithmen aus Abschnitt 1.1 beim Ansatz von Gauß-Funktionen ergeben. Im Zusammenhang mit der NFFT wurden solche Fehlerabschätzungen erstmals in [31] betrachtet und mit unserem allgemeinen Ansatz in [116, 104] wie im Folgenden verbessert. Im univariaten Fall werden die Gauß-Funktionen φ durch

$$\varphi(x) := \frac{1}{\sqrt{\pi b}} \, \mathrm{e}^{-(\sigma N x)^2/b} \quad (b \in \mathbb{R}^+) \tag{1.32}$$

definiert. Nutzen wir die Fourier-Transformierte der normierten Gauß-Funktion e^{-x^2} (siehe z.B. [23, Example 2.6])

$$\int_{\mathbb{R}} \mathrm{e}^{-x^2} \, \mathrm{e}^{-2\pi \mathrm{i} k x} \, \mathrm{d}x = \sqrt{\pi} \, \mathrm{e}^{-\pi^2 k^2} \, , \tag{1.33}$$

so folgt mit der Definition von (1.9) die Fourier-Transformierte von (1.32)

$$c_k(\tilde{\varphi}) = \hat{\varphi}(k) = \frac{1}{\sigma N} \, \mathrm{e}^{-\left(\frac{\pi k}{\sigma N}\right)^2 b}. \tag{1.34}$$

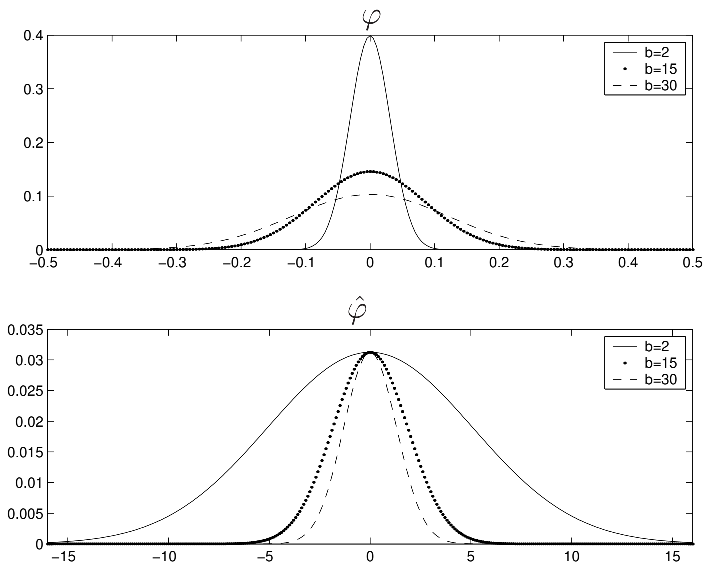

Abbildung 1.1: Gauß-Funktionen φ in (1.32) (oben) und $\hat{\varphi}$ (unten) für $\sigma N = 32$ und verschiedene Parameter $b \in \{2, 15, 30\}$.

An dieser Stelle tritt natürlich die Frage auf, wie wir den Parameter b wählen müssen. Es ist klar, dass wir mit diesem Parameter die Orts- und Frequenzlokalität steuern können (siehe Abbildung 1.1).
Algorithmen zur NFFT, die die Gauß-Funktionen benutzen, wurden erstmals in [31] angegeben, wobei dort der Parameter b durch numerische Untersuchungen bestimmt wurde. Wir wählen b so, dass die Fehler E_{t} und E_{a} von der gleichen Größenordung sind. Diese Wahl folgt aus den Fehlerabschätzungen von Satz 1.7. In [116] ist belegt, dass diese theoretischen Untersuchungen schnellere bzw. genauere Algorithmen zur Folge haben (siehe auch [29]). Wir beweisen die Fehlerabschätzung für die Gauß-Funktionen im folgenden Satz.

Satz 1.7 *Zur Berechnung der Summe* (1.1) *durch den NFFT-Algorithmus* 1.1 *sei für den univariaten Fall* $d = 1$ *die Ansatzfunktion* φ *in* (1.32) *gegeben. Es gelte* $\sigma \geq 3/2$ *und* $b := \frac{2\sigma}{2\sigma-1}\frac{m}{\pi}$. *Dann kann der maximale Approximationsfehler für alle Vektoren* $\hat{\boldsymbol{f}} := (\hat{f}_k)_{k \in I_N^1}$ *durch*

$$E_\infty = \max_{j \in I_M^1} E(x_j) \leq 4 \|\hat{\boldsymbol{f}}\|_1 \, \mathrm{e}^{-m\pi\left(1 - \frac{1}{2\sigma-1}\right)}$$

abgeschätzt werden.

Beweis: Für positive Zahlen a und c wird die Abschätzung

$$\int\limits_a^\infty \mathrm{e}^{-cx^2}\,\mathrm{d}x = \int\limits_0^\infty \mathrm{e}^{-c(x+a)^2}\,\mathrm{d}x \leq \mathrm{e}^{-ca^2} \int\limits_0^\infty \mathrm{e}^{-2acx}\,\mathrm{d}x = \frac{\mathrm{e}^{-ca^2}}{2ac} \tag{1.35}$$

benutzt. Wir betrachten zunächst den Überlappungsfehler E_a. Mit der Fourier-Transformierten $\hat{\varphi}$ der Gauß-Funktion (1.34) ergibt sich in (1.23)

$$E_\mathrm{a}(x_j) \le \|\hat{\boldsymbol{f}}\|_1 \max_{k\in I_N^1} \sum_{r\in\mathbb{Z}\setminus\{0\}} \mathrm{e}^{-b\pi^2\left(\frac{2k}{\sigma N}r+r^2\right)}.$$

Da $\mathrm{e}^u + \mathrm{e}^{-u}$ monoton wachsend für $u \ge 0$ und monoton fallend für $u \le 0$ ist, folgt

$$\begin{aligned}
E_\mathrm{a}(x_j) &\le \|\hat{\boldsymbol{f}}\|_1 \sum_{r=1}^{\infty} \left(\mathrm{e}^{-b\pi^2\left(r^2-\frac{N}{\sigma N}r\right)} + \mathrm{e}^{-b\pi^2\left(r^2+\frac{N}{\sigma N}r\right)}\right) \\
&\le ||\hat{\boldsymbol{f}}||_1 \left(\mathrm{e}^{-b\pi^2\left(1-\frac{1}{\sigma}\right)}\left(1+\mathrm{e}^{-\frac{2b\pi^2}{\sigma}}\right) + \mathrm{e}^{b\left(\frac{\pi}{2\sigma}\right)^2} \sum_{r=2}^{\infty}\left(\mathrm{e}^{-b\pi^2\left(r-\frac{1}{2\sigma}\right)^2} + \mathrm{e}^{-b\pi^2\left(r+\frac{1}{2\sigma}\right)^2}\right)\right) \\
&\le ||\hat{\boldsymbol{f}}||_1 \left(\mathrm{e}^{-b\pi^2\left(1-\frac{1}{\sigma}\right)}\left(1+\mathrm{e}^{-\frac{2b\pi^2}{\sigma}}\right) + \mathrm{e}^{b\left(\frac{\pi}{2\sigma}\right)^2} \int_1^{\infty} \mathrm{e}^{-b\pi^2\left(x-\frac{1}{2\sigma}\right)^2} + \mathrm{e}^{-b\pi^2\left(x+\frac{1}{2\sigma}\right)^2}\,\mathrm{d}x\right)
\end{aligned}$$

und schließlich mit Hilfe der Formel (1.35)

$$E_\mathrm{a}(x_j) \le ||\hat{\boldsymbol{f}}||_1\, \mathrm{e}^{-b\pi^2\left(1-\frac{1}{\sigma}\right)}\left(1+\frac{\sigma}{(2\sigma-1)b\pi^2} + \mathrm{e}^{-2b\pi^2/\sigma}\left(1+\frac{\sigma}{(2\sigma+1)b\pi^2}\right)\right).$$

Um den Abschneidefehler E_t abzuschätzen, substituieren wir (1.32) und (1.34) in (1.27) und erhalten unter Berücksichtigung der Definition (1.15) von ψ die Ungleichung

$$\begin{aligned}
E_\mathrm{t}(x_j) &\le ||\hat{\boldsymbol{f}}||_1 \frac{2}{\sqrt{\pi b}}\, \mathrm{e}^{b\left(\frac{\pi}{2\sigma}\right)^2} \sum_{r=m}^{\infty} \mathrm{e}^{-r^2/b} \\
&\le ||\hat{\boldsymbol{f}}||_1 \frac{2}{\sqrt{\pi b}}\, \mathrm{e}^{b\left(\frac{\pi}{2\sigma}\right)^2}\left(\mathrm{e}^{-m^2/b} + \int_m^{\infty} \mathrm{e}^{-x^2/b}\,\mathrm{d}x\right).
\end{aligned}$$

Mit (1.35) ergibt sich

$$E_\mathrm{t}(x_j) \le ||\hat{\boldsymbol{f}}||_1 \frac{2}{\sqrt{\pi b}}\left(1+\frac{b}{2m}\right)\mathrm{e}^{-b\pi^2\left(\left(\frac{m}{b\pi}\right)^2-\left(\frac{1}{2\sigma}\right)^2\right)}.$$

Da wir $b = \frac{2\sigma}{2\sigma-1}\frac{m}{\pi}$ gewählt haben, ist

$$\left(\frac{m}{b\pi}\right)^2 - \left(\frac{1}{2\sigma}\right)^2 = 1 - \frac{1}{\sigma}$$

und somit

$$E_\mathrm{t}(x_j) \le ||\hat{\boldsymbol{f}}||_1 \frac{2}{\sqrt{\pi b}}\left(1+\frac{\sigma}{(2\sigma-1)\pi}\right)\mathrm{e}^{-b\pi^2\left(1-\frac{1}{\sigma}\right)}.$$

Demzufolge bekommen wir für den Gesamtfehler $E \le E_\mathrm{t} + E_\mathrm{a}$ die Ungleichung

$$E_\infty \le ||\hat{\boldsymbol{f}}||_1 \, \mathrm{e}^{-b\pi^2\left(1-\frac{1}{\sigma}\right)} \left[1 + \frac{1}{2m\pi} + \mathrm{e}^{\frac{-4m\pi}{2\sigma-1}} \left(1 + \frac{2\sigma-1}{2(2\sigma+1)m\pi}\right) + \frac{1+\frac{\sigma}{(2\sigma-1)\pi}}{\sqrt{\frac{\sigma m}{2(2\sigma-1)}}}\right].$$

Der Ausdruck in den eckigen Klammern ist für festes $m \ge 1$ monoton wachsend für wachsendes σ, und monoton fallend für festes σ und wachsendes m. Da für $\sigma \to \infty$ dieser Ausdruck gegen

$$2 + \frac{\sqrt{m} + m + 2m\pi}{m^{3/2}\pi}$$

strebt, erhalten wir für $m \ge 2$ die Behauptung

$$\begin{aligned} E_\infty &\le 4\,||\hat{\boldsymbol{f}}||_1 \, \mathrm{e}^{-b\pi^2\left(1-\frac{1}{\sigma}\right)} \\ &= 4\,||\hat{\boldsymbol{f}}||_1 \, \mathrm{e}^{-m\pi\left(1-\frac{1}{2\sigma-1}\right)}. \end{aligned}$$

■

1.3.3 Ansatz mit Sinc- und Kaiser-Bessel-Funktionen (Frequenzbeschränkte Funktionen)

In diesem Abschnitt wollen wir Ansatzfunktionen so wählen, dass der Überlappungsfehler E_a verschwindet. Es ist klar, dass wir dazu frequenzlokale Funktionen, d.h. trigonometrische Polynome, verwenden müssen. Zwei verschiedene Möglichkeiten, die Ansatzfunktion φ zu wählen, werden beschrieben.

Ansatz mit Sinc-Funktionen

Im Folgenden wollen wir wieder einen B-Spline-Ansatz beschreiben. Diesmal soll aber der Fehler E_a vollständig verschwinden, d.h., es wird verlangt, dass $\hat{\varphi}(k)$ einen kompakten Träger im Frequenzbereich hat. Die Fourier-Transformierte der Ansatzfunktion φ sei durch

$$\hat{\varphi}(k) := M_{2m}\left(\frac{2mk}{(2\sigma-1)N}\right) \tag{1.36}$$

definiert. Da der Träger des B-Splines (vgl. die Definition (1.28)) durch $\mathrm{supp} M_{2m} = [-m, m]$ gegeben ist, folgt, dass $\hat{\varphi}(k) = 0$ für $|k| \ge \sigma N \left(1 - \frac{1}{2\sigma}\right)$ ist und damit ist $E_\mathrm{a}(x_j) = 0$ in (1.23), denn $\hat{\varphi}(k + \sigma N r) = 0$ für $-N/2 \le k < N/2$ und $r \ne 0$. Jetzt wird die Funktion φ so berechnet, dass die Fourier-Transformierte von φ durch (1.36) gegeben ist. Dazu starten wir mit der Identität (vgl. (1.30))

$$\int_{\mathbb{R}} M_{2m}(Nx)\,\mathrm{e}^{-2\pi\mathrm{i}xw}\,\mathrm{d}x = \frac{1}{N}\left(\mathrm{sinc}\left(\frac{\pi w}{N}\right)\right)^{2m},$$

substituieren $x = \frac{2mk}{(2\sigma-1)N^2}$, setzen $w = \frac{N^2 s(2\sigma-1)}{2m}$ und erhalten

$$\int_{\mathbb{R}} M_{2m}\left(\frac{2mk}{(2\sigma-1)N}\right) \mathrm{e}^{-2\pi\mathrm{i}ks}\,\mathrm{d}k = \frac{N(2\sigma-1)}{2m}\left(\operatorname{sinc}\left(\frac{\pi N s(2\sigma-1)}{2m}\right)\right)^{2m}.$$

Also berechnet sich die Ansatzfunktion φ durch die Formel

$$\varphi(x) = \frac{N(2\sigma-1)}{2m}\left(\operatorname{sinc}\left(\frac{\pi N x(2\sigma-1)}{2m}\right)\right)^{2m}. \tag{1.37}$$

Satz 1.8 *Zur Berechnung der Summe* (1.1) *durch den NFFT-Algorithmus 1.1 sei für den univariaten Fall $d = 1$ die Ansatzfunktion φ in* (1.37) *gegeben. Es gelte $\sigma > 1$. Dann kann der maximale Approximationsfehler für alle Vektoren $\hat{\boldsymbol{f}} := (\hat{f}_k)_{k\in I_N^1}$ durch*

$$E_\infty := \max_{j\in I_M^1} E(x_j) \le ||\hat{\boldsymbol{f}}||_1 \frac{1}{2m-1}\left(\frac{4}{\sigma^{2m}} + \left(\frac{\sigma}{2\sigma-1}\right)^{2m-1}\right)$$

abgeschätzt werden.

Beweis: Da φ einen kompakten Träger im Frequenzbereich hat, verschwindet der Überlappungsfehler E_{a} in (1.23). Deshalb müssen wir nur noch den Abschneidefehler E_{t} in (1.27) abschätzen. Wir beginnen mit der Berechnung von

$$\max_{k\in I_N^1} \frac{1}{\hat{\varphi}(k)} = \frac{1}{\hat{\varphi}\left(\frac{N}{2}\right)} = \frac{1}{M_{2m}\left(\frac{m}{2\sigma-1}\right)}$$

und schreiben den Funktionswert des B-Splines als Integral

$$M_{2m}\left(\frac{2mk}{(2\sigma-1)N}\right) = \frac{N(2\sigma-1)}{2m}\int_{\mathbb{R}}\left(\operatorname{sinc}\left(\frac{\pi N s(2\sigma-1)}{2m}\right)\right)^{2m} \mathrm{e}^{2\pi\mathrm{i}ks}\,\mathrm{d}s.$$

Wegen

$$\hat{\varphi}\left(\frac{N}{2}\right) = M_{2m}\left(\frac{m}{2\sigma-1}\right) = \frac{N(2\sigma-1)}{m}\int_{\mathbb{R}}\left(\operatorname{sinc}\left(\frac{\pi N s(2\sigma-1)}{m}\right)\right)^{2m} \mathrm{e}^{\pi\mathrm{i}Ns}\,\mathrm{d}s$$

gilt mit der Ungleichung $\sin(x) \ge \frac{2}{\pi}x$ für $x \in [0, \pi/2]$

$$\begin{aligned} M_{2m}\left(\frac{m}{2\sigma-1}\right) &\ge \frac{N(2\sigma-1)}{m}\int_{-\frac{m}{2(2\sigma-1)N}}^{\frac{m}{2(2\sigma-1)N}}\left(\frac{2}{\pi}\right)^{2m}\mathrm{d}s \\ &= \left(\frac{2}{\pi}\right)^{2m} \end{aligned}$$

und schließlich
$$\max_{k\in I_N^1} \frac{1}{\hat{\varphi}(k)} \le \left(\frac{\pi}{2}\right)^{2m}.$$

Weiterhin folgt

$$\begin{aligned}
\int_m^\infty \varphi\left(\frac{x}{\sigma N}\right) \mathrm{d}x &= \frac{N(2\sigma-1)}{2m} \int_m^\infty \left(\operatorname{sinc}\left(\frac{\pi x(2\sigma-1)}{2m\sigma}\right)\right)^{2m} \mathrm{d}x \\
&\le \frac{N(2\sigma-1)}{2m} \int_m^\infty \left(\frac{2\sigma m}{\pi x(2\sigma-1)}\right)^{2m} \mathrm{d}x \\
&= \left(\frac{2}{\pi}\right)^{2m} \frac{2\sigma-1}{2} \frac{N}{2m-1} \left(\frac{\sigma}{2\sigma-1}\right)^{2m} \\
&= \left(\frac{2}{\pi}\right)^{2m} \frac{N\sigma}{4m-2} \left(\frac{\sigma}{2\sigma-1}\right)^{2m-1}.
\end{aligned}$$

Nun schätzen wir noch $\varphi\left(\frac{m}{\sigma N}\right)$ ab. Auf Grund der Definition (1.37) ist

$$\varphi\left(\frac{m}{\sigma N}\right) = \frac{N(2\sigma-1)}{2m} \left(\operatorname{sinc}\frac{(2\sigma-1)\pi}{2\sigma}\right)^{2m}.$$

Da für $\sigma > 1$ die Abschätzung $\frac{\pi}{2} < \frac{2\sigma-1}{2\sigma}\pi < \pi$ und die Ungleichung

$$|\sin(x)| < -\frac{4}{\pi^2}x^2 + \frac{4}{\pi}x \quad \text{für} \quad x \in (\frac{\pi}{2}, \pi)$$

gilt, erhalten wir

$$\left(\operatorname{sinc}\left(\frac{2\sigma-1}{2\sigma}\pi\right)\right)^{2m} \quad < \quad \left(\frac{2}{\pi\sigma}\right)^{2m}$$

und damit
$$\varphi\left(\frac{m}{\sigma N}\right) < \frac{N(2\sigma-1)}{2m} \left(\frac{2}{\pi\sigma}\right)^{2m}.$$

Hieraus folgt für die Ungleichung (1.27)

$$\begin{aligned}
E_\infty &\le 2||\hat{\boldsymbol{f}}||_1 \frac{1}{\sigma N} \left(\frac{\pi}{2}\right)^{2m} \left[\frac{N(2\sigma-1)}{2m}\left(\frac{2}{\pi\sigma}\right)^{2m} + \left(\frac{2}{\pi}\right)^{2m} \frac{N\sigma}{4m-2}\left(\frac{\sigma}{2\sigma-1}\right)^{2m-1}\right] \\
&= ||\hat{\boldsymbol{f}}||_1 \left[\frac{2\sigma-1}{m}\frac{1}{\sigma^{2m+1}} + \frac{1}{2m-1}\left(\frac{\sigma}{2\sigma-1}\right)^{2m-1}\right]
\end{aligned}$$

und somit die Behauptung. ∎

Ansatz mit Kaiser-Bessel-Funktionen

In diesem Abschnitt wollen wir einen Ansatz mit Kaiser-Bessel-Funktionen beschreiben. Im Zusammenhang mit der NFFT wurden diese Funktion erstmals in [44, 45] betrachtet. Wir definieren sie als

$$\varphi(x) := \begin{cases} \dfrac{\sinh(b\sqrt{m^2-(\sigma N)^2x^2})}{\pi\sqrt{m^2-(\sigma N)^2x^2}} & \text{für} \quad |x| < \frac{m}{\sigma N} \qquad \left(b := \pi\left(2-\frac{1}{\sigma}\right)\right), \\ \dfrac{b}{\pi} & \text{für} \quad |x| = \frac{m}{\sigma N}, \\ \dfrac{\sin(b\sqrt{(\sigma N)^2x^2-m^2})}{\pi\sqrt{(\sigma N)^2x^2-m^2}} & \text{sonst} \end{cases} \tag{1.38}$$

und können die Fourier-Transformierte für gerade N unmittelbar, mit Hilfe der modifizierten Bessel-Funktion nullter Ordnung I_0, angeben (vgl. beispielsweise [90])

$$\hat{\varphi}(k) = \begin{cases} \frac{1}{\sigma N} I_0(m\sqrt{b^2-(2\pi k/(\sigma N))^2}) & \text{für } k = -\sigma N\left(1-\frac{1}{2\sigma}\right), \ldots, \sigma N\left(1-\frac{1}{2\sigma}\right), \\ 0 & \text{sonst.} \end{cases} \tag{1.39}$$

Aus dieser Definition folgt sofort, dass der Überlappungsfehler E_{a} verschwindet, denn $\hat{\varphi}(k+\sigma Nr) = 0$ für $r \neq 0$ und $-N/2 \leq k < N/2$ und somit $E_{\mathrm{a}}(x_j) = 0$ in (1.23).
Bevor wir den Abschneidefehler E_{t} abschätzen können, geben wir folgendes Lemma an. Für den Beweis verweisen wir auf [44].

Lemma 1.9 *(vgl. [44, Satz 2.5.11]) Für $b = \pi\left(2-\frac{1}{\sigma}\right)$ mit $\sigma > 1$ und mit $t \in \mathbb{R}$ gilt*

$$\left| \sum_{|r|>m} \frac{\sin(b\sqrt{r^2-m^2})}{\pi\sqrt{r^2-m^2}}\, \mathrm{e}^{2\pi\mathrm{i}tr} \right| \leq 2(1+\sqrt{m}).$$

Satz 1.10 *Zur Berechnung der Summe* (1.1) *durch den NFFT-Algorithmus* 1.1 *sei für den univariaten Fall $d = 1$ die Ansatzfunktion φ in* (1.38) *gegeben. Es gelte $\sigma > 1$. Dann kann der maximale Approximationsfehler für alle Vektoren $\hat{\boldsymbol{f}} := (\hat{f}_k)_{k\in I_N^1}$ durch*

$$E_\infty := \max_{j\in I_M^1} E(x_j) \leq \|\hat{\boldsymbol{f}}\|_1\, 4\pi(\sqrt{m}+m)\sqrt[4]{1-\frac{1}{\sigma}}\; \mathrm{e}^{-2\pi m\sqrt{1-1/\sigma}}$$

abgeschätzt werden.

Beweis: Wir haben wieder nur den Abschneidefehler E_{t} in (1.27) abzuschätzen, denn der Überlappungsfehler E_{a} verschwindet. Da $\hat{\varphi}(k)$, gegeben in (1.39), für wachsende k monoton fallend ist, folgt

$$\max_{k\in I_N^1} \frac{1}{|\hat{\varphi}(k)|} = \frac{1}{|\hat{\varphi}\left(\frac{N}{2}\right)|} = \frac{\sigma N}{I_0\left(2m\pi\sqrt{1-\frac{1}{\sigma}}\right)}.$$

Nun ist $\varphi\left(\frac{m}{\sigma N}\right) = \frac{b}{\pi}$ und wegen Lemma 1.9 können wir mit $b = \pi\left(2 - \frac{1}{\sigma}\right)$ den Approximationsfehler wegen (1.26) durch

$$E_\infty \leq \|\hat{\boldsymbol{f}}\|_1(2+2\sqrt{m})/I_0\left(2m\pi\sqrt{1-\frac{1}{\sigma}}\right)$$

abschätzen. Mit Hilfe der Formel $I_0(x) > \frac{1}{\sqrt{2\pi x}}\,\mathrm{e}^x$ für $x > 1$ (vgl. [43, S.160]) folgt die Behauptung. ■

Die Ergebnisse dieses Abschnitts wollen wir in einem Satz zusammenfassen.

Satz 1.11 *Der Approximationsfehler E in* (1.22)*, der im univariaten Fall durch die näherungsweise Berechnung der Summe* (1.1) *durch s in* (1.16) *mittels Algorithmus 1.1 entsteht, kann für alle $j \in I_M^1$ durch*

$$|f(x_j) - s(x_j)| \leq C(\sigma, m)||\hat{\boldsymbol{f}}||_1$$

mit

$$C(\sigma,m) := \begin{cases} 4\left(\frac{1}{2\sigma-1}\right)^{2m} & \text{für die B-Splines (1.31)},\\ 4\,\mathrm{e}^{-m\pi(1-1/(2\sigma-1))} & \text{für die Gauß-Funktionen (1.32)},\\ \frac{3}{m-1}\left(\frac{\sigma}{2\sigma-1}\right)^{2m-1} & \text{für die Sinc-Funktionen (1.37)},\\ 4\pi(\sqrt{m}+m)\sqrt[4]{1-\frac{1}{\sigma}}\,\mathrm{e}^{-m2\pi\sqrt{1-1/\sigma}} & \text{für die Kaiser-Bessel-Funktionen (1.38)} \end{cases}$$

abgeschätzt werden.

Beweis: Aus Satz 1.5, Satz 1.7, Satz 1.8 und Satz 1.10 folgt sofort die Behauptung. ■

Um eine vorgegebene Genauigkeit ϵ des relativen Approximationsfehlers zu erhalten, müssen wir für festes $\sigma > 1$ den Abschneideparameter m mindestens wie $m \sim \log(1/\epsilon)$ wählen. Damit beträgt die arithmetische Komplexität des Algorithmus 1.1 im univariaten Fall $\mathcal{O}(\sigma N \log(\sigma N) + mM)$=$\mathcal{O}(N \log N + \log(1/\epsilon)M)$.
Wir bestätigen durch numerische Testbeispiele in Abschnitt 1.8 (vgl. Beispiel 1.1), dass der Fehler E für wachsendes m exponentiell fällt.

1.4 FFT-Verfahren für beliebige Knoten und beliebige Frequenzen

Die in Abschnitt 1.1 beschriebenen Algorithmen sind FFT-Verfahren für beliebige Knoten und ganzzahlige Frequenzen. Wir wollen diese Verfahren weiter verallgemeinern und einen Algorithmus für beliebige Knoten und beliebige Frequenzen vorstellen, d.h., einen Algorithmus für die schnelle Berechnung der Summe

$$f(\boldsymbol{v}_j) = \sum_{k \in I_N^1} f_k\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{x}_k\boldsymbol{v}_j} \qquad (j \in I_M^1)\,, \tag{1.40}$$

wobei $\boldsymbol{x}_k \in \mathbb{T}^d$, $\boldsymbol{v}_j \in N\mathbb{T}^d$ sind. Diesen Algorithmus bezeichnen wir als NNFFT. Erstmals wurde dieser Algorithmus von B. Elbel und G. Steidl in [35, 34] vorgeschlagen (siehe auch [104]). Es stellt sich heraus, dass der NNFFT-Algorithmus eine Kombination aus dem NFFT-Algorithmus 1.1 und dem NFFT$^{\mathrm{T}}$-Algorithmus 1.3 darstellt.
Es sei $\varphi_1 \in L_2(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$ eine hinreichend glatte Funktion, wobei die Fourier-Transformierte durch

$$\hat{\varphi}_1(\boldsymbol{v}) = \int\limits_{\mathbb{R}^d} \varphi_1(\boldsymbol{x}) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{v}\boldsymbol{x}} \mathrm{d}\boldsymbol{x}$$

gegeben ist und der Bedingung $\hat{\varphi}_1(\boldsymbol{v}) \neq 0$ für alle $\boldsymbol{v} \in N\mathbb{T}^d$ genügt. Damit erhalten wir für die Funktion

$$G(\boldsymbol{x}) := \sum_{k \in I_N^1} f_k\, \varphi_1(\boldsymbol{x} - \boldsymbol{x}_k)$$

die Fourier-Transformierte

$$\hat{G}(\boldsymbol{v}) = \sum_{k \in I_N^1} f_k\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{x}_k\boldsymbol{v}}\, \hat{\varphi}_1(\boldsymbol{v})$$

und es gilt

$$f(\boldsymbol{v}_j) = \frac{\hat{G}(\boldsymbol{v}_j)}{\hat{\varphi}_1(\boldsymbol{v}_j)} \qquad (j \in I_N^1).$$

Also müssen wir für gegebene Fourier-Transformierte $\hat{\varphi}_1$ die Funktion $\hat{G}$ an den Knoten $\boldsymbol{v}_j$ $(j \in I_N^1)$ berechnen.
Es sei $N_1 := \sigma_1 N$ $(\sigma_1 > 1)$, $m_1 \in \mathbb{N}$ $(2m_1 \ll N_1)$. Außerdem führen wir den Parameter $a \in \mathbb{R}^+$ ein, um a-periodische Funktionen zu benutzen. Wir schreiben $\hat{G}(\boldsymbol{v})$ als

$$\hat{G}(\boldsymbol{v}) \quad = \quad \sum_{k \in I_N^1} f_k \int\limits_{\mathbb{R}^d} \varphi_1(\boldsymbol{x} - \boldsymbol{x}_k)\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{x}\boldsymbol{v}}\, \mathrm{d}\boldsymbol{x}$$

und dürfen die Summation und Integration wegen [20, Lemma 5.3] vertauschen, so dass

$$\hat{G}(\boldsymbol{v}) \quad = \quad \sum_{k \in I_N^1} f_k \int\limits_{a\mathbb{T}^d} \sum_{\boldsymbol{r} \in \mathbb{Z}^d} \varphi_1(\boldsymbol{x} + a\boldsymbol{r} - \boldsymbol{x}_k)\, \mathrm{e}^{-2\pi\mathrm{i}(\boldsymbol{x}+a\boldsymbol{r})\boldsymbol{v}}\, \mathrm{d}\boldsymbol{x} \tag{1.41}$$

gilt. Anschließend diskretisieren wir dieses Integral wieder mit der Rechteckregel

$$\hat{G}(\boldsymbol{v}) \approx S_1(\boldsymbol{v}) := \sum_{k \in I_N^1} f_k\, N_1^{-d} \sum_{\boldsymbol{t} \in I_{aN_1}^d} \sum_{\boldsymbol{r} \in \mathbb{Z}^d} \varphi_1(\frac{\boldsymbol{t}}{N_1} + a\boldsymbol{r} - \boldsymbol{x}_k)\, \mathrm{e}^{-2\pi\mathrm{i}(\frac{\boldsymbol{t}}{N_1}+a\boldsymbol{r})\boldsymbol{v}}. \tag{1.42}$$

Analog zu Abschnitt 1.1 wird die Funktion φ_1 durch eine Funktion ψ_1 mit $\operatorname{supp}\psi_1 \subseteq [-\frac{2m_1}{N_1}, \frac{2m_1}{N_1}]^d$ approximiert. Damit enthält die innere Summe in (1.42) nur für $r = 0$ einen Summanden ungleich Null. Vertauschen wir die Reihenfolge der Summation, so ergibt sich

$$S_1(\boldsymbol{v}) \approx S_2(\boldsymbol{v}) := N_1^{-d} \sum_{\boldsymbol{t} \in I_{aN_1}^d} \left(\sum_{k \in I_N^1} f_k\, \psi_1(\frac{\boldsymbol{t}}{N_1} - \boldsymbol{x}_k) \right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{t}\boldsymbol{v}/N_1}.$$

Nachdem wir die innere Summe über $k \in I_N^1$ berechnet haben, können wir die Auswertung der äußeren Summe mit Hilfe des NFFT-Algorithmus 1.1 schnell berechnen. Die zugehörigen Ansatzfunktionen und Parameter indizieren wir mit dem Index 2 und fassen den Algorithmus zusammen.

Algorithmus 1.12 (Schnelle Berechnung der NNFFT (1.40))

Eingabe: $N \in \mathbb{N}$, $\sigma_1 > 1$, $\sigma_2 > 1$, $N_1 := \sigma_1 N$, $a := 1 + \frac{2m_1}{N_1}$, $N_2 := \sigma_1\sigma_2 aN$,
$\boldsymbol{x}_k \in \mathbb{T}^d$, $f_k \in \mathbb{C}$ $(k \in I_N^1)$, $\boldsymbol{v}_j \in N\mathbb{T}^d$, $(j \in I_M^1)$.

Vorberechnung: *(i)* $\hat{\varphi}_1(\boldsymbol{v}_j)$ $(j \in I_N^1)$, *wobei* $\hat{\varphi}_1(\boldsymbol{v}_j) \neq 0$ *vorausgesetzt wird,*
(ii) $\psi_1(\frac{\boldsymbol{t}}{N_1} - \boldsymbol{x}_k)$ $(k \in I_{N_1,m}^{\mathrm{T}}(\boldsymbol{t}), \boldsymbol{t} \in I_{aN_1,m_1}(\boldsymbol{x}_k))$,
(iii) $c_{\boldsymbol{t}}(\varphi_2)$ $(\boldsymbol{t} \in I_{aN_1}^d)$, *wobei* $c_{\boldsymbol{t}}(\varphi_2) \neq 0$ *vorausgesetzt wird,*
(iv) $\psi_2(\frac{\boldsymbol{v}_j}{N_1} - \frac{\boldsymbol{l}}{N_2})$ $(j \in I_N^1, \boldsymbol{l} \in I_{N_2,m_2}(\frac{\boldsymbol{v}_j}{N_1}))$.

1. *Berechne*
$$F(\boldsymbol{t}) := \sum_{k \in I_{N_1,m}^{\mathrm{T}}} f_k\, \psi_1\left(\frac{\boldsymbol{t}}{N_1} - \boldsymbol{x}_k\right) \quad (\boldsymbol{t} \in I_{aN_1,m_1})\,.$$

2. *Setze* $\hat{g}_{\boldsymbol{t}} := F(\boldsymbol{t})/c_{\boldsymbol{t}}(\varphi_2)$ $\quad(\boldsymbol{t} \in I_{aN_1}^d)$.
3. *Berechne mit der* d*-variaten* FFT
$$g_{\boldsymbol{l}} := N_2^{-d} \sum_{\boldsymbol{t} \in I_{aN_1}^d} \hat{g}_{\boldsymbol{t}}\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{t}\boldsymbol{l}/N_2} \quad (\boldsymbol{l} \in I_{N_2}^d).$$

4. *Bilde*
$$s(\boldsymbol{v}_j) := N_1^{-d} \sum_{\boldsymbol{l} \in I_{N_2,m_2}(\boldsymbol{v}_j/N_1)} g_{\boldsymbol{l}}\, \psi_2\left(\frac{\boldsymbol{v}_j}{N_1} - \frac{\boldsymbol{l}}{N_2}\right) \quad (j \in I_N^1)\,.$$

5. *Berechne* $S(\boldsymbol{v}_j) := s(\boldsymbol{v}_j)/\hat{\varphi}_1(\boldsymbol{v}_j)$ $(j \in I_N^1)$.

Ausgabe: $S(\boldsymbol{v}_j)$ *Näherungswerte für* $f(\boldsymbol{v}_j)$ $(j \in I_N^1)$.

Komplexität: $\mathcal{O}((\sigma_1\sigma_2 aN)^d \log(\sigma_1\sigma_2 aN)) = \mathcal{O}(N^d \log N)$.

Den Approximationsfehler von Algorithmus 1.12 berechnen wir durch
$$E(\boldsymbol{v}_j) \;\leq\; E_{\mathrm{a}}(\boldsymbol{v}_j) + E_{\mathrm{t}}(\boldsymbol{v}_j) + E_3(\boldsymbol{v}_j)$$
mit $E_{\mathrm{a}}(\boldsymbol{v}_j) := |f(\boldsymbol{v}_j) - \frac{S_1(\boldsymbol{v}_j)}{\hat{\varphi}_1(\boldsymbol{v}_j)}|$, $E_{\mathrm{t}}(\boldsymbol{v}_j) := |\frac{S_1(\boldsymbol{v}_j) - S_2(\boldsymbol{v}_j)}{\hat{\varphi}_1(\boldsymbol{v}_j)}|$, und $E_3(\boldsymbol{v}_j) := |\frac{S_2(\boldsymbol{v}_j) - s(\boldsymbol{v}_j)}{\hat{\varphi}_1(\boldsymbol{v}_j)}|$. Der Fehler $E_3(\boldsymbol{v}_j)$ ist das Produkt aus dem Approximationsfehler des NFFT-Algorithmus 1.1 und $|\hat{\varphi}_1(\boldsymbol{v}_j)|^{-1}$. Der Abschneidefehler $E_{\mathrm{t}}(\boldsymbol{v}_j)$ verhält sich wie der Abschneidefehler in Algorithmus 1.1. Der Fehler $E_{\mathrm{a}}(\boldsymbol{v}_j)$ entsteht aus der Diskretisierung des Integrals (1.41) und kann mittels der Aliasing-Formel abgeschätzt werden (siehe [35]).

1.5 Diskrete trigonometrische Transformationen

Neben der DFT sind eine Reihe diskreter trigonometrischer Transformationen von praktischem Interesse. Sie stehen in engem Zusammenhang mit der DFT und lassen sich ebenfalls mittels schneller Algorithmen berechnen. Eine herausragende Rolle spielen dabei die diskreten Kosinus-Transformationen, die breite Anwendung in der numerischen Analysis und in der digitalen Signalverarbeitung finden [118]. In diesem Abschnitt entwickeln wir schnelle diskrete trigonometrische Transformationen an beliebigen Knoten. Zunächst sei, wie in [127], die DCT-I (Discrete Cosine Transform of Type I) definiert. Unter der DCT vom Typ I der Länge $N+1$ (DCT-I($N+1$)) verstehen wir diejenige lineare Abbildung von $\mathbb{R}^{N+1}$ in sich, die jedem Vektor $\hat{\boldsymbol{x}} = (\hat{x}_j)_{j=0}^N \in \mathbb{R}^{N+1}$ den Vektor $\boldsymbol{x} = (x_k)_{k=0}^N \in \mathbb{R}^{N+1}$ mit

$$x_k := \sum_{j=0}^{N} \varepsilon_{N,j} \hat{x}_j \cos \frac{jk\pi}{N}$$

zuordnet, wobei wir $\varepsilon_{N,0} = \varepsilon_{N,N} = 1/2$, $\varepsilon_{N,j} = 1$ $(j = 1, \ldots, N-1)$ gesetzt haben. Daraus kann die Matrix-Vektor-Darstellung der DCT-I

$$\boldsymbol{x} = \boldsymbol{C}_{N+1}^I \hat{\boldsymbol{x}}, \qquad \boldsymbol{C}_{N+1}^I := \left(\varepsilon_{N,j} \cos \frac{jk\pi}{N} \right)_{k,j=0}^N \tag{1.43}$$

abgeleitet werden.

Es existieren verschiedene Möglichkeiten einer schnellen Realisierung dieser Transformationen in $\mathcal{O}(N \log N)$ arithmetischen Rechenoperationen. Hintergrund dieser schnellen Algorithmen, die dann nur reelle Additionen und Multiplikationen benötigen, sind die Additionstheoreme, die eine Teile-und-Herrsche-Strategie ermöglichen (siehe hierzu z.B. [115, 1, 117, 118]).

Wir wollen nun einen schnellen Algorithmus, die NFCT (Nonequispaced Fast Cosine Transform), für die nichtäquidistante diskrete Kosinus-Transformation NDCT (Nonequispaced Discrete Cosine Transform) herleiten. Diese Algorithmen werden neue Anwendungen und schnelle numerische Verfahren ermöglichen (siehe z.B. [57, 123]). Dazu gehen wir ähnlich wie in Abschnitt 1.1 vor, beschränken uns aber auf den univariaten Fall, d.h. $d = 1$. Drei verschiedene Methoden für die NFCT wurden bereits in [95] veröffentlicht. Die besten numerischen Ergebnisse konnten wir wieder mit dem Algorithmus erzielen, der auf der Approximation durch Translate basiert. Diese Herleitung beruht auf Eigenschaften des sogenannten Chebyshev-Shiftes. Hier wird ein einfacherer Zugang als in [40] beschrieben. Algorithmen für die nichtäquidistante Hartley-Transformation sollen hier nicht untersucht werden. Sie ergeben sich aber in ähnlicher Weise wie die Algorithmen für die NFCT und NFST.

1.5.1 Schnelle Kosinus-Transformationen für nichtäquidistante Daten

Wir suchen eine Methode, die Funktion

$$f^{\mathrm{C}}(x) := \sum_{k=0}^{N-1} \hat{f}_k^{\mathrm{C}} \cos(2\pi k x) \tag{1.44}$$

schnell an beliebigen Knoten $x_j \in [0,1/2]$ $(j=0,\ldots,M-1)$ auszuwerten. Natürlich kann diese Summe wieder als Matrix-Vektor-Produkt $\boldsymbol{A}\hat{\boldsymbol{f}}$ mit den Bezeichnungen $\boldsymbol{A} := (\cos(2\pi k x_j))_{j=0,k=0}^{M-1,N-1}$ und $\hat{\boldsymbol{f}} = (\hat{f}_k^{\mathrm{C}})_{k=0}^{N-1}$ aufgefasst werden. Ein schneller Algorithmus für die NDCT leitet sich aus dem NFFT-Algorithmus 1.1 ab.
Der Oversamplingfaktor $\sigma > 1$ $(\sigma N \in \mathbb{N})$ sei gegeben und wie in (1.6) werden 1-periodisierte Versionen $\tilde{\varphi}$ einer geraden Funktion $\varphi \in L^2(\mathbb{R}) \cap L^1(\mathbb{R})$ mit gleichmäßig konvergenter Fourier-Reihe betrachtet. Ziel ist es nun, die Koeffizienten $g_l \in \mathbb{R}$ $(l = 0,\ldots,\sigma N)$ der Linearkombination

$$s_1(x) := \sum_{l=0}^{\sigma N} g_l \tilde{\varphi}\left(x - \frac{l}{2\sigma N}\right) \tag{1.45}$$

so zu bestimmen, dass s_1 die Funktion f^{C} gut annähert. Dazu wählen wir in

$$f(x) := \sum_{k=-N}^{N-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k x} \tag{1.46}$$

(siehe auch (1.1)) die Koeffizienten $\hat{f}_k \in \mathbb{R}$ $(k=0,\ldots,N-1)$ mit $\hat{f}_k = \hat{f}_{-k}$ und $\hat{f}_{-N} = 0$. Dann gilt die Identität $f^{\mathrm{C}}(x) = f(x)$, falls $\hat{f}_k^{\mathrm{C}} = 2\varepsilon_{N,k}\hat{f}_k$. Da $\tilde{\varphi}$ eine gerade Funktion ist, erhalten wir für die Fourier-Koeffizienten die Eigenschaft $c_k(\tilde{\varphi}) = c_{-k}(\tilde{\varphi})$ und mit (1.13) auch $\hat{g}_k = \hat{g}_{-k}$. Beachten wir diese Symmetrie in Schritt 2 vom NFFT-Algorithmus 1.1, so können die Koeffizienten g_l einfacher durch

$$g_l = \mathrm{Re}(g_l) = \frac{1}{\sigma N} \sum_{k=0}^{\sigma N} \varepsilon_{\sigma N,k}\, \hat{g}_k \cos(\frac{2\pi k l}{\sigma N}) \quad (l = 0,\ldots,\sigma N)$$

berechnet werden. Es gilt dann $g_l = g_{2\sigma N r - l}$ $(r \in \mathbb{Z})$, d.h., die Koeffizienten g_l in (1.45) ergeben sich mit einer schnellen Kosinus-Transformation vom Typ-I der Länge σN. Anschließend approximieren wir s_1 wieder durch

$$s(x) := \sum_{l=\lfloor 2\sigma N x\rfloor - m}^{\lceil 2\sigma N x\rceil + m} g_l \tilde{\psi}\left(x - \frac{l}{2\sigma N}\right). \tag{1.47}$$

Für einen festen Knoten x_j enthält die Summe (1.47) höchstens $2m+2$ von Null verschiedene Summanden. Damit kann die Summe (1.44) an den Knoten x_j $(j=0,\ldots,M-1)$ wegen

$$f(x) \approx s_1(x) \approx s(x)$$

näherungsweise durch Funktionsauswertung von $s(x_j)$ $(j=0,\ldots,M-1)$ berechnet werden.

Zusammenfassend erhalten wir folgenden schnellen Algorithmus, den wir mit NFCT bezeichnen.

Algorithmus 1.13 (NFCT)

Eingabe: $N, M \in \mathbb{N}$, $\sigma > 1$, $m \in \mathbb{N}$, $x_j \in [0, 1/2]$ $(j = 0, \ldots, M-1)$,
$\hat{f}_k^{\mathrm{C}} \in \mathbb{R}$ $(k = 0, \ldots, N-1)$.

Vorberechnung: (i) *Berechne die Fourier-Koeffizienten* $c_k(\tilde{\varphi})$ *für* $k = 0, \ldots, N-1$, *wobei* $c_k(\tilde{\varphi}) \neq 0$ *vorausgesetzt wird.*
(ii) *Berechne die Funktionswerte* $\tilde{\psi}\left(x_j - \frac{l}{2\sigma N}\right)$ *für* $j = 0, \ldots, M-1$ *und* $l \in I_{\sigma N, m}(x_j)$.

1. Setze

$$\hat{g}_k := \begin{cases} \dfrac{\hat{f}_k^{\mathrm{C}}}{2\varepsilon_{\sigma N,k}\, c_k(\tilde{\varphi})} & \text{für } k = 0, \ldots, N-1, \\ 0 & \text{für } k = N, \ldots, \sigma N. \end{cases}$$

2. Berechne mit Hilfe der DCT-I die Daten

$$g_l := \frac{1}{\sigma N} \sum_{k=0}^{\sigma N} \varepsilon_{\sigma N,k}\, \hat{g}_k \cos\left(\frac{\pi k l}{\sigma N}\right) \qquad (l = 0, \ldots, \sigma N).$$

3. Setze

$$s(x_j) := \sum_{l=\lfloor 2\sigma N x_j \rfloor - m}^{\lceil 2\sigma N x_j \rceil + m} g_l \tilde{\psi}\left(x_j - \frac{l}{2\sigma N}\right) \qquad (j = 0, \ldots, M-1).$$

Ausgabe: $s(x_j)$ $(j = 0, \ldots, M-1)$ *Näherungswerte für* $f^{\mathrm{C}}(x_j)$ *in* (1.44).

Komplexität: $\mathcal{O}(N \log N + mM)$.

Wir diskutieren jetzt einen Algorithmus für das „transponierte“ Problem, d.h. für die schnelle Berechnung von

$$h(k) := \sum_{j=0}^{M-1} h_j \cos(2\pi k x_j) \qquad (k = 0, \ldots, N-1)\,. \tag{1.48}$$

Dazu schreiben wir Algorithmus 1.13 in Matrix-Vektor-Form, denn die Berechnung der Summe (1.44) an den Knoten x_j bedeutet eine Matrix-Vektor-Multiplikation mit der Matrix $\boldsymbol{A}^{\mathrm{T}}$. Die Matrix $\boldsymbol{A}$ kann durch das Produkt $\boldsymbol{B}\boldsymbol{C}^I_{N+1,t}\boldsymbol{D}$ näherungsweise faktorisiert werden. Jede der folgenden Matrizen korrespondiert zu einem Schritt im Algorithmus 1.13:

1. $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ ist eine Diagonal-Matrix

$$\boldsymbol{D} := \ \mathrm{diag}\ \left(1/(2\varepsilon_{\sigma N,k}\, c_k(\tilde{\varphi}))\right)_{k=0}^{N-1}.$$

2. $\boldsymbol{C}^I_{N+1,t} \in \mathbb{R}^{\sigma N \times N}$ ist eine abgeschnittene Kosinus-I Matrix

$$\boldsymbol{C}^I_{N+1,t} := \left(\frac{\varepsilon_{\sigma N,k}}{\sigma N} \cos \frac{\pi k l}{\sigma N}\right)_{l=0,k=0}^{\sigma N-1, N-1}.$$

3. $\boldsymbol{B} \in \mathbb{R}^{M\times\sigma N}$ ist eine dünn besetzte Bandmatrix mit höchstens $2m+1$ Einträgen ungleich Null pro Zeile

$$\boldsymbol{B} := (b_{j,l})_{j=0,l=0}^{M-1,\sigma N-1},$$

wobei

$$b_{j,l} = \begin{cases} \tilde{\psi}\left(x_j - \frac{l}{2\sigma N}\right) & \text{falls} \quad l \in \{\lfloor 2\sigma N x_j \rfloor - m, \ldots, \lceil 2\sigma N x_j \rceil + m\}, \\ 0 & \text{sonst.} \end{cases}$$

Der NDCT$^{\mathrm{T}}$ Algorithmus

Die Faktorisierung für $\boldsymbol{A}$ erlaubt es uns, einen schnellen Algorithmus für die Berechnung von (1.48) herzuleiten, denn

$$\begin{aligned} \boldsymbol{g} := (h(k))_{k=0}^{N-1} &= \boldsymbol{A}^{\mathrm{T}}(h_j)_{j=0}^{M-1} \\ &\approx \boldsymbol{D}^{\mathrm{T}}(\boldsymbol{C}_{N+1,t}^{I})^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}(h_j)_{j=0}^{M-1}. \end{aligned}$$

Daraus folgt unmittelbar der folgende Algorithmus.

Algorithmus 1.14 (NFCT$^{\mathrm{T}}$)
Eingabe: $N \in \mathbb{N}, \sigma > 1, m \in \mathbb{N}$, $x_j \in [0, 1/2]$ $(j = 0, \ldots, M-1)$,
$h_j \in \mathbb{R}$ $(j = 0, \ldots, M-1)$.

Vorberechnung: (i) Berechne die Fourier-Koeffizienten $c_k(\tilde{\varphi})$ für $k = 0, \ldots, N-1$, wobei $c_k(\tilde{\varphi}) \neq 0$ vorausgesetzt wird.
(ii) Berechne die Funktionswerte $\tilde{\psi}\left(x_j - \frac{l}{\sigma N}\right)$ für $l \in I_{\sigma N}^1$ und $j \in I_{\sigma N,m}^{\mathrm{T}}(l)$.

1. *Setze $\boldsymbol{g} := \boldsymbol{B}^{\mathrm{T}}\boldsymbol{h}$ mittels*

   for $l = 0 \ldots, \sigma N$
   $\quad g_l := 0$
   end
   for $j = 0, \ldots, M-1$
   $\quad$ for $l = \lfloor \sigma N x_j \rfloor - m, \ldots, \lceil \sigma N x_j \rceil + m$
   $\quad\quad g_l := g_l + h_j\, \tilde{\psi}(x_j - \frac{l}{2\sigma N})$
   $\quad$ end
   end.

2. *Berechne mit Hilfe der DCT-I die Daten*

$$\hat{g}_k := \frac{1}{\sigma N} \sum_{l=0}^{\sigma N} \varepsilon_{\sigma N,l}\, g_l \cos \frac{\pi k l}{\sigma N}.$$

3. *Setze $\tilde{h}(k) := \hat{g}_k / (2\varepsilon_{\sigma N,k}\, c_k(\tilde{\varphi}))$ für $k = 0, 1, \ldots, N-1$.*

Ausgabe: $\tilde{h}(k)$ $(k = 0, \ldots, N-1)$ Näherungswerte für $h(k)$ in (1.48).

Komplexität: $\mathcal{O}(N \log N + mM)$.

1.5.2 Schnelle Sinus-Transformationen für nichtäquidistante Daten

Die Berechnung der NFFT wird jetzt so modifiziert, dass die Funktion

$$f^{\mathrm{S}}(x) = \sum_{k=1}^{N-1} \hat{f}_k^{\mathrm{S}} \sin(2\pi k x) \tag{1.49}$$

an den beliebigen Knoten $x_j \in [0, 1/2]$ schnell auswertbar ist. Dazu beginnen wir wieder mit der Formel (1.46), setzen jetzt aber voraus, dass $\hat{f}_k \in \mathbb{R}$ mit $\hat{f}_{-k} = -\hat{f}_k$ ($k = 1, \ldots, N-1$) und $\hat{f}_0 = \hat{f}_{-N} = 0$ gilt. Dann folgt für das trigonometrische Polynom in (1.46) die Gleichung

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

Wir erhalten für $\hat{f}_k^{\mathrm{S}} = 2\hat{f}_k$, dass $f^{\mathrm{S}}(x) = \mathrm{i} f(x)$ ist. Damit folgt für die Koeffizienten g_k aus (1.13) die Beziehung $\hat{g}_k = -\hat{g}_{-k}$ ($k = 1, \ldots, \sigma N - 1$) und es ergibt sich für $l = 0, \ldots, \sigma N$ analog zu (1.14) die Gleichung

$$-\mathrm{i} g_l = \frac{-\mathrm{i}}{2\sigma N} \sum_{k=-\sigma N}^{\sigma N-1} \hat{g}_k \, \mathrm{e}^{\pi \mathrm{i} k l/(\sigma N)} = \frac{1}{\sigma N} \sum_{k=1}^{\sigma N-1} \hat{g}_k \sin\left(\frac{\pi k l}{\sigma N}\right). \tag{1.50}$$

Insbesondere gilt die Eigenschaft $g_{2\sigma N r - l} = -g_l$ ($r \in \mathbb{Z}$). Schließlich berechnen wir wie in (1.16) die Summen

$$\mathrm{i} s(x_j) := \sum_{l=\lfloor 2\sigma N x_j \rfloor - m}^{\lceil 2\sigma N x_j \rceil + m} \mathrm{i} g_l \, \tilde{\varphi}\left(x_j - \frac{l}{2\sigma N}\right) \tag{1.51}$$

und haben damit eine gute Näherung für $f^{\mathrm{S}}(x_j) = \mathrm{i} f(x_j) \approx \mathrm{i} s(x_j)$.
Zusammenfassend liest sich der Algorithmus für die schnelle Berechnung der NDST wie folgt:

Algorithmus 1.15 (NFST)

Eingabe: $N \in \mathbb{N}$, $\sigma > 1$, $m \in \mathbb{N}$, $x_j \in [0, 1/2]$, ($j = 0, \ldots, M-1$),
$\hat{f}_k^{\mathrm{S}} \in \mathbb{R}$ ($k = 1, \ldots, N-1$).

Vorberechnung: *(i) Berechne die Fourier-Koeffizienten* $c_k(\tilde{\varphi})$ *für* $k = 0, \ldots, N-1$, *wobei* $c_k(\tilde{\varphi}) \neq 0$ *vorausgesetzt wird.*
(ii) Berechne die Funktionswerte $\tilde{\psi}\left(x_j - \frac{l}{2\sigma N}\right)$ *für* $j = 0, \ldots, M-1$ *und* $l \in I_{\sigma N, m}(x_j)$.

1. Setze

$$\hat{g}_k := \begin{cases} \dfrac{\hat{f}_k^{\mathrm{S}}}{2\, c_k(\tilde{\varphi})} & \text{für } k = 1, \ldots, N-1, \\ 0 & \text{für } k = 0; k = N, \ldots, \sigma N. \end{cases}$$

2. Berechne mit Hilfe der DST-I die Daten

$$g_l := \frac{1}{\sigma N} \sum_{k=1}^{\sigma N-1} \hat{g}_k \sin\left(\frac{\pi k l}{\sigma N}\right) \quad (l = 0, \dots, \sigma N).$$

3. Setze für

$$s(x_j) := \sum_{l=\lfloor 2\sigma N x_j \rfloor - m}^{\lceil 2\sigma N x_j \rceil + m} g_l \tilde{\psi}\left(x_j - \frac{l}{2\sigma N}\right) \quad (j = 0, \dots, M-1).$$

Ausgabe: $s(x_j)$ $(j = 0, \dots, M-1)$ *Näherungswerte für* $f^{\mathrm{S}}(x_j)$ *in* (1.49).

Komplexität: $\mathcal{O}(N \log N + mM)$.

Den Algorithmus zur schnellen Berechnung von

$$h(k) := \sum_{j=0}^{M-1} h_j \sin(2\pi k x_j), \tag{1.52}$$

der unmittelbar aus dem Transponieren des Matrix-Vektor-Produktes folgt, bezeichnen wir als $\mathrm{NFST}^{\mathrm{T}}$ (vergleiche Algorithmus 1.14). Beispiel 2.9 nutzt die Algorithmen 1.13, 1.14, 1.15 und die $\mathrm{NFST}^{\mathrm{T}}$ zur schnellen Summation.

1.6 Rundungsfehleranalyse

Neben dem Approximationsfehler entsteht durch die numerische Berechnung der NFFT noch ein Rundungsfehler. In diesem Abschnitt werden wir zeigen, dass ähnlich wie die FFT [107, 66], unser Algorithmus robust bezüglich der Rundungsfehler ist. Wir untersuchen hier nur den NFFT-Algorithmus 1.1 für den univariaten Fall $d = 1$ und $M = N$. Für eine ausführliche Diskussion von Rundungsfehlern der schnellen Fourier-Transformation und der schnellen diskreten trigonometrischen Transformationen sei auf die Dissertation [112] und den Übersichtsartikel [122] verwiesen.
Im Folgenden nutzen wir das Standard-Modell der reellen Gleitkommaarithmetik (siehe [66, S. 44]): Für beliebige $\xi, \eta \in \mathbb{R}$ und eine Operation $\circ \in \{+, -, \times, /\}$ besteht zwischen dem exakten Wert $\xi \circ \eta$ und dem berechneten Wert $\mathrm{fl}(\xi \circ \eta)$ der Zusammenhang

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

wobei wir mit u die Maschinengenauigkeit (unit roundoff) bezeichnen. Im Fall der einfachen Genauigkeit (24 Bits für die Mantisse (mit einem Bit für das Vorzeichen), 8 Bits für den Exponenten) erhalten wir

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

und im Fall der doppelten Genauigkeit (53 Bits für die Mantisse (mit einem Bit für das Vorzeichen), 11 Bits für den Exponenten) ergibt sich

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

Da meistens die komplexe Arithmetik mit Hilfe der reellen Arithmetik implementiert ist (siehe [66, S. 78 – 80]), folgt für beliebige $\xi, \eta \in \mathbb{C}$

$$\begin{aligned} \mathrm{fl}(\xi + \eta) &= (\xi + \eta)\,(1 + \delta) \quad (|\delta| \le u)\;, & (1.53)\\ \mathrm{fl}(\xi\,\eta) &= \xi\,\eta\,(1 + \delta) \quad (|\delta| \le \frac{2\sqrt{2}\,u}{1 - 2u})\;. & (1.54) \end{aligned}$$

Insbesondere erhalten wir für $\xi \in \mathbb{R} \cup \mathrm{i}\mathbb{R}$ und $\eta \in \mathbb{C}$ den Zusammenhang

$$\mathrm{fl}(\xi\eta) = \xi\eta\,(1 + \delta) \quad (|\delta| \le u)\;. \tag{1.55}$$

Um die folgende Rundungsfehleranalyse zu vereinfachen, setzen wir voraus, dass alle Einträge der Matrizen $\boldsymbol{A}$, $\boldsymbol{B}$ und $\boldsymbol{D}$ in (1.18) exakt vorberechnet wurden. Außerdem sollen die Eingabevektoren $\hat{\boldsymbol{f}} \in \mathbb{R}^N$ sein. Wenn wir $\boldsymbol{f} := \boldsymbol{A}\hat{\boldsymbol{f}}$ mit einfacher Kaskadensummation berechnen, erhalten wir mit (1.53), (1.55) und [66, S. 70] die komponentenweise Abschätzung

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

und unter Nutzung der Euklidischen Norm

$$\|\mathrm{fl}(\boldsymbol{f}) - \boldsymbol{f}\|_2 \le \big(u\,N\,(\lceil \log_2 N \rceil + 1) \,+\, \mathcal{O}(u^2)\big)\; \|\hat{\boldsymbol{f}}\|_2\;.$$

Insbesondere ergibt sich für $\boldsymbol{f} := \boldsymbol{F}_N^1 \hat{\boldsymbol{f}}$ mit der Fourier-Matrix $\boldsymbol{F}_N^1 = \boldsymbol{F}_{N,N}^1$ die Abschätzung

$$\|\mathrm{fl}(\boldsymbol{F}_N^1 \hat{\boldsymbol{f}}) - \boldsymbol{F}_N^1 \hat{\boldsymbol{f}}\|_2 \le \big(u\,N\,(\lceil \log_2 N \rceil + 1) \,+\, \mathcal{O}(u^2)\big)\; \|\hat{\boldsymbol{f}}\|_2\;. \tag{1.56}$$

Falls wir das Matrix-Vektor-Produkt $\boldsymbol{f} = \boldsymbol{F}_N^1 \hat{\boldsymbol{f}}$ ($\hat{\boldsymbol{f}} \in \mathbb{R}^N$, N ist eine Zweierpotenz) mit dem Cooley-Tukey-Algorithmus realisieren, erhalten wir unter Nutzung der Ideen aus dem Beweis in [129] und Beachtung von (1.53) – (1.54), die Rundungsfehlerabschätzung

$$\|\mathrm{fl}(\boldsymbol{F}_N^1 \hat{\boldsymbol{f}}) - \boldsymbol{F}_N^1 \hat{\boldsymbol{f}}\|_2 \le \Big(u\,(4 + \sqrt{2})\sqrt{N}\log_2 N \,+\, \mathcal{O}(u^2)\Big)\; \|\hat{\boldsymbol{f}}\|_2\;. \tag{1.57}$$

Der nachfolgende Satz zeigt, dass der Rundungsfehler, der durch den NFFT-Algorithmus 1.1 erzeugt wird, abgeschätzt werden kann und sich ähnlich wie der Rundungsfehler der gewöhnlichen FFT (1.56) verhält. Es wird nur ein weiterer konstanter Faktor, der nur von dem Abschneideparameter m und von dem Oversamplingfaktor σ abhängt, benötigt.

Satz 1.16 *Es seien $m, N \in \mathbb{N}$ und es sei σN ($\sigma > 1$) eine Zweierpotenz mit $2m \ll N$. Sei $h \in L^2(\mathbb{R}) \cap L^1(\mathbb{R})$ eine nichtnegative reelle, gerade Funktion, welche für $x \ge 0$ monoton fällt, so dass $\tilde{\varphi}$ und $\tilde{\psi}$ durch*

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

gegeben sind. Weiterhin setzen wir voraus, dass $\tilde{\varphi}$ eine gleichmäßig konvergente Fourier-Reihe mit monoton fallenden Fourier-Koeffizienten

$$c_k(\tilde{\varphi}) = \frac{1}{\sigma N}\,\hat{h}(\frac{2\pi k}{\sigma N}) \quad (k \in \mathbb{Z})$$

hat. Die Knoten $x_j \in [-\frac{1}{2}, \frac{1}{2})$ $(j \in I_N^1)$ seien so verteilt, dass in jedem „Fenster“ $[-\frac{m}{\sigma N} + \frac{l}{\sigma N}, \frac{m}{\sigma N} + \frac{l}{\sigma N})$ $(l \in I_{\sigma N}^1)$ höchstens γ/σ Knoten enthalten sind. Falls die Summe (1.1) *im univariaten Fall mit Hilfe vom NFFT-Algorithmus 1.1 mit den Funktionen $\tilde{\varphi}, \tilde{\psi}$ berechnet wird, d.h.*

$$\tilde{\boldsymbol{f}} := \boldsymbol{B}\,\boldsymbol{F}_{\sigma N}^1\,\boldsymbol{D}\,\hat{\boldsymbol{f}} \quad (\hat{\boldsymbol{f}} \in \mathbb{R}^N)\,,$$

wobei $\boldsymbol{D} \in \mathbb{C}^{\sigma N \times N}$ und $\boldsymbol{B} \in \mathbb{R}^{N \times \sigma N}$ durch (1.19) – (1.21) *gegeben sind, kann der Rundungsfehler vom NFFT-Algorithmus* 1.1 *durch die Ungleichung*

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

beschrieben werden, wobei

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

ist.

Beweis: 1. Zunächst schätzen wir die Spektralnorm von $\boldsymbol{D}$ und $\boldsymbol{B}$ ab ([66, S. 120]). Nach Voraussetzung und mit (1.19) sehen wir sofort

$$\|\boldsymbol{D}\|_2 \;=\; \max_{k \in I_N^1} |\sigma N\, c_k(\tilde{\varphi})|^{-1} \;=\; (\sigma N\,|c_{N/2}(\tilde{\varphi})|)^{-1} \;=\; |\hat{h}(\pi/\sigma)|^{-1}\,. \tag{1.58}$$

Weil $\tilde{\psi}$ eine gerade, monoton fallende Funktion für $x \in [0, \frac{1}{2}]$ ist, gilt die Abschätzung

$$\frac{1}{\sigma N} \sum_{l \in I_{\sigma N}^1} \tilde{\psi}^2\left(x_j - \frac{l}{\sigma N}\right) \le \frac{1}{\sigma N}\,\tilde{\psi}^2(0) \;+\; \int_{-1/2}^{1/2} \tilde{\psi}^2(x)\,\mathrm{d}x\,.$$

Damit folgt aus der Definition von $\tilde{\psi}$ die Ungleichung

$$\begin{aligned} \sum_{l \in I_{\sigma N}^1} \tilde{\psi}^2(x_j - \frac{l}{\sigma N}) \;&\le\; h^2(0) + \sigma N \int_{-m/(\sigma N)}^{m/(\sigma N)} h^2(\sigma N x)\,\mathrm{d}x \\ &\le\; h^2(0) + ||h||_{L_2}^2\,. \end{aligned} \tag{1.59}$$

Aus der Definition (1.21) für die schwach besetzte Matrix

$$\boldsymbol{B} = (b_{j,k})_{j=-N/2,k=-\sigma N/2}^{N/2-1,\sigma N/2-1}\,, \quad b_{j,k} := \tilde{\psi}\left(x_j - \frac{k}{\sigma N}\right)\,,$$

ergibt sich für die j-te Komponente $(\boldsymbol{By})_j$ von $\boldsymbol{By}$ ($\boldsymbol{y} = (y_k)_{k=-\sigma N/2}^{\sigma N/2-1} \in \mathbb{C}^{\sigma N}$) die Abschätzung

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

Benutzen wir die Ungleichung (1.59), so erhalten wir

$$\sum_{r=1}^{2m} b_{j,k_r}^2 \;\leq\; \sum_{k\in I_{\sigma N}^1} \tilde{\psi}(x_j - \frac{k}{\sigma N})^2 \;\leq\; h^2(0) + ||h||_{L_2}^2$$

und damit

$$|(\boldsymbol{By})_j|^2 \leq (h^2(0) + ||h||_{L_2}^2) \sum_{r=1}^{2m} y_{k_r}^2\,. \tag{1.60}$$

Nach der Voraussetzung des Satzes enthält jedes „Fenster“ $[-\frac{m}{\sigma N} + \frac{l}{\sigma N}, \frac{m}{\sigma N} + \frac{l}{\sigma N})$ ($l \in I_n^1$) höchstens γ/σ Knoten x_j ($j \in I_N^1$). Damit enthält jede Spalte von $\boldsymbol{B}$ höchstens γ/σ Einträge ungleich Null, so dass mit (1.60) die Ungleichung

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

und schließlich

$$\|\boldsymbol{B}\|_2 \;\leq\; \sqrt{\frac{\gamma}{\sigma}\,\left(h^2(0) + ||h||_{L_2}^2\right)} \;=:\tilde{\beta} \tag{1.61}$$

gilt.
2. Es ist einfach einzusehen, dass mit (1.55) und (1.58) die Ungleichung

$$\|\mathrm{fl}(\boldsymbol{D}\hat{\boldsymbol{f}}) - \boldsymbol{D}\hat{\boldsymbol{f}}\|_2 \;\leq u\,|\hat{h}(\pi/\sigma)|^{-1}\,\|\hat{\boldsymbol{f}}\|_2 \tag{1.62}$$

gilt. Aus (1.58) ergibt sich

$$\|\mathrm{fl}(\boldsymbol{D}\hat{\boldsymbol{f}})\|_2 \;\leq \|\mathrm{fl}(\boldsymbol{D}\hat{\boldsymbol{f}}) - \boldsymbol{D}\hat{\boldsymbol{f}}\|_2 + \|\boldsymbol{D}\hat{\boldsymbol{f}}\|_2 \leq |\hat{h}(\pi/\sigma)|^{-1}(u+1)\,\|\hat{\boldsymbol{f}}\|_2\,. \tag{1.63}$$

3. Setzen wir $\hat{\boldsymbol{y}} := \mathrm{fl}(\boldsymbol{F}_{\sigma N}(\mathrm{fl}(\boldsymbol{D}\boldsymbol{f}))$ und $\boldsymbol{y} := \boldsymbol{F}_{\sigma N}\boldsymbol{D}\hat{\boldsymbol{f}}$, dann können wir

$$\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2 \leq \|\hat{\boldsymbol{y}} - \boldsymbol{F}_{\sigma N}(\mathrm{fl}(\boldsymbol{D}\hat{\boldsymbol{f}}))\|_2 \,+\, \|\boldsymbol{F}_{\sigma N}(\mathrm{fl}(\boldsymbol{D}\hat{\boldsymbol{f}})) - \boldsymbol{F}_{\sigma N}\boldsymbol{D}\hat{\boldsymbol{f}}\|_2$$

abschätzen, so dass mit (1.57), (1.62) und (1.63) die Ungleichung

$$\begin{aligned} \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2 &\leq \left(u(4+\sqrt{2})\sqrt{\sigma N}\log_2(\sigma N) + \mathcal{O}(u^2)\right)\|\mathrm{fl}(\boldsymbol{D}\hat{\boldsymbol{f}})\|_2 \\ &\quad + \sqrt{\sigma N}\ \|\mathrm{fl}(\boldsymbol{D}\hat{\boldsymbol{f}}) - \boldsymbol{D}\hat{\boldsymbol{f}}\|_2 \\ &\leq |\hat{h}(\pi/\sigma)|^{-1}\left(u(4+\sqrt{2})\sqrt{\sigma N}\log_2(\sigma N) + \sqrt{\sigma N}u + \mathcal{O}(u^2)\right)\|\hat{\boldsymbol{f}}\|_2 \end{aligned} \tag{1.64}$$

folgt.
4. Schließlich betrachten wir den Fehler zwischen $\mathrm{fl}(\tilde{\boldsymbol{f}}) := \mathrm{fl}(\boldsymbol{B}\hat{\boldsymbol{y}})$ und $\tilde{\boldsymbol{f}} := \boldsymbol{B}\boldsymbol{y}$. Mittels (1.61) und (1.64) erhalten wir

$$\begin{aligned}\|\mathrm{fl}(\tilde{\boldsymbol{f}}) - \tilde{\boldsymbol{f}}\|_2 &\leq \|\mathrm{fl}(\boldsymbol{B}\hat{\boldsymbol{y}}) - \boldsymbol{B}\hat{\boldsymbol{y}}\|_2 + \|\boldsymbol{B}(\hat{\boldsymbol{y}} - \boldsymbol{y})\|_2 \\ &\leq \|\mathrm{fl}(\boldsymbol{B}\hat{\boldsymbol{y}}) - \boldsymbol{B}\hat{\boldsymbol{y}}\|_2 + \tilde{\beta}\,|\hat{h}(\pi/\sigma)|^{-1} \times \\ &\quad \left(u(4+\sqrt{2})\sqrt{\sigma N}\log_2(\sigma N) + \sqrt{\sigma N}u + \mathcal{O}(u^2)\right)\|\boldsymbol{f}\|_2 . \end{aligned} \tag{1.65}$$

Aus (1.53), (1.55) und (1.21) folgt mit [66, S. 76], dass

$$|\mathrm{fl}(\boldsymbol{B}\hat{\boldsymbol{y}}) - \boldsymbol{B}\hat{\boldsymbol{y}}| \leq \frac{2mu}{1-2mu}\,\boldsymbol{B}\left(|\hat{y}_k|\right)_{k=-\sigma N/2}^{\sigma N/2-1}$$

und mit (1.61) schließlich die Ungleichung

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

Mit den Ungleichungen (1.63) und (1.58) wird abgeschätzt

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

Folglich gilt

$$\|\mathrm{fl}(\boldsymbol{B}\hat{\boldsymbol{y}}) - \boldsymbol{B}\hat{\boldsymbol{y}}\|_2 \leq \left(2m\,\tilde{\beta}\,\sqrt{\sigma N}\,|\hat{h}(\pi/\sigma)|^{-1}\,u + \mathcal{O}(u^2)\right)\|\hat{\boldsymbol{f}}\|_2 . \tag{1.66}$$

Zusammen mit (1.65) ergibt dies die Behauptung des Satzes. ■

Für „gleichmäßig" verteilte Knoten x_j ist stets $\gamma \approx 2m$.
Ein Algorithmus für die Berechnung der NDFT heißt robust, falls für alle $\hat{\boldsymbol{f}} \in \mathbb{R}^N$ eine positive Konstante k_N mit $k_N\,u \ll 1$ existiert, so dass

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

Aus Satz 1.16 folgt, dass der NFFT-Algorithmus 1.1 robust ist. In Abschnitt 1.8, Beispiel 1.4 werden wir diese theoretische Aussage durch numerische Ergebnisse bestätigen.

1.7 Iterative Rekonstruktion

In den vorigen Abschnitten wurden ausschließlich Matrix-Vektor-Multiplikationen behandelt und die Frage der „inversen Transformation" noch nicht erläutert. Wir beschränken uns in diesem Abschnitt wieder auf die NFFT. Analoge Aussagen gelten aber auch für die NFCT und die NFST. Um die Problematik zu erklären, beginnen wir mit dem eindimensionalen Problem. Zu den gegebenen Daten $f_j := f(x_j)$ $(j = -M/2, \ldots, M/2-1)$ des trigonometrischen Polynoms

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

sollen die Fourier-Koeffizienten $\hat{f}_k$ $(k = -N/2, \ldots, N/2-1)$ berechnet werden. Für den Spezialfall $N = M$ bedeutet dies, dass ein lineares Gleichungssystem mit der nichtäquidistanten Fourier-Matrix

$$\boldsymbol{A} := \left(\mathrm{e}^{2\pi \mathrm{i} k x_j / N} \right)_{j=-N/2, k=-N/2}^{N/2-1, N/2-1} \qquad \in \mathbb{C}^{N \times N}$$

zu lösen ist. Für äquidistante Punkte $x_j := j\pi/N$ erhalten wir die Fourier-Matrix $\boldsymbol{F}_N^1 = N \bar{\boldsymbol{F}}_{N,N}^1$ und damit ist $\boldsymbol{A} = \bar{\boldsymbol{F}}_N^1$ fast unitär. Für sehr kleine Änderungen dieser äquidistanten Punkte bleibt die Konditionszahl der Matrix $\boldsymbol{A}$ in Bereichen, die ein numerisches Invertieren der Matrix erlauben. Schnelle Algorithmen wurden dazu mit Hilfe der Multipol-Methode entwickelt (siehe [32]). Numerische Ergebnisse zeigen aber, dass diese Algorithmen instabil werden, wenn die Punkte zu weit vom äquidistanten Gitter $j\pi/N$ $(j = -N/2, \ldots, N/2-1)$ abweichen (siehe [28]). Verallgemeinerungen der Ergebnisse von [32] auf den mehrdimensionalen Fall sind nicht bekannt. Außerdem ist für viele Anwendungen die Forderung $M = N$ zu einschränkend.

Deshalb wollen wir einen anderen Lösungsweg vorschlagen und formulieren diesen gleich für den d-dimensionalen Fall:

$$\text{Gesucht ist ein } \hat{\boldsymbol{f}} \in \mathbb{C}^{N^d} \text{ mit } \|\boldsymbol{f} - \boldsymbol{A}\hat{\boldsymbol{f}}\|_2 \le \|\boldsymbol{f} - \boldsymbol{A}\boldsymbol{g}\|_2 \text{ für alle } \boldsymbol{g} \in \mathbb{C}^{N^d}\,. \tag{1.67}$$

Dabei ist der Vektor $\hat{\boldsymbol{f}}$ durch (1.3), die nichtäquidistante Fourier-Matrix $\boldsymbol{A}$ durch (1.4) definiert und $\boldsymbol{f}$ ist der Vektor der gegebenen Daten $\boldsymbol{f} = (f(\boldsymbol{x}_j))_{j \in I_M^1}$.
Falls die Dimension N^d kleiner als die Anzahl M der gegebenen Punkte, d.h. falls $N^d \le M$ ist, ergibt sich die Lösung des Approximationsproblems (1.67) durch die Normalengleichung (siehe [12, Kapitel 1])

$$\boldsymbol{A}^{\mathrm{H}} \boldsymbol{A} \hat{\boldsymbol{f}} = \boldsymbol{A}^{\mathrm{H}} \boldsymbol{f}\,. \tag{1.68}$$

Nehmen wir an, dass die Matrix $\boldsymbol{A}$ vollen Rang besitzt, so erhalten wir die Lösung $\hat{\boldsymbol{f}}$ für das Kleinste-Quadrate-Problem durch Invertieren der Matrix $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{A}$.
Im Fall $N^d \ge M$ formulieren wir das Approximationsproblem (1.67) zu einem speziellen Optimierungsproblem (siehe [12, Kapitel 1]):

$$\text{Gesucht ist ein } \hat{\boldsymbol{f}} \in \mathbb{C}^{N^d} \text{ mit } \min_{\hat{\boldsymbol{f}} \in \mathbb{C}^{N^d}} \|\hat{\boldsymbol{f}}\|_2 \text{ unter der Nebenbedingung } \boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{f}\,.$$

Es sei auch hier wieder vorausgesetzt, dass die Matrix $\boldsymbol{A}$ vollen Rang besitzt, so ergibt sich die Lösung $\hat{\boldsymbol{f}}$ eindeutig aus der Lösung $\boldsymbol{y}$ des linearen Gleichungssystems

$$\boldsymbol{A}\boldsymbol{A}^{\mathrm{H}} \boldsymbol{y} = \boldsymbol{f} \text{ mit } \hat{\boldsymbol{f}} = \boldsymbol{A}^{\mathrm{H}} \boldsymbol{y}\,. \tag{1.69}$$

Für viele Anwendungen ist es weder möglich noch nötig obige Gleichungssysteme exakt zu lösen. Da wir eine schnelle Matrix-Vektor-Multiplikation für die Matrizen $\boldsymbol{A}$ bzw. $\boldsymbol{A}^{\mathrm{T}}$ in Algorithmus 1.1 bzw. 1.3 zur Verfügung haben, bietet sich ein iteratives Lösen der Gleichungssysteme (1.68) und (1.69) an.
Vom System (1.68) abgeleitete iterative Methoden werden oft mit den Buchstaben NR (N für „Normal" und R für „Residual") bezeichnet, während Methoden die vom System (1.69) ausgehen, mit den Buchstaben NE (N für „Normal" und E für „Error") bezeichnet werden. Damit ist CGNR das Verfahren der konjugierten Gradienten angewandt auf das System (1.68) und CGNE das Verfahren der konjugierten Gradienten angewandt auf das System (1.69) (siehe [109]).
Es gibt verschiedene mathematisch äquivalente Möglichkeiten, das CG-Verfahren für die Normalengleichungen zu implementieren. In exakter Arithmetik erzeugen diese Verfahren alle die gleiche Folge von Approximationen an die Lösung, doch in endlicher Arithmetik können sie sehr unterschiedliche Näherungen liefern. Es sei hier betont, dass ein CG-Verfahren für symmetrisch positiv definite Matrizen nicht direkt auf eine schlecht konditionierte Normalengleichung angewendet werden soll (siehe dazu die Diskussion in [92, Sec. 7.1]).
Die beiden Methoden CGNR und CGNE liefern Approximationen aus den gleichen affinen Krylov-Unterräumen, aber jeweils mit verschiedenen Optimalitätseigenschaften: kleinstes Residuum für CGNR und kleinster Fehler für CGNE.
Es sei an dieser Stelle bemerkt, dass der CGNR-Algorithmus außerdem ein „fehlerreduzierender" Algorithmus [80] ist. Ein Beispiel für eine iterative Rekonstruktion geben wir in Beispiel 1.3 an, verweisen aber auch auf die Diskussion am Ende des Kapitels sowie auf Bemerkung 3.4.

1.8 Numerische Ergebnisse

In diesem Abschnitt präsentieren wir einige numerische Beispiele. Die NFFT-Algorithmen (Algorithmus 1.1 und Algorithmus 1.3) wurden von uns in C für die Fälle $d = 1, 2, 3$ implementiert und sind als Softwarepaket von der NFFT-Hompage

`http://www.math.uni-luebeck.de/potts/nfft`

abrufbar [76]. Dieses Programmpaket benutzt die FFTW-Bibliothek (siehe [47]).
Unser Softwarepaket testeten wir auf verschiedenen Plattformen. In den anschließenden Beispielen beziehen sich die numerischen Ergebnisse auf Tests, die auf einem Pentium 3 Rechner mit 256 MB und SuSe-Linux 8.0 durchgeführt wurden. In allen Fällen sind die Knoten $\boldsymbol{x}_j$ und die Fourier-Koeffizienten $\hat{f}_{\boldsymbol{k}}$ als Zufallszahlen mit $\boldsymbol{x}_j \in [-0.5, 0.5]^d$ und $\hat{f}_{\boldsymbol{k}} \in [0, 1]$ gewählt. Eine ausführliche Beschreibung von Testbeispielen haben wir bereits in der Dokumentation zu unserem Softwarepaket in [78] veröffentlicht.
Insbesondere sind wir in der Lage, durch wenige Modifikationen im Programmcode verschiedene Ansatzfunktionen φ zu testen. Dies erlaubt eine einfache Anpassung für eine Reihe von Anwendungen. Im Folgenden wird sich auf die vier Ansatz-Funktionen, für die wir auch in Abschnitt 1.3 die Fehlerabschätzungen angegeben haben, beschränkt.

Beispiel 1.1 (Genauigkeit)
Wir interessieren uns für die Fehler

$$E_2 := \frac{\|\boldsymbol{f}-\boldsymbol{s}\|_2}{\|\boldsymbol{f}\|_2} = \left(\sum_{j\in I_M^1} |f_j - s(\boldsymbol{x}_j)|^2 / \sum_{\boldsymbol{j}\in I_M^1} |f_j|^2 \right)^{\frac{1}{2}}$$

und

$$E_\infty := \frac{\|\boldsymbol{f}-\boldsymbol{s}\|_\infty}{\|\hat{\boldsymbol{f}}\|_1} = \max_{j\in I_M^1} |f_j - s(\boldsymbol{x}_j)| / \sum_{\boldsymbol{k}\in I_N^d} |\hat{f}_{\boldsymbol{k}}|$$

von Algorithmus 1.1. Algorithmus 1.3 liefert den gleichen Approximationsfehler (siehe Abschnitt 1.3). Wir untersuchen neben dem E_2-Fehler auch den E_∞-Fehler, weil wir in Satz 1.5, Satz 1.7, Satz 1.8 und Satz 1.10 genau diesen Fehler beschrieben haben.
Abbildung 1.2 und Abbildung 1.3 zeigen die Fehler E_2 und E_∞ im univariaten Fall für doppelte und einfache Rechengenauigkeit. Wir haben $N = 1024$, $M = 2000$, $\sigma = 2$ und verschiedene Parameter m gewählt. Abbildung 1.4 und Abbildung 1.5 zeigen die Fehler im bivariaten Fall $d = 2$ und im Fall $d = 3$. Um die Resultate der Abbildungen nachvollziehbar zu gestalten, haben wir die Testprogramme in unserem Softwarepaket zur NFFT [76] im Unterverzeichnis `./test_cases/test_error.c` abgelegt. Für alle vier Ansatz-Funktionen φ, die B-Splines (1.31), die Gauß-Funktionen (1.32), die Sinc-Funktionen (1.37) und die Kaiser-Bessel-Funktionen (1.38), beobachten wir einen exponentiellen Abfall der Fehler E_2 und E_∞ für wachsenden Abschneideparameter m. Damit bestätigen sich unsere Fehlerabschätzungen in Abschnitt 1.3 (vgl. Satz 1.11) durch numerische Tests. □

Beispiel 1.2 (CPU-Zeiten)
Im Folgenden sind wir an CPU-Zeiten unserer Algorithmen interessiert. Wir vergleichen die Laufzeiten der Routinen NDFT und NFFT in doppelter Genauigkeit für die Funktionen sinc^{2m} mit den Parametern $m = 5$, $\sigma = 2$. Mit der Routine NDFT bezeichnen wir dabei eine Realisierung des langsamen Algorithmus. Die Ergebnisse für diesen Test zeigen wir in Abbildung 1.6 für den univariaten Fall $d = 1$ (links) und den bivariaten Fall $d = 2$ mit $M = N$, wobei keine Werte vorberechnet wurden (rechts). Diese Tests kann man wieder in [76] unter `./test_cases/test_time.c` finden. Die Abbildungen belegen, dass sich in unseren Implementierungen die Laufzeit der Routinen proportional zur angegebenen arithmetischen Komplexität verhält.
Da sich die Algorithmen 1.1 und 1.3 mit verschiedenen Ansatzfunktionen nur im ersten und dritten Schritt unterscheiden, untersuchen wir die CPU-Zeiten für diese Schritte mit verschiedenen Ansatzfunktionen gesondert (siehe Abbildung 1.7). Diese Testbeispiele zeigen, dass wir mit den schnellen Algorithmen zur NFFT erstmals Problemgrößen behandeln können, die mit der NDFT zur Zeit undenkbar sind. □

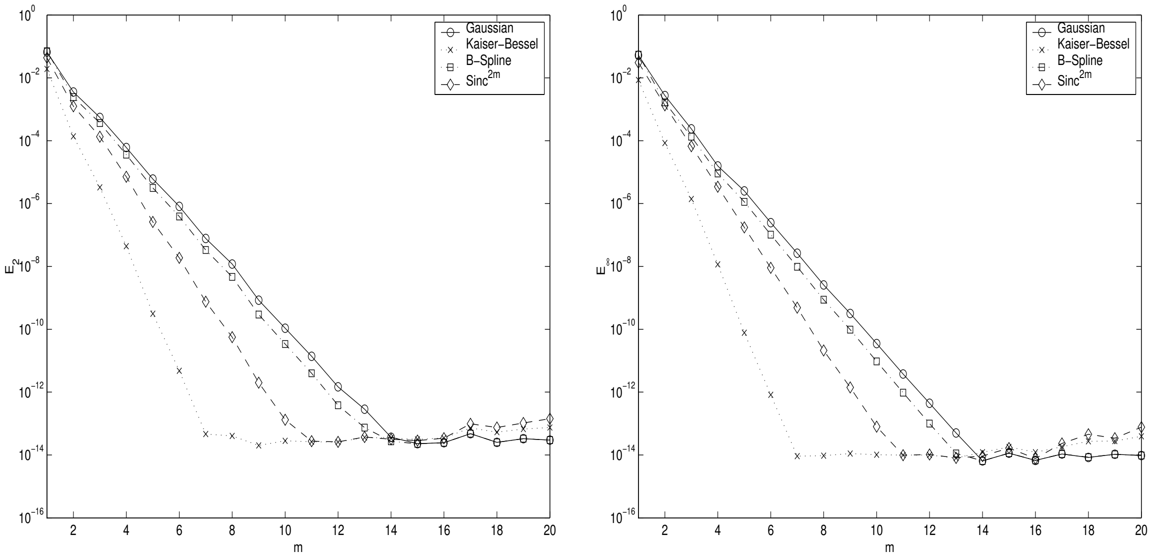

Abbildung 1.2: Die Fehler E_2 (links) und E_∞ (rechts) für den univariaten Fall $d = 1$ mit den Parametern $N = 1024$, $M = 2000$, $\sigma = 2$ bei Rechnungen mit doppelter Genauigkeit.

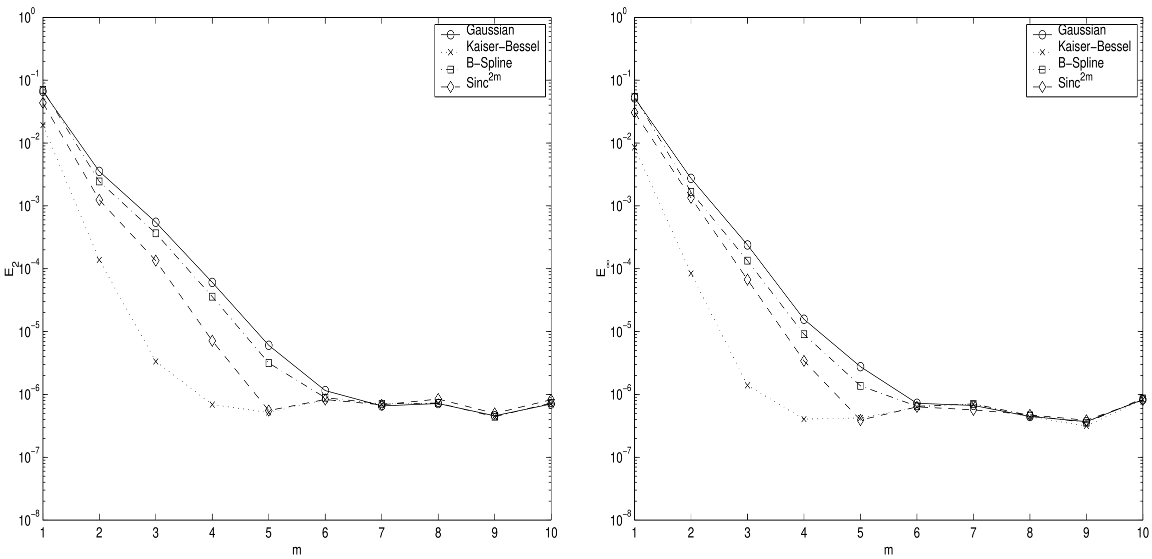

Abbildung 1.3: Die Fehler E_2 (links) und E_∞ (rechts) für den univariaten Fall $d = 1$ mit den Parametern $N = 1024$, $M = 2000$, $\sigma = 2$ bei Rechnungen mit einfacher Genauigkeit.

Beispiel 1.3 (Fehlende Pixel)
In diesem Beispiel testen wir den CGNR Algorithmus, um ein Bild mit fehlenden Pixeln zu rekonstruieren (siehe [58]), wobei wir die Matrix-Vektor-Multiplikation wieder

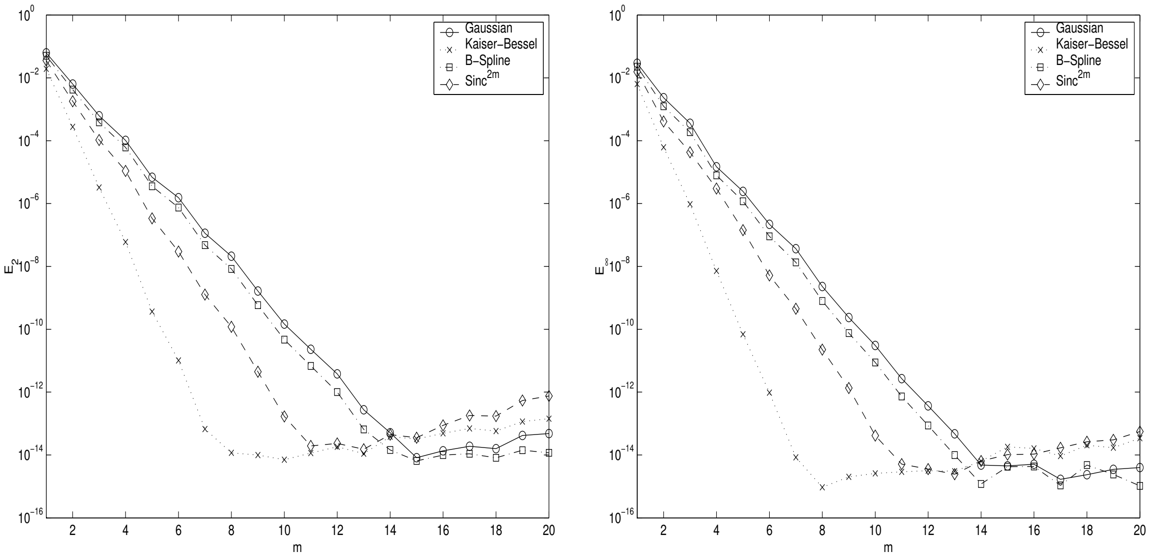

Abbildung 1.4: Die Fehler E_2 (links) und E_∞ (rechts) für den bivariaten Fall $d = 2$ mit den Parametern $N = 128$, $M = 10000$, $\sigma = 2$ bei Rechnungen mit doppelter Genauigkeit.

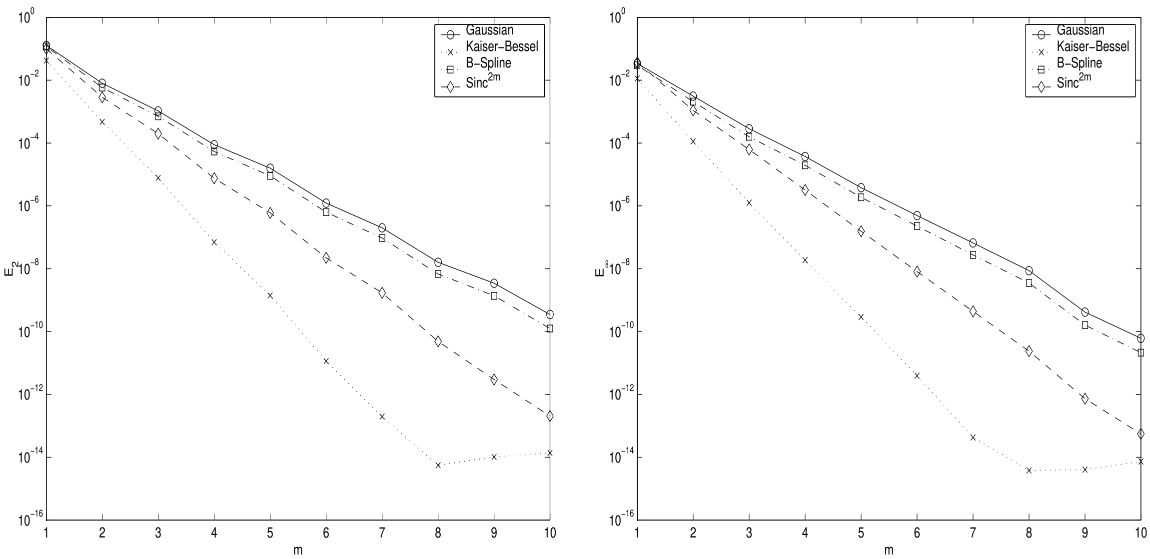

Abbildung 1.5: Die Fehler E_2 (links) und E_∞ (rechts) für den Fall $d = 3$ mit den Parametern $N = 16$, $M = 4000$, $\sigma = 2$ bei Rechnungen mit doppelter Genauigkeit.

mit Algorithmus 1.1 und 1.3 realisieren. Unser Testbild (siehe Abbildung 1.8 (links)) ist ein Ausschnitt der Größe 256×256 eines Bildes der Gatlinburg Konferenz (siehe online Reference in MATLAB). Das Bild ist an 40000 zufällig gewählten Punkten ab-

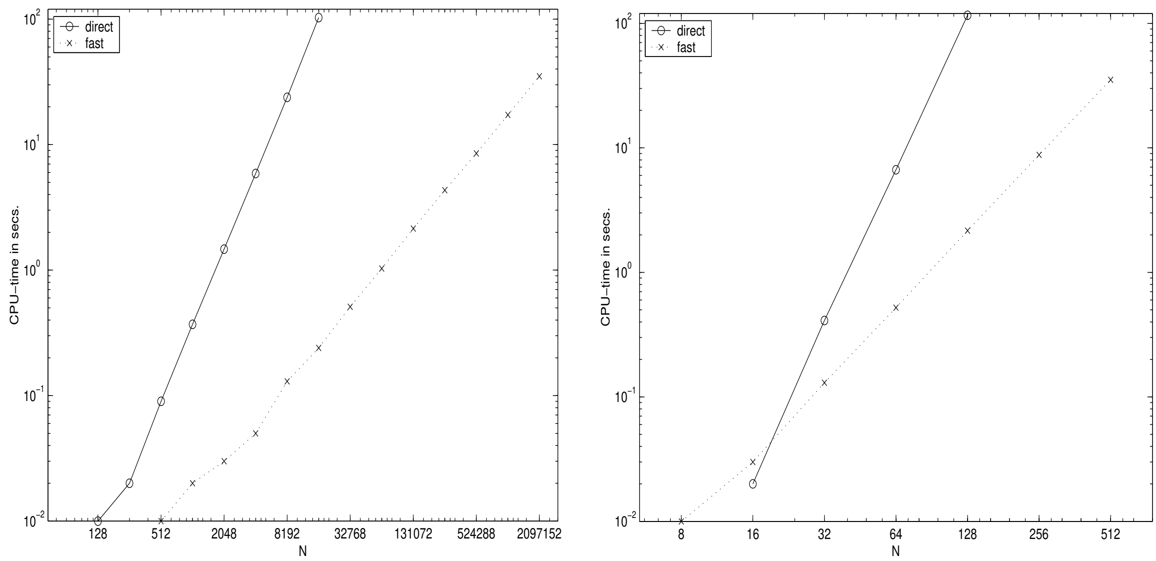

Abbildung 1.6: Die CPU-Zeiten mit den Optionen doppelte Genauigkeit, $\varphi = \mathrm{sinc}^{2m}$, den Parametern $m = 5$, $\sigma = 2$ und keiner Vorberechnung; für den univariaten Fall $d = 1$, `FFTW_OUT_OF_PLACE` und $M = N$ (links) und den bivariaten Fall $d = 2$, `FFTW_IN_PLACE` und $M = N^2$ (rechts).

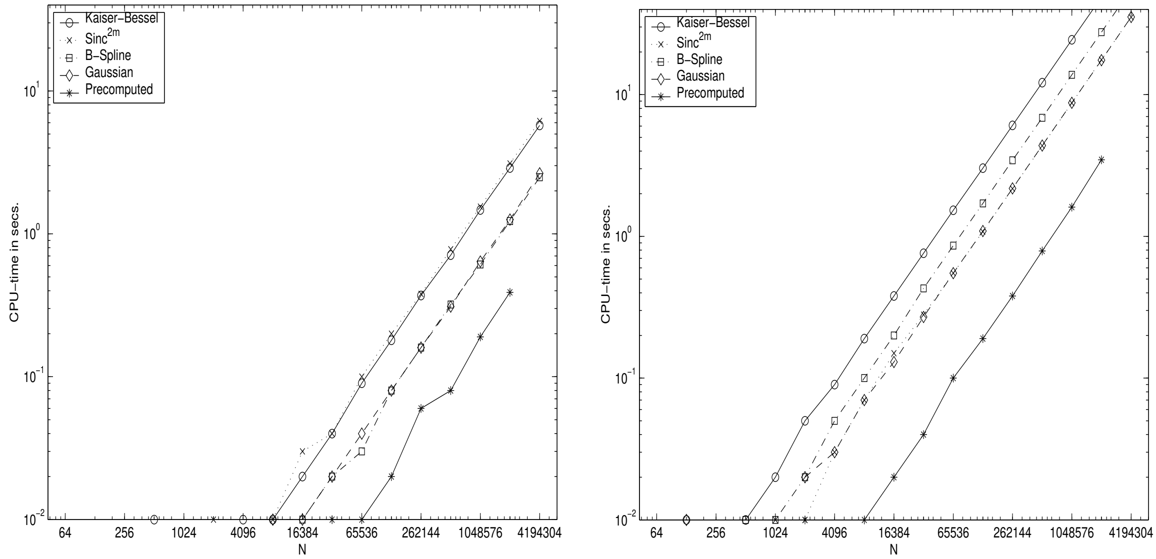

Abbildung 1.7: Die CPU-Zeiten mit den Optionen doppelte Genauigkeit, $d = 1$, den Parametern $m = 5$, $\sigma = 2$, $M = N$ und keiner Vorberechnung; für Schritt 1 von Algorithmus 1.1 (links) und Schritt 3 von Algorithmus 1.1 (rechts).

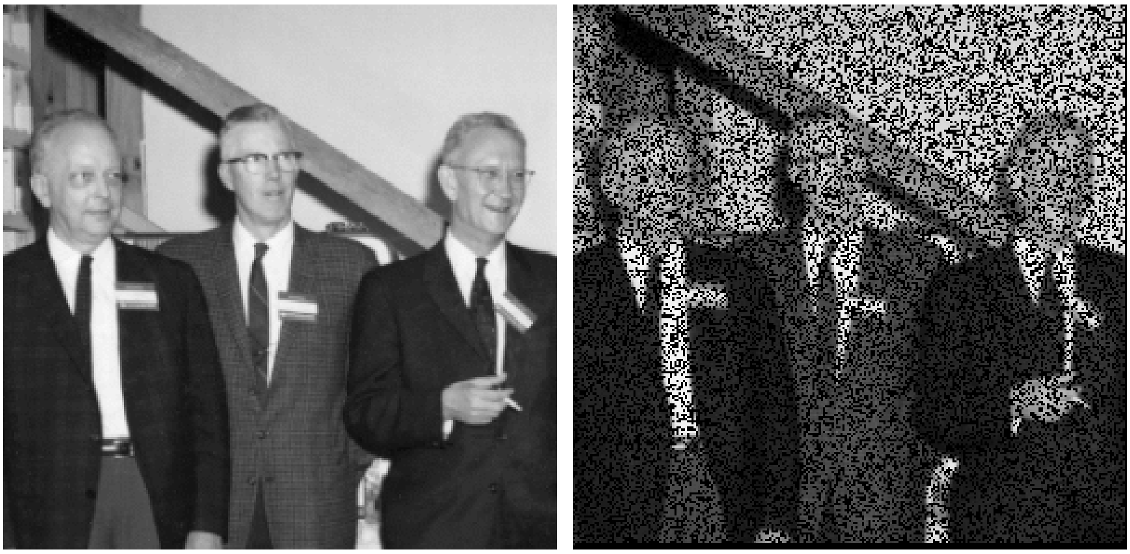

Abbildung 1.8: Originalbild (links), 39 % fehlende Bildpunkte (rechts).

Abbildung 1.9: Rekonstruiertes Bild nach 10 Iterationen.

getastet worden, d.h. etwa 39% der Bildpunkte blieben unberücksichtigt. Diese sind als schwarze Punkte in Abbildung 1.8 (rechts) gedruckt. Nach 10 Iterationen berechnen wir mittels bivariater FFT das rekonstruierte Bild (siehe Abbildung 1.9). Die Randeffekte, die durch den Ansatz mit periodischen trigonometrischen Polynomen entstehen, lassen sich vermeiden, wenn wir Kosinus-Polynome auf $[0, \pi]$ benutzen. $\square$

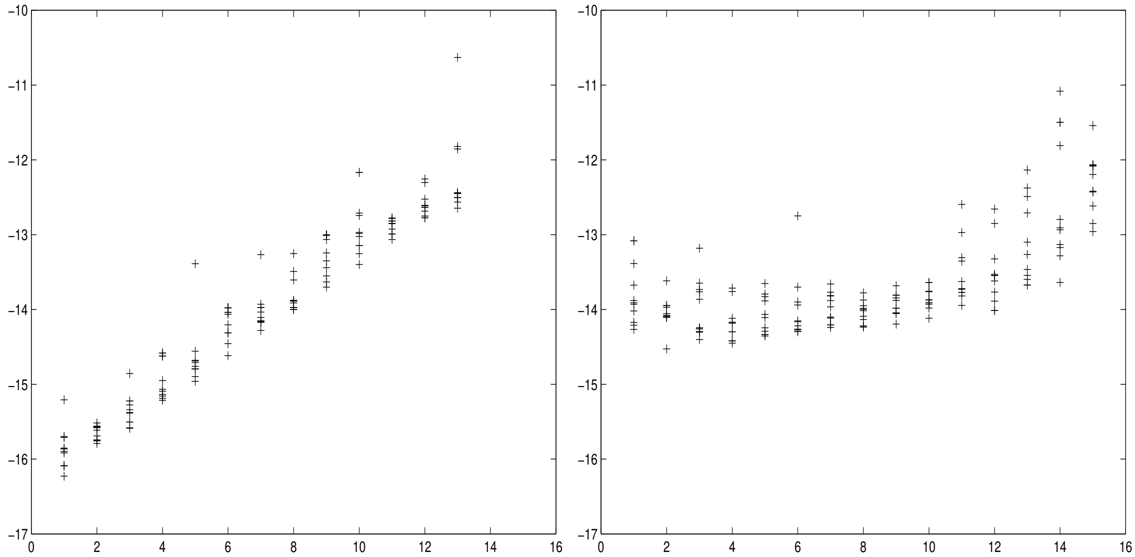

Abbildung 1.10: Links: $(t, E_{\mathrm{C}}(t))$ für $t = 1, \ldots, 13$, rechts: $(t, E_{\mathrm{NFFT}}(t))$ für $t = 1, \ldots, 15$.

Beispiel 1.4 (Rundungsfehler)
Schließlich wird gezeigt, dass wir die theoretischen Ergebnisse aus Satz 1.16 durch numerische Berechnungen bestätigen können. Wir benutzen in Algorithmus 1.1 die Gauß-Funktionen und setzen $\sigma = 2$. Weiterhin sei $\tilde{\boldsymbol{f}}_{\mathrm{C}} \in \mathbb{C}^{2^t}$ der Vektor, welcher mittels Kaskadensummation in (1.1) berechnet wird. Wir definieren

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

und zeigen in Abbildung 1.10 (links) den Fehler $E_{\mathrm{C}}(t)$ für 10 numerische Tests mit verschiedenen, zufällig gewählten Knoten x_j als Funktion der Transformationslänge $N = 2^t$. Zum Vergleich zeigt Abbildung 1.10 (rechts) den entsprechenden Fehler

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

der durch den Algorithmus 1.1 entsteht (vgl. [105]). □

Weitere Themen und Literaturhinweise

NFFT

In den letzten Jahren sind sehr viele Arbeiten zur schnellen Approximation für die NFFT [31, 8, 93, 35, 34, 128, 29, 87, 42, 45, 89] entstanden. Der in Abschnitt 1.1 beschriebene Ansatz basiert auf den Arbeiten [116, 104] und erlaubt erstmalig einen einheitlichen Zugang zur NFFT. Damit ist es möglich, die vorhandenen Arbeiten einzuordnen, zu vergleichen und neue Ansätze zu beschreiben. In verschiedenen Arbeiten wurden weitere Ansatzfunktionen φ betrachtet, so z.B.:

- Gauß-Funktionen mit einem Hanning-Fenster [29],
- Gauß-Funktionen kombiniert mit sinc-Kernen [93],
- besonders optimierte Fensterfunktionen [67, 29].

Weiterhin wurden spezielle Zugänge, basierend auf „Skalierungsvektoren“ [87], basierend auf Minimierung der Frobenius-Norm bestimmter Fehlermatrizen [89] oder basierend auf Min-Max-Interpolationen [42], vorgeschlagen. Numerische Ergebnisse in [89, 42] zeigen aber, dass diese Zugänge unserem Algorithmus 1.1 mit Kaiser-Bessel-Funktionen als Ansatzfunktionen φ nicht überlegen sind.
Zweifellos ist es ein Verdienst von A. Dutt, V. Rokhlin [31] und G. Beylkin [8], erstmals den genauen Zusammenhang zwischen arithmetischer Komplexität und Approximationseigenschaften angegeben zu haben. Dennoch wurden ähnliche Ideen, die als Gridding-Methoden bekannt sind, schon in früheren Arbeiten veröffentlicht (siehe z.B. [91, 114, 67, 111]).
In den letzten Jahren erschienen auch umfangreiche numerische Tests der NFFT (siehe dazu auch die Diplom- bzw. Studienarbeiten [28, 34, 75, 13]), wobei diese Algorithmen in verschiedenen Bereichen angewandt werden. Wir gehen in den folgenden Kapiteln darauf ein. Nach vielen weiteren umfangreichen Tests haben wir uns entschlossen, die verschiedenen implementierten Algorithmen konzeptionell zu überarbeiten, zu erweitern und als Softwarepaket in [76] zu veröffentlichen. Weitere Software für die NFFT, basierend auf [42], ist als Matlab-Paket von [41] abrufbar.

Iterative Rekonstruktion

Für die iterative Rekonstruktion (siehe Abschnitt 1.7) ist die Existenz schneller Algorithmen eine Voraussetzung. Die unregelmäßige Abtastung von Signalen führt sehr oft zu schlecht konditionierten Systemen. Im Allgemeinen tritt dieses Phänomen auf, wenn die Abtastmenge große „Löcher“ hat. Um diese großen linearen Gleichungssysteme zu lösen, ist die CG-Methode ein sehr effizientes Verfahren. Außerdem haben viele CG-ähnliche Methoden regularisierende Eigenschaften [62].

Schnelle, direkte und effiziente Algorithmen zur Lösung von (1.67) mit reellen Funktionen f wurden in [37] für den univariaten Fall $d = 1$ entwickelt. Die Algorithmen basieren auf der Lösung von Eigenwertproblemen und benötigen $\mathcal{O}(MN)$ arithmetische Operationen. In [108] wurde ein Zugang mittels QR-Zerlegung veröffentlicht. H. Feichtinger, K. Gröchenig und T. Strohmer veröffentlichten in [39] eine neue (“second generation”) Rekonstruktionsmethode für irregulär abgetastete Daten. Diese so genannte ACT-Methode ist eine Kombination der adaptiven Gewichtsmethode [38] und der CG-Methode zur Lösung der positiv definiten Gleichungssysteme (1.68). Die Autoren nutzten den Fakt, dass für den univariaten Fall $d = 1$ die Matrix $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{W}\boldsymbol{A}$ ($\boldsymbol{W}$ ist eine Diagonalmatrix von Gewichten) eine Toeplitz-Matrix ist (siehe auch [31]). Insbesondere konnten sie die Konditionszahl für den univariaten Fall $d = 1$ und in einigen besonderen Fällen für den bivariaten Fall $d = 2$ (siehe [58]) abschätzen. Im Gegensatz zur ACT-Methode verzichtet unsere Methode auf das explizite Aufstellen der Normalengleichung (siehe auch [96]).

2 Schnelle Summation radialer Funktionen

In diesem Kapitel entwickeln wir neue Algorithmen für die schnelle, approximative Berechnung der Summen

$$f(\boldsymbol{y}) = \sum_{k=1}^{N} \alpha_k \mathcal{K}(\boldsymbol{y} - \boldsymbol{x}_k) \tag{2.1}$$

an den Punkten $\boldsymbol{y} = \boldsymbol{y}_j \in \mathbb{R}^d$ $(j = 1, \ldots, M)$ für gegebene Knoten $\boldsymbol{x}_k \in \mathbb{R}^d$ $(k = 1, \ldots, N)$ und gegebene Koeffizienten $\alpha_k \in \mathbb{C}$ $(k = 1, \ldots, N)$. Wir werden uns hier auf reellwertige, radialsymmetrische Kerne $\mathcal{K}$, d.h. $\mathcal{K}(\boldsymbol{x}) = K(\|\boldsymbol{x}\|_2)$, beschränken. Dabei ist K eine univariate Funktion.

Die schnelle Berechnung von speziell strukturierten, diskreten Summen ist eine häufig wiederkehrende Grundaufgabe der numerischen Mathematik. Bekannte Kerne, die z.B. beim Lösen von Integralgleichungen auftreten, sind die Singularitätenfunktionen $K(x) = \log|x|$, $K(x) = 1/x$ oder $K(x) = x^2 \log(x)$ des zwei- bzw. dreidimensionalen Laplaceoperators oder des biharmonischen Operators. Weitere bekannte Kerne wie z.B. die Multiquadric $K(x) = \sqrt{x^2 + c^2}$ oder die inverse Multiquadric $K(x) = 1/\sqrt{x^2 + c^2}$ treten im Zusammenhang mit der Approximation durch radiale Basisfunktionen auf (siehe z.B. [110]). Ein anderer, häufig benutzter glatter Kern, der Gauß-Kern $K(x) = \mathrm{e}^{-\delta x^2}$, wird beim Lösen der Wärmeleitungsgleichung [52], in der Bildverarbeitung [36], in der Strömungsmechanik oder in der Finanzmathematik [17] verwendet. Die entsprechende Transformation (2.1) heißt dann die diskrete Gauß-Transformation.

Die bekanntesten Methoden zur effizienten Berechnung von (2.1) sind die schnellen Multipol-Methoden (Fast Multipole Methode, FMM). Die FMM entwickelten ursprünglich L. Greengard und V. Rokhlin [51, 53]. Diese Algorithmen benötigen nur $\mathcal{O}((N + M)(\log(1/\epsilon))$ arithmetische Operationen, wobei ϵ die gewünschte Genauigkeit der Berechnung ist. Um die Summen (2.1) mit der FMM schnell berechnen zu können, müssen die Multipol-Ideen für jeden Kern neu angepasst werden. Insbesondere haben R. Beatson u.a. die FMM für verschiedene Kerne, z.B. für den Thin Plate Spline $K(x) = x^2 \log|x|$, die Multiquadric $K(x) = \sqrt{x^2 + c^2}$ oder die inverse Multiquadric $K(x) = 1/\sqrt{x^2 + c^2}$ [6, 4] entwickelt. In den letzten Jahren wurden viele Modifikationen und Verbesserungen für die FMM vorgeschlagen (siehe z.B. [131, 121, 7, 22]). Algorithmen zur schnellen Berechnung der diskreten Gauß-Transformation sind in [54, 55, 4, 120] veröffentlicht.

Den gemeinsamen Hintergrund dieser Algorithmen bildet die Zerlegung des Kerns $\mathcal{K}$ in der Form

$$\mathcal{K}(\boldsymbol{y} - \boldsymbol{x}) = \sum_{k=1}^{q} \varphi_k(\boldsymbol{x}) \psi_k(\boldsymbol{y}) \; + \; \tilde{\mathcal{K}}(\boldsymbol{x}, \boldsymbol{y}), \tag{2.2}$$

wobei $\tilde{\mathcal{K}}$ klein für hinreichend große $q \ll N$ und die meisten Knoten $\boldsymbol{x}_k$ und $\boldsymbol{y}_j$ wird. Die Summe mit den separierten Funktionen an Stelle von $\mathcal{K}$ in (2.1), erlaubt dann eine effiziente Berechnung von f in (2.1) an den Knoten $\boldsymbol{y}_j$ $(j = 1, \ldots, M)$. Im Fall singulärer Kerne muss noch eine sogenannte Nahfeld-Berechnung $\tilde{\mathcal{K}}$ addiert werden, um Funktionsauswertungen bei sehr kleinem Abstand $\|\boldsymbol{y}_j - \boldsymbol{x}_k\|_2$ in (2.1) zu summieren.

So benutzt die zweidimensionale Multipol-Methode Taylor-Entwicklungen in (2.2), d.h. Polynome für φ_k und ψ_k, während die schnelle Gauß-Transformation in [54] Hermite-Taylor-Entwicklungen, d.h. Polynome und Hermite-Funktionen, verwendet.

In unseren Algorithmen zur schnellen Summation nutzen wir Fourier-Summen. Genauer, der Kern $\mathcal{K}$ wird in die Summe von Funktionen $\mathcal{K}_{\mathrm{NE}}$ ($\mathcal{K}$ im Nahfeld) mit einem kleinen Träger und glatten 1-periodischen Funktionen $\mathcal{K}_{\mathrm{R}}$ zerlegt. Den Kern $\mathcal{K}_{\mathrm{R}}$ nähern wir durch eine Fourier-Summe $\mathcal{K}_{\mathrm{RF}} := \sum_{\boldsymbol{k} \in I_n^d} b_{\boldsymbol{k}} \, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \cdot}$ an, wobei $\boldsymbol{k}$ aus der endlichen Index-Menge I_n^d ist. Dann verwenden wir die herausragende Eigenschaft der Exponentialfunktion $\mathrm{e}^{2\pi\mathrm{i}(\boldsymbol{y}-\boldsymbol{x})} = \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{y}} \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{x}}$, um die geforderte Separation der Knoten $\boldsymbol{x}$ und $\boldsymbol{y}$ zu erhalten. Wir können die Grundidee unseres neuen schnellen Summationsalgorithmus in der Formel

$$\mathcal{K}(\boldsymbol{y}-\boldsymbol{x}) \approx \sum_{\boldsymbol{k} \in I_n^d} b_{\boldsymbol{k}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{y}} \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} + \mathcal{K}_{\mathrm{NE}}(\boldsymbol{y}-\boldsymbol{x})$$

ausdrücken. Im Gegensatz zu (2.2) läuft der Summationsindex $\boldsymbol{k}$ über eine große Index-Menge I_n^d. Wir sind nun aber in der Lage, die Summe in (2.1) unter Nutzung der NFFT in $\mathcal{O}(M + N \log N)$ arithmetischen Operationen zu berechnen. Für glatte Kerne reduziert sich die arithmetische Komplexität sogar zu $\mathcal{O}(M+N)$. Die Konstante in diesen Komplexitätsaussagen hängt nur von der geforderten Exaktheit des Ergebnisses ab.

Aus Sicht der linearen Algebra lässt sich die Summe (2.1) als Matrix-Vektor-Produkt mit der speziell strukturierten, aber i.a. voll besetzten Matrix

$$\boldsymbol{K} := \left(K(\|\boldsymbol{y}_j - \boldsymbol{x}_k\|_2)\right)_{j \in I_M^1, k \in I_N^1}$$

schreiben.

Unser Zugang ist eine konsequente Erweiterung des Summationsalgorithmus für den äquidistanten Fall. Im univariaten, äquidistanten Fall ist die Matrix $\boldsymbol{K}$ eine Toeplitz-Matrix. Der Standard-Algorithmus zur schnellen Matrix-Vektor-Multiplikation beruht auf der Einbettung der Toeplitz-Matrix $\boldsymbol{K}$ in eine zirkulante Matrix, die dann durch die Fourier-Matrix diagonalisiert wird. Die Realisierung dieses Algorithmus erfolgt durch den Aufruf einer FFT, einer Multiplikation mit der Diagonal-Matrix und der Anwendung einer weiteren FFT. In unserem neuen Summationsalgorithmus ersetzen wir die FFT durch die NFFT. Neben der einfachen algorithmischen Umsetzung ist ein weiterer wesentlicher Vorteil unserer Algorithmen, dass sie in einfacher Weise für sehr viele Kerne $\mathcal{K}$ benutzt werden können.

Weitere Methoden, die auch zur schnellen Matrix-Vektor-Multiplikation mit speziell strukturierten Matrizen herangezogen werden, sind z.B. Paneel-Methoden [61], Wavelet-Methoden [9], $\mathcal{H}$- und $\mathcal{H}^2$-Matrix-Methoden [59, 60] oder Mosaik-Skeleton Approximationen [125, 49].

Einige Teile dieses Kapitels wurden bereits in [101, 88] veröffentlicht.

Dieses Kapitel ist wie folgt gegliedert:

In Abschnitt 2.1 stellen wir einen Algorithmus zur schnellen Berechnung der Summe (2.1) für äquidistante Daten vor, der eine Verallgemeinerung auf den nichtäquidistanten Fall erlaubt. Im folgenden Abschnitt 2.2 werden dann die schnellen Summationsalgorithmen präsentiert. Wir beginnen zunächst mit dem schnellen Summationsalgorithmus für singuläre Kerne (siehe Algorithmus 2.3) und vereinfachen diesen neuen Algorithmus dann auf nichtsinguläre Kerne (siehe Algorithmus 2.6). Da es sich um approximative Algorithmen handelt, beweisen wir in Abschnitt 2.3 die Fehlerabschätzungen. Aus diesen Ergebnissen lässt sich folgern, wie gewisse Parameter im Summationsalgorithmus günstig zu wählen sind. Diese theoretischen Resultate belegen wir durch umfangreiche numerische Tests in Abschnitt 2.4. Das Kapitel schließt mit einigen weiteren Bemerkungen und Literaturhinweisen.

2.1 Schnelle Summation für äquidistante Knoten

Wir betrachten zunächst den bekannten Standard-Algorithmus zur schnellen Summation, falls die Knoten äquidistant verteilt sind. In diesem Fall ist die Matrix $\boldsymbol{K}$ eine (multilevel) Toeplitz-Matrix. Da die Grundidee unseres neuen schnellen Summationsalgorithmus von Methoden zur schnellen Matrix-Vektor-Multiplikation mit einer Toeplitz-Matrix abstammt, formulieren wir diesen Algorithmus in einer Form, die eine Verallgemeinerung auf den nichtäquidistanten Fall zulässt. Hintergrund dieses schnellen Algorithmus ist, dass die Toeplitz-Matrix in eine zirkulante Matrix eingebettet werden kann und diese mit Hilfe der Fourier-Matrix diagonalisiert wird.
Es seien $\boldsymbol{x}_k$ und $\boldsymbol{y}_j$ äquidistante Punkte aus dem Quader $[-1/4, 1/4)^d$, z.B. die Punkte $\boldsymbol{k}/(2N)$ ($\boldsymbol{k} \in I_N^d$) und $\boldsymbol{j}/(2M)$ ($\boldsymbol{j} \in I_M^d$). Im Folgenden setzen wir $\mathcal{K}(\boldsymbol{0}) := 0$, falls $\mathcal{K}$ eine Singularität im Ursprung hat. Für die schnelle Summation, d.h. für die Berechnung von

$$f\left(\frac{\boldsymbol{j}}{2M}\right) := \sum_{\boldsymbol{k}\in I_N^d} \alpha_{\boldsymbol{k}} \mathcal{K}\left(\frac{\boldsymbol{j}}{2M} - \frac{\boldsymbol{k}}{2N}\right) \qquad (\boldsymbol{j} \in I_M^d) \tag{2.3}$$

wenden wir die folgende Aliasing-Formel (Überlappungsformel) an.

Satz 2.1 *(Aliasing-Formel)*
Es sei $f \in L_2(\mathbb{T}^d)$ eine multivariate, 1-periodische Funktion mit absolut konvergenter Fourier-Reihe

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

und mit den Fourier-Koeffizienten $c_{\boldsymbol{k}}(f)$ (vgl. Definition (1.8))

$$c_{\boldsymbol{k}}(f) := \int_{\mathbb{T}^d} f(\boldsymbol{x})\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{kx}}\, \mathrm{d}\boldsymbol{x}.$$

Dann gilt für die Approximation der Fourier-Koeffizienten $c_{\boldsymbol{k}}(f)$ durch die Rechteckregel

$$\hat{f}_{\boldsymbol{k}} := \frac{1}{n^d} \sum_{\boldsymbol{j}\in I_n^d} f\left(\frac{\boldsymbol{j}}{n}\right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{jk}/n}$$

der Zusammenhang

$$\hat{f}_{\boldsymbol{k}} = c_{\boldsymbol{k}}(f) + \sum_{\substack{\boldsymbol{r}\in\mathbb{Z}^d \\ \boldsymbol{r}\neq\boldsymbol{0}}} c_{\boldsymbol{k}+\boldsymbol{r}n}(f).$$

Es sei $n := 2\,\mathrm{kgV}(N,M)$ das Doppelte des kleinsten gemeinsamen Vielfachen der natürlichen Zahlen N und M. Außerdem sei K_{R} eine glatte, multivariate, 1-periodische Funktion mit $K_{\mathrm{R}}(\boldsymbol{j}/n) = \mathcal{K}(\boldsymbol{j}/n)$ für $\boldsymbol{j} \in I_n^d$. Dann folgt mit obiger Aliasing-Formel

$$\begin{aligned} K_{\mathrm{R}}(\boldsymbol{x}) &= \sum_{\boldsymbol{l}\in\mathbb{Z}^d} c_{\boldsymbol{l}}(K_{\mathrm{R}})\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}} \\ &= \sum_{\boldsymbol{l}\in I_n^d} c_{\boldsymbol{l}}(K_{\mathrm{R}})\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}} + \sum_{\boldsymbol{l}\in I_n^d} \sum_{\substack{\boldsymbol{r}\in\mathbb{Z}^d \\ \boldsymbol{r}\neq\boldsymbol{0}}} c_{\boldsymbol{l}+n\boldsymbol{r}}(K_{\mathrm{R}})\,\mathrm{e}^{2\pi\mathrm{i}(\boldsymbol{l}+n\boldsymbol{r})\boldsymbol{x}} \\ &= \sum_{\boldsymbol{l}\in I_n^d} b_{\boldsymbol{l}}\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}} + \sum_{\boldsymbol{l}\in I_n^d} \sum_{\substack{\boldsymbol{r}\in\mathbb{Z}^d \\ \boldsymbol{r}\neq\boldsymbol{0}}} c_{\boldsymbol{l}+n\boldsymbol{r}}(K_{\mathrm{R}})\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}}(\,\mathrm{e}^{2\pi\mathrm{i}n\boldsymbol{r}\boldsymbol{x}} - 1), \end{aligned} \tag{2.4}$$

wobei wir mit $b_{\boldsymbol{l}}$ abkürzen

$$b_{\boldsymbol{l}} := \frac{1}{n^d} \sum_{\boldsymbol{j}\in I_n^d} K\left(\frac{\boldsymbol{j}}{n}\right)\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{j}\boldsymbol{l}/n}. \tag{2.5}$$

Die Formel (2.4) ist Voraussetzung für die Herleitung und Fehlerabschätzung des nichtäquidistanten Summationsalgorithmus (siehe Abschnitte 2.2 und 2.3). Für die Differenz der äquidistanten Knoten $\boldsymbol{x} := \boldsymbol{j}/(2M) - \boldsymbol{k}/(2N)$ folgt aus der Formel (2.4) und der Definition von n, dass der Ausdruck $\mathrm{e}^{2\pi\mathrm{i}n\boldsymbol{r}\boldsymbol{x}} - 1$ für alle $\boldsymbol{r} \in \mathbb{Z}^d$ verschwindet. Also ist

$$K_{\mathrm{R}}\left(\frac{\boldsymbol{j}}{2M} - \frac{\boldsymbol{k}}{2N}\right) = \mathcal{K}\left(\frac{\boldsymbol{j}}{2M} - \frac{\boldsymbol{k}}{2N}\right) = \sum_{\boldsymbol{l}\in I_n^d} b_{\boldsymbol{l}}\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}(\boldsymbol{j}/(2M)-\boldsymbol{k}/(2N))}$$

und mit (2.3) folgt

$$\begin{aligned} f\left(\frac{\boldsymbol{j}}{2M}\right) &= \sum_{\boldsymbol{k}\in I_N^d} \alpha_{\boldsymbol{k}} \sum_{\boldsymbol{l}\in I_n^d} b_{\boldsymbol{l}}\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}(\boldsymbol{j}/(2M)-\boldsymbol{k}/(2N))} \\ &= \sum_{\boldsymbol{l}\in I_n^d} b_{\boldsymbol{l}} \left(\sum_{\boldsymbol{k}\in I_N^d} \alpha_{\boldsymbol{k}}\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{k}/(2N)}\right)\,\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{j}/(2M)}. \end{aligned}$$

Damit können wir folgenden Algorithmus für die schnelle Summation von (2.3) angeben.

Algorithmus 2.2 (Schnelle (multilevel) Matrix-Vektor-Multiplikation für Toeplitz-Matrizen)

Eingabe: $N, M \in \mathbb{N}$, $\alpha_{\boldsymbol{k}} \in \mathbb{C}$ $(\boldsymbol{k} \in I_N^d)$.

Vorberechnung: Berechnung von $b_{\boldsymbol{l}}$ $(\boldsymbol{l} \in I_n^d)$ *in (2.5) mittels* d*-variater FFT*$(n^{\otimes d})$.

1. *Berechne die Summe*

$$a_{\boldsymbol{l}} := \sum_{\boldsymbol{k}\in I_N^d} \alpha_{\boldsymbol{k}}\, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{l}\boldsymbol{k}/(2N)} \quad (\boldsymbol{l} \in I_n^d)$$

mittels einer d-variaten FFT$((2N)^{\otimes d})$*. Benutze den Zusammenhang* $a_{\boldsymbol{l}+2N\boldsymbol{s}} = a_{\boldsymbol{l}}$ *für* $\boldsymbol{s} = -(\boldsymbol{n}/2 - \mathbf{1}N)/(2N), \ldots, (\boldsymbol{n}/2 - \mathbf{1}N)/(2N)$.

2. *Bilde für* $\boldsymbol{l} \in I_n^d$ *die Produkte*

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

3. *Berechne für* $\boldsymbol{j} \in I_M^d$ *die Summe*

$$f\left(\frac{\boldsymbol{j}}{2M}\right) = \sum_{\boldsymbol{l}\in I_n^d} d_{\boldsymbol{l}}\, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{l}\boldsymbol{j}/(2M)} = \sum_{\boldsymbol{l}\in I_M^d} \left(\sum_{2M\boldsymbol{s}\in I_{n-2M}^d} d_{\boldsymbol{l}+2M\boldsymbol{s}} \right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{l}\boldsymbol{j}/(2M)}$$

mittels d-variater FFT$((2M)^{\otimes d})$*.*

Ausgabe: Funktionswerte $f\left(\frac{\boldsymbol{j}}{2M}\right)$ $(\boldsymbol{j} \in I_M^d)$ *von* (2.1).

Komplexität: $\mathcal{O}(N^d \log N + M^d \log M)$.

Im univariaten Fall $d = 1$ und für $M = N$ ist die Matrix $\boldsymbol{K} := \left(K(\frac{j-k}{2N})\right)_{j,k=-N/2}^{N/2-1} \in \mathbb{R}^{N\times N}$ eine Toeplitz-Matrix der Größe (N, N). Damit ist Algorithmus 2.2 der Standard-Algorithmus für die Matrix-Vektor-Multiplikation mit einer Toeplitz-Matrix, basierend auf der Einbettung von $\boldsymbol{K}$ in eine zirkulante Matrix der Größe $(2N, 2N)$. Die Multiplikation des Vektors mit der zirkulanten Matrix erfolgt dann mittels schneller Fourier-Transformation (siehe z.B. [19]). Es sind schnellere Algorithmen für die Multiplikation eines Vektors mit einer reellen Toeplitz-Matrix bekannt, die auf trigonometrischen Transformationen basieren (siehe [11, 15, 97, 94, 65]). Diese Algorithmen können in ähnlicher Weise hergeleitet werden, wenn man Chebyshev-Reihen an Stelle der Fourier-Reihen benutzt. Es gibt für reelle Eingabedaten $\alpha_{\boldsymbol{k}} \in \mathbb{R}$ im nichtäquidistanten Fall aber auch Algorithmen, die auf nichtäquidistanten trigonometrischen Transformationen beruhen (vgl. Beispiel 2.9).

2.2 Schnelle Summation für nichtäquidistante Knoten

In diesem Abschnitt verallgemeinern wir Algorithmus 2.2 für nichtäquidistante Knoten, d.h., wir werden schnelle Algorithmen zur Berechnung von Summen der Form

$$f(\boldsymbol{y}) := \sum_{k=1}^{N} \alpha_k \mathcal{K}(\boldsymbol{y} - \boldsymbol{x}_k) = \sum_{k=1}^{N} \alpha_k K(\|\boldsymbol{y} - \boldsymbol{x}_k\|_2), \tag{2.6}$$

an M beliebigen Knoten $\boldsymbol{y} = \boldsymbol{y}_j \in \mathbb{R}^d$ $(j = 1, \ldots, M)$ vorstellen. Dabei ist $f : \mathbb{R}^d \to \mathbb{R}$. Da wir mit 1-periodischen Funktionen arbeiten wollen, skalieren wir die Knoten $\boldsymbol{x}_k$ und

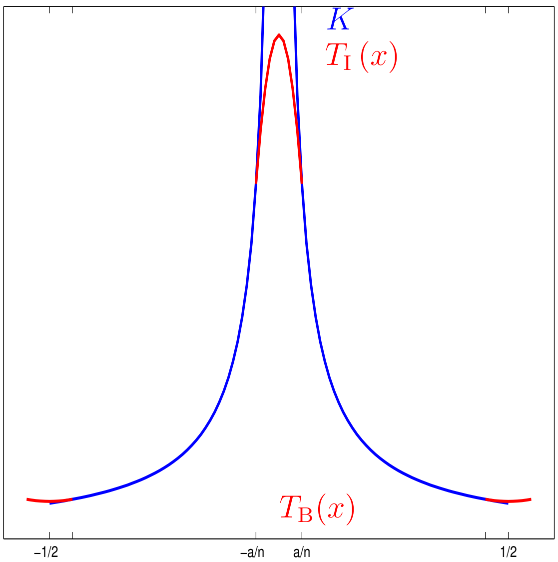

Abbildung 2.1: Singulärer Kern K mit der Regularisierung T_{I} und T_{B}.

$\boldsymbol{y}_j$ so, dass $||\boldsymbol{x}_k||_2 \leq 1/4 - 1/32 = 7/32$ und $||\boldsymbol{y}_j||_2 \leq 7/32$ ist. Somit muss für alle Abstände zwischen Punkten $\|\boldsymbol{x}_k - \boldsymbol{y}_j\|_2 \leq 7/16 = 1/2 - 1/16$ gelten. Weiterhin sollen die Punkte $\boldsymbol{x}_k$ $(k = 1, \ldots, N)$ in der d-dimensionalen Kugel $\{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\|_2 \leq 7/32\}$ approximativ gleichmäßig verteilt sein.

Singuläre Kerne

Zunächst betrachten wir Kerne K, die mit Ausnahme vom Nullpunkt beliebig glatt sind. Hier lassen wir zu, dass K oder Ableitungen von K eine Singularität im Nullpunkt haben. Beispiele für solche Kerne sind

$$\frac{1}{x^2},\ \frac{1}{x},\ \frac{1}{|x|},\ \log x,\ x^2 \log |x|. \tag{2.7}$$

Die Idee besteht darin, die singuläre Funktion in der Nähe der Singularität (im Nahfeld) und am Rand zu regularisieren, so dass wir eine glatte (p-mal stetig differenzierbare, 1-periodische) Funktion $\mathcal{K}_{\mathrm{R}}$ erhalten (siehe Abbildung 2.1).

Die Funktion $\mathcal{K}_{\mathrm{R}}$ wird dann durch eine Fourier-Summe $\mathcal{K}_{\mathrm{RF}}$ angenähert. Da die Funktion $\mathcal{K}_{\mathrm{R}}$ aber p-mal stetig differenzierbar ist, sind wir in der Lage, in Abschnitt 2.3 den Fehler $\|\mathcal{K}_{\mathrm{R}} - \mathcal{K}_{\mathrm{RF}}\|_\infty$ abzuschätzen. Der Kern $\mathcal{K}$ kann also durch $\mathcal{K}_{\mathrm{RF}}$ und einen Kern $\mathcal{K}_{\mathrm{NE}}$, dessen Träger nur im Nahfeld liegt, approximiert werden. Im Folgenden geben wir die genaue Konstruktion von $\mathcal{K}_{\mathrm{R}}$ sowie $\mathcal{K}_{\mathrm{RF}}$ an und leiten einen schnellen Algorithmus zur Berechnung von (2.6) an den beliebigen Knoten $\boldsymbol{y}_j$ $(j = 1, \ldots, M)$ her.

Wir regularisieren $\mathcal{K}$ in der Nähe von $\mathbf{0}$ und in der Nähe des Randes von $\|\boldsymbol{x}\|_2 = 1/2$

durch einen glatten 1-periodischen Kern

$$\mathcal{K}_{\mathrm{R}}(\boldsymbol{x}) := \begin{cases} T_{\mathrm{I}}(\|\boldsymbol{x}\|_2) & \text{falls} \quad \|\boldsymbol{x}\|_2 \le \frac{a}{n}, \\ T_B(\|\boldsymbol{x}\|_2) & \text{falls} \quad \frac{7}{16} < \|\boldsymbol{x}\|_2 < \frac{1}{2}, \\ T_B(\frac{1}{2}) & \text{falls} \quad \frac{1}{2} \le \|\boldsymbol{x}\|_2, \\ K(\|\boldsymbol{x}\|_2) & \text{sonst,} \end{cases} \tag{2.8}$$

wobei $a \in \mathbb{N}$ ein zuvor gewählter Parameter ist, und die Polynome T_{I} und T_{B} durch die Definition

$$T_{\mathrm{I}}(x) := \sum_{j=0}^{p-1} a_j^{\mathrm{I}} \cos \frac{\pi n j}{2a} x, \tag{2.9}$$

und

$$T_{\mathrm{B}}(x) := \sum_{j=0}^{p_{\mathrm{B}}-1} a_j^{\mathrm{B}} \cos(8\pi j\,(x-1/2)) \qquad (x \in (7/16, 1/2]) \tag{2.10}$$

gegeben sind. Jetzt soll die Bedingung $K_{\mathrm{R}}(\|\boldsymbol{x}\|_2) := \mathcal{K}_{\mathrm{R}}(\boldsymbol{x})$ für $\|\boldsymbol{x}\|_2 \le 1/2$ mit $K_{\mathrm{R}} \in C^p(\mathbb{T})$ erfüllt werden. Wir setzen

$$p_{\mathrm{B}} := \begin{cases} p & \text{für } d = 1, \\ p + \lfloor (p-1)/2 \rfloor & \text{für } d \ge 2 \end{cases}$$

und bestimmen die Koeffizienten a_j^{I} und a_j^{B} durch die Forderung, dass die Ableitungen von T_{I} und K in a/n bis zur Ordnung $p-1$ identisch sind, also durch

$$T_{\mathrm{I}}^{(r)}\left(\frac{a}{n}\right) = K^{(r)}\left(\frac{a}{n}\right) \qquad (r = 0, \ldots, p-1), \tag{2.11}$$

und dass die Ableitungen von T_{B} und K in $7/16$ bis zur Ordnung $p-1$ identisch sind, d.h.

$$T_{\mathrm{B}}^{(r)}\left(\frac{7}{16}\right) \;=\; K^{(r)}\left(\frac{7}{16}\right) \qquad (r = 0, \ldots, p-1). \tag{2.12}$$

Damit wir im multivariaten Fall eine „glatte Periodisierung“ $\mathcal{K}_{\mathrm{R}}$ erhalten, fordern wir für $d \ge 2$ zusätzlich

$$T_{\mathrm{B}}^{(2r)}\left(\frac{1}{2}\right) \;=\; 0 \qquad (r = 1, \ldots, \lfloor (p-1)/2 \rfloor)\,. \tag{2.13}$$

Aus der Definition des trigonometrischen Polynoms T_{B} in (2.10) ergibt sich $T_{\mathrm{B}}^{(2r+1)}(1/2) = 0$ für alle $r \in \mathbb{N}_0$. Starten wir mit dem Ansatz (2.9) und benutzen die Bedingung (2.11), so können wir die Koeffizienten $\boldsymbol{a}_{\mathrm{even}}^{\mathrm{I}} := (a_{2j}^{\mathrm{I}})_{j=1}^{\lfloor \frac{p-1}{2} \rfloor}$ und $\boldsymbol{a}_{\mathrm{odd}}^{\mathrm{I}} := (a_{2j+1}^{\mathrm{I}})_{j=0}^{\lfloor \frac{p-2}{2} \rfloor}$ eindeutig als Lösung der zwei linearen Gleichungssysteme

$$\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}}\, \boldsymbol{a}_{\mathrm{even}}^{\mathrm{I}} = \left(\left(\frac{2a}{\pi n}\right)^{2r} K^{(2r)}\left(\frac{a}{n}\right) \right)_{r=1}^{\lfloor \frac{p-1}{2} \rfloor}, \tag{2.14}$$

$$\boldsymbol{V}_{\mathrm{odd}}^{\mathrm{I}}\,\boldsymbol{a}_{\mathrm{odd}}^{\mathrm{I}} = \left(-\left(\frac{2a}{\pi n}\right)^{2r+1} K^{(2r+1)}\left(\frac{a}{n}\right)\right)_{r=0}^{\lfloor\frac{p-2}{2}\rfloor} \tag{2.15}$$

berechnen, wobei $\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}}$ und $\boldsymbol{V}_{\mathrm{odd}}^{\mathrm{I}}$ die regulären Mosaik-Vandermonde-Matrizen

$$\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}} := \left((-1)^{r+j}(2j)^{2r}\right)_{r,j=1}^{\lfloor\frac{p-1}{2}\rfloor}, \quad \boldsymbol{V}_{\mathrm{odd}}^{\mathrm{I}} := \left((-1)^{r+j}(2j+1)^{2r+1}\right)_{r,j=0}^{\lfloor\frac{p-2}{2}\rfloor} \tag{2.16}$$

sind.
Natürlich sind diese Matrizen für große $p \in \mathbb{N}$ sehr schlecht konditioniert. In unseren Algorithmen müssen wir die Gleichungssysteme aber nur für kleine Parameter $p \leq 16$ lösen, was ohne Probleme möglich ist.
In ähnlicher Weise folgt aus der Definition (2.10) für das trigonometrische Polynom T_{B} mit den Forderungen (2.12) und (2.13), dass die Koeffizienten $\boldsymbol{a}_{\mathrm{even}}^{\mathrm{B}} := (a_{2j}^{\mathrm{B}})_{j=1}^{\lfloor\frac{p_B-1}{2}\rfloor}$ und $\boldsymbol{a}_{\mathrm{odd}}^{\mathrm{B}} := (a_{2j+1}^{\mathrm{B}})_{j=0}^{\lfloor\frac{p_{\mathrm{B}}-2}{2}\rfloor}$ die eindeutig bestimmte Lösung des linearen Gleichungssystems

$$\begin{pmatrix} \boldsymbol{V}_{\mathrm{even}}^{\mathrm{B}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{V}_{\mathrm{odd}}^{\mathrm{B}} \\ \boldsymbol{V}_{\mathrm{e}}^{\mathrm{B}} & \boldsymbol{V}_{\mathrm{o}}^{\mathrm{B}} \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_{\mathrm{even}}^{\mathrm{B}} \\ \boldsymbol{a}_{\mathrm{odd}}^{\mathrm{B}} \end{pmatrix} = \begin{pmatrix} \left(K^{(2r)}(7/16)\right)_{r=1}^{\lfloor\frac{p-1}{2}\rfloor} \\ \left(K^{(2r+1)}(7/16)\right)_{r=0}^{\lfloor\frac{p-2}{2}\rfloor} \\ \mathbf{0} \end{pmatrix}, \tag{2.17}$$

mit

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

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

sind.
Im univariaten Fall folgt die Eindeutigkeit der Lösung des trigonometrischen Hermite-Interpolationsproblems (2.10) mit den Forderungen (2.12) wieder aus der Regularität der Vandermonde-Matrizen. Die Regularität der Matrix in der Gleichung (2.17) haben wir für $p \leq 16$ elementar nachgerechnet.

Alternativ zu der vorgeschlagenen Regularisierung mittels trigonometrischer Polynome können z.B. zwei-Punkte Taylor-Interpolationen [26, S. 37] oder Splines benutzt werden.

Jetzt approximieren wir die glatte Funktion $\mathcal{K}_{\mathrm{R}}$ durch das Lagrange-Interpolationspolynom

$$\mathcal{K}_{\mathrm{RF}}(\boldsymbol{x}) := \sum_{\boldsymbol{l} \in I_n^d} b_{\boldsymbol{l}}\, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}}, \tag{2.18}$$

wobei die diskreten Fourier-Koeffizienten $b_{\boldsymbol{l}}$ durch Abtasten des regularisierten, 1-periodischen Kerns $\mathcal{K}_{\mathrm{R}}$, also durch

$$b_{\boldsymbol{l}} := \frac{1}{n^d} \sum_{\boldsymbol{j} \in I_n^d} \mathcal{K}_{\mathrm{R}}(\boldsymbol{j}/n)\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{j}\boldsymbol{l}/n} \quad (\boldsymbol{l} \in I_n^d) \tag{2.19}$$

gegeben sind. Wir zerlegen unseren Kern $\mathcal{K}$ in der Form

$$\mathcal{K} = (\mathcal{K} - \mathcal{K}_{\mathrm{R}}) + (\mathcal{K}_{\mathrm{R}} - \mathcal{K}_{\mathrm{RF}}) + \mathcal{K}_{\mathrm{RF}} = \mathcal{K}_{\mathrm{NE}} + \mathcal{K}_{\mathrm{ER}} + \mathcal{K}_{\mathrm{RF}} \tag{2.20}$$

mit den Definitionen $\mathcal{K}_{\mathrm{NE}} := \mathcal{K} - \mathcal{K}_{\mathrm{R}}$ und $\mathcal{K}_{\mathrm{ER}} := \mathcal{K}_{\mathrm{R}} - \mathcal{K}_{\mathrm{RF}}$. Die Abkürzungen $\mathcal{K}_{\mathrm{NE}}$ verweisen auf das Nahfeld (Nearfield) und $\mathcal{K}_{\mathrm{ER}}$ auf den Fehler (Error), der zwischen dem regularisierten Kern $\mathcal{K}_{\mathrm{R}}$ und dessen Fourier-Summe $\mathcal{K}_{\mathrm{RF}}$ entsteht. Weil der 1-periodische, regularisierte Kern $\mathcal{K}_{\mathrm{R}}$ sehr glatt (p-mal stetig differenzierbar) ist, erwarten wir, dass die Approximation $\mathcal{K}_{\mathrm{RF}}$ nur einen sehr kleinen Fehler $\mathcal{K}_{\mathrm{ER}}$ erzeugt. Wir vernachlässigen diesen Fehler und approximieren f durch

$$\tilde{f}(\boldsymbol{x}) := f_{\mathrm{NE}}(\boldsymbol{x}) + f_{\mathrm{RF}}(\boldsymbol{x}), \tag{2.21}$$

wobei

$$f_{\mathrm{NE}}(\boldsymbol{x}) := \sum_{k=1}^{N} \alpha_k \mathcal{K}_{\mathrm{NE}}(\boldsymbol{x} - \boldsymbol{x}_k)\,, \tag{2.22}$$

$$f_{\mathrm{RF}}(\boldsymbol{x}) := \sum_{k=1}^{N} \alpha_k \mathcal{K}_{\mathrm{RF}}(\boldsymbol{x} - \boldsymbol{x}_k) \tag{2.23}$$

ist.
Nun wird nicht f, sondern $\tilde{f}$ an den Punkten $\boldsymbol{y}_j$ berechnet. Dies kann sehr schnell erfolgen, nämlich:

1. *Berechnung des Nahfeldes* (2.22)

   Nach Definition (2.8) von $\mathcal{K}_{\mathrm{R}}$ hat die Funktion $\mathcal{K}_{\mathrm{NE}}$ einen kleinen Träger in der Einheitskugel mit dem Radius a/n um den Nullpunkt $\mathbf{0}$. Der Kreisring $7/16 < \|\boldsymbol{x}\|_2 = 1/2$ ist für die Berechnung uninteressant, da wir die Punkte so skaliert haben, dass $\|\boldsymbol{x}_k - \boldsymbol{y}_j\|_2 \le 7/16$ gilt. Wir setzen voraus, dass die Knoten $\boldsymbol{x}_k$ ($k = 0, \ldots, N$) in $||\boldsymbol{x}||_2 \le 7/32$ so verteilt sind, dass jede Kugel mit dem Radius $1/n$ höchstens ν Punkte $\boldsymbol{x}_k$ ($k = 0, \ldots, N$) enthält. Wir wählen die Länge der Fourier-Summe in (2.18)

   $$n = \mathcal{O}(N^{1/d}). \tag{2.24}$$

   Die Summe (2.22) enthält für festes $\boldsymbol{y}_j$ nur $a^d \nu$ Summanden, so dass die Berechnung an den M verschiedenen Knoten nur $\mathcal{O}(a^d \nu M)$ arithmetische Operationen benötigt.

2. *Berechnung der Summe* f_{RF} (2.23)

   Substituieren wir die Fourier-Summe (2.18) des Kerns $\mathcal{K}_{\mathrm{RF}}$ in die Summe (2.23), so erhalten wir

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

Der Ausdruck in den Klammern kann durch eine d-variate $\mathrm{NFFT}^{\mathrm{T}}(n^{\otimes d})$ berechnet werden. Dann multiplizieren wir das Ergebnis mit den n^d Daten $b_{\boldsymbol{l}}$ ($\boldsymbol{l} \in I_n^d$) und berechnen schließlich mit einer d-variaten $\mathrm{NFFT}(n^{\otimes d})$ die äußere Summe. Wenn wir wieder m als den Abschneideparameter und den Oversamplingfaktor $\sigma = 2$ für die NFFT- und $\mathrm{NFFT}^{\mathrm{T}}$-Algorithmen (Algorithmus 1.1 und Algorithmus 1.3) wählen, dann können wir f_{RF} an den Punkten $\boldsymbol{y}_j$ ($j = 1, \ldots, M$) in $\mathcal{O}(m^d(N + M) + N \log N)$ arithmetischen Operationen berechnen.

Zusammenfassend folgt der Algorithmus:

Algorithmus 2.3 (Schnelle Summation für singuläre Kerne)

Eingabe: $a, p, n \in \mathbb{N}$, $\alpha_k \in \mathbb{C}$ $(k = 1, \ldots, N)$, $\boldsymbol{x}_k \in \mathbb{R}^d$ $(k = 1, \ldots, N)$, $\boldsymbol{y}_j \in \mathbb{R}^d$ $(j = 1, \ldots, M)$.

Vorberechnung: (i) *Berechnung von* a_j^{I} $(j = 0, \ldots, p-1)$ *in* (2.14) *und* a_j^B $(j = 0, \ldots, p_{\mathrm{B}} - 1)$ *in* (2.15).

(ii) *Berechnung von* $(b_{\boldsymbol{l}})_{\boldsymbol{l} \in I_n^d}$ *mittels FFT durch* (2.19) *und* (2.8).

(iii) *Berechnung von* $K_{\mathrm{NE}}(\boldsymbol{y}_j - \boldsymbol{x}_k)$ *für alle* $(j = 1, \ldots, M)$ *und* $k \in I_{n,a}^{\mathrm{NE}}(j)$, *wobei* $I_{n,a}^{\mathrm{NE}}(j) := \{k \in \{1, \ldots, N\} : \|\boldsymbol{y}_j - \boldsymbol{x}_k\|_2 < \frac{a}{n}\}$. *Beachte auch die Bemerkung* 2.4.

1. *Berechne für* $\boldsymbol{l} \in I_n^d$ *die Summen*

$$a_{\boldsymbol{l}} := \sum_{k=1}^{N} \alpha_k \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{l} \boldsymbol{x}_k}$$

mit einer d-variaten $\mathrm{NFFT}^{\mathrm{T}}(n^{\otimes d})$ *unter Nutzung von Algorithmus* 1.3.

2. *Bilde für alle* $\boldsymbol{l} \in I_n^d$ *die Produkte*

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

3. *Berechne für* $j = 1, \ldots, M$ *die Summen*

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

mit einer d-variaten NFFT($n^{\otimes d}$*) unter Nutzung von Algorithmus* 1.1.

4. *Berechne für* $j = 1, \ldots, M$ *die Nahfeldsummation*

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

5. *Bilde für* $j = 1, \ldots, M$ *die Summen*

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

Ausgabe: $\tilde{f}(\boldsymbol{y}_j)$ $(j = 0, \ldots, M)$ *Näherungswerte für* $f(\boldsymbol{y}_j)$ *in* (2.6).

Komplexität: $\mathcal{O}(N \log N + m^d(M + N) + a^d \nu M) = \mathcal{O}(N \log N + M)$.

Bemerkung 2.4 Bis jetzt wurde vorausgesetzt, dass die Werte $\mathcal{K}_{\mathrm{NE}}(\boldsymbol{y}_j-\boldsymbol{x}_k)$ in Schritt 4 von Algorithmus 2.3 durch $\mathcal{K}_{\mathrm{NE}}(\boldsymbol{y}_j-\boldsymbol{x}_k)=\mathcal{K}(\boldsymbol{y}_j-\boldsymbol{x}_k)-\mathcal{K}_{\mathrm{R}}(\boldsymbol{y}_j-\boldsymbol{x}_k)=\mathcal{K}(\boldsymbol{y}_j-\boldsymbol{x}_k)-T_{\mathrm{I}}(\|\boldsymbol{y}_j-\boldsymbol{x}_k\|_2)$ ($k\in I_{n,a}^{\mathrm{NE}}(j)$; $j=1,\ldots,M$) berechenbar sind. Alternativ können wir aber wie folgt vorgehen: Wir betrachten die Aufspaltung

$$f_{\mathrm{NE}}(\boldsymbol{y}_j)=\sum_{k\in I_{n,a}^{\mathrm{NE}}(j)}\alpha_k\mathcal{K}(\boldsymbol{y}_j-\boldsymbol{x}_k)\;-\sum_{k\in I_{n,a}^{\mathrm{NE}}(j)}\alpha_k T_{\mathrm{I}}(\|\boldsymbol{y}_j-\boldsymbol{x}_k\|_2)\,,$$

wobei die erste Summe in $\mathcal{O}(a\nu M)$ Schritten berechnet werden kann. Nach Definition (2.9) von T_{I} würde die zweite Summe $\mathcal{O}(pa\nu M)$ Operationen benötigen. Um diese Summe effizient zu bestimmen, approximieren wir sie wie folgt: Wir berechnen $T_{\mathrm{I}}(\frac{a}{n}\frac{s}{ap})$ ($s=-ap,\ldots,ap$) im Vorberechnungsschritt und bestimmen $T_{\mathrm{I}}(\|\boldsymbol{y}_j-\boldsymbol{x}_k\|_2)$ mittels kubischer Spline-Interpolation. Nun kann die (näherungsweise) Berechnung der zweiten Summe mit der gleichen arithmetischen Komplexität wie die der ersten Summe erfolgen. □

Bemerkung 2.5 (Matrix-Vektor-Schreibweise von Algorithmus 2.3)
Es seien die Matrix $\boldsymbol{K}:=\left(\mathcal{K}(\boldsymbol{y}_j-\boldsymbol{x}_k)\right)_{j=1,k=1}^{M,N}$ und der Vektor $\boldsymbol{\alpha}:=(\alpha_k)_{k=1}^N$ definiert. Dann kann Algorithmus 2.3 in Matrix-Vektor-Schreibweise

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

mit $\boldsymbol{D}_{\mathcal{K}_{\mathrm{R}}}:=\operatorname{diag}(b_{\boldsymbol{l}})_{\boldsymbol{l}\in I_n^d}$, $\boldsymbol{A}_N:=\left(\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}_k}\right)_{k=1,\ldots,N;\boldsymbol{l}\in I_n^d}$, $\boldsymbol{A}_M:=\left(\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{y}_j}\right)_{j=1,\ldots,M;\boldsymbol{l}\in I_n^d}$ und $\boldsymbol{K}_{\mathrm{NE}}:=\left(\mathcal{K}_{\mathrm{NE}}(\boldsymbol{y}_j-\boldsymbol{x}_k)\right)_{j=1,k=1}^{M,N}$ geschrieben werden. Nutzen wir die Matrix-Vektor-Notation (1.18) für die NFFT/NFFT$^{\mathrm{T}}$-Algorithmen, so können wir $\boldsymbol{K\alpha}$ in der Form

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

schreiben, wobei $\boldsymbol{C}:=\boldsymbol{F}_{\sigma n,n}^d\boldsymbol{D}\boldsymbol{D}_{\mathcal{K}_{\mathrm{R}}}\boldsymbol{D}(\boldsymbol{F}_{\sigma n,n}^d)^{\mathrm{T}}$ eine Multilevel-zirkulante Matrix ist. □

Nichtsinguläre Kerne

Im Folgenden wollen wir glatte Kerne wie z.B.

$$(x^2+c^2)^{\pm 1/2},\ \mathrm{e}^{-\delta x^2} \tag{2.25}$$

betrachten. Diese Funktionen sind als Multiquadric, inverse Multiquadric und als Gauß-Kerne bekannt. Multipol-Algorithmen sind für die schnelle Gauß-Transformation schon länger bekannt [54], während schnelle Algorithmen für die Multiquadric in [6, 4, 7] entwickelt wurden. Wir erwähnen nochmals, dass unsere Algorithmen einen neuen, einheitlichen Zugang für eine große Klasse von Kernen erlaubt. Insbesondere zeichnen sich diese Verfahren durch einfach strukturierte Algorithmen (FFT, NFFT) aus.
Für die Kerne (2.25) ist keine Regularisierung im Nullpunkt nötig und somit entfällt die Nahfeldberechnung. Falls der Kern K sehr klein am Rand ist, z.B. bei großen Werten δ

für den Gauß-Kern, ist auch die Regularisierung am Rand nicht nötig, d.h. wir können einfach $\mathcal{K}_{\mathrm{R}} := \mathcal{K}$ definieren. Ansonsten setzen wir

$$\mathcal{K}_{\mathrm{R}}(\boldsymbol{x}) := \begin{cases} T_{\mathrm{B}}(\|\boldsymbol{x}\|_2) & \text{falls} \quad \frac{7}{16} < \|\boldsymbol{x}\|_2 < \frac{1}{2}, \\ T_{\mathrm{B}}(\frac{1}{2}) & \text{falls} \quad \frac{1}{2} \leq \|\boldsymbol{x}\|_2, \\ K(\|\boldsymbol{x}\|_2) & \text{sonst.} \end{cases} \tag{2.26}$$

Damit vereinfacht sich Algorithmus 2.3 wie folgt:

Algorithmus 2.6 (Schnelle Summation für nichtsinguläre Kerne)

Eingabe: $a, p, n \in \mathbb{N}$, $\alpha_k \in \mathbb{C}$ $(k = 1, \ldots, N)$, $\boldsymbol{x}_k \in \mathbb{R}^d$ $(k = 1, \ldots, N)$, $\boldsymbol{y}_j \in \mathbb{R}^d$ $(j = 1, \ldots, M)$.

Vorberechnung: *(i) Berechnung von* a_j^B $(j = 0, \ldots, p_{\mathrm{B}} - 1)$, *falls eine Regularisierung am Rand nötig ist.*
(ii) Berechnung von $(b_{\boldsymbol{l}})_{\boldsymbol{l} \in I_n^d}$ *mittels FFT durch* (2.19) *und* (2.26).

1. *Berechne für* $\boldsymbol{l} \in I_n^d$ *die Summen*

$$a_{\boldsymbol{l}} := \sum_{k=1}^{N} \alpha_k \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{l}\boldsymbol{x}_k}$$

mit einer d*-variaten NFFT*$^{\mathrm{T}}(n^{\otimes d})$ *unter Nutzung von Algorithmus* 1.3.

2. *Bilde für alle* $\boldsymbol{l} \in I_n^d$ *die Produkte*

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

3. *Berechne für* $j = 1, \ldots, M$ *die Summe*

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

mit einer d*-variaten NFFT*$(n^{\otimes d})$ *unter Nutzung von Algorithmus* 1.1.

Ausgabe: $\tilde{f}(\boldsymbol{y}_j)$ $(j = 0, \ldots, M)$ *Näherungswerte für* $f(\boldsymbol{y}_j)$ *in* (2.6).

Komplexität: $\mathcal{O}(m^d(M + N) + a^d \nu M) = \mathcal{O}(M + N)$.

Da wir keine Nahfeldberechnung durchführen müssen, hängt die Größe n der NFFT/NFFT$^{\mathrm{T}}$ Algorithmen im Gegensatz zu Algorithmus 2.3 (vgl. Formel (2.24)) nur vom Kern K, aber nicht von der Anzahl N der Knoten ab. Deshalb benötigt Algorithmus 2.6 nur $\mathcal{O}(n^d \log n + m^d(M + N)) = \mathcal{O}(M + N)$ arithmetische Operationen.

Bemerkung 2.7 (Zusammenhang zum Multiresolutionszugang von Beylkin und Cramer [10])
Kürzlich haben G. Beylkin und R. Cramer einen schnellen Summationsalgorithmus, basierend auf einem Mehrskalenzugang, vorgestellt. Dieser Algorithmus benötigt auch $\mathcal{O}(N \log N)$ arithmetische Operationen. Wir wollen den Zusammenhang zu unseren Verfahren für den univariaten Fall $d = 1$ verdeutlichen. Die Idee in [10] fassen wir wie folgt zusammen: Für $|y - x| > R$ wird gezeigt, dass der Kern K mit einer Genauigkeit, die von R abhängt, durch

$$K(x-y) \approx \sum_{r,s} t_{r-s}\, \varphi\left(x - \frac{r}{2^j}\right) \varphi\left(y - \frac{s}{2^j}\right) \tag{2.27}$$

approximiert werden kann. Dabei bezeichnet φ eine Skalierungsfunktion mit kleinem Träger. Wie in [8] bevorzugen die Autoren kardinale B-Splines für φ (siehe auch Abschnitt 1.3). Die Koeffizienten t_k müssen durch Lösen eines großen linearen Gleichungssystems berechnet werden. Dann kann die Summe (2.6) durch

$$\begin{aligned} f(y) \;\approx\; & \sum_{k=1}^{N} \alpha_k \sum_{r,s} t_{r-s}\, \varphi\left(x_k - \frac{r}{2^j}\right) \varphi\left(y - \frac{s}{2^j}\right) \\ & - \sum_{k \in I_y} \alpha_k \sum_{r,s} t_{r-s}\, \varphi\left(x_k - \frac{r}{2^j}\right) \varphi\left(y - \frac{s}{2^j}\right) + \sum_{k \in I_y} \alpha_k K(y - x_k) \end{aligned}$$

mit $I_y := \{x_k : |y - x_k| \le R\}$ approximiert werden. Die Berechnung der letzten zwei Summen benötigt wie Schritt 4 von Algorithmus 2.3 nur $\mathcal{O}(M)$ arithmetische Operationen, wobei aber die Konstante in $\mathcal{O}(M)$ von R abhängt. Die erste Summe wird in der Form

$$\sum_s \sum_r t_{r-s} \sum_{k=1}^{N} \alpha_k\, \varphi\left(x_k - \frac{r}{2^j}\right)\; \varphi\left(y - \frac{s}{2^j}\right)$$

geschrieben. Daraus ergibt sich folgende Berechnungsmöglichkeit:

- Berechne für alle r die Summe

  $$g_r := \sum_{k=1}^{N} \alpha_k\, \varphi\left(x_k - \frac{r}{2^j}\right) .$$

  Da φ einen kleinen Träger hat, erfordert die Berechnung für alle r zusammen nur $\mathcal{O}(N)$ arithmetische Operationen.

- Berechnung von $\tilde{g}_s := \sum_r g_r t_{r-s}$ für alle s mittels schneller Toeplitz-Matrix-Vektor-Multiplikation. Dies kann mit Hilfe der FFT in $\mathcal{O}(N \log N)$ arithmetischen Operationen (siehe Algorithmus 2.2) realisiert werden.

- Berechnung von $\sum_s \tilde{g}_s\, \varphi\left(y - \frac{s}{2^j}\right)$ in $\mathcal{O}(M)$ arithmetischen Operationen.

Der Zusammenhang zur NFFT-Summation wird deutlich, wenn wir beachten, dass die Funktionen $\mathrm{e}^{-2\pi\mathrm{i}kx}$ durch

$$\mathrm{e}^{-2\pi\mathrm{i}kx} \approx \frac{1}{\sigma n \hat{\varphi}(k)} \sum_r \tilde{\psi}\left(x - \frac{r}{\sigma n}\right) \mathrm{e}^{-2\pi\mathrm{i}kr/(\sigma n)}$$

approximiert werden können. Falls φ einen kompakten Träger hat, z.B. wenn φ die B-Splines sind, setzen wir $\psi = \varphi$ (siehe Abschnitt 1.3). Wir erhalten aus der Definition (2.18) für $|y-x| > a/n$ (und $|y-x| \le 1/2 - a/n$) mit einer Genauigkeit, die von a/n abhängt

$$\begin{aligned} K_{\mathrm{RF}}(y-x) &\approx \sum_{l=-n/2}^{n/2-1} b_l \,\mathrm{e}^{-2\pi\mathrm{i}lx}\,\mathrm{e}^{2\pi\mathrm{i}ly} \\ &\approx \sum_{l=-n/2}^{n/2-1} \frac{b_l}{(\sigma n)^2|\hat{\varphi}(l)|^2} \sum_{r,s} \tilde{\psi}\left(x - \frac{r}{\sigma n}\right) \mathrm{e}^{-2\pi\mathrm{i}lr/(\sigma n)}\, \tilde{\psi}\left(y - \frac{s}{\sigma n}\right) \mathrm{e}^{2\pi\mathrm{i}ls/(\sigma n)} \\ &= \sum_{r,s} \tilde{t}_{r-s}\, \tilde{\psi}\left(x - \frac{r}{\sigma n}\right) \tilde{\psi}\left(y - \frac{s}{\sigma n}\right), \end{aligned}$$

wobei

$$\tilde{t}_j := \sum_{l=-n/2}^{n/2-1} \frac{b_l}{(\sigma n)^2|\hat{\varphi}(l)|^2}\, \mathrm{e}^{-2\pi\mathrm{i}lj}$$

ist. Dies zeigt den Zusammenhang zu Formel (2.27). Dennoch scheint Algorithmus 2.3 einfacher zu sein. Insbesondere ist die Regularisierung leichter als in der Veröffentlichung [10] zu berechnen. Weiterhin ist unser Zugang nicht auf Funktionen φ beschränkt, die einer Zwei-Skalen-Relation genügen. □

2.3 Fehlerabschätzungen für die schnellen Summationsalgorithmen

In diesem Abschnitt wird bewiesen, dass der Fehler für die approximativen Summationsalgorithmen (Algorithmus 2.3 und Algorithmus 2.6), in Abhängigkeit von den Parametern a und p, klein werden kann. In Abschnitt 1.3 haben wir uns schon mit dem Approximationsfehler beschäftigt, der durch die NFFT-Algorithmen erzeugt wird. Jetzt wird der Fehler untersucht, der durch die Approximation von f durch $\tilde{f}$ in (2.21) entsteht. Das Ziel dieses Abschnitts ist wieder, den genauen Zusammenhang zwischen Approximationsfehler und arithmetischer Komplexität des Algorithmus zu beschreiben. Mit der Zerlegung des Kerns in Formel (2.20) und der Definition von $\tilde{f}$ in (2.21) erhalten wir

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

Schließlich ergibt sich mit der Hölder-Ungleichung

$$|f(\boldsymbol{y}_j) - \tilde{f}(\boldsymbol{y}_j)| \le \|\boldsymbol{\alpha}\|_1 \left\| \left(\mathcal{K}_{\mathrm{ER}}(\boldsymbol{y}_j - \boldsymbol{x}_k)\right)_{k=1}^{N} \right\|_\infty . \tag{2.28}$$

Im Folgenden wollen wir $\mathcal{K}_{\mathrm{ER}}$ in der Maximum-Norm

$$||\mathcal{K}_{\mathrm{ER}}||_\infty := \max_{\|\boldsymbol{x}\|_2 \le \frac{1}{2} - \frac{a}{n}} |\mathcal{K}_{\mathrm{ER}}(\boldsymbol{x})|$$

abschätztzen. Diese Form der Fehlerabschätzung wird auch immer im Zusammenhang mit den schnellen Multipol-Algorithmen untersucht. Die Hölder-Ungleichung (2.28) ist in dem Sinne scharf, dass die Gleichheit angenommen wird, falls $\alpha_k = 0$ für alle Indizes k mit Ausnahme der Indizes, für die $|\mathcal{K}_{\mathrm{ER}}(\boldsymbol{y}_j - \boldsymbol{x}_k)|$ den größten Wert annimmt. Unsere Fehlerabschätzungen basieren auf der bekannten Formel für den Aliasing-Fehler (Satz 2.1).

Anschließend untersuchen wir zunächst den Approximationsfehler für den Algorithmus 2.3 und danach den Approximationsfehler für Algorithmus 2.6.

Singuläre Kerne

In diesem Teilabschnitt werden wir Kerne K betrachten, die mit Ausnahme des Ursprungs Funktionen aus $C^\infty(\mathbb{R})$ sind und die einer Abfallbedingung der Form

$$|K^{(r)}(x)| \le C_{\mathrm{K}} \frac{(r+\alpha-1)!}{|x|^{r+\alpha}} \quad (x \in [-1/2, 1/2]\backslash\{0\}), \tag{2.29}$$

für alle $r = 1, \ldots, p$ $(p \ge 2)$ mit einer Konstanten $C_{\mathrm{K}} \ge 0$ und $\alpha \in \mathbb{N}_0$ genügen. Die Ungleichung (2.29) ist z.B. für homogene Kerne $K(x) = 1/|x|^\alpha$ oder für $K(x) = \log|x|$ mit $\alpha = 0$ erfüllt.

Der anschließende Satz gibt die Fehlerabschätzungen für den univariaten Fall $d = 1$ an. Im bivariaten Fall $d = 2$ sind einige neue Beweisideen nötig, die in der Veröffentlichung [88] herausgearbeitet wurden. Wir beweisen jetzt den folgenden Satz:

Satz 2.8 *Es sei K ein gerader Kern, der die Abfallbedingung* (2.29) *erfüllt. Dann gilt für* $2 \le p \le 50$ *mit* $K_{\mathrm{ER}} := K_{\mathrm{R}} - K_{\mathrm{RF}}$ *die Abschätzung*

$$\max_{x \in [-1/2, 1/2]} |K_{\mathrm{ER}}(x)| \le C \, \frac{n^\alpha}{a^{\alpha-1}} \, \frac{(p+\alpha+\delta_{0,\alpha}-2)!}{p-1} \left(\frac{\beta\pi}{4a}\right)^p,$$

wobei $\beta := \sqrt{1 + (2/\pi)^2}$ *und* $\delta_{0,\alpha}$ *das Kronecker-Symbol ist.*

Die Einschränkung $p \le 50$ ist wahrscheinlich nicht notwendig. Sie tritt nur deshalb auf, weil in unserem Beweis numerische Berechnungen bis $p \le 50$ mit einfließen. Wir benötigen aber im Algorithmus 2.3 nur Werte $p \le 16$, da diese bereits doppelte Genauigkeit des approximativen Algorithmus garantieren.

Beweis: In diesem Beweis sind C jeweils verschiedene Konstanten, die nicht von der Glattheit p, der Größe des Nahfeldes gesteuert durch a, der Länge der Fourier-Summe n und dem Kern-Parameter α (vgl. (2.29)) abhängen. Wir gehen wie in Abschnitt 2.1

vor, d.h., wir benutzen die Fourier-Summe von K_{R} und die Aliasing-Formel, so dass wir schließlich wie in Formel (2.4) für den Fehler die Identität

$$K_{\mathrm{ER}}(x) = \sum_{k=-n/2}^{n/2-1} \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} c_{k+rn}(K_{\mathrm{R}})\, \mathrm{e}^{2\pi \mathrm{i}kx}(\, \mathrm{e}^{2\pi \mathrm{i}rnx} - 1)$$

erhalten. Unter zusätzlichen Voraussetzungen an n (siehe Abschnitt 2.1) war dieser Fehler K_{ER} für äquidistante Knoten gleich Null. Für beliebige Knoten $x \in [-1/2, 1/2]$ schätzen wir diesen Fehler durch

$$|K_{\mathrm{ER}}(x)| \leq 2 \left(2 \sum_{k=n/2+1}^{\infty} |c_k(K_{\mathrm{R}})| + |c_{n/2}(K_{\mathrm{R}})| \right)$$

ab. Nach Konstruktion ist K_{R} p-mal stetig differenzierbar, so dass wir durch partielle Integration

$$c_k(K_{\mathrm{R}}) = (2\pi \mathrm{i}k)^{-p}\, c_k(K_{\mathrm{R}}^{(p)})$$

erhalten und damit

$$|K_{\mathrm{ER}}(x)| \leq 2 \left(2 \sum_{k=n/2+1}^{\infty} (2\pi k)^{-p} + (\pi n)^{-p} \right) \int_{-1/2}^{1/2} |K_{\mathrm{R}}^{(p)}(x)|\, \mathrm{d}x$$

folgern. Für $p \geq 2$ kann die obige Summe durch ein Integral vergrößert werden. Dies führt zu

$$|K_{\mathrm{ER}}(x)| \leq \frac{C}{(p-1)\pi^p n^{p-1}} \int_{-1/2}^{1/2} |K_{\mathrm{R}}^{(p)}(x)|\, \mathrm{d}x$$

und weil T_{I}, T_{B} und K gerade Funktionen sind, erhalten wir zusammen mit der Definition in (2.8)

$$|K_{\mathrm{ER}}(x)| \leq \frac{C}{(p-1)\pi^p n^{p-1}} \left(\int_0^{\frac{a}{n}} |T_{\mathrm{I}}^{(p)}(x)|\, \mathrm{d}x + \int_{\frac{a}{n}}^{\frac{7}{16}} |K^{(p)}(x)|\, \mathrm{d}x + \int_{\frac{7}{16}}^{\frac{1}{2}} |T_{\mathrm{B}}^{(p)}(x)|\, \mathrm{d}x \right). \tag{2.30}$$

Auf Grund der Abfallbedingung (2.29) folgt für das mittlere Integral von (2.30) die Ungleichung

$$\begin{aligned} \int_{\frac{a}{n}}^{\frac{7}{16}} |K^{(p)}(x)|\, \mathrm{d}x &\leq C\,(p+\alpha-1)! \int_{\frac{a}{n}}^{\frac{7}{16}} x^{-(p+\alpha)}\, \mathrm{d}x \\ &\leq C\,(p+\alpha-2)! \left(\frac{n}{a}\right)^{p-1+\alpha}. \end{aligned} \tag{2.31}$$

Für die Abschätzung des ersten und des letzten Integrals in (2.30) konzentrieren wir uns nur auf den Fall, dass p gerade ist. Wir setzen dazu $q := p/2-1$. Der Beweis für ungerade Zahlen p verläuft ähnlich. Aus der Definition (2.9) des trigonometrischen Polynoms T_{I} folgt

$$\begin{aligned} |T_{\mathrm{I}}^{(p)}(x)| &= \left(\frac{\pi n}{2a}\right)^p \left|\sum_{j=1}^{p-1} j^p a_j^{\mathrm{I}} \cos\frac{\pi n j}{2a} x\right| \\ &\leq \left(\frac{\pi n}{2a}\right)^p \sum_{j=1}^{p-1} j^p |a_j^{\mathrm{I}}| \end{aligned}$$

und damit

$$\int_0^{\frac{a}{n}} |T_{\mathrm{I}}^{(p)}(x)| \,\mathrm{d}x \leq \left(\frac{\pi}{2}\right)^p \left(\frac{n}{a}\right)^{p-1} \sum_{j=1}^{p-1} j^p |a_j^{\mathrm{I}}| \,. \tag{2.32}$$

Die Koeffizienten a_{2j}^{I} und a_{2j+1}^{I} sind durch das lineare Gleichungssystem (2.14) und (2.15) mit den entsprechenden Matrizen (2.16) bestimmt. Wir betrachten die Gleichung

$$\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}} \, (c_{k,l})_{k,l=1}^{q} = \boldsymbol{I}_q$$

wobei wir mit $\boldsymbol{I}_q$ die Einheitsmatrix der Größe (q,q) bezeichnen. Nun wird die Beziehung zwischen den Vandermonde-Matrizen und der polynomialen Interpolation verwendet. Wir können folgern, dass $c_{k,l}$ die Koeffizienten der geraden Lagrange-Polynome

$$L_{2l,p}(x) := \sum_{k=1}^{q} (-1)^{k+l} c_{k,l} \, x^{2k} = \frac{x^2}{(2l)^2} \prod_{\substack{s=1\\ s\neq l}}^{q} \frac{x^2-(2s)^2}{(2l)^2-(2s)^2} \qquad (l=1,\ldots,q) \tag{2.33}$$

sind. Beachten wir den Satz von Vieta, so sehen wir, dass die Koeffizienten $c_{k,l}$ positiv sind. Zusammen mit $(-1)^{k+l}\, c_{k,l} = L_{2l,p}^{(2k)}(0)/(2k)!$ folgt $c_{k,l} = |L_{2l,p}^{(2k)}(0)|/(2k)!$ und damit

$$((\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}})^{\mathrm{T}})^{-1} = \left(\frac{|L_{2l,p}^{(2k)}(0)|}{(2k)!}\right)_{l,k=1}^{q} .$$

Analog erhalten wir

$$((\boldsymbol{V}_{\mathrm{odd}}^{\mathrm{I}})^{\mathrm{T}})^{-1} = \left(\frac{|L_{2l+1,p}^{(2k+1)}(0)|}{(2k+1)!}\right)_{l,k=0}^{q} ,$$

wobei jetzt die Lagrange-Polynome durch

$$L_{2l+1,p}(x) := \frac{x}{(2l+1)} \prod_{\substack{s=0\\ s\neq l}}^{q} \frac{x^2-(2s+1)^2}{(2l+1)^2-(2s+1)^2} \qquad (l=0,\ldots,q) \tag{2.34}$$

gegeben sind. Damit folgt für die Koeffizienten a_{2j}^{I} $(j = 1, \ldots, q)$ auf Grund der Forderung (2.14) die Beziehung

$$(a_{2j}^{\mathrm{I}})_{j=1}^{q} = ((\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}})^{\mathrm{T}})^{-1} \left(\left(\frac{2a}{\pi n} \right)^{2r} K^{(2r)} \left(\frac{a}{n} \right) \right)_{r=1}^{q}, \tag{2.35}$$

was zusammen mit der Abfallbedingung (2.29) die Ungleichung

$$\begin{aligned} |a_{2j}^{\mathrm{I}}| &\leq C \left(\frac{n}{a} \right)^{\alpha} \sum_{k=1}^{q} \left(\frac{2}{\pi} \right)^{2k} \frac{(2k+\alpha-1)!}{(2k)!} \, |L_{2j,p}^{(2k)}(0)| \\ &\leq C \, \frac{(p-3+\alpha+\delta_{0,\alpha})!}{(p-2)!} \left(\frac{n}{a} \right)^{\alpha} \sum_{k=1}^{q} \left(\frac{2}{\pi} \right)^{2k} |L_{2j,p}^{(2k)}(0)| \end{aligned}$$

ergibt. Auf dem gleichen Weg erhalten wir für die ungeraden Indizes die Abschätzung

$$|a_{2j+1}^{\mathrm{I}}| \leq C \, \frac{(p-2+\alpha+\delta_{0,\alpha})!}{(p-1)!} \left(\frac{n}{a} \right)^{\alpha} \sum_{k=0}^{q} \left(\frac{2}{\pi} \right)^{2k+1} |L_{2j+1,p}^{(2k+1)}(0)|.$$

Schließlich ergibt sich mit der Aussage (2.32) die Ungleichung

$$\int_0^{\frac{a}{n}} |T_{\mathrm{I}}^{(p)}(x)| \, \mathrm{d}x \leq C \left(\frac{\pi}{2} \right)^{p} \left(\frac{n}{a} \right)^{p-1+\alpha} \frac{(p+\alpha-2+\delta_{0,\alpha})!}{(p-1)!} \left(S_{\mathrm{even}}(p) + S_{\mathrm{odd}}(p) \right),$$

wobei wir die Summen mit

$$S_{\mathrm{even}}(p) := \sum_{k=1}^{q} \left(\frac{2}{\pi} \right)^{2k} \sum_{j=1}^{q} (2j)^p \, |L_{2j,p}^{(2k)}(0)| \,, \tag{2.36}$$

$$S_{\mathrm{odd}}(p) := \sum_{k=0}^{q} \left(\frac{2}{\pi} \right)^{2k+1} \sum_{j=0}^{q} (2j+1)^p |L_{2j+1,p}^{(2k+1)}(0)| \tag{2.37}$$

abkürzen. Die Abschätzung für diese Summen $S_{\mathrm{even}}(p)$ und $S_{\mathrm{odd}}(p)$ werden wir in Lemma 2.9 geben. Nutzen wir das dortige Ergebnis (2.41), so folgt

$$\int_0^{\frac{a}{n}} |T_{\mathrm{I}}^{(p)}(x)| \, \mathrm{d}x \leq C \, (p-2+\alpha+\delta_{0,\alpha})! \left(\frac{\pi}{2} \right)^{p} \left(\frac{n}{a} \right)^{p-1+\alpha} \left(\frac{\beta\pi}{2} \right)^{p}. \tag{2.38}$$

Das dritte Integral in (2.30) kann in analoger Weise wie das erste Integral in (2.30) behandelt werden, allerdings mit den Koeffizienten a_j^{B} anstatt a_j^{I}. Die Koeffizienten a_j^{B} erhalten wir wie in (2.35), nur mit kleineren Werten auf der rechten Seite von (2.15). Weil die Inverse unserer Vandermonde-ähnlichen Matrix $(\boldsymbol{V}_{\mathrm{even}}^{\mathrm{I}})^{\mathrm{T}}$ nur positive Einträge enthält, folgt daraus, dass das dritte Integral in (2.30) kleiner als das erste Integral ist. Wir beenden den Beweis durch Kombination der Ergebnisse in (2.38), (2.31) und (2.30). ∎

Im obigen Satz haben wir bereits folgendes Lemma benutzt.

Lemma 2.9 *Es sei $p \geq 2$ eine gerade natürliche Zahl und q durch $q := p/2 - 1$ gegeben. Weiterhin seien $L_{2j,p}$ und $L_{2j+1,p}$ durch (2.33) und (2.34) definiert. Die Summen $S_{\mathrm{even}}(p)$ und $S_{\mathrm{odd}}(p)$ in (2.36) und (2.37) können durch*

$$
\begin{aligned}
S_{\mathrm{even}}(p) &\leq \left(\frac{2}{\pi}\right)^2 \frac{1}{2} \left(\frac{\beta}{2}\right)^{p-4} \sum_{k=0}^{q-1} (p-2k-2)^p \, \frac{\left[\frac{2^k q!}{(q-k)!}\right]^2}{k! \frac{(p-k-2)!}{(p-2k-2)!}} \,, && (2.39)\\
S_{\mathrm{odd}}(p) &\leq \left(\frac{2}{\pi}\right) \left(\frac{\beta}{2}\right)^{p-2} \sum_{k=0}^{q} (p-2k-1)^p \, \frac{\left[\frac{(p-1)!(q-k)!}{2^k (p-2k-1)! q!}\right]^2}{k! \frac{(p-k-1)!}{(p-2k-1)!}} && (2.40)
\end{aligned}
$$

abgeschätzt werden. Für $p \leq 50$ gilt die Ungleichung

$$
S_{\mathrm{even/odd}}(p) < \frac{1}{2} \left(\frac{\beta\pi}{2}\right)^p (p-1)! \,. \tag{2.41}
$$

Beweis: Wir beschränken uns hier auf den Beweis von $S_{\mathrm{even}}(p)$. Die Abschätzung für $S_{\mathrm{odd}}(p)$ kann in ähnlicher Weise erfolgen. Mit der Definition (2.33) ergibt sich

$$
L_{p-2,p}(x) = \frac{x^2}{(p-2)^2} \prod_{s=1}^{q-1} \frac{x^2 - (2s)^2}{(p-2)^2 - (2s)^2} = \frac{1}{(p-2)!} \frac{1}{2^{p-3}} \prod_{s=0}^{q-1} (x^2 - (2s)^2)
$$

sowie

$$
L_{2j,p}(x) = L_{2j,p-2}(x) \frac{x^2 - (p-2)^2}{(2j)^2 - (p-2)^2} \qquad (j = 1, \ldots, q-1) \,.
$$

Für $k = 1, \ldots, q$ folgt aus der Regel von Leibniz

$$
\begin{aligned}
L_{2j,p}^{(2k)}(x) &= L_{2j,p-2}^{(2k)}(x) \frac{x^2 - (p-2)^2}{(2j)^2 - (p-2)^2} + \frac{2x\, 2k}{(2j)^2 - (p-2)^2} L_{2j,p-2}^{(2k-1)}(x) \\
&\quad + \frac{(2k)(2k-1)}{(2j)^2 - (p-2)^2} L_{2j,p-2}^{(2k-2)}(x)
\end{aligned}
$$

und insbesondere für $x = 0$ die Identität

$$
L_{2j,p}^{(2k)}(0) = L_{2j,p-2}^{(2k)}(0) \frac{(p-2)^2}{(p-2)^2 - (2j)^2} - L_{2j,p-2}^{(2k-2)}(0) \frac{2k(2k-1)}{(p-2)^2 - (2j)^2} \,. \tag{2.42}
$$

Wir substituieren dies in $S_{\mathrm{even}}(p)$ und beachten, dass $L_{2j,p-2}^{(2k)}(0)$ und $L_{2j,p-2}^{(2k-2)}(0)$ unterschiedliche Vorzeichen haben, so dass die Abschätzung

$$\begin{aligned}
S_{\text{even}}(p) &= \sum_{k=1}^{q}\left(\frac{2}{\pi}\right)^{2k}(p-2)^p|L^{(2k)}_{p-2,p}(0)| \\
&+ (p-2)^2\sum_{k=1}^{q}\left(\frac{2}{\pi}\right)^{2k}\sum_{j=1}^{q-1}\frac{(2j)^p}{(p-2)^2-(2j)^2}|L^{(2k)}_{2j,p-2}(0)| \\
&+ \sum_{k=1}^{q}\left(\frac{2}{\pi}\right)^{2k}2k(2k-1)\sum_{j=1}^{q-1}\frac{(2j)^p}{(p-2)^2-(2j)^2}|L^{(2k-2)}_{2j,p-2}(0)| \\
&\leq (p-2)^p\sum_{k=1}^{q}\left(\frac{2}{\pi}\right)^{2k}|L^{(2k)}_{p-2,p}(0)| \\
&+ \left(1+\left(\frac{2}{\pi}\right)^2\right)(p-2)^2\sum_{k=1}^{q-1}\left(\frac{2}{\pi}\right)^{2k}\sum_{j=1}^{q-1}\frac{(2j)^p}{(p-2)^2-(2j)^2}|L^{(2k)}_{2j,p-2}(0)|
\end{aligned}$$

folgt. Nach Anwendung der Gleichung (2.42) mit $p-2$ anstelle von p in der letzten Summe wird dieses Vorgehen iteriert. Zur Abkürzung sei

$$R(p) := \sum_{k=1}^{q}\left(\frac{2}{\pi}\right)^{2k}|L^{(2k)}_{p-2,p}(0)|.$$

Wir erhalten auf diesem Weg die Summe

$$\begin{aligned}
S_{\text{even}}(p) &\leq (p-2)^pR(p)+(p-2)^2\,\beta^2\,\frac{(p-4)^p}{(p-2)^2-(p-4)^2}R(p-2) \\
&+ (p-2)^2(p-4)^2\,\beta^4\,\frac{(p-6)^p}{((p-6)^2-(p-2)^2)((p-6)^2-(p-4)^2)}R(p-4)\;+\;\ldots \\
&+ (p-2)^2\ldots 4^2\,\beta^{p-4}\,\frac{2^p}{(2^2-(p-2)^2)(2^2-(p-4)^2)\ldots(2^2-4^2)}R(4) \\
&= \sum_{k=0}^{q-1}\beta^{2k}\,(p-2k-2)^p\,R(p-2k)\,\frac{\left[\frac{q!}{(q-k)!}\right]^2}{k!\,\frac{(p-k-2)!}{(p-2k-2)!}}\,. \qquad (2.43)
\end{aligned}$$

Nun betrachten wir

$$R(p) = \frac{1}{(p-2)!}\frac{1}{2^{p-3}}\sum_{k=1}^{q}\left(\frac{2}{\pi}\right)^{2k}|w_p^{(2k)}(0)|\,,$$

wobei

$$w_p(x) = \prod_{s=0}^{q-1}(x^2-(2s)^2)\,.$$

Da aber

$$w_p(x) = w_{p-2}(x)(x^2-(p-4)^2)$$

ist, folgt aus der Regel von Leibniz

$$w_p^{(2k)}(0) = -w_{p-2}^{(2k)}(0)(p-4)^2 + 2k(2k-1)w_{p-2}^{(2k-2)}(0).$$

Damit kann die Summe $R(p)$ in der Form

$$\begin{aligned} R(p) &\le \frac{1}{(p-2)!}\frac{1}{2^{p-3}}\left((p-4)^2\sum_{k=1}^{q-1}\left(\frac{2}{\pi}\right)^{2k}|w_{p-2}^{(2k)}(0)|\right. \\ &\qquad \left. + (p-2)(p-3)\left(\frac{2}{\pi}\right)^2\sum_{k=1}^{q-1}\left(\frac{2}{\pi}\right)^{2k}|w_{p-2}^{(2k)}(0)|\right) \\ &\le \frac{1}{(p-4)!}\frac{\beta^2}{2^{p-3}}\sum_{k=1}^{q-1}\left(\frac{2}{\pi}\right)^{2k}|w_{p-2}^{(2k)}(0)| \;=\; \left(\frac{\beta}{2}\right)^2 R(p-2) \\ &\le \left(\frac{\beta}{2}\right)^{p-4} R(4) \;=\; \left(\frac{\beta}{2}\right)^{p-4}\left(\frac{2}{\pi}\right)^2\frac{1}{2} \end{aligned}$$

geschrieben werden. Substitution des oberen Ausdrucks für $R(p-2k)$ in (2.43) ergibt die Behauptung (2.39).
Die Abschätzung (2.41) erhalten wir einfach aus den Ungleichungen (2.39) und (2.40) mit Hilfe eines Computers. Damit ist der Beweis vollständig. ■

Folgerung 2.10 *Mit der Hölder-Ungleichung* (2.28), *Satz 2.8 und der Stirling-Formel*

$$\lambda! < 1.1\left(\frac{\lambda}{\mathrm{e}}\right)^{\lambda}\sqrt{2\pi\lambda}$$

erhalten wir die Fehlerabschätzung

$$|f(\boldsymbol{y}_j) - \tilde{f}(\boldsymbol{y}_j)| \le C\,\|\boldsymbol{\alpha}\|_1 n^{\alpha}\, a^{\delta_{0,\alpha}}\,\frac{\sqrt{p(p+\alpha+\delta_{0,\alpha}-2)}}{p-2}\left(\frac{p+\alpha+\delta_{0,\alpha}-2}{a}\,\frac{\beta\pi}{4\,\mathrm{e}}\right)^{p+\alpha+\delta_{0,\alpha}-2}.$$

Der Fehler fällt also exponentiell mit p, *falls*

$$\frac{\beta\pi}{4\,\mathrm{e}}\,\frac{p+\alpha+\delta_{0,\alpha}-2}{a} < 1 \qquad \left(\frac{\beta\pi}{4\,\mathrm{e}} \approx \frac{1}{3}\right).$$

In unseren numerischen Beispielen setzen wir einfach $a = p$. *Bessere Näherungswerte ergeben sich natürlich für größere Werte von* a, *jedoch steigt damit auch die arithmetische Komplexität zur Berechnung des „Nahfeldes“ in unserem Summationsalgorithmus 2.3.*

Umfangreiche numerische Beispiele in Abschnitt 2.4 bestätigen (siehe insbesondere Beispiel 2.1 sowie Abbildung 2.2) die vorangegangenen theoretischen Aussagen.

Nichtsinguläre Kerne

Im Folgenden leiten wir die Fehlerabschätzung für die schnelle Gauß-Transformation, die mit Hilfe von Algorithmus 2.6 ohne Regularisierung berechnet wird, her. An diesem Beispiel werden Fehlerabschätzungen im bivariaten Fall, d.h. für $d = 2$, ausgearbeitet. Nutzen wir wieder die Hölder-Ungleichung (2.28), so genügt es, den Fehler zwischen $\mathcal{K}$ und der Summe $\mathcal{K}_{\mathrm{RF}}$ zu berechnen. Wir formulieren den folgenden Satz.

Satz 2.11 *Es sei* $\delta \geq 2$, $\mathcal{K}(\boldsymbol{x}) := \mathrm{e}^{-\delta\|\boldsymbol{x}\|_2^2}$ *und* $\boldsymbol{x} \in \mathbb{R}^d$ *mit* $d = 2$. *Weiterhin sei die Differenz* $\mathcal{K}_{\mathrm{ER}} := \mathcal{K} - \mathcal{K}_{\mathrm{RF}}$, *wobei* $\mathcal{K}_{\mathrm{RF}}$ *die endliche Fourier-Summe* (2.18) *von* $\mathcal{K}$ *ist. Mit* $\eta := \frac{\pi n}{2\sqrt{\delta}} \geq 1$, *gilt dann die Abschätzung*

$$\|\mathcal{K}_{\mathrm{ER}}\|_\infty \leq 20 \max\{\frac{1}{\eta}, \frac{1}{\sqrt{\delta}}\}\ \mathrm{e}^{-\eta^2} \ + \ 40 \frac{\sqrt{\delta}}{\eta}\ \mathrm{e}^{-\delta/4}. \tag{2.44}$$

Beweis: Die Fourier-Transformierte der univariaten Gauß-Funktion $\mathrm{e}^{-\delta x^2}$ ist in (1.33) durch

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

gegeben. Für den weiteren Beweis werden wir die folgenden Abschätzungen benutzen:

$$\sum_{k=1}^{n/2} \frac{1}{k^2} \ < \ \sum_{k=1}^{\infty} \frac{1}{k^2} \ = \ \frac{\pi^2}{6}, \tag{2.45}$$

$$\sum_{k=1}^{n/2} \mathrm{e}^{-k^2\pi^2/\delta} \ < \ \int_0^{\infty} \mathrm{e}^{-x^2\pi^2/\delta}\, \mathrm{d}x \ = \ \frac{1}{2}\sqrt{\frac{\delta}{\pi}}, \tag{2.46}$$

$$\sum_{k=n/2+1}^{\infty} \frac{1}{k^2} \ < \ \int_{n/2}^{\infty} \frac{1}{x^2}\, \mathrm{d}x \ = \ \frac{2}{n}, \tag{2.47}$$

$$\sum_{k=n/2+1}^{\infty} \mathrm{e}^{-k^2\pi^2/\delta} \ < \ \int_{n/2}^{\infty} \mathrm{e}^{-x^2\pi^2/\delta}\, \mathrm{d}x \ < \ \frac{\delta}{n\pi^2}\ \mathrm{e}^{-\pi^2 n^2/(4\delta)}, \tag{2.48}$$

wobei die letzte Ungleichung aus (1.35) folgt.
Wir wenden partielle Integration zweimal auf das folgende Integral an und erhalten für $k \neq 0$ die Identität

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

und damit für $\delta \geq 2$ die Abschätzung

$$
\begin{aligned}
|c_k\left(\mathrm{e}^{-\delta x^2}\right)| &\leq \frac{\delta}{2\pi^2 k^2}\,\mathrm{e}^{-\delta/4} + \sqrt{\frac{\pi}{\delta}}\,\mathrm{e}^{-k^2\pi^2/\delta} + \frac{\delta}{\pi^2 k^2}\int\limits_{1/2}^{\infty}(2\delta x^2-1)\,\mathrm{e}^{-\delta x^2}\,\mathrm{d}x \\
&= \sqrt{\frac{\pi}{\delta}}\,\mathrm{e}^{-k^2\pi^2/\delta} + \frac{\delta}{\pi^2 k^2}\,\mathrm{e}^{-\delta/4}. \qquad (2.49)
\end{aligned}
$$

Da wir keine Regularisierung für den Gauß-Kern benutzen, folgt für $\mathcal{K}_{\mathrm{R}} = \mathcal{K}$ aus der Aliasingformel (2.4) die Ungleichung

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

Es verbleiben die Summen S_1 und S_2 abzuschätzen. Die Tensorproduktstruktur der bivariaten Gauß-Funktion ergibt für $\boldsymbol{k} := (k_1,k_2)^{\mathrm{T}}$ und $\boldsymbol{x} := (x_1,x_2)^{\mathrm{T}}$ die Identität

$$c_{\boldsymbol{k}}(\mathcal{K}) = c_{k_1}(\mathrm{e}^{-\delta x_1^2})c_{k_2}(\mathrm{e}^{-\delta x_2^2}).$$

Vergrößern wir S_1 und verwenden wir (2.49), so erhalten wir

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

und weiter mit (2.45) und (2.46)

$$S_1 \leq 2\,C\left(\sqrt{\frac{\pi}{\delta}}\,\mathrm{e}^{-\eta^2} + \frac{1}{\eta^2}\,\mathrm{e}^{-\delta/4}\right),$$

wobei $C := \left(1 + \frac{\delta}{3}\,\mathrm{e}^{-\delta/4} + \sqrt{\frac{\pi}{\delta}}\right)$ ist. Die zweite Summe S_2 kann in der Form

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

geschrieben werden. Vergrößern wir die rechte Seite mit (2.49), (2.47) und (2.48), ergibt sich

$$S_2 \leq 4\,A(n,\delta)\;(A(n,\delta) + C),$$

wobei

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

ist. Mit den Definitionen

$$C_1 := \max\{4\,C\sqrt{\pi}, 4(A(n,\delta)+C)/(\sqrt{\pi})\}\ ,\ C_2 := \max\{8\,C/(\pi n), 8(A(n,\delta)+C)/(\pi)\}$$

folgt die Abschätzung

$$\|\mathcal{K}_{\mathrm{ER}}\|_\infty \le C_1 \max\{\frac{1}{\eta}, \frac{1}{\sqrt{\delta}}\}\ \mathrm{e}^{-\eta^2}\ +\ C_2 \frac{\sqrt{\delta}}{\eta}\ \mathrm{e}^{-\delta/4}\,.$$

Die Behauptung des Satzes ergibt sich mit den Ungleichungen $C < 2.7$ und $A(n,\delta) < 0.4$. ■

Der erste Summand in (2.44) fällt mit wachsendem η. Der zweite Summand ist für große δ zu vernachlässigen, d.h., wir haben $\sqrt{\delta}\ \mathrm{e}^{-\delta/4} < 2.7 \cdot 10^{-6}$ für $\delta \ge 60$. Diese Fehlerabschätzungen bestätigen wir durch numerische Tests in Beispiel 2.7. Insbesondere wird deutlich, dass wir für $\delta < 60$ eine Randregularisierung benötigen. In Tabelle 2.9 geben wir numerische Ergebnisse für $\delta = 1$ mit Randregularisierung an.

2.4 Numerische Ergebnisse

In diesem Abschnitt präsentieren wir einige numerische Beispiele. Die Algorithmen sind in C implementiert und auf einer SGI O2 mit doppelter Genauigkeit getestet.
Wir beginnen mit der Berechnung der Summe (2.1) für den univariaten Fall, d.h. $d = 1$, mit $M = N$ und $y_j = x_j \in \mathbb{R}$ $(j = 1, \dots, N)$. Die Summe

$$f(x_j) = \sum_{\substack{k=1\\k\neq j}}^{N} \alpha_k K(x_j - x_k) \qquad (j = 1, \dots, N)$$

soll für die verschiedenen Kerne (2.7) mittels Algorithmus 2.3 berechnet werden. In allen Berechnungen wählen wir $n = N$, so dass unser Algorithmus 2.3 eine arithmetische Komplexität $\mathcal{O}(N \log N + mN + aN) = \mathcal{O}(N \log N)$ hat. Dabei ist m die Anzahl der Summanden in (1.16) und a die Anzahl der Summanden für die Nahfeldberechnung. Wir wenden die NFFT/NFFT$^{\mathrm{T}}$ mit Kaiser-Bessel-Funktionen φ (siehe Abschnitt 1.3.2) und einem Oversamplingfaktor $\sigma = 2$ an. Diese Berechnungen beruhen noch auf FFT Implementierungen aus „Numerical Recipes in C" [106, S. 523 ff]. Die Laufzeiten unseres Summationsalgorithmus werden sich weiter verbessern, wenn wir das Programmpaket [76], welches auf der FFTW basiert, verwenden.
Die Koeffizienten α_k wurden zufällig in [0,1] und die Knoten x_k wurden zufällig in $[-1/4 + a/(2n), 1/4 - a/(2n)]$ gewählt. Jeder Test zeigt den Durchschnitt von 20 verschiedenen Berechnungen. Wir wollen das Verhalten der Fehler

$$E_\infty := \max_{j=1,\dots,N} \frac{|f(\boldsymbol{x}_j) - \tilde{f}(\boldsymbol{x}_j)|}{|f(\boldsymbol{x}_j)|} \tag{2.50}$$

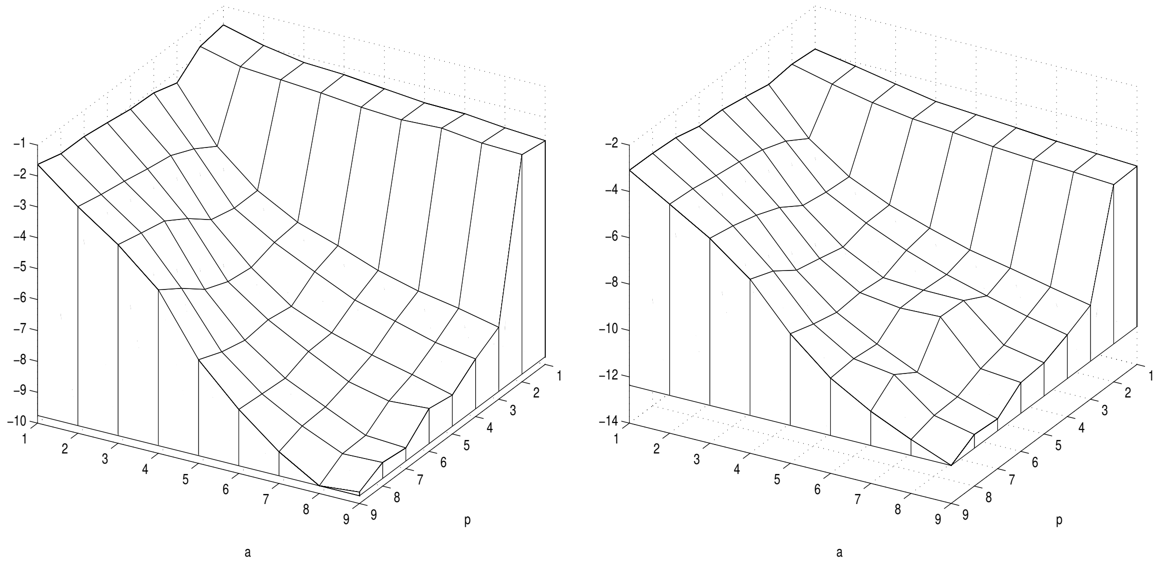

Abbildung 2.2: Fehler $\log_{10} E_\infty$ für $K(x) = 1/|x|$ (links) und $K(x) = 1/|x|^2$ (rechts) für $N = 512$, $m = 12$ und $(p, a) \in \{1, \dots, 9\}^2$, $d = 1$.

und

$$E_1 := \frac{1}{N} \sum_{j=1}^{N} \frac{|f(\boldsymbol{x}_j) - \tilde{f}(\boldsymbol{x}_j)|}{|f(\boldsymbol{x}_j)|} \tag{2.51}$$

untersuchen. Dabei sind $f(x_j)$ $(j = 1, \dots, N)$ die Funktionswerte von f, falls wir einen naiven Summationsalgorithmus der Komplexität $\mathcal{O}(N^2)$ anwenden und $\tilde{f}(x_j)$ die Funktionswerte, falls wir den schnellen approximativen Summationsalgorithmus 2.3 benutzen.

Beispiel 2.1 (Parameterwahl von p und a)
Wir betrachten in diesem Beispiel die Kerne $K(x) = 1/|x|^\alpha$ $(\alpha = 1, 2)$ genauer. Ähnliche Resultate ergeben sich für die anderen Kerne in Formel (2.7). Abbildung 2.2 zeigt den Fehler $\log_{10} E_\infty$ von Algorithmus 2.3 für $(p, a) \in \{1, \dots, 9\}^2$ und $N = 512$. Zunächst setzen wir $m = 12$, damit kein zusätzlicher Fehler durch die NFFT/NFFT$^{\mathrm{T}}$ entsteht. Aus der Abbildung entnehmen wir, dass $a = p$ eine gute Wahl ist. Dies wurde auch durch die theoretischen Untersuchungen in Abschnitt 2.3 bestätigt. □

Beispiel 2.2 (Parameterwahl von m)
Im Folgenden wollen wir den Parameter m so klein wie möglich wählen, damit der konstante Faktor in der arithmetischen Komplexität der NFFT/NFFT$^{\mathrm{T}}$ möglichst klein ist. Abbildung 2.3 zeigt den Fehler $\log_{10} E_\infty$ für $a = p \in \{1, \dots, 9\}$ und $m \in \{1, \dots, 9\}$ für die Kerne $K(x) = 1/|x|$ (links) und $K(x) = 1/|x|^2$ (rechts), wobei $N = 512$ ist. Wir sehen, dass für $p \le 5$ die Wahl $a = p = m$ sehr gute Resultate liefert, während für größere Werte von p die Wahl $m = 5$ ausreichend ist. □

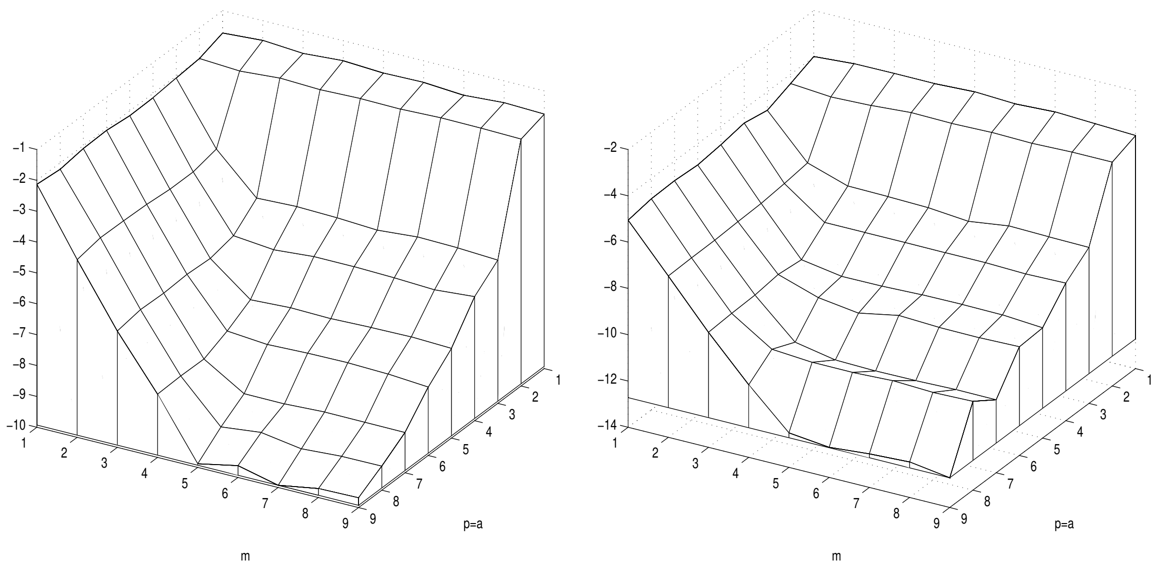

Abbildung 2.3: Fehler $\log_{10} E_\infty$ für $K(x) = 1/|x|$ (links) und $K(x) = 1/|x|^2$ (rechts) für $(a, m) \in \{1, \ldots, 9\}^2$ und $a = p$, $N = 512$, $d = 1$.

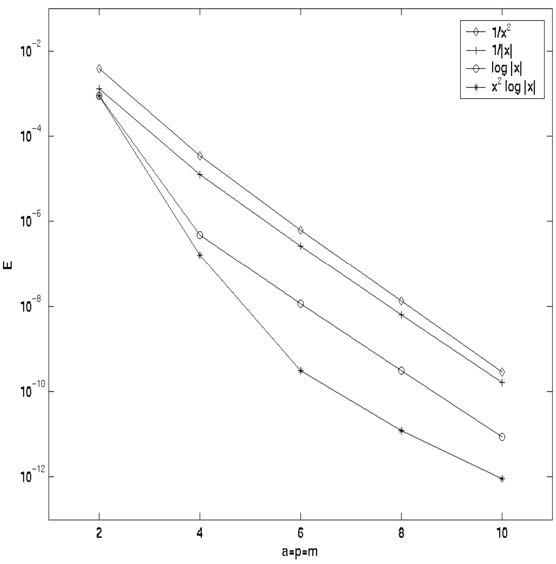

Abbildung 2.4: Fehler E_∞ für verschiedene Kerne mit $a = p = m \in \{2, 4, \ldots, 10\}$, wobei $N = 512$, $d = 1$.

Beispiel 2.3 (Genauigkeit im univariaten Fall)
Die Abbildung 2.4 zeigt numerische Ergebnisse für die Kerne (2.7). Wir betrachten den Fehler E_∞ als Funktion von $p = a = m$. Diese Ergebnisse bestätigen den exponentiellen Abfall von E_∞ mit wachsendem p (siehe Satz 2.8 und Folgerung 2.10). $\square$

	CPU-Zeiten				Fehler	
N	t_{slow}	t_{NE}	t_{app}	t_{FMM}	E_∞	E_{FMM}
64	1.152e-03	1.193e-03	2.952e-03	1.584e-03	1.634e-06	1.060e-05
128	4.608e-03	2.429e-03	5.911e-03	3.243e-03	6.778e-06	1.580e-05
256	1.849e-02	4.938e-03	1.194e-02	6.738e-03	4.521e-06	1.223e-05
512	7.396e-02	9.557e-03	2.419e-02	1.450e-02	6.366e-06	5.0164e-06
1024	3.010e-01	1.956e-02	5.034e-02	3.940e-02	9.184e-06	4.010e-06
2048	1.524e+00	3.953e-02	1.171e-01	8.093e-02	9.483e-06	5.676e-06
4096	6.702e+00	7.843e-02	2.559e-01	2.263e-01	4.256e-06	1.770e-06
8192	2.730e+01	1.590e-01	5.997e-01	1.112e+00	5.449e-06	1.372e-06

Tabelle 2.5: Vergleich der CPU-Zeiten und der Approximationsfehler von Algorithmus 2.3 mit $a = p = m = 4$ und der FMM für den Kern $K(x) = 1/|x|$.

Beispiel 2.4 (CPU-Zeiten und Vergleich mit FMM im univariaten Fall)
Schließlich zeigt Tabelle 2.5 einen Vergleich der CPU-Zeiten von unserem Algorithmus mit der FMM, die in [30, Algorithmus 3.2] vorgeschlagen wurde. Unsere Implementierung der FMM benutzt die Chebyshev-Polynome bis zum Grad 5, um einen Fehler $E \leq 10^{-5}$ zu erhalten. Es sei erwähnt, dass kürzlich ein etwa doppelt so schneller FMM-Algorithmus (siehe [131]) vorgeschlagen wurde. Mit t_{slow}, t_{app} und t_{FMM} bezeichnen wir die CPU-Zeiten für die naive langsame Summation, für Algorithmus 2.3 und für den Algorithmus, basierend auf der FMM. Außerdem zeigen wir in Spalte 3 die CPU-Zeit t_{NE} für Schritt 4 von Algorithmus 2.3, unter Beachtung von Bemerkung 2.4. Die sechste und siebente Spalte zeigen den Approximationsfehler E_∞ von Algorithmus 2.3 und von der FMM. □

In den folgenden Beispielen wollen wir den schnellen Summationsalgorithmus für den bivariaten Fall testen. Dazu wird für die NFFT/NFFT$^{\text{T}}$ ein Tensorprodukt aus Kaiser-Bessel-Funktionen für die Ansatzfunktion φ gewählt.

Beispiel 2.5 (Genauigkeit im bivariaten Fall)
Abbildung 2.6 zeigt den Fehler E_∞ für die Parameter $a = p$ und für die verschiedenen Kerne (2.7). Wir haben $n = 256$, $N = 40000$ und $m = 8$ gewählt. Der große Wert von m stellt sicher, dass die NFFT keinen zusätzlichen Approximationsfehler liefert. Auch hier im bivariaten Fall fällt der Fehler exponentiell mit wachsendem p. Weiterhin wächst der Fehler mit wachsender Ordnung der Singularität von $\mathcal{K}$. □

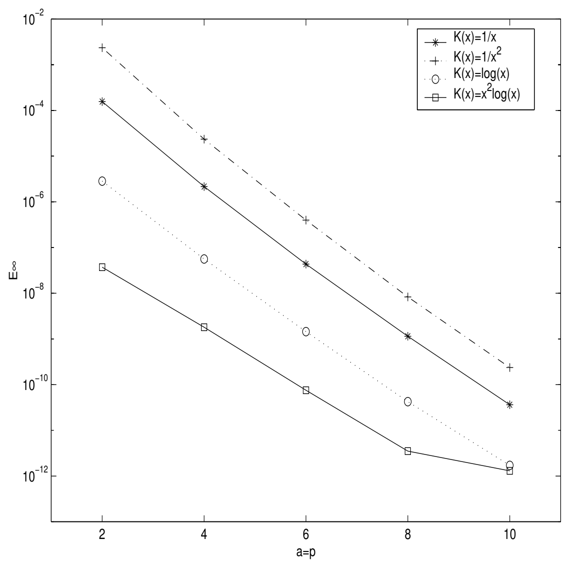

Abbildung 2.6: Fehler E_∞ von Algorithmus 2.3 in Abhängigkeit von $a = p$ für die Kerne (2.7), wobei $n = 256$, $N = 40000$, $m = 8$ und $d = 2$ gewählt wurde.

Parameter		CPU-Zeiten		Fehler
n	N	t_{slow}	t_{app}	E_∞
32	1000	2.950e-01	5.700e-01	1.184e-05
64	4000	4.755e+00	2.305e+00	4.820e-06
128	16000	7.699e+01	1.166e+01	2.815e-06
256	65000	1.502e+03	5.144e+01	1.757e-06
512	65000	1.496e+03	3.314e+01	1.754e-06
512	260000	2.885e+04	2.138e+02	1.026e-06

Tabelle 2.7: Vergleich von CPU-Zeiten für die Approximation von Algorithmus 2.3 für $\mathcal{K}(\boldsymbol{x}) = 1/||\boldsymbol{x}||_2$ mit $a = p = m = 4$, $d = 2$.

Beispiel 2.6 (CPU-Zeiten im bivariaten Fall)
In diesem Beispiel werden die CPU-Zeiten im bivariaten Fall untersucht. Wir testen dies am Beispiel für den Kern $\mathcal{K}(\boldsymbol{x}) = 1/\|\boldsymbol{x}\|_2$. Mit t_{slow} und t_{app} bezeichnen wir wieder die CPU-Zeiten für den naiven, langsamen Algorithmus und für Algorithmus 2.3. Man beachte, dass in t_{app} auch die Zeit für das Suchen aller Punkte im Nahfeld enthalten sind. Dies kann aber auch für jeden Punkt in $\mathcal{O}(\log N)$ Schritten erfolgen. $\square$

In den beiden folgenden Beispielen wird Algorithmus 2.6 mit den glatten Kernen (siehe (2.25))

$$\mathrm{e}^{-\delta x^2}, (x^2 + c^2)^{\pm 1/2}$$

getestet.

Beispiel 2.7 (Schnelle Gauß-Transformation)
Wir beginnen mit der Gauß-Funktion $\mathcal{K}(\boldsymbol{x}) = \mathrm{e}^{-\delta \|2\boldsymbol{x}\|_2^2}$. Hier wurde der Faktor 2 eingeführt, weil die Berechnung in [54] mit $||\boldsymbol{x}||_2 \leq 1/2$ an Stelle von $||\boldsymbol{x}||_2 \leq 1/4$ stattfand. Tabelle 2.8 zeigt sowohl die Approximationsfehler als auch die CPU-Zeiten unseres Algorithmus. Mit t_{slow} und t_{app} bezeichnen wir wieder die CPU-Zeiten für den langsamen Algorithmus bzw. für Algorithmus 2.6. In Übereinstimmung mit Algorithmus 2.6 sehen wir, dass die arithmetische Komplexität $\mathcal{O}(N)$ ist. Weiterhin zeigen die numerischen Experimente, dass wir $n^2 \approx \delta$ setzen sollten, um einen festen Wert für den Fehler E_∞ zu erhalten. Damit bestätigen die numerischen Tests die theoretische Aussage von Satz 2.11.
Natürlich kann für $\delta = 1$ der Approximationsfehler weiter verbessert werden, wenn wir auch an den Rändern regularisieren. Für festes n benötigt die Randregularisierung keine weitere Rechenzeit. Tabelle 2.9 zeigt den Approximationsfehler von Algorithmus 2.6 mit Randregularisierung für verschiedene Parameter n und die entsprechend besten Parameter p. □

Beispiel 2.8 ((Inverse) Multiquadric)
In diesem Beispiel wenden wir Algorithmus 2.6 mit Randregularisierung für die Multiquadric und die inverse Multiquadric $\mathcal{K}(\boldsymbol{x}) = (\|\boldsymbol{x}\|_2^2 + c^2)^{\pm 1/2}$ an. Weil die CPU-Zeiten für diese Kerne die gleichen sind wie in Beispiel 2.7, wird nur der Approximationsfehler untersucht. Abbildung 2.10 zeigt den Approximationsfehler in Abhängigkeit vom Parameter p für die Randregularisierung.
Für den Parameter $c = 0$ stimmt die Multiquadric und die inverse Multiquadric mit den Funktionen $K(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^{\pm 1}$ überein. Dann haben die Kerne bzw. deren Ableitungen Singularitäten im Nullpunkt. Dies erklärt, warum der Approximationsfehler für kleiner werdende c wächst. Für sehr kleine Parameterwerte von c müssen wir unseren Kern wieder in der Nähe von Null regularisieren, d.h., wir müssen Algorithmus 2.3 an Stelle von Algorithmus 2.6 anwenden. Dies ist z.B. für den Fall $c = 1/\sqrt{N}$ nötig (siehe [22]). □

Beispiel 2.9 (Schnelle reelle Summation)
In diesem Beispiel wollen wir den schnellen Summationsalgorithmus 2.3 modifizieren. Wir verbessern den Algorithmus für reelle Eingabedaten. Dazu nutzen wir die Algorithmen zur schnellen Kosinus- und Sinus-Transformation für nichtäquidistante Knoten (siehe die Algorithmen 1.13, 1.14 und 1.15). Wir wenden diese Algorithmen zur schnellen Berechnung der Summe

$$f(y) = \sum_{k=1}^{N} \alpha_k K(y - x_k)$$

Parameter			CPU-Zeiten		Fehler
δ	n	N	t_{slow}	t_{app}	E_∞
1	32	25000	7.384e+01	1.340e-01	3.659e-05
1	32	50000	2.965e+02	2.700e-01	3.808e-05
1	32	100000	1.187e+03	5.400e-01	3.647e-05
15	16	25000	7.400e+01	1.360e-01	4.319e-07
15	16	50000	2.971e+02	2.700e-01	4.567e-07
15	16	100000	1.189e+03	5.420e-01	4.447e-07
50	32	25000	7.400e+01	1.340e-01	3.383e-07
50	32	50000	2.971e+02	2.700e-01	3.485e-07
50	32	100000	1.192e+03	5.400e-01	3.577e-07
100	64	25000	7.392e+01	1.560e-01	3.354e-07
100	64	50000	2.968e+02	2.960e-01	3.407e-07
100	64	100000	1.189e+03	5.780e-01	3.525e-07
200	64	25000	7.399e+01	1.560e-01	3.832e-07
200	64	50000	2.972e+02	2.980e-01	3.696e-07
200	64	100000	1.190e+03	5.800e-01	3.790e-07
800	128	25000	7.398e+01	4.800e-01	5.449e-07
800	128	50000	2.972e+02	6.800e-01	4.769e-07
800	128	100000	1.189e+03	1.046e+00	4.213e-07
2400	256	25000	9.014e+01	1.726e+00	3.480e-07
2400	256	50000	3.614e+02	1.924e+00	3.435e-07
2400	256	100000	1.447e+03	2.490e+00	3.388e-07
10000	512	25000	1.238e+02	7.372e+00	3.538e-07
10000	512	50000	4.977e+02	7.484e+00	3.384e-07
10000	512	100000	1.983e+03	8.242e+00	3.523e-07

Tabelle 2.8: Vergleich der Approximationsfehler von Algorithmus 2.6 ohne Randregularisierung für den Kern $\mathcal{K}(\boldsymbol{x}) = \mathrm{e}^{-\delta\|2\boldsymbol{x}\|_2^2}$, $m = 4$ und $d = 2$.

Parameter		Fehler	
n	p	E_1	E_∞
32	2	1.699e-06	6.418e-06
64	4	1.474e-08	1.666e-07
128	6	2.268e-10	2.074e-09
256	8	4.027e-13	3.739e-12

Tabelle 2.9: Vergleich der Approximationsfehler von Algorithmus 2.6 mit Randregularisierung für den Kern $\mathcal{K}(\boldsymbol{x}) = \mathrm{e}^{-\|2\boldsymbol{x}\|_2^2}$, $N = 10000$, $m = 8$ und $d = 2$.

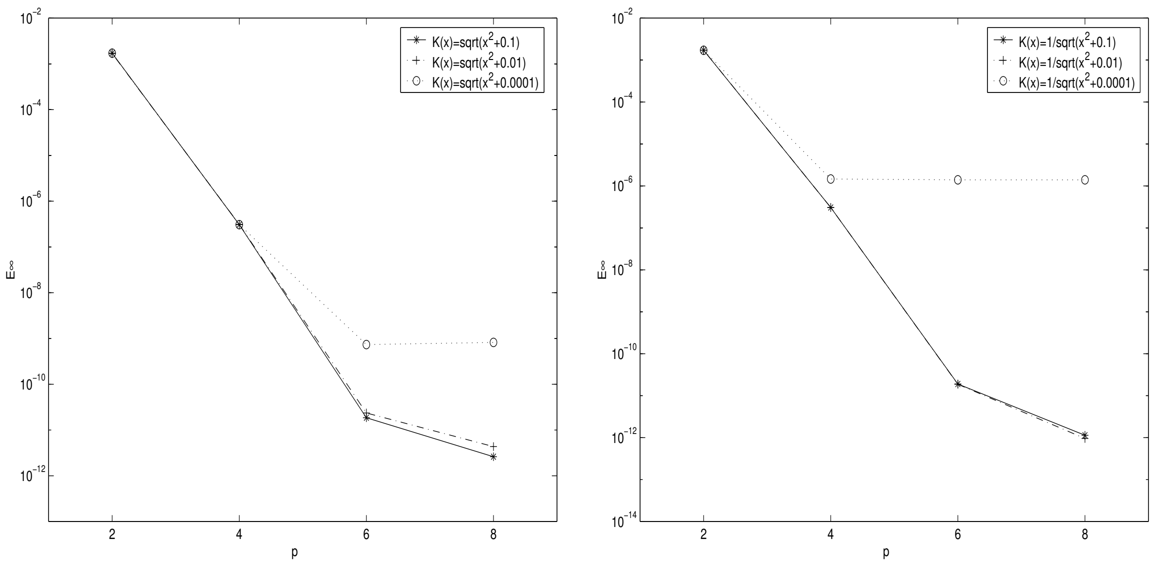

Abbildung 2.10: Fehler E_∞ für Algorithmus 2.6 in Abhängigkeit vom Regularisationsparameter p für die Multiquadric (links) und für die inverse Multiquadric (rechts), wobei $n = 256$, $N = 40000$ und $m = 8$, $d = 2$ gewählt wurden.

an den Knoten $y_j \in \mathbb{R}$ $(j = 1, \ldots, M)$ an. Der glatte Anteil des Kerns K wird im Gegensatz zu Formel (2.18) in der Form

$$K_{\mathrm{RC}}(x) := \sum_{l=0}^{n-1} b_l \cos(2\pi l x)$$

geschrieben. Die Koeffizienten b_l $(l = 0, \ldots, n-1)$ berechnen wir ähnlich wie in Abschnitt 2.2 durch

$$b_l := \frac{2\varepsilon_{n,l}}{n} \sum_{j=0}^{n} \varepsilon_{n,j} K_{\mathrm{R}}\left(\frac{j}{2n}\right) \cos\left(\frac{\pi l j}{n}\right). \tag{2.52}$$

Dazu ist K_{R} wieder der regularisierte Kern wie in (2.8). Wir gehen jetzt analog wie in Abschnitt 2.2 vor und zerlegen den Kern K in die Form $K = K_{\mathrm{NE}} + K_{\mathrm{ER}} + K_{\mathrm{RC}}$ mit $K_{\mathrm{NE}} := K - K_{\mathrm{R}}$, $K_{\mathrm{ER}} := K_{\mathrm{R}} - K_{\mathrm{RC}}$ und vernachlässigen wieder K_{ER}. Also approximieren wir K durch $K_{\mathrm{NE}} + K_{\mathrm{RC}}$ und damit f durch $\tilde{f}(y) := f_{\mathrm{NE}}(y) + f_{\mathrm{RC}}(y)$, wobei

$$f_{\mathrm{RC}}(y) := \sum_{k=1}^{N} \alpha_k K_{\mathrm{RC}}(y - x_k),$$

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

gesetzt wurden. Die Summe f_{NE} kann analog zur Summe (2.22) sehr schnell berechnet werden. Für die Summe f_{RC} gilt

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

Der Ausdruck in den inneren Klammern von

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

kann mit der $\mathrm{NFCT}^{\mathrm{T}}/\mathrm{NFST}^{\mathrm{T}}$ sehr effizient berechnet werden. Dann folgt die Multiplikation mit den Daten b_l und schließlich eine NFCT/NFST für die Berechnung der äußeren Summen. Zusammenfassend erhalten wir einen modifizierten Algorithmus 2.3 (siehe Algorithmus 3.1 in [40]). Im nächsten numerischen Test vergleichen wir die CPU-Zeiten für diesen reellen Summationsalgorithmus und Algorithmus 2.3. Da eine effiziente Implementierung der DCT bzw. DST von großer Bedeutung für einen Laufzeitvergleich ist, benutzen wir, wie auch in Algorithmus 2.3 für die FFT, für die DCT und die DST Implementierungen von „Numerical Recipes in C" [106]. Die Algorithmen wurden in C implementiert und auf einem Rechner AMD Athlon(tm) XP1800+, 512MB RAM, SuSe–Linux mit doppelter Genauigkeit getestet. Die Koeffizienten α_k sind zufällig in $[0, 1]$ verteilt und die Knoten $x_k = y_k$ $(k = 1, \ldots, N; M = N)$ haben wir zufällig in $[0, \frac{1}{2} - \frac{a}{2n}]$ gewählt. In diesen Algorithmen ist $n = N/2$ gesetzt worden und jedes Resultat der Durchschnitt von 20 Testläufen. Abbildung 2.11 vergleicht die CPU-Zeiten des schnellen Summationsalgorithmus, basierend auf der NFCT/NFST, mit Algorithmus 2.3 für den Kern $K(x) = 1/|x|$. In den Algorithmen für die schnellen nichtäquidistanten Transformationen haben wir $m = a = p = 4$ und als Ansatzfunktion φ die Kaiser-Bessel-Funktion gewählt. Der Approximationsfehler

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

für diese schnellen approximativen Algorithmen ist dann etwa 10^{-5}. $\square$

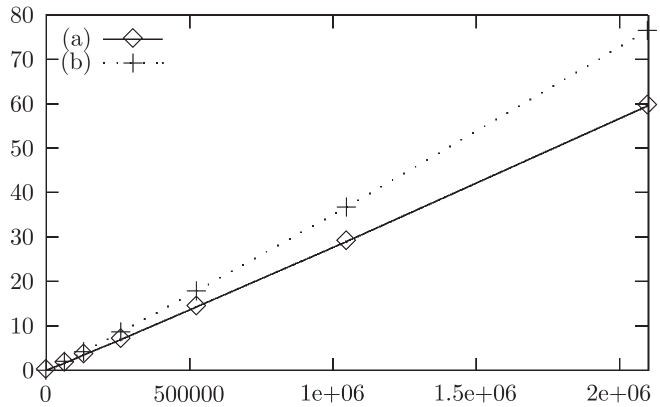

Abbildung 2.11: CPU-Zeiten (in Sekunden) von (a) Algorithmus basierend auf NFCT/NFST und (b) Algorithmus 2.3 in Abhängigkeit von N.

Weitere Themen und Literaturhinweise

Die multivariate Interpolation mit radialen Basisfunktionen ist auf Grund ihrer Flexibilität eine sehr populäre Methode und wird in verschiedenen Bereichen benutzt.

Radiale Basisfunktionen

Die neuen Summationsalgorithmen beruhen auf einer Verallgemeinerung von äquidistanten Daten auf nichtäquidistante Daten. Damit sind schnelle Matrix-Vektor-Multiplikationen sehr effizient möglich. Um eine schnelle Konvergenz von iterativen Lösern zu erreichen, ist die Konstruktion von Vorkonditionierern eine weitere Aufgabe. Im Zusammenhang mit der Multipol-Methode wurde dieses Thema bisher in [3, 5] behandelt. Da für äquidistante Daten der Zusammenhang zu Toeplitz-Matrizen in Abschnitt 2.2 verdeutlicht wurde, ist ein Übertragen der Konstruktion von Vorkonditionierern für Toeplitz-Matrizen (siehe z.B. [19, 94]) auf den nichtäquidistanten Fall von großem Interesse. Durch die Nutzung der Theorie der Toeplitz-Operatoren (siehe z.B. [56]) hat B. Baxter [2] effiziente Vorkonditionierer für Toeplitz-Matrizen, die durch Interpolation von äquidistanten Daten mit der Gauß-Funktion und der Multiquadric entstehen, konstruiert und deren Effizienz nachgewiesen.

Numerische Lösung von Operatorgleichungen

Weitere sehr interessante Fragestellungen ergeben sich im Kontext der numerischen Lösung von Integralgleichungen. Wir verweisen in diesem Zusammenhang auf das sehr interessante Buch von M.A. Goldberg und C.S. Chen [48]. Schnelle Summationsalgorithmen sind z.B. in Verbindung mit der Methode der Fundamentallösung von großem Interesse. Viel Aufmerksamkeit erlangten in den letzten Jahren die sogenannten „gitterlosen Methoden“ (meshless methods), da die Gittergenerierung in vielen Anwendungen die meiste Rechenzeit benötigt. Für einen Vergleich zweier gitterloser Methoden verweisen wir auf [81]. Die Anwendungen unserer schnellen Summationsalgorithmen auf das näherungsweise Lösen von Operatorgleichungen soll in Zukunft untersucht werden.

3 Fourier-Rekonstruktion in der Computer-Tomographie

Ein Spezialfall der Computer-Tomographie besteht in der Rekonstruktion einer Scheibe (gr. Tomos) eines Körpers durch Messung der Absorption von Röntgenstrahlen, die den Körper durchdringen. Dieser Vorgang kann durch die Radon-Transformation modelliert werden. Eine Rekonstruktion des Bildes aus den Messdaten entspricht dann einer numerischen Invertierung der Radon-Transformation.

Die zu besprechenden Verfahren sind auf die parallele, äquidistante Anordnung der Linienintegrale aus der jeweiligen Richtung ausgelegt (Standard-Parallelgeometrie). Ein als Projektionssatz oder Fourier-Slice-Theorem bekanntes Ergebnis besagt, dass die eindimensionale Fourier-Transformation einer Parallel-Projektion der Fourier-Transformation des Objektes auf einem Zentralstrahl entspricht, der senkrecht zur Projektionsrichtung liegt. Die Diskretisierung der Integrale dieses Satzes und Realisierung wird als Fourier-Rekonstruktion bezeichnet.

In diesem Abschnitt werden wir zwei neue Fourier-Rekonstruktions-Algorithmen für digitale Bilder mit N Zeilen und N Spalten vorstellen. Für eine umfassende mathematische Darstellung der Radon-Transformation und daraus resultierender Algorithmen verweisen wir auf [85, 86].

Der Standard-Rekonstruktions-Algorithmus, zur Zeit in fast allen Computer-Tomographen implementiert, ist als gefilterte Rückprojektion bekannt und liefert qualitativ gute Bilder. Allerdings ist die arithmetische Komplexität der Algorithmen $\mathcal{O}(N^3)$. Fourier-Rekonstruktions-Algorithmen reduzieren die Anzahl der arithmetischen Operationen auf $\mathcal{O}(N^2 \log N)$. Ein naiver Ansatz für die Fourier-Rekonstruktion liefert allerdings Bilder mit nicht zu akzeptierenden Artefakten, so dass diese Algorithmen in der Praxis nutzlos sind. Eine bessere Qualität der rekonstruierten Bilder kann mit Hilfe von Linogramm-Algorithmen [33, 113], mit Gridding-Algorithmen [91, 111], mit UFR-Algorithmen [71, 72] oder mit Algorithmen von K. Fourmont [44], J. Walden [126] oder D. Gottlieb u.a. [50] erreicht werden.

Im Folgenden schlagen wir zwei neue Fourier-Rekonstruktions-Algorithmen vor, die gleiche oder bessere Rekonstruktionsergebnisse als die gefilterte Rückprojektion liefern. Diese Verfahren basieren auf den NFFT-Algorithmen (Algorithmus 1.1 und Algorithmus 1.3) und benötigen nur $\mathcal{O}(N^2 \log N)$ arithmetische Operationen. Die Algorithmen entwickeln wir für zwei verschiedene Abtastgeometrien im Fourier-Bereich des Bildes, für ein Polar-Gitter und für ein Linogramm-Gitter. Der erste Algorithmus verwendet eine bivariate NFFT auf dem Polar-Gitter und der zweite Algorithmus benutzt mehrere univariate NFFT auf dem Linogramm-Gitter.

Einige Teile dieses Kapitels wurden bereits in [98, 99] veröffentlicht.

Dieses Kapitel ist wie folgt gegliedert:
Im Abschnitt 3.1 werden die beiden neuen Fourier-Rekonstruktions-Algorithmen vorgestellt und im Abschnitt 3.2 präsentieren wir numerische Ergebnisse. Es wird gezeigt, dass die neuen Algorithmen zu guten Rekonstruktionsergebnissen führen. Wir belegen dies

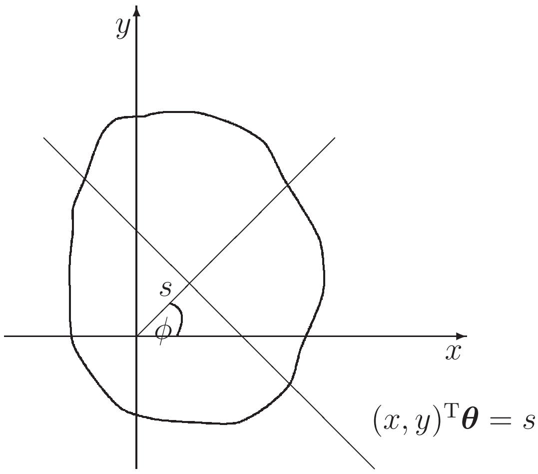

Abbildung 3.1: Röntgenstrahl in der Standard-Parallelgeometrie.

auch mit Daten vom Computer-Tomographen „Somatom Plus 4“. Das Kapitel schließt mit einigen weiteren Bemerkungen und Literaturhinweisen.

3.1 Anwendung der NFFT-Algorithmen in der Computer-Tomographie

In diesem Abschnitt präsentieren wir zwei neue Fourier-Rekonstruktions-Algorithmen, wobei wir uns auf die Standard-Parallel-Geometrie beschränken.
Die Radon-Transformation ordnet einer Funktion aus dem Schwartz-Raum $S(\mathbb{R}^d)$ die Menge ihrer Integrale über Hyperebenen des Raumes $\mathbb{R}^d$ zu. Wir betrachten im weiteren Verlauf nur den zweidimensionalen Fall $d = 2$. Die Radon-Transformation beschreibt dann für festes $\boldsymbol{\theta} := (\cos\phi, \sin\phi)^{\mathrm{T}}$ und s (siehe Abbildung 3.1) das Linienintegral von f entlang der Geraden, die senkrecht zu $\boldsymbol{\theta}$ mit Abstand s zum Ursprung verläuft.
Die Radon-Transformation ist für $f \in S(\mathbb{R}^2)$ durch

$$\mathcal{R}f(s,\phi) := \int\limits_{\{\boldsymbol{x}:\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\theta}=s\}} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \qquad (\boldsymbol{\theta} := (\cos\phi, \sin\phi)^T)$$

definiert. Da $f \in S(\mathbb{R}^2) \subset L_1(\mathbb{R}^2)$ ist, existiert die Fourier-Transformierte von f. Wir sind an der Inversion der Radon-Transformation, basierend auf dem *Projektionssatz* (Fourier-Slice-Theorem)

$$\hat{f}(\varsigma\boldsymbol{\theta}) = \int\limits_{-\infty}^{\infty} \mathcal{R}f(s,\phi)\mathrm{e}^{-2\pi\mathrm{i}s\varsigma}\,\mathrm{d}s = \hat{\mathcal{R}f}(\varsigma,\phi)\,, \tag{3.1}$$

interessiert. Dieser Satz stellt einen Zusammenhang zwischen der Radon- und der Fourier-Transformation für Funktionen aus dem Schwartz-Raum $S(\mathbb{R}^2)$ her. Hierbei ist auf der linken Seite die zweidimensionale Fourier-Transformation, auf der rechten Seite hingegen die eindimensionale Fourier-Transformation nach dem ersten Parameter gemeint. Für einen Beweis des Satzes siehe z.B. [85, Theorem 1.1]. Wir setzen voraus, dass für den Träger von f gilt $\operatorname{supp} f \subseteq \Omega := \{\boldsymbol{x} \in \mathbb{R}^2 : ||\boldsymbol{x}||_2 \leq 1\}$. Die Funktion f wollen wir auf dem Gitter

$$(x_j, y_k) := \left(j\frac{2}{N}, k\frac{2}{N}\right) \qquad (j,k \in I_N^1) \tag{3.2}$$

rekonstruieren. Dazu sei die Radon-Transformierte $\mathcal{R}f$ auf dem Gitter

$$(s_r, \phi_t) := \left(r\frac{2}{R}, t\frac{\pi}{T}\right) \qquad (r = -\frac{R}{2}, \dots, \frac{R}{2} - 1;\ t = 0, \dots, T-1)$$

gegeben, wobei auf Grund des Shannon'schen Abtastsatzes [69] die Ungleichungen $R \geq N$ und $T \geq \frac{\pi R}{2}$ gelten sollen.
Die Standard-Fourier-Rekonstruktionsmethode folgt direkt aus der Diskretisierung des Fourier-Slice-Theorems (3.1) und besteht aus den folgenden drei Schritten:

Algorithmus 3.1 (Fourier-Rekonstruktion)
Eingabe: $T, R, N \in \mathbb{N}$, $\mathcal{R}f(r\frac{2}{R}, t\frac{\pi}{T})$ $(r = -\frac{R}{2}, \dots, \frac{R}{2} - 1; t = 0, \dots, T-1)$.

1. *Berechne für alle* $t = 0, \dots, T-1$ *die Werte* $(m = -\frac{R\gamma}{4}, \dots, \frac{R\gamma}{4} - 1)$

$$\hat{f}(\frac{m}{\gamma}\boldsymbol{\theta}_t) = \hat{\mathcal{R}f}(\frac{m}{\gamma}, t\frac{\pi}{T}) := \frac{2}{R}\sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} \mathcal{R}f(r\frac{2}{R}, t\frac{\pi}{T})\mathrm{e}^{-2\pi \mathrm{i} rm/(\frac{R\gamma}{2})}$$

   mittels T *univariaten FFT der Länge* $\frac{R\gamma}{2}$. *Hier ist* $\frac{\gamma}{2} \geq 1$ *ein Oversamplingfaktor und* $\boldsymbol{\theta}_t := (\cos(t\frac{\pi}{T}), \sin(t\frac{\pi}{T}))^{\mathrm{T}}$.

2. *Interpoliere vom Polar-Gitter zum kartesischen Gitter.*

3. *Berechne* $f(x_j, y_k)$ $(j,k \in I_N^1)$ *mit einer bivariaten FFT der Länge* $\frac{\gamma}{2}N$.

Ausgabe: $f(x_j, y_k)$ *für* $j,k \in I_N^1$.
Komplexität: $\mathcal{O}(TR\log R + N^2 \log N)$.

Der obige Algorithmus ist für die Praxis unbrauchbar, da er viele Artefakte produziert. In [84] hat F. Natterer bewiesen, dass diese Artefakte durch eine Interpolation in radialer Richtung hervorgerufen werden. Dies führte zu einer Reihe von weiteren Untersuchungen und neuen Algorithmen (siehe auch [85, 86]).

- *Algorithmus von Fourmont* [44, 86, 126]
  Durch Anwendung von T univariaten NFFT der Länge N in Schritt 1 benötigt der Algorithmus in Schritt 2 nur eine lineare Interpolation in Winkelrichtung.

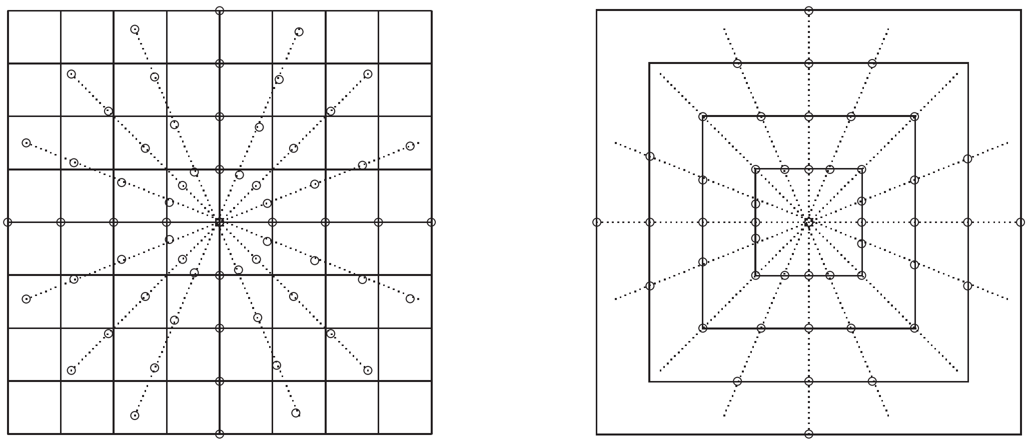

Abbildung 3.2: Polar- und kartesisches Gitter von Algorithmus 3.2 (links), Linogramm von Algorithmus 3.3 (rechts).

- *Gridding-Algorithmus / UFR-Algorithmus* [91, 111, 71, 72, 113, 86]
  Dabei wird Schritt 1 von Algorithmus 3.1 mit einem Oversamplingfaktor berechnet, d.h. $\frac{\gamma}{2} > 1$, z.B. $\frac{\gamma}{2} = 4$ [113]. An Stelle von Schritt 2 tritt ein Gridding-Schritt auf. Dieser approximiert die Werte $\hat{f}$ auf einem kartesischen Gitter. Schritt 3 stimmt dann wieder mit dem entsprechenden Schritt von Algorithmus 3.1 überein.

- *Linogramm-Algorithmen* [33, 113]
  Mittels T Chirp-z Transformationen der Länge $\frac{R\gamma}{2}$ in Schritt 1 benötigt der Algorithmus nur lineare Interpolation in x- bzw. in y-Richtung, d.h., Interpolationen „fast nur“ in Winkelrichtung. Eine andere Version eines Linogramm-Algorithmus [33] berechnet lineare Interpolationen im Radon-Bereich (Sinogramm). Damit kann in Schritt 2 die lineare Interpolation im Fourier-Bereich vermieden werden und statt dessen eine Chirp-z Transformation in Schritt 3 benutzt werden.

Im Folgenden werden zwei Fourier-Rekonstruktions-Algorithmen vorgeschlagen, welche *die lineare Interpolation vollständig* vermeiden. Beide Algorithmen basieren auf der NFFT und haben eine arithmetische Komplexität von $\mathcal{O}(N^2 \log N)$. Wir geben den Algorithmus zunächst an und erklären anschließend die Details.
Der erste Algorithmus benutzt eine bivariate NFFT$^{\mathrm{T}}$. Ein Algorithmus, der ähnlich zu Algorithmus 3.2 ist, wird auch in [45] vorgeschlagen.

Algorithmus 3.2 (Fourier-Rekonstruktion (Polar-Gitter))
Eingabe: $T, R, N \in \mathbb{N}$, $\mathcal{R}f(r\frac{2}{R}, t\frac{\pi}{T})$ $(r = -\frac{R}{2}, \ldots, \frac{R}{2} - 1; t = 0, \ldots, T-1)$.

1. *Berechne für alle* $t = 0, \ldots, T-1$ *die Werte*
$$\hat{f}(\frac{m}{\gamma}\boldsymbol{\theta}_t) = \hat{\mathcal{R}}f(\frac{m}{\gamma}, t\frac{\pi}{T}) \quad (m = -\frac{R\gamma}{4}, \ldots, \frac{R\gamma}{4} - 1)$$
mittels T *univariaten FFT der Länge* $\frac{R\gamma}{2}$ $(\frac{\gamma}{2} \geq 1)$.

2. *Berechne für $j,k \in I_N^1$ die Werte*

$$f(x_j,y_k) := \frac{\pi}{\gamma^2 T} \sum_{m=0}^{\frac{R\gamma}{4}-1} \sum_{t=-T}^{T-1} \nu_m \, \hat{f}(\frac{m}{\gamma}\boldsymbol{\theta}_t) \, \mathrm{e}^{2\pi\mathrm{i}(jm\cos(t\frac{\pi}{T})+km\sin(t\frac{\pi}{T}))/\left(\frac{\gamma N}{2}\right)}$$

mit einer bivariaten $NFFT^{\mathrm{T}}$, wobei ν_m durch

$$\nu_m := \begin{cases} \frac{1}{12} & m = 0\,, \\ m & m = 1,\ldots,\frac{R\gamma}{4}-1 \end{cases} \tag{3.3}$$

gegeben ist.

Ausgabe: $f(x_j,y_k)$ *für* $(j,k \in I_N^1)$.
Komplexität: $\mathcal{O}(TR\log R + N^2 \log N)$.

Der erste Schritt des Algorithmus stimmt mit dem ersten Schritt von Algorithmus 3.1 überein. Man beachte, dass $\frac{\gamma}{2}$ wieder als Oversamplingfaktor angesehen werden kann. Für praktische Anwendungen ist es oft ausreichend, $\frac{\gamma}{2} = 1$ zu wählen.

Der zweite Algorithmus basiert auf der Linogramm-Geometrie. Wir kombinieren dazu T univariate NFFT und $\frac{\gamma N}{2}$ univariate $\mathrm{NFFT}^{\mathrm{T}}$.
Im Folgenden setzen wir voraus, dass T durch 4 teilbar ist.

Algorithmus 3.3 (Fourier-Rekonstruktion (Linogramm-Gitter))
Eingabe: $T,R,N \in \mathbb{N}$, $\mathcal{R}f(r\frac{2}{R},\phi_t)$ $(r = -\frac{R}{2},\ldots,\frac{R}{2}-1; t = 0,\ldots,T-1)$ *mit* $\phi_t := t\frac{\pi}{T}$.

1. *Berechne für alle $t = 0,\ldots,\frac{T}{4},\frac{3T}{4},\ldots,T-1$ die Werte*

$$\hat{f}(\frac{m}{\gamma}\frac{1}{\cos\phi_t}\boldsymbol{\theta}_t) = \hat{\mathcal{R}f}(\frac{m}{\gamma}\frac{1}{\cos\phi_t},\phi_t) := \frac{2}{R}\sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} \mathcal{R}f(r\frac{2}{R},\phi_t)\mathrm{e}^{-2\pi\mathrm{i}rm\frac{1}{\cos\phi_t}/(\frac{R\gamma}{2})}$$

für $m = \lceil -\frac{R\gamma}{4}\cos\phi_t\rceil,\ldots,\lceil\frac{R\gamma}{4}\cos\phi_t\rceil - 1$ mittels NFFT.
Berechne für alle $t = \frac{T}{4}+1,\ldots,\frac{3T}{4}-1$ die Werte

$$\hat{f}(\frac{m}{\gamma}\frac{1}{\sin\phi_t}\boldsymbol{\theta}_t) = \hat{\mathcal{R}f}(\frac{m}{\gamma}\frac{1}{\sin\phi_t},\phi_t) := \frac{2}{R}\sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} \mathcal{R}f(r\frac{2}{R},\phi_t)\mathrm{e}^{-2\pi\mathrm{i}rm\frac{1}{\sin\phi_t}/(\frac{R\gamma}{2})}$$

für $m = \lceil -\frac{R\gamma}{4}\sin\phi_t\rceil,\ldots,\lceil\frac{R\gamma}{4}\sin\phi_t\rceil - 1$ mittels NFFT. Die anderen Werte $\hat{f}(\frac{m}{\gamma}\frac{1}{\cos\phi_t}\boldsymbol{\theta}_t)$ und $\hat{f}(\frac{m}{\gamma}\frac{1}{\sin\phi_t}\boldsymbol{\theta}_t)$ $(m \in [-\frac{R\gamma}{4},\ldots,\frac{R\gamma}{4}-1]; m \in \mathbb{Z})$ seien Null.

2. *Berechne*

$$f_1(x_j, y_k) = \frac{\pi}{\gamma^2 T} \sum_{m=-\frac{R\gamma}{4}}^{\frac{R\gamma}{4}-1} \nu_m \sum_{t=-\frac{T}{4}}^{\frac{T}{4}-1} \frac{1}{\cos^2 \phi_t} \hat{f}(\frac{m}{\gamma}, \frac{m}{\gamma}\frac{\sin \phi_t}{\cos \phi_t}) \mathrm{e}^{2\pi\mathrm{i}\frac{\sin\phi_t}{\cos\phi_t} mk/(\frac{N\gamma}{2})} \mathrm{e}^{2\pi\mathrm{i} jm/(\frac{N\gamma}{2})}$$

$$f_2(x_j, y_k) = \frac{\pi}{\gamma^2 T} \sum_{m=-\frac{R\gamma}{4}}^{\frac{R\gamma}{4}-1} \nu_m \sum_{t=-\frac{T}{4}+1}^{\frac{T}{4}} \frac{1}{\cos^2 \phi_t} \hat{f}(\frac{m}{\gamma}\frac{\sin \phi_t}{\cos \phi_t}, \frac{m}{\gamma}) \mathrm{e}^{2\pi\mathrm{i}\frac{\sin\phi_t}{\cos\phi_t} mj/(\frac{N\gamma}{2})} \mathrm{e}^{2\pi\mathrm{i} km/(\frac{N\gamma}{2})}$$

($j, k \in I_N^1$) mittels $\frac{R\gamma}{2}$ univariaten NFFT$^{\mathrm{T}}$ der Länge $\frac{N\gamma}{2}$ für die inneren Summen und N univariaten FFT der Länge $\frac{N\gamma}{2}$ für die äußeren Summen. Setze dann

$$f(x_j, y_k) = \frac{1}{2} \mathrm{Re}(f_1(x_j, y_k) + f_2(x_j, y_k)) .$$

Ausgabe: $f(x_j, y_k)$ *für* ($j, k \in I_N^1$).

Komplexität: $\mathcal{O}(TR \log R + N^2 \log N)$.

Bei beiden Algorithmen wird die Tatsache

$$\hat{f}(-\varsigma\boldsymbol{\theta}) = \overline{\hat{f}(\varsigma\boldsymbol{\theta})} \tag{3.4}$$

ausgenutzt. Basierend auf (3.4) sind Vereinfachungen in unserer Implementierung möglich. Weiterhin führen wir standardmäßig einen „Filter-Schritt“ im Fourier-Bereich nach Schritt 1 durch. Im Folgenden geben wir einige Erläuterungen zu den Algorithmen.

Erster Schritt

Für beliebige feste Winkel $\phi_t := t\frac{\pi}{T}$, $(t = 0, \ldots, T-1)$ sei $h(s) := \mathcal{R}f(s, \phi_t)$ und $\hat{h}(\varsigma) := \hat{\mathcal{R}f}(\varsigma, \phi_t)$. Im ersten Schritt von Algorithmus 3.2 und 3.3 wird das Integral

$$\hat{h}(\varsigma) = \int_{-1}^{1} h(s)\mathrm{e}^{-2\pi\mathrm{i}s\varsigma}\,\mathrm{d}s$$

diskretisiert. Nach der Poissonschen Summenformel erhalten wir

$$\hat{h}(\varsigma) + \sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \hat{h}(\varsigma + n\frac{R}{2}) = \frac{2}{R} \sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} h(r\frac{R}{2})\mathrm{e}^{-2\pi\mathrm{i}r\varsigma/(\frac{R}{2})} . \tag{3.5}$$

Da nur Details von f mit Absolutbetrag größer $\frac{2}{N}$ und $R \geq N$ rekonstruiert werden sollen, können wir nach dem Abtastsatz von Shannon die Werte $\hat{h}(\varsigma)$ für $|\varsigma| > \frac{R}{4}$ vernachlässigen. Damit ist die rechte Seite von (3.5) eine gute Approximation an $\hat{h}(\varsigma)$ für

$\varsigma \in [-\frac{R}{4}, \frac{R}{4}]$.

Zweiter Schritt

Im zweiten Schritt von Algorithmus 3.2 und Algorithmus 3.3 berechnen wir eine diskretisierte Form des Integrals

$$\begin{aligned} f(x,y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{f}(u,v)\mathrm{e}^{2\pi\mathrm{i}(xu+yv)}\,\mathrm{d}u\mathrm{d}v \\ &= \int_{0}^{\infty} \varsigma \int_{-\pi}^{\pi} \hat{f}(\varsigma\cos\phi, \varsigma\sin\phi)\mathrm{e}^{2\pi\mathrm{i}\varsigma(x\cos\phi+y\sin\phi)}\,\mathrm{d}\phi\mathrm{d}\varsigma. \end{aligned} \tag{3.6}$$

Da das innere Integral, betrachtet als Funktion von ς, gerade ist, kann obige Formel als

$$f(x,y) = \frac{1}{2}\int_{-\pi}^{\pi}\int_{-\infty}^{\infty} |\varsigma|\hat{f}(\varsigma\cos\phi, \varsigma\sin\phi)\mathrm{e}^{2\pi\mathrm{i}\varsigma(x\cos\phi+y\sin\phi)}\,\mathrm{d}\varsigma\,\mathrm{d}\phi \tag{3.7}$$

umgeschrieben werden. Wir diskretisieren zunächst das innere Integral. Für beliebig, aber fest gewählte $(x,y) \in [-1,1]^2$ und $\phi \in [-\pi, \pi]$ setzen wir

$$\begin{aligned} g(\varsigma) &:= |\varsigma|\hat{f}(\varsigma\cos\phi, \varsigma\sin\phi)\mathrm{e}^{2\pi\mathrm{i}\varsigma(x\cos\phi+y\sin\phi)}, \\ \hat{g}(v) &:= \int_{-\infty}^{\infty} g(\varsigma)\mathrm{e}^{-2\pi\mathrm{i}v\varsigma}\,\mathrm{d}\varsigma. \end{aligned}$$

Wir erhalten dann mit Hilfe der Poissonschen Summenformel für die Diskretisierung auf dem Polar-Gitter

$$\hat{g}(0) + \sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \hat{g}(\gamma n) = \frac{1}{\gamma}\sum_{m\in\mathbb{Z}} g\left(\frac{m}{\gamma}\right),$$

und für das Linogramm-Gitter

$$\hat{g}(0) + \sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \hat{g}(\gamma n\cos\phi) = \frac{1}{\gamma\cos\phi}\sum_{m\in\mathbb{Z}} g\left(\frac{m}{\gamma\cos\phi}\right) \qquad (\phi \in [-\frac{\pi}{4}, \frac{\pi}{4}]).$$

Da wir $\hat{f}(\varsigma\cos\phi, \varsigma\sin\phi)$ für $|\varsigma| > \frac{R}{4}$ wieder vernachlässigen können, erhalten wir

$$\hat{g}(0) + \sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \hat{g}(\gamma n) \approx \frac{1}{\gamma}\sum_{m=-\frac{R\gamma}{4}}^{\frac{R\gamma}{4}-1} g\left(\frac{m}{\gamma}\right) \tag{3.8}$$

bzw.

$$\hat{g}(0) + \sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \hat{g}(\gamma n\cos\phi) \approx \frac{1}{\gamma\cos\phi}\sum_{m=\lceil -\frac{R\gamma}{4}\cos\phi\rceil}^{\lceil\frac{R\gamma}{4}\cos\phi\rceil-1} g\left(\frac{m}{\phi\cos\phi}\right) \qquad (\phi \in [-\frac{\pi}{4}, \frac{\pi}{4}]). \tag{3.9}$$

Der Aliasing-Fehler der sich duch das Näherungszeichen in den den Gleichungen (3.8) und (3.9) ausdrückt, wird kleiner für wachsenden Oversamplingfaktor γ.
Weiterhin ist $\cos\phi \geq \frac{\sqrt{2}}{2}$ ($\phi \in [-\frac{\pi}{4}, \frac{\pi}{4}]$). Falls wir $\gamma = \gamma_p$ in (3.8) wählen wollen, sollten wir $\gamma = \sqrt{2}\gamma_p$ in (3.9) setzen, um einen vergleichbaren Aliasing-Fehler zu erhalten. Unter der Voraussetzung, dass $\tilde{g}(\varsigma) := \varsigma \hat{f}(\varsigma\cos\phi, \varsigma\sin\phi)\mathrm{e}^{2\pi\mathrm{i}\varsigma(x\cos\phi + y\sin\phi)}$ eine „quasi“ Bandweite (essentially band-limited) $\ll \gamma$ bzw. $\ll \gamma\cos\phi$ hat, kann der Aliasing-Fehler wie in [86] durch

$$\sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \hat{g}(\gamma n) \approx \frac{1}{6\gamma^2}\hat{f}(0,0), \tag{3.10}$$

bzw.

$$\sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \hat{g}(\gamma n\cos\phi) \approx \frac{1}{6\gamma^2\cos^2\phi}\hat{f}(0,0) \tag{3.11}$$

abgeschätzt werden. Da wir nur Details von f mit Absolutbetrag größer $\frac{2}{N}$ rekonstruieren wollen, diskretisieren wir das äußere Integral in (3.7) mit einem kleineren Aliasing-Fehler durch die Trapez-Regel an den Knoten $\phi_t = t\frac{\pi}{T}$ $(t = -T, \ldots, T-1)$, falls $T \geq \pi\frac{R}{2}$. Damit erhalten wir Schritt 2 von Algorithmus 3.2 und 3.3. Insbesondere (3.10) und (3.11) erklären den Koeffizienten vor $\hat{f}(0,0)$ im Schritt 2 unserer Algorithmen. Man beachte, dass der Koeffizient $\frac{2\pi}{12\gamma^2}$ vor $\hat{f}(0,0)$ in Algorithmus 3.2 etwa die Fläche eines Kreises mit dem Radius $\frac{1}{\sqrt{6}\gamma}$ ist. Der Koeffizient vor $\hat{f}(0,0)$ in Algorithmus 3.3 ist etwa $\frac{1}{\gamma^2}$, also die Fläche von einem Quadrat der Seitenlänge $\frac{1}{\gamma}$.

Bemerkung 3.4 Wir haben in unseren Algorithmen die Formel (3.6) diskretisiert mit dem Ziel, die gesuchte Funktion f auf dem Gitter (3.2) zu berechnen. Eine andere Möglichkeit besteht in der Diskretisierung der Formel

$$\hat{f}(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\mathrm{e}^{-2\pi\mathrm{i}(ux+vy)}\,\mathrm{d}x\mathrm{d}y\,.$$

Dann können die Daten $\hat{f}$ im Fourier-Bereich aus der Formel (3.5) bzw. Schritt 1 von Algorithmus 3.2 berechnet werden. Um die Funktion f auf einem Gitter zu rekonstruieren, müssen wir ein lineares Gleichungssystem der Form

$$\hat{\boldsymbol{f}} = \boldsymbol{A}\boldsymbol{f}$$

mit einer Matrix $\boldsymbol{A}$ der bivariaten NFFT und gegebenen Daten $\hat{\boldsymbol{f}}$ lösen. Dieses Problem wurde in Abschnitt 1.7 besprochen. Unsere Rekonstruktion, d.h. Schritt 2 von Algorithmus 3.2, entspricht der Matrix-Vektor-Multiplikation von $\hat{\boldsymbol{f}}$ mit $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{W}$, wobei $\boldsymbol{W}$ die Matrix mit den entsprechenden Quadraturgewichten (3.3) ist. In unserem Fall gilt

$$\boldsymbol{A}^{\mathrm{H}}\boldsymbol{W}\hat{\boldsymbol{f}} = \boldsymbol{A}^{\mathrm{H}}\boldsymbol{W}\boldsymbol{A}\boldsymbol{f} \approx \boldsymbol{f}.$$

Fügen wir zusätzlich zur iterativen Rekonstruktion die Matrix $\boldsymbol{W}$ ein, so ist unser Schritt 2 von Algorithmus 3.2 der Initialisierungsschritt im CGNR-Verfahren. Diese iterative Rekonstruktion ist besonders im Zusammenhang mit MRT-Daten interessant [18]. □

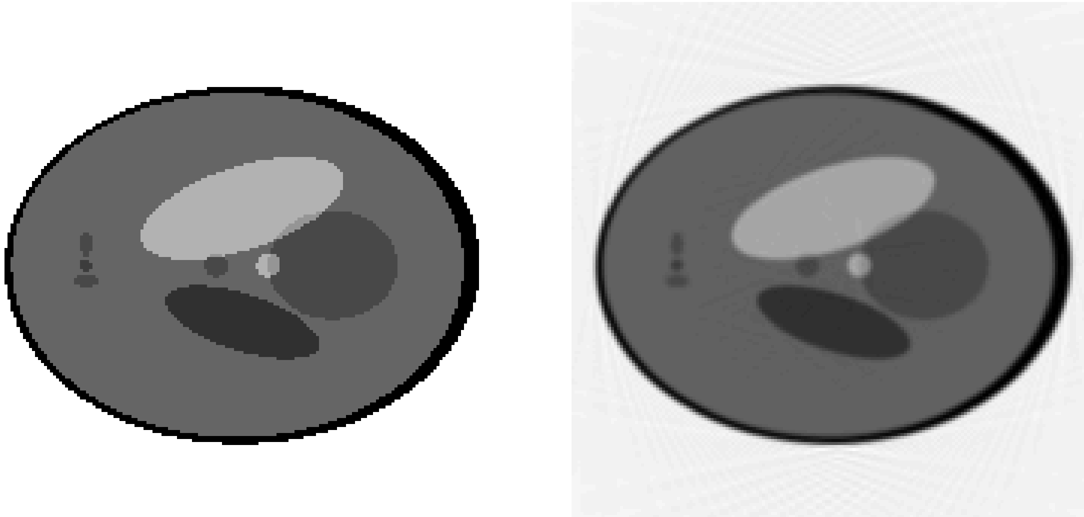

Abbildung 3.3: Shepp-Logan-Phantom, Original (links), rekonstruiertes Bild mittels gefilterter Rückprojektion (rechts).

3.2 Numerische Ergebnisse

Ein oft benutztes Modell in der Computer-Tomographie ist das Shepp-Logan-Phantom eines Gehirns. Dieses Modell besteht aus verschiedenen Ellipsen, so dass die Radon-Transformierte analytisch berechnet werden kann. Um diskrete Werte für das Phantom und deren Radon-Transformierte zu erhalten, verwenden wir das Softwarepaket „RadonAna“ [124]. Das Originalbild (siehe Abbildung 3.3 (links)) ist von der Größe $N \times N = 170 \times 170$ und das Sinogramm ist von der Größe $R \times T = 170 \times 600$. Abbildung 3.3 (rechts) zeigt das rekonstruierte Bild, welches wir mit Hilfe der gefilterten Rückprojektion (FB) erhalten haben. Dazu wurde das Softwarepaket „iradon“ [124] benutzt.

Beispiel 3.1 (Rekonstruktionsqualität bekannter Algorithmen)
Die rekonstruierten Bilder in Abbildung 3.4 wurden durch Anwenden von Algorithmus 3.1 (Abbildung 3.4 (links)) und durch Anwendung des Linogramm-Algorithmus (NFFTL, wir benutzen Algorithmus 1.1) mit linearer Interpolation in x- und y-Richtung [113] (Abbildung 3.4 (rechts)) erstellt. Als Filter haben wir sinc^3 im Fourier-Bereich gewählt. Der Algorithmus ist etwa 14 mal schneller als die der gefilterten Rückprojektion. Allerdings ist die Bildqualität deutlich schlechter. □

Beispiel 3.2 (Rekonstruktionsqualität der neuen Algorithmen)
Diese numerischen Beispiele zeigen, dass unsere Algorithmen 3.2 und 3.3 qualitativ gute Bilder ähnlich zur gefilterten Rückprojektion liefern. In beiden Algorithmen (wie auch in den obigen drei beschriebenen Algorithmen) haben wir mit einem Oversamplingfaktor $\frac{\gamma}{2} = \frac{256}{170}$ ($\approx \sqrt{2}$) gearbeitet. Die Algorithmen wurden in C auf einer SGI O2 getestet.

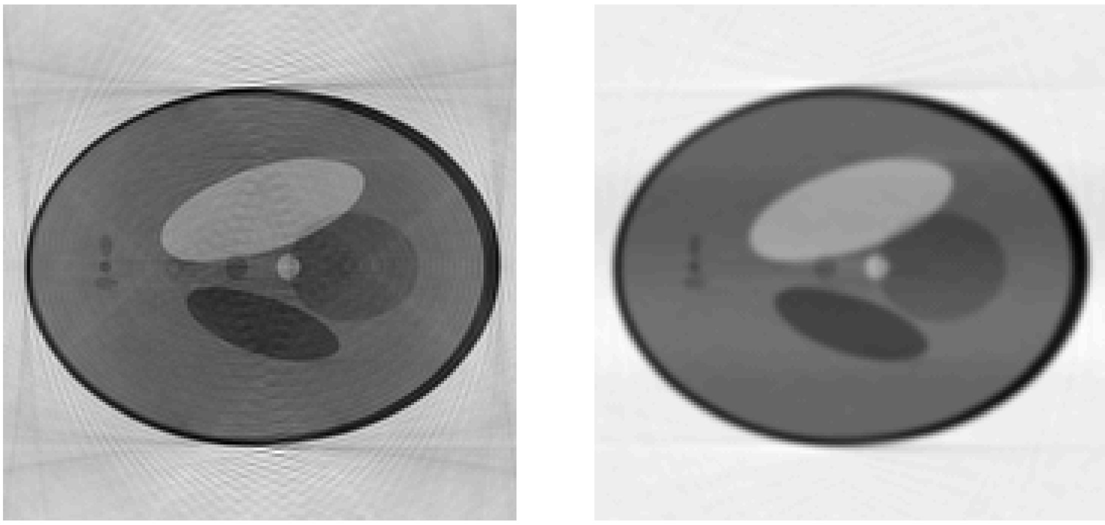

Abbildung 3.4: Rekonstruiertes Bild mittels Algorithmus 3.1 (links) und mittels Linogramm-Algorithmus (rechts).

Wir haben Schritt 2 von Algorithmus 3.2 durch Anwendung von Algorithmus 1.3 mit $d = 2$, Oversamplingfaktor $\sigma = 2$, $m = 3$ und $b = \frac{2\sigma m}{(2\sigma-1)\pi} = \frac{4}{\pi}$ getestet. Weiterhin wurde für die Ansatzfunktion φ im Algorithmus 1.1 ein Tensorprodukt von Gauß-Funktionen (siehe Abschnitt 1.3.2) gewählt.
Schritt 1 von Algorithmus 3.3 nutzt Algorithmus 1.1 mit $d = 1$, Oversamplingfaktor $\sigma = 2$, $m = 5$ und $b = \frac{20}{3\pi}$. Als Ansatz-Funktion φ haben wir, wie oben, die Gauß-Funktionen gewählt. Im 2. Schritt von Algorithmus 3.3 verwenden wir wieder Algorithmus 1.3 mit den gleichen Parametern wie in Schritt 1. Im Fourier-Bereich wird ein sinc-Filter benutzt. Abbildung 3.5 zeigt die rekonstruierten Bilder, welche wir durch Anwendung von Algorithmus 3.3 und Algorithmus 3.2 erhalten haben. □

Beispiel 3.3 (CPU-Zeiten)
In diesem Beispiel vergleichen wir CPU-Zeiten. Es sei aber bemerkt, dass die NFFT noch nicht auf dem effizienteren Programmpaket [76] beruhen. Weil die bivariate NFFT etwas teurer ist, benötigt der Algorithmus 3.3 etwa dreimal soviel Zeit wie Algorithmus 3.2, welcher etwa die gleiche Zeit wie der Linogramm-Algorithmus (NFFTL) benötigt. In Tabelle 3.6 geben wir die Rechenzeiten für verschiedene Rekonstruktions-Algorithmen an. □

Man beachte, dass wir die Unterschiede in der Qualität der rekonstruierten Bilder viel besser an Farbbildern ablesen können. Aus diesem Grund verweisen wir auf

`http://www.math.uni-luebeck.de/potts/radon/spie.html`.

Weitere Vergleiche sind in der Arbeit [99] veröffentlicht, in der insbesondere verschiedene Linogramm-Algorithmen analysiert wurden.

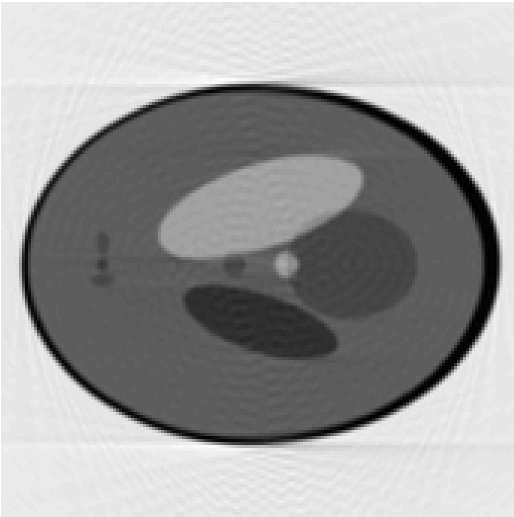


Abbildung 3.5: Rekonstruiertes Bild mittels Algorithmus 3.2 (links) und mittels Algorithmus 3.3 (rechts).

	R	T	N	Zeit in s	% FFT
FB	180	600	180	20.2	11.7
NFFTL	180	600	180	2.08	43.6
Algorithmus 3.3	180	600	180	3.5	32.6
FB	362	900	362	127.81	3.49
NFFTL	362	900	362	8.44	45.6
Algorithmus 3.3	362	900	362	10.59	36.3

Tabelle 3.6: Rechenzeiten für verschiedene Rekonstruktions-Algorithmen.

Beispiel 3.4 („Somatom Plus 4“– Daten)
Die folgenden Abbildungen 3.7 - 3.8 zeigen Ergebnisse, die mittels Algorithmus 3.2 nach einem Rebinning [86, S. 42f] der Daten, rekonstruiert wurden. Das Sinogramm stammt aus dem DKFZ Heidelberg, Abt. Medizinische Physik. In dankenswerter Weise hat M. Ebert uns diese Daten zur Verfügung gestellt. Sie wurden mit einem „Somatom Plus 4“-Gerät aufgenommen und zeigen einen Brustkorb im Spiral-Scan. Weitere rekonstruierte Bilder haben wir auf der Internetseite

`http://www.math.uni-luebeck.de/potts/projekte/TomBiFou.sql`

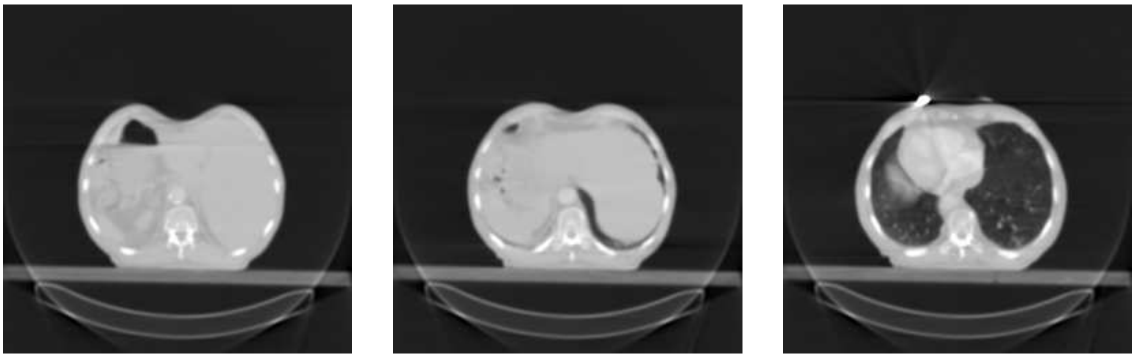

Abbildung 3.7: Rekonstruierte Bilder mittels Algorithmus 3.2 mit Daten des Computer-Tomographen „Somatom Plus 4“.

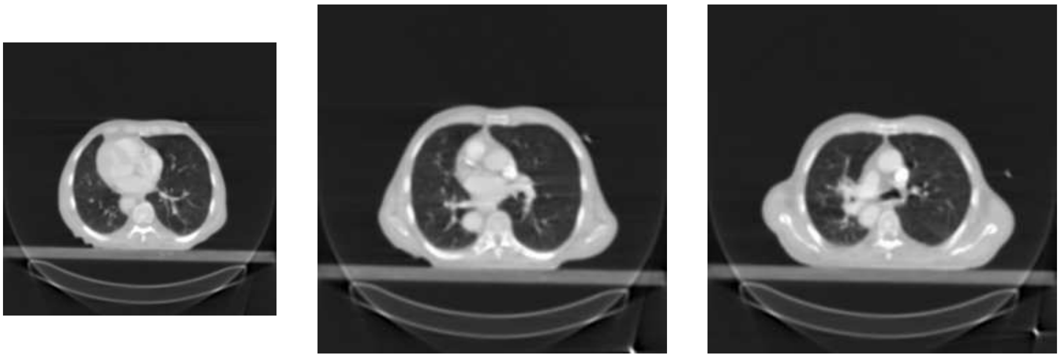

Abbildung 3.8: Rekonstruierte Bilder mittels Algorithmus 3.2 mit Daten des Computer-Tomographen „Somatom Plus 4“.

veröffentlicht. □

Mit den Beispielen 3.1 - 3.4 wurde gezeigt, dass die neuen vorgeschlagenen Algorithmen sehr gute Rekonstruktionsergebnisse liefern und eine geringere arithmetische Komplexität als die gefilterte Rückprojektion aufweisen.

Weitere Themen und Literaturhinweise

Rekonstruktions-Algorithmen von Nicht-Standard-Abtastungen im Radon-Bereich

Die Fourier-Rekonstruktions-Algorithmen haben wir in [100] erstmalig auch für Interlaced-Daten entwickelt. Dabei handelt es sich um ein Abtastschema, bei dem, gegenüber der Standard-Abtastung, nur die Hälfte der Daten gemessen werden. A. Cormack hat im Jahre 1978 durch eine heuristische Überlegung diese Abtastschemata gefunden. Sie können aber durch einen verallgemeinerten Abtastsatz auch mathematisch begründet werden. Für diese Nicht-Standard-Abtastungen wurden bereits ein gefilterter Rückprojektions-algorithmus von H. Kruse [74] und ein algebraischer Rekonstruktionsalgorithmus von W. Klaverkamp [73] angegeben.

Rekonstruktionsalgorithmen ohne Rebinning

Ein noch offenes Problem ist die Entwicklung von schnellen Fourier-Rekonstruktions-Algorithmen ohne Rebinning. Diese Rekonstruktion von sogenannten Fan-Beam-Projektionen wurde z.B. für die gefilterte Rückprojektion in [70] veröffentlicht. Eine Verallgemeinerung der obigen Algorithmen auf den 3D-Fall eröffnet viele theoretisch und praktisch interessante Fragestellungen.

Weitere Rekonstruktionsalgorithmen

In diesem Kapitel haben wir ausschließlich Fourier-Rekonstruktions-Algorithmen betrachtet. Für einen umfassenden Überblick auf aktuelle Rekonstruktionsalgorithmen verweisen wir auf das Buch von F. Natterer und F. Wübbeling [86].

4 Fourier-Algorithmen auf der Sphäre

Die Fourier-Analysis auf der Sphäre $\mathbb{S}^2 \subset \mathbb{R}^3$ hat z.B. praktische Bedeutung in der Tomographie, Geophysik, Seismologie, Meteorologie oder Kristallographie. Zur numerischen Berechnung werden oft sphärische Fourier-Summen benutzt. Diese haben ähnlich bemerkenswerte Eigenschaften wie die Fourier-Summen auf dem Torus. Viele in der numerischen Meteorologie eingesetzte Lösungsverfahren für partielle Differentialgleichungen basieren auf Spektralmethoden. Der Großteil der Rechenzeit nimmt dabei die Berechnung der sphärischen Fourier-Summen [16, S. 402ff] ein. Eine numerisch stabile, schnelle diskrete sphärische Fourier-Transformation ist deshalb von großem Interesse.

Wir sagen, eine Funktion $f \in L^2(\mathbb{S}^2)$ hat Bandbreite N, falls sich f als sphärische Fourier-Summe

$$f = \sum_{k=0}^{N} \sum_{n=-k}^{k} a_k^n Y_k^n \tag{4.1}$$

schreiben läßt. Dabei sind

$$a_k^n = a_k^n(f) := \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi) Y_k^n(\theta, \phi) \sin(\theta)\ \mathrm{d}\theta\, \mathrm{d}\phi \tag{4.2}$$

die sphärischen Fourier-Koeffizienten von f bezüglich der orthogonalen Basis der Kugelflächenfunktionen (spherical harmonics) Y_k^n (vgl. Formel (4.8)). In diesem Kapitel sei f von der Form (4.1).

Im Folgenden leiten wir einen Algorithmus für die schnelle und stabile Auswertung

$$f\left(\theta_l, \phi_l\right) = \sum_{k=0}^{N} \sum_{n=-k}^{k} a_k^n Y_k^n\left(\theta_l, \phi_l\right) \quad (l = 0, \ldots, M-1) \tag{4.3}$$

an beliebigen Knoten $(\theta_l, \phi_l) \in \mathbb{S}^2$ mit gegebenen Koeffizienten $a_k^n \in \mathbb{C}$ her. Weiterhin sind wir an der Berechnung des „transponierten“ Problems

$$\breve{a}_k^n := \sum_{l=0}^{M-1} f_l Y_k^n\left(\theta_l, \phi_l\right) \quad (k = 0, \ldots, N;\ n = -k, \ldots, k) \tag{4.4}$$

für gegebene Daten $f_l \in \mathbb{C}$ interessiert.

Falls wir einen schnellen Algorithmus für das Problem (4.3) konstruiert haben, können wir sofort einen Algorithmus für das „transponierte“ Problem (4.4) angeben. Ein schneller Algorithmus für eine Matrix-Vektor-Multiplikation lässt sich nämlich als Produkt schwach besetzter Matrizen schreiben. Eine effiziente Realisierung zur Multiplikation mit der transponierten Matrix folgt dann sofort durch Transponieren der Faktorisierung. Aus diesem Grund konzentrieren wir uns hier auf die Herleitung eines schnellen Algorithmus für das Problem (4.3). Den „transponierten“ Algorithmus zur schnellen Berechnung von

(4.4) benötigen wir z.B. für die näherungsweise Berechnung der sphärischen Fourier-Koeffizienten (4.2) der Funktion f, unter Verwendung geeigneter Quadraturformeln und Quadraturgewichten [46, 82].

Für M beliebig verteilte Punkte auf der Sphäre hat die direkte Berechnung der diskreten sphärischen Fourier-Transformation (4.3) eine Komplexität von $\mathcal{O}(N^2M)$. Nur für spezielle äquidistante Gitter, z.B. Gitter der Form

$$\left(\frac{s\pi}{T}, \frac{t\pi}{2T}\right) \in \mathbb{S}^2 \quad (s = 0, \ldots, T;\ t = 0, \ldots, 2T-1;\ T := 2^{\lceil \log_2 N \rceil}) \tag{4.5}$$

sind schnelle Realisierungen der DSFT (Discrete Spherical Fourier Transform) bekannt. Erstmals wurden schnelle Algorithmen für die diskrete sphärische Fourier-Transformation 1994 von J. Driscoll und D. Healy in [27] veröffentlicht. Diese Arbeit motivierte viele Autoren, Weiterentwicklungen und Verbesserungen zur schnellen Summation von Kugelflächenfunktionen zu suchen (siehe [103, 83, 16, 119, 63]). Schnelle Algorithmen zur Realisierung der DSFT auf Gittern bezeichnen wir als FSFT (Fast Spherical Fourier Transform). Die FSFT reduziert die arithmetische Komplexität der DSFT für spezielle Gitter (4.5) von $\mathcal{O}(N^3)$ auf $\mathcal{O}(N^2 \log^2 N)$.
Bisher sind lediglich schnelle sphärische Algorithmen für spezielle Gitter bekannt. Ziel des Kapitels ist die konsequente Weiterentwicklung dieser schnellen sphärischen Fourier-Transformationen von Gittern (FSFT) auf beliebig verteilte Daten auf der Kugeloberfläche. Diese Algorithmen werden dann mit NFSFT (Nonequispaced Fast Spherical Fourier Transform) bezeichnet. Zur besseren Orientierung geben wir die namentliche Korrespondenz zu den Algorithmen auf dem Torus an.

Torus	Sphäre
DFT	DSFT (Discrete Spherical Fourier Transform)
FFT	FSFT (Fast Spherical Fourier Transform)
NDFT	NDSFT (Nonequispaced Discrete Spherical Fourier Transform)
NFFT	NFSFT (Nonequispaced Fast Spherical Fourier Transform)

Die grundlegende Idee ist die schnelle Realisierung einer Basistransformation, so dass f in (4.1) in der Form

$$f(\theta, \phi) = \sum_{n=-N}^{N} \sum_{k=-N}^{N} c_k^n \, \mathrm{e}^{\mathrm{i}k\theta} \, \mathrm{e}^{\mathrm{i}n\phi} \tag{4.6}$$

mit komplexen Koeffizienten $c_k^n \in \mathbb{C}$ repräsentiert wird. Dann kann die schnelle Berechnung von f an den beliebigen Knoten $(\theta_l, \phi_l) \in \mathbb{S}^2$ $(l = 0, \ldots, M-1)$ mit einer bivariaten NFFT (Algorithmus 1.1) erfolgen.

In [21] nutzt H.-B. Cheong Fourier-Summen der Form (4.6) zur Approximation der Lösung von partiellen Differentialgleichungen in sphärischen Koordinaten.
Für eine gegebene bandbeschränkte Funktion auf der Sphäre gilt, dass alle Gebiete gleichmäßig gut aufgelöst werden (siehe z.B. [68, 16, S. 400]). Deshalb scheint es nicht

sinnvoll zu sein, mit Gittern (4.5) wie in der FSFT zu rechnen, die besonders viele Punkte in der Nähe der Pole erzeugen.

Teile der folgenden Ergebnisse wurden bereits in [77] veröffentlicht.

Dieses Kapitel ist wie folgt gegliedert:
Im Abschnitt 4.1 führen wir zunächst die notwendigen Bezeichnungen ein und geben den Algorithmus für die diskrete sphärische Fourier-Transformation an beliebigen Punkten an. Diese einfache, naive Implementierung wird in Algorithmus 4.1 formuliert. Im anschließenden Abschnitt 4.2 präsentieren wir einen schnellen Algorithmus für die diskrete sphärische Fourier-Transformation (FSFT) auf speziellen Gittern. In Abschnitt 4.3 werden unsere neuen approximativen Algorithmen zur schnellen Berechnung der nichtäquidistanten DSFT (siehe Algorithmus 4.4) hergeleitet. Numerische Ergebnisse beschreiben wir in Abschnitt 4.4 und schließen dieses Kapitel wieder mit einigen weiteren Bemerkungen und Literaturhinweisen.

4.1 Diskrete sphärische Fourier-Transformationen

Zunächst werden einige Bezeichnungen eingeführt. Wir beginnen mit der Definition der Legendre-Polynome

$$P_k(x) := \frac{1}{2^k k!}\frac{\mathrm{d}^k}{\mathrm{d}x^k}(x^2-1)^k \quad (x \in [-1,1]; k \in \mathbb{N}_0)$$

und führen die assoziierten Legendre-Funktionen P_k^n ($n \in \mathbb{N}_0; k = n, n+1, \dots$) durch

$$P_k^n(x) := \left(\frac{(k-n)!}{(k+n)!}\right)^{1/2} (1-x^2)^{n/2}\frac{\mathrm{d}^n}{\mathrm{d}x^n}P_k(x) \quad (x \in [-1,1]) \tag{4.7}$$

ein. Die Kugelflächenfunktionen Y_k^n sind auf der Einheitssphäre $\mathbb{S}^2$ durch

$$Y_k^n(\theta,\phi) := P_k^{|n|}(\cos\theta)\,\mathrm{e}^{\mathrm{i}n\phi} \tag{4.8}$$

definiert. Für gegebene sphärische Fourier-Koeffizienten $a_k^n \in \mathbb{C}$ ($k = 0,\dots,N; n = -k,\dots,k$) in (4.1) sind wir an der Berechnung der Funktionswerte $f(\theta_l,\phi_l)$ ($l = 0,\dots,M-1$) interessiert. Vertauschen der Summationsreihenfolge in (4.1) ergibt mit der Definition (4.8) und mit

$$h_n(x) := \sum_{k=|n|}^{M} a_k^n P_k^{|n|}(x) \quad (n = -M,\dots,M) \tag{4.9}$$

die Formel

$$\begin{aligned} f(\theta_l,\phi_l) &= \sum_{k=0}^{N}\sum_{n=-k}^{k} a_k^n Y_k^n(\theta_l,\phi_l) \\ &= \sum_{n=-N}^{N} h_n(\cos\theta_l)\,\mathrm{e}^{\mathrm{i}n\phi_l}\,. \end{aligned} \tag{4.10}$$

Die Berechnung der diskreten sphärischen Fourier-Transformation für beliebige Daten auf $\mathbb{S}^2$ fassen wir im folgenden Algorithmus zusammen und bezeichnen diesen Algorithmus als nichtäquidistante diskrete sphärische Fourier-Transformation (NDSFT).

Algorithmus 4.1 (NDSFT)

Eingabe: $N, M \in \mathbb{N}$, $a_k^n \in \mathbb{C}$ $(k = 0, \dots, N; n = -k, \dots, k)$,
$(\theta_l, \phi_l) \in \mathbb{S}^2$ $(l = 0, \dots, M-1)$.

Vorberechnung: $x_l := \cos\theta_l$ $(l = 0, \dots, M-1)$.

1. *Berechne für alle* $n = -N, \dots, N$ *jeweils mittels Clenshaw-Algorithmus (siehe* [24]*) die Daten*
$$h_n(x_l) = \sum_{k=|n|}^{N} a_k^n P_k^{|n|}(x_l) \quad (l = 0, \dots, M-1).$$
2. *Berechne für alle* $l = 0, \dots, M-1$ *die Daten*
$$f(\theta_l, \phi_l) = \sum_{n=-N}^{N} h_n(\cos\theta_l)\, \mathrm{e}^{\mathrm{i}n\phi_l}\,.$$

Ausgabe: $f(\theta_l, \phi_l) \in \mathbb{C}$ $(l = 0, \dots, M-1)$ *Funktionswerte von* (4.1).

Komplexität: $\mathcal{O}(MN^2)$.

4.2 Schnelle sphärische Fourier-Transformationen

In diesem Abschnitt wollen wir den NDSFT-Algorithmus 4.1 für spezielle Gitter vereinfachen. Dieser Algorithmus wird mit DSFT bezeichnet. Damit sind wir in der Lage, die Tensorproduktstruktur der Kugelflächenfunktionen Y_k^n in (4.8) auszunutzen. Zusätzlich dazu werden noch schnelle eindimensionale Algorithmen bezüglich θ und ϕ (Zeilen-Spalten-Verfahren) benutzt und wir erhalten asymptotisch schnellere Algorithmen (FSFT).
Wir beginnen mit dem Problem, die Funktion f in (4.1) für gegebene sphärische Fourier-Koeffizienten a_k^n $(k = 0, \dots, N; n = -k, \dots, k)$ auf einem Gitter (4.5) zu berechnen. Durch die Separation der Variablen und Berechnung von

$$h_{s,n} := h_n\left(\cos\frac{s\pi}{T}\right) \quad (s = 0, \dots, T) \tag{4.11}$$

für $n = -N, \dots, N$, gefolgt von $T+1$ diskreten Fourier-Transformationen der Länge $2T$

$$f\left(\frac{s\pi}{T}, \frac{t\pi}{2T}\right) = \sum_{n=-N}^{N} h_{s,n}\, \mathrm{e}^{\mathrm{i}nt/(2T)} \quad (t = 0, \dots, 2T-1) \tag{4.12}$$

für alle $s = 0, \dots, T$ erhalten wir einen Algorithmus, den wir mit DSFT bezeichnen. Damit hat die DSFT eine arithmetische Komplexität von $\mathcal{O}(N^3)$. Um einen schnellen

Algorithmus für die DSFT, die FSFT, zu erhalten, wird für jedes $n = -N, \ldots, N$ die Summe in (4.11) mit Hilfe einer schnellen Legendre-Funktionen-Transformation (FLT) berechnet. Wir benutzen hier für jedes n den Algorithmus [103, Algorithmus 4.2]. Dieses Verfahren beruht auf einer stabilisierten Version einer diskreten Polynomtransformation (siehe [102]) und benötigt $\mathcal{O}(N^2 \log^2 N)$ arithmetische Operationen. Die Berechnung der Summe in (4.12) wird für jedes $s = 0, \ldots, T$ mit einer FFT der Länge $2T$ realisiert. Dieses Vorgehen fassen wir im folgenden Algorithmus zusammen.

Algorithmus 4.2 (FSFT auf speziellen Gittern (4.5))

Eingabe: $N \in \mathbb{N}$, $a_k^n \in \mathbb{C}$ $(k = 0, \ldots, N; n = -k, \ldots, k)$.

Vorberechnung: $T := 2^{\lceil \log_2 N \rceil}$.

1. *Berechne für alle* $n = -N, \ldots, N$ *mit der schnellen FLT die Daten*

$$h_{s,n} = \sum_{k=|n|}^{N} a_k^n P_k^{|n|} \left(\cos(\frac{s\pi}{T}) \right) \quad (s = 0, \ldots, T-1)\,.$$

2. *Berechne für alle* $s = 0, \ldots, T-1$ *mittels FFT die Daten*

$$f(\frac{s\pi}{T}, \frac{t\pi}{2T}) = \sum_{n=-N}^{N} h_{s,n}\, \mathrm{e}^{\mathrm{i}nt\pi/(2T)} \quad (t = 0, \ldots, 2T-1)\,.$$

Ausgabe: $f\left(\frac{s\pi}{T}, \frac{t\pi}{2T}\right)$ $(s = 0, \ldots, T;\ t = 0, \ldots, 2T-1)$ *Funktionswerte von* (4.1) *auf dem Gitter* (4.5).

Komplexität: $\mathcal{O}(N^2 \log^2 N)$.

Bemerkung 4.3 (Schnelle Legendre-Funktionen-Transformation (FLT))
Die Idee zur Realisierung einer schnellen Legendre-Funktionen-Transformation stammt von J. Driscoll und D. Healy [27]. Diese Transformation ist aber numerisch instabil für große Transformationslängen. Deshalb wurden stabilisierte Versionen [103, 119, 63] veröffentlicht. Ein anderer approximativer Algorithmus zur Berechnung der FLT wurde von M. Mohlenkamp [83] vorgeschlagen. □

4.3 Schnelle sphärische Fourier-Transformationen für nichtäquidistante Daten

In diesem Abschnitt leiten wir einen schnellen Algorithmus für die NDSFT (vgl. Algorithmus 4.1) her und bezeichnen diesen Algorithmus als NFSFT. Zunächst wird wieder die schnelle Legendre-Funktionen-Transformation benutzt, um einen Basiswechsel zu

berechnen, der unabhängig von den Knoten ist. Anschließend nutzen wir die NFFT (Algorithmus 1.1 im bivariaten Fall, d.h. $d = 2$), um die Funktionswerte an allen Stellen (θ_l, ϕ_l) $(l = 0, \ldots, M-1)$ zu berechnen.
Für beliebige $(\theta, \phi) \in \mathbb{S}^2$ kann die sphärische Fourier-Summe (4.1) in der Form

$$f(\theta, \phi) = \sum_{n=-N}^{N} h_n(\cos\theta)\, \mathrm{e}^{\mathrm{i}n\phi} \tag{4.13}$$

geschrieben werden, wobei h_n in (4.9) gegeben ist.
Wir definieren den polynomialen Anteil von h_n, durch die Polynome vom Grad N

$$g_n(x) := \sum_{k=|n|}^{N} a_k^n P_k^{|n|}(x) \in \Pi_N\,, \tag{4.14}$$

falls $|n|$ eine gerade natürliche Zahl, und die Polynome vom Grad $N-1$ durch

$$g_n(x) := \frac{1}{\sqrt{1-x^2}} \sum_{k=|n|}^{N} a_k^n P_k^{|n|}(x) \in \Pi_{N-1}\,, \tag{4.15}$$

falls $|n|$ eine ungerade natürliche Zahl ist. Mit Π_N bezeichnen wir dabei den Raum der Polynome vom Grad kleiner oder gleich N.
Dann ist der Zusammenhang zur Definition (4.9) durch

$$h_n(\cos\theta) = \begin{cases} g_n(\cos\theta) & \text{für gerade } n, \\ (\sin\theta)\, g_n(\cos\theta) & \text{für ungerade } n \end{cases} \tag{4.16}$$

gegeben. Wir berechnen, wie im Schritt 1 von Algorithmus 4.2, für alle $n = -N, \ldots, N$ mit der schnellen Legendre-Funktionen-Transformation die Daten

$$g_{s,n} := g_n(\cos\frac{s\pi}{T}) \quad (s = 0, \ldots, T;\ T := 2^{\lceil \log_2 N \rceil}) \tag{4.17}$$

und erhalten anschließend die Chebyshev-Koeffizienten $\tilde{a}_k^n \in \mathbb{C}$ $(k = 0, \ldots, N)$ in

$$g_n(\cos\theta) = \sum_{k=0}^{N} \tilde{a}_k^n T_k(\cos\theta) \tag{4.18}$$

für gerade n und

$$g_n(\cos\theta) = \sum_{k=0}^{N-1} \tilde{a}_k^n T_k(\cos\theta) \tag{4.19}$$

für ungerade n unter Nutzung einer diskreten Kosinus-Transformation vom Typ I. Beachten wir, dass g_n Polynome vom Grad N bzw. $N-1$ sind, so können wir g_n unter Nutzung von

$$T_k(\cos\theta) = \cos(k\theta) = \frac{1}{2}\left(\mathrm{e}^{\mathrm{i}k\theta} + \mathrm{e}^{-\mathrm{i}k\theta}\right)$$

als Fourier-Summe

$$g_n(\cos\theta) = \sum_{k=-N}^{N} b_k^n \, \mathrm{e}^{\mathrm{i}k\theta}$$

mit den diskreten Fourier-Koeffizienten

$$b_k^n := \begin{cases} \tilde{a}_{|k|}^n & \text{für } k = 0, \\ \frac{\tilde{a}_{|k|}^n}{2} & \text{sonst} \end{cases} \tag{4.20}$$

schreiben. Für ungerade n folgt, unter Beachtung, dass $g_n(\cos\cdot)$ ein trigonometrisches Polynom vom Grad $N-1$ ist,

$$\begin{aligned} h_n(\cos\theta) = \sin(\theta) \sum_{k=-N+1}^{N-1} b_k^n \, \mathrm{e}^{\mathrm{i}k\theta} &= \frac{1}{2\mathrm{i}} \left(\mathrm{e}^{\mathrm{i}\theta} - \mathrm{e}^{-\mathrm{i}\theta}\right) \sum_{k=-N+1}^{N-1} b_k^n \, \mathrm{e}^{\mathrm{i}k\theta} \\ &= \sum_{k=-N}^{N} \tilde{b}_k^n \, \mathrm{e}^{\mathrm{i}k\theta} \end{aligned}$$

mit

$$\tilde{b}_k^n := \frac{1}{2\mathrm{i}} \times \begin{cases} -b_{k+1}^n & \text{für } k = -N, -N+1, \\ b_{k-1}^n & \text{für } k = N-1, N, \\ b_{k-1}^n - b_{k+1}^n & \text{sonst.} \end{cases} \tag{4.21}$$

Demzufolge kann h_n in der Form

$$h_n(\cos\theta) = \sum_{k=-N}^{N} c_k^n \, \mathrm{e}^{\mathrm{i}k\theta} \tag{4.22}$$

mit den diskreten Fourier-Koeffizienten

$$c_k^n := \begin{cases} b_k^n & \text{für gerade } n, \\ \tilde{b}_k^n & \text{für ungerade } n \end{cases} \tag{4.23}$$

dargestellt werden. Wir setzen diese Summe (4.22) in (4.13) ein und erhalten schließlich die Darstellung der Funktion f in der Form

$$f(\theta,\phi) = \sum_{n=-N}^{N} \sum_{k=-N}^{N} c_k^n \, \mathrm{e}^{\mathrm{i}k\theta} \, \mathrm{e}^{\mathrm{i}n\phi} \tag{4.24}$$

mit komplexen Koeffizienten $c_k^n \in \mathbb{C}$. Aus gegebenen sphärischen Fourier-Koeffizienten a_k^n in (4.1) haben wir die (gewöhnlichen) diskreten Fourier-Koeffizienten c_k^n in (4.24) mit einem exakten Algorithmus der Komplexität $\mathcal{O}\left(N^2 \log^2 N\right)$ unabhängig von den Knoten (θ_l, ϕ_l) berechnet.

Die geometrische Interpretation besagt, dass die Sphäre auf die äußere Hälfte des Torus abgebildet wird, während die innere Hälfte glatt (siehe Abbildung 4.1) fortgesetzt wird.

Abbildung 4.1: Topographie der Erde (links) und die „äußere Hälfte“ auf dem Torus (rechts).

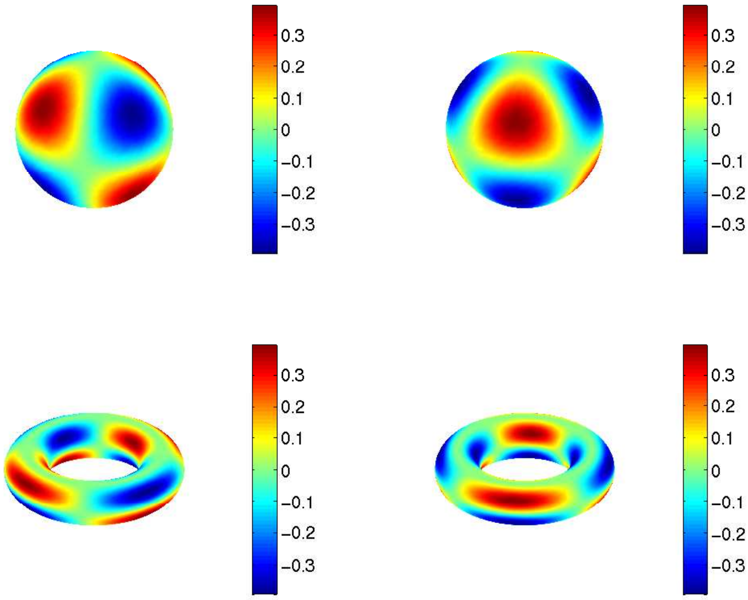

Abbildung 4.2: Die normalisierten Kugelflächenfunktionen $Y_3^{-2}(\theta, \phi) = \frac{1}{4}\sqrt{\frac{105}{2\pi}} \sin^2\theta \cos\theta \, \mathrm{e}^{-2\mathrm{i}\phi}$, oben: Realteil (links) und Imaginärteil (rechts), unten: die Kugelflächenfunktion auf dem Torus, der innere Teil ist glatt am Nord- bzw. Südpol fortgesetzt.

Zunächst teilen wir f in einen geraden und einen ungeraden Teil (relativ zu den Polen) und konstruieren die geraden und ungeraden Fortsetzungen. Abbildung 4.2 illustriert diesen Fall für die Kugelflächenfunktion Y_3^{-2}. Auf dem Torus gilt natürlich $\theta \in [0, 2\pi)$.

Im letzten Schritt wird f an den beliebigen Knoten mit einer bivariaten NFFT (Algorithmus 1.1) berechnet. Wir fassen die Herleitung im folgenden Algorithmus zusammen.

Algorithmus 4.4 (NFSFT)

Eingabe: $N, M \in \mathbb{N}$, $a_k^n \in \mathbb{C}$ $(k = 0, \dots, N; n = -k, \dots, n)$,

$(\theta_l, \phi_l) \in \mathbb{S}^2$ $(l = 0, \ldots, M-1)$.
Vorberechnung: $T := 2^{\lceil \log_2 N \rceil}$.

1. *Berechne für alle* $n = -N, \ldots, N$ *mit der schnellen Legendre-Funktionen-Transformation (FLT) und* $s = 0, \ldots, T$ *die Daten*

$$g_n(\cos \frac{s\pi}{T}) = \begin{cases} \sum_{k=|n|}^{N} a_k^n P_k^{|n|}(\cos \frac{s\pi}{T}) & \textit{falls } n \textit{ gerade ist,} \\ \frac{1}{\sin \frac{s\pi}{T}} \sum_{k=|n|}^{N} a_k^n P_k^{|n|}(\cos \frac{s\pi}{T}) & \textit{falls } n \textit{ ungerade ist.} \end{cases}$$

2. *Berechne für* $n = -N, \ldots, N$ *mit einer DCT-I die Chebyshev-Koeffizienten* $\tilde{a}_k^n$ *in* (4.18) *und* (4.19).

3. *Berechne für* $k = -N, \ldots, N$ *und* $n = -N, \ldots, N$ *die diskreten Fourier-Koeffizienten* c_k^n *nach den Formeln* (4.20), (4.21) *und* (4.23).

4. *Berechne die Funktionswerte* $f(\theta_l, \phi_l)$ *in* (4.24)

$$f(\theta_l, \phi_l) = \sum_{n=-N}^{N} \sum_{k=-N}^{N} c_k^n \, \mathrm{e}^{\mathrm{i}(k\theta_l + n\phi_l)}$$

mit einer bivariaten NFFT (Algorithmus 1.1).

Ausgabe: $f(\theta_l, \phi_l)$ $(l = 0, \ldots, M-1)$.

Komplexität: $\mathcal{O}(N^2 \log^2 N + m^2 M)$.

Bemerkung 4.5 Oftmals ist man an einer reellen Darstellung von Funktionen auf $\mathbb{S}^2$ interessiert. Die reelle Fourier-Reihe der Funktion $f \in L^2(\mathbb{S}^2)$ mit Bandbreite N ist dann an Stelle von (4.1) durch

$$f(\theta, \phi) = \sum_{k=0}^{N} \left(a_k^0 P_k(\cos\theta) + \sum_{n=1}^{k} \left(a_k^n \cos(n\phi) + b_k^n \sin(n\phi) \right) \, P_k^n(\cos\theta) \right)$$

mit reellen Koeffizienten a_k^n und b_k^n gegeben. Zur schnellen Auswertung für diese Funktionen können wir ähnliche Algorithmen wie in diesem Abschnitt herleiten. An Stelle der NFFT werden dann die NFCT (Algorithmus 1.13) und die NFST (Algorithmus 1.15) benutzt. □

m	1	2	3	4	5	6	7	8
E_∞	5.0e-02	7.7e-03	3.0e-04	1.9e-05	7.1e-06	5.8e-07	5.1e-08	2.3e-08

Tabelle 4.3: Relativer Fehler E_∞ für das System (I) ($N = 128$, $\sigma = 2$ und $M = 100$).

4.4 Numerische Ergebnisse

Wir haben die oben beschriebenen Algorithmen in C implementiert und auf den beiden Systemen
(I) Intel-Celeron 64MB RAM, SuSe-Linux und
(II) Sun-SPARC 768MB RAM, SunOS 5.6
getestet. Dabei wird der Stabilisierungsparameter der FLT auf 10000 gesetzt (siehe [75, 77]). In den ersten Beispielen untersuchen wir die Komplexität und Genauigkeit unseres neuen Algorithmus im Vergleich zur direkten Berechnung. Im letzten Beispiel wird unser Algorithmus am EGM96 Datensatz getestet.

Beispiel 4.1 (Genauigkeit)
Das Interesse in diesem Beispiel ist auf das Verhältnis zwischen Abschneideparameter m und dem relativen Fehler

$$E_\infty := \frac{\max\limits_{l=0,\ldots,M-1} |f_{\mathrm{app}}(\theta_l, \phi_l) - f_{\mathrm{slow}}(\theta_l, \phi_l)|}{\max\limits_{l=0,\ldots,M-1} |f_{\mathrm{slow}}(\theta_l, \phi_l)|}$$

gerichtet. Hier bezeichnet $f_{\mathrm{app}}(\theta_l, \phi_l)$ den Funktionswert von Algorithmus 4.4 und $f_{\mathrm{slow}}(\theta_l, \phi_l)$ den Funktionswert, der mit Algorithmus 4.1 berechnet wird. Wir beobachten einen exponentiellen Abfall des Fehlers für wachsendes m (siehe Tabelle 4.4). □

Beispiel 4.2 (CPU-Zeiten)
Zunächst wählen wir zufällige sphärische Fourier-Koeffizienten $a_k^n \in [0, 1]$ und zufällige Knoten $(\theta_l, \phi_l) \in \mathbb{S}^2$. Tabelle 4.4 vergleicht die CPU-Zeit von Algorithmus 4.1 und Algorithmus 4.4 für verschiedene Bandbreiten und gewisse Anzahlen von Knoten (Oversamplingfaktor $\sigma = 2$, Abschneideparameter $m = 4$ mit dem Gauß-Kern, siehe Algorithmus 1.1). □

Beispiel 4.3 (Komplexität)
In diesem Beispiel wird die Rechenzeit bezüglich einer gegebenen Bandbreite N und einer bestimmten Anzahl von Knoten M untersucht. Wir wählen $M = N^2$. Abbildung 4.5 zeigt einen log-log-Plot der verstrichenen CPU-Zeit. Man beachte, dass z.B. für eine Bandbreite von $N = 500$ unser schneller Algorithmus 4.4 nur 92 Sekunden und die direkte Berechnung (Algorithmus 4.1) mehr als 7 Stunden benötigt. □

N	64		128		256	
M	Alg. 4.1	Alg. 4.4	Alg. 4.1	Alg. 4.4	Alg. 4.1	Alg. 4.4
1	0.01	0.37	0.01	1.80	0.01	8.61
10	0.01	0.38	0.05	1.82	0.20	8.61
100	0.13	0.42	0.49	1.84	1.82	8.67
1000	1.27	0.43	4.82	1.88	18.24	8.71
10000	12.61	0.53	49.04	1.96	176.41	8.81
20000	25.44	0.65	98.20	2.09	352.75	8.95

Tabelle 4.4: CPU-Zeit in Sekunden für das System (I).

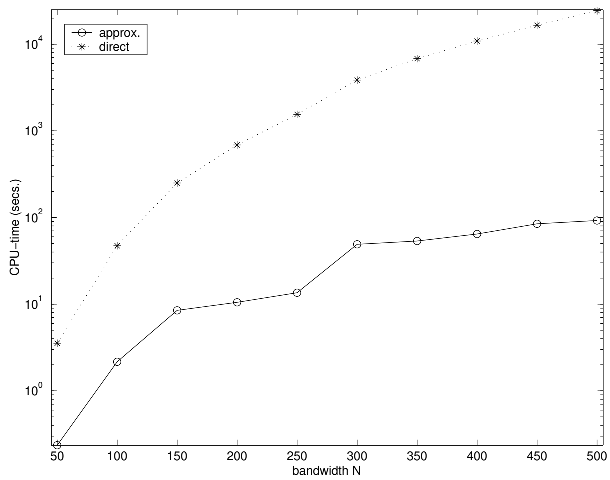

Abbildung 4.5: CPU-Zeit für das System (II) ($\sigma = 2$, $m = 4$).

Beispiel 4.4 (EGM96 Datensatz)
In diesem Beispiel zeigen wir eine Anwendung von Algorithmus 4.4 auf den EGM96 Datensatz. Die Daten besteht aus 360^2 speziell normierten sphärischen Fourier-Koeffizienten a_k^n und repräsentiert das Gravitationspotential der Erde (siehe [79]). Abbildung 4.6 und Abbildung 4.7 stellen die Rekonstruktion der gesamten Erdoberfläche dar, während Abbildung 4.8 einen Ausschnitt („zoom-in"-Eigenschaft) hervorhebt. Damit sind wir in der Lage, aus dem gegebenen globalen Modell Ausschnitte beliebiger Regionen zu rekonstruieren. □

Weitere Themen und Literaturhinweise

Das Lösen von partiellen Differentialgleichungen auf der Sphäre hat in vielen Bereichen, wie z.B. bei der Wettervorhersage oder in Klimamodellen [16, Section 18.7], eine große Bedeutung. Insbesondere werden dazu sphärische Fourier-Reihen der Form (4.1) benutzt. Durch die Manipulation dieser Reihen (Integration oder Multiplikation) wächst

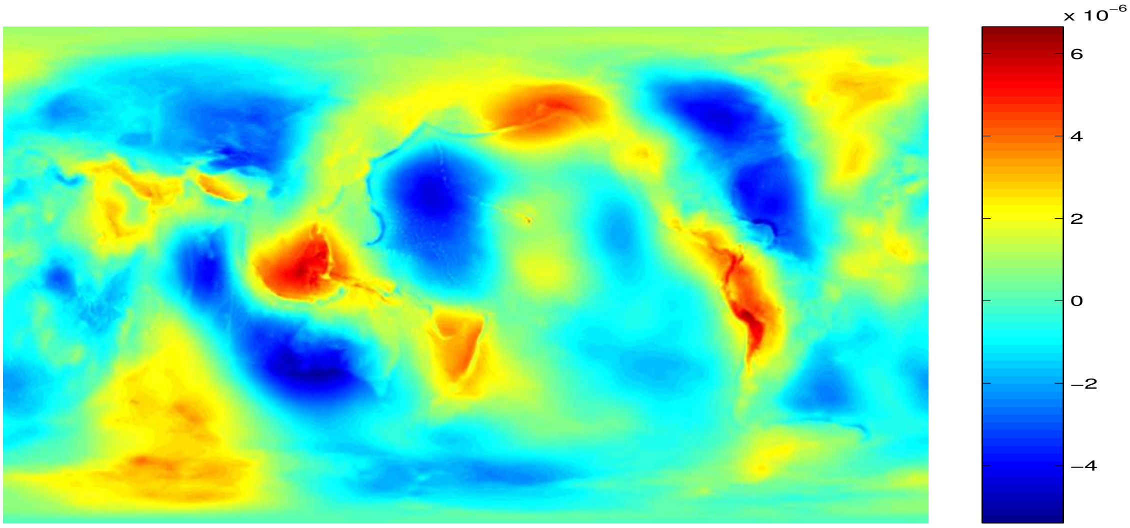

Abbildung 4.6: Rekonstruktion von EGM96 Daten für $k \geq 4$.

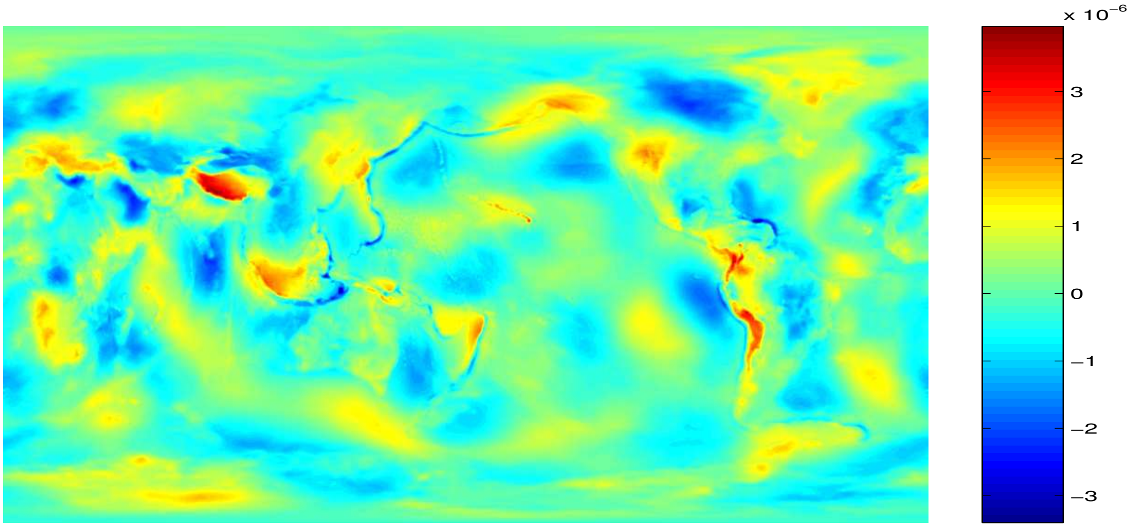

Abbildung 4.7: Rekonstruktion von EGM96 Daten für $k \geq 8$.

die Bandbreite. Als Teilproblem ergibt sich beim Lösen von partiellen Differentialgleichungen auf der Sphäre das Projizieren auf den Raum der bandbeschränkten Funktionen mit kleinerer Bandbreite. Diese Aufgabe ist als „sphärisches Filtern“ [68] bekannt. Da die arithmetische Komplexität sphärischen Filterns $\mathcal{O}(N^3)$ für $\mathcal{O}(N^2)$ Punkte auf der Sphäre ist, sind schnelle Algorithmen von großem Interesse. R. Jakob-Chien und K. Alpert [68] präsentierten erstmals einen schnellen Algorithmus der Komplexität $\mathcal{O}(N^2 \log N)$, basierend auf der schnellen Multipol-Methode. Dieser Algorithmus wurde von N. Yarvin und V. Rokhlin [130] verbessert. Wir haben diese sphärischen Filter mittels unseres schnellen

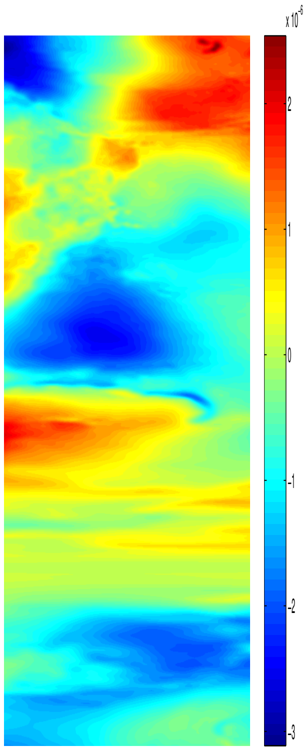

Abbildung 4.8: Östlichen Teils Südamerikas als Ausschnitt von Abbildung 4.6.

Algorithmus 2.3 mit dem Kern $1/x$ realisiert [13, 14]. Unser Vorgehen ist einfacher zu implementieren, basiert auf der NFFT und hat auch nur eine arithmetische Komplexität von $\mathcal{O}(N^2 \log N)$. Weiterhin benutzten wir diese Methode zur schnellen Berechnung von Wavelet-Entwicklungen auf der Sphäre [13, 14].

Literatur

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

[2] B. Baxter. Preconditioned conjugate gradients, radial basis functions, and Toeplitz matrices. *Comput. Math. Appl.*, 43:305 – 318, 2002.

[3] R. K. Beatson, J. B. Cherrie, and C. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration. *Adv. Comput. Math.*, 11:253 – 270, 1999.

[4] R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: methods for 2–dimensional polyharmonic splines. IMA *J. Numer. Anal.*, 17:343 – 372, 1997.

[5] R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. SIAM *J. Sci. Comput.*, pages 1717 – 1740, 2000.

[6] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: I. *Comput. Math. Appl.*, 24:7 – 19, 1992.

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

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

[9] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure and Appl. Math.*, 44:141 – 183, 1991.

[10] G. Beylkin and R. Cramer. A multiresolution approach to regularization of singular operators and fast summation. SIAM *J. Sci. Comput.*, 24:81 – 117, 2002.

[11] D. Bini and F. Di Benedetto. A new preconditioner for the parallel solution of positive definite Toeplitz systems. In *Proc. Second* ACM *Symp. on Parallel Algorithms and Architectures*, pages 220 – 223, Crete, 1990.

[12] A. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[13] M. Böhme. A fast algorithm for filtering and wavelet decomposition on the sphere. Diplomarbeit, Universität zu Lübeck, Report B-02-08, 2002.

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

[15] E. Boman and I. Koltracht. Fast transform based preconditioners for Toeplitz equations. SIAM *J. Matrix Anal. Appl.*, 16:628 – 645, 1995.

[16] J. P. Boyd. *Chebyshev and Fourier Spectral Methods.* Dover Press, New York, second edition, 2000.

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

[18] M. Bronstein, A. Bronstein, and M. Zibulevsky. The non-uniform FFT and its applications. *Report, Israel Institute of Technology*, 2002.

[19] R. H. Chan and M. K. Ng. Conjugate gradient methods of Toeplitz systems. SIAM *Rev.*, 38:427 – 482, 1996.

[20] K. Chandrasenkharan. *Classical Fourier Transforms.* Spinger-Verlag, Berlin, 1989.

[21] H.-B. Cheong. Application of double Fourier series to the shallow–water equations on the sphere. *J. Comput. Phys.*, 165:261 – 287, 2000.

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

[23] C. K. Chui. *An Introduction to Wavelets.* Academic Press, Boston, 1992.

[24] C. W. Clenshaw. A note on the summation of Chebyshev series. *Math. Comput.*, 9:118 – 120, 1955.

[25] J. W. Cooley and J. W. Tukey. An algorithm for machine calculation of complex Fourier series. *Math. Comput.*, 19:297 – 301, 1965.

[26] P. Davis. *Interpolation and Approximation.* Dover, New York, 1975.

[27] J. Driscoll and D. Healy. Computing Fourier transforms and convolutions on the 2–sphere. *Adv. Appl. Math.*, 15:202 – 250, 1994.

[28] J. O. Droese. Verfahren zur schnellen Fourier–Transformation mit nichtäquidistanten Knoten. Diplomarbeit, TU Darmstadt, 1996.

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

[30] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration and differentiation. SIAM *J. Numer. Anal.*, 33:1689 – 1711, 1996.

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

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

[33] P. Edholm and G. Herman. Linograms in image reconstruction from projections. IEEE *Trans. Med. Imag.*, 6:301 – 307, 1987.

[34] B. Elbel. Mehrdimensionale Fouriertransformation für nichtäquidistante Daten. Diplomarbeit, TH Darmstadt, 1998.

[35] B. Elbel and G. Steidl. Fast Fourier transform for nonequispaced data. In C. K. Chui and L. L. Schumaker, editors, *Approximation Theory IX*, Nashville, 1998. Vanderbilt University Press.

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

[37] H. Faßbender. On numerical methods for discrete least-squares approximation by trigonometric polynomials. *Math. Comput.*, 66:719 – 741, 1997.

[38] H. Feichtinger and K. Gröchenig. Theory and practice of irregular sampling. In J. Benedetto and M. Frazier, editors, *Wavelets: Mathematics and Applications*, pages 305 – 363, CRC Press, 1993.

[39] H. Feichtinger, K. Gröchenig, and T. Strohmer. Efficient numerical methods in non-uniform sampling theory. *Numer. Math.*, 69:423 – 440, 1995.

[40] M. Fenn and D. Potts. Fast summation based on fast trigonometric transforms at nonequispaced nodes. *Numer. Linear Algebra Appl.*, to appear.

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

[42] J. A. Fessler and B. P. Sutton. Nonuniform fast Fourier transforms using min-max interpolation. IEEE *Trans. Signal Process.*, 51:560 – 574, 2003.

[43] G. B. Folland. *Fourier Analysis and its Applications*. Brooks/Cole, Albany, 1992.

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

[45] K. Fourmont. Non equispaced fast Fourier transforms with applications to tomography. Preprint Universität Münster, 2002.

[46] W. Freeden, T. Gervens, and M. Schreiner. *Constructive Approximation on the Sphere*. Oxford University Press, Oxford, 1998.

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

[48] M. A. Golberg and C. S. Chen. *Discrete Projection Methods for Integral Equations*. Computational Mechanics Publications, Southampton, 1997.

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

[50] D. Gottlieb, B. Gustafsson, and P. Forssen. On the direct Fourier method for computer tomography. IEEE *Trans. Med. Imag.*, 9:223 – 232, 2000.

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

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

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

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

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

[56] U. Grenander and G. Szegö. *Toeplitz Forms and Their Applications*. University of California Press, Los Angeles, 1958.

[57] D. Grishin and T. Strohmer. Fast multi-dimensional scattered data approximation with Neumann boundary conditions. *submitted*, 2002.

[58] K. Gröchenig and T. Strohmer. Numerical and theoretical aspects of non-uniform sampling of band-limited images. In F. Marvasti, editor, *Theory and Practice of Nonuniform Sampling*. Kluwer/Plenum, 2001.

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

[60] W. Hackbusch, B. N. 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*, pages 9 – 29, Berlin, 2000. Springer-Verlag.

[61] W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math.*, 54:463 – 491, 1989.

[62] M. Hanke. *Conjugate gradient type method for ill–posed problems*. Wiley, New York, 1995.

[63] D. Healy, P. Kostelec, S. S. B. Moore, and D. Rockmore. FFTs for the 2-sphere – Improvements and variations. *J. Fourier Anal. Appl.*, 9, 2003.

[64] M. T. Heideman, D. H. Johnson, and C. S. Burrus. Gauss and the history of the fast Fourier transform. *Arch. Hist. Exact Sci.*, 34:265 – 277, 1985.

[65] G. Heinig and K. Rost. Representations of Toeplitz–plus–Hankel matrices using trigonometric transforms with application to fast matrix–vector multiplication. *Linear Algebra Appl.*, 275:225 – 248, 1998.

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

[67] J. I. Jackson. Selection of a convolution function for Fourier inversion using gridding. IEEE *Trans. Med. Imag.*, 10:473 – 478, 1991.

[68] R. Jakob-Chien and B. K. Alpert. A fast spherical filter with uniform resolution. *J. Comput. Phys.*, 136:580 – 584, 1997.

[69] A. J. Jerri. The Shannon sampling theorem – its various extensions and applications: A tutorial review. *Proc. IEEE*, pages 1565 – 1596, 1977.

[70] A. C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging.* IEEE Press, New York, 1987.

[71] M. Kaveh and M. Soumekh. Computer–assisted diffraction tomography. In H. Stark, editor, *Image Recovery: Theory and Applications*, pages 369 – 413, Academic Press, Orlando, 1987.

[72] M. Kaveh, M. Soumekh, and J. Greenleaf. Signal processing for diffraction tomography. IEEE *Trans. Sonics Ultrasonics*, 31, 1984.

[73] W. Klaverkamp. Tomographische Bildrekonstruction mit direkten algebraischen Verfahren. Dissertation, Universität Münster, 1991.

[74] H. Kruse. Resolution of reconstruction methods in computerized tomography. SIAM *J. Sci. Stat. Comput.*, 10:447 – 474, 1989.

[75] S. Kunis. Schnelle Fourieralgorithmen auf der Sphäre. Studienarbeit, Universität zu Lübeck, Report B-02-01, 2002.

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

[77] S. Kunis and D. Potts. Fast spherical Fourier algorithms. *J. Comput. Appl. Math*, 161:75 – 98, 2003.

[78] S. Kunis, D. Potts, and G. Steidl. Fast Fourier transforms for nonequispaced data – A user's guide to a C-library. Preprint B-02-13, Universität zu Lübeck, 2002.

[79] F. G. Lemoine and E. C. Pavlis et.al. The development of the joint NASA GSFC and NIMA geopotential model EGM96. Technical report, NASA Goddard Space Flight Center, Greenbelt, Maryland, 20771 USA, 1998.

[80] C. Li. CGNR is an error reducing algorithm. SIAM *J. Sci. Comput.*, 22:2109 – 2112, 2001.

[81] J. Li, Y. C. Hon, and C. S. Chen. Numerical comparisons of two meshless methods using radial basis functions. *Eng. Anal. Bound. Elem.*, 26:205 – 225, 2002.

[82] H. Mhaskar, F. Narcowich, and J. Ward. Quadrature formulas on spheres using scattered data. *Math. Comput.*, to appear.

[83] M. J. Mohlenkamp. A fast transform for spherical harmonics. *J. Fourier Anal. Appl.*, 5:159 – 184, 1999.

[84] F. Natterer. Fourier reconstruction in tomography. *Numer. Math.*, 47:343 – 353, 1985.

[85] F. Natterer. *The Mathematics of Computerized Tomography.* Teubner, Stuttgart, 1986.

[86] F. Natterer and F. Wübbeling. *Mathematical Methods in Image Reconstruction.* SIAM, Philadelphia, 2000.

[87] N. Nguyen and Q. H. Liu. The regular Fourier matrices and nonuniform fast Fourier transforms. SIAM *J. Sci. Comput.*, 21:283 – 293, 1999.

[88] A. Nieslony, D. Potts, and G. Steidl. Rapid evaluation of radial functions by fast Fourier transforms at nonequispaced knots. Preprint A-02-11, Universität zu Lübeck, 2002.

[89] A. Nieslony and G. Steidl. Approximate factorizations of Fourier matrices with nonequispaced knots. *Linear Algebra Appl.*, 266:337 – 351, 2003.

[90] F. Oberhettinger. *Tables of Fourier Transforms and Fourier Transforms of Distributions.* Springer, Berlin, 1990.

[91] J. O'Sullivan. A fast sinc function gridding algorithm for Fourier inversion in computer tomography. IEEE *Trans. Med. Imag.*, 4:200 – 207, 1985.

[92] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8:43 – 71, 1982.

[93] J. Pelt. Fast computation of trigonometric sums with applications to frequency analysis of astronomical data. In D. Maoz, A. Sternberg, and E. Leibowitz, editors, *Astronomical Time Series*, pages 179 – 182, Kluwer, 1997.

[94] D. Potts. Schnelle Polynomtransformation und Vorkonditionierer für Toeplitz–Matrizen. Dissertation, Universität Rostock, 1998.

[95] D. Potts. Fast algorithms for discrete polynomial transforms on arbitrary grids. *Linear Algebra Appl.*, 366:353 – 370, 2003.

[96] D. Potts. Approximation of scattered data by trigonometric polynomials on the torus and the 2-sphere. *Adv. Comput. Math.*, to appear.

[97] D. Potts and G. Steidl. Optimal trigonometric preconditioners for nonsymmetric Toeplitz systems. *Linear Algebra Appl.*, 281:265 – 292, 1998.

[98] D. Potts and G. Steidl. New Fourier reconstruction algorithms for computerized tomography. In A. Aldroubi, A. Laine, and M. Unser, editors, *Proceedings of SPIE: Wavelet Applications in Signal and Image Processing VIII*, volume 4119, pages 13 – 23, 2000.

[99] D. Potts and G. Steidl. A new linogram algorithm for computerized tomography. IMA *J. Numer. Anal.*, 21:769 – 782, 2001.

[100] D. Potts and G. Steidl. Fourier reconstruction of functions from their nonstandard sampled Radon transform. *J. Fourier Anal. Appl.*, 8:513 – 533, 2002.

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

[102] D. Potts, G. Steidl, and M. Tasche. Fast algorithms for discrete polynomial transforms. *Math. Comput.*, 67:1577 – 1590, 1998.

[103] D. Potts, G. Steidl, and M. Tasche. Fast and stable algorithms for discrete spherical Fourier transforms. *Linear Algebra Appl.*, 275:433 – 450, 1998.

[104] D. Potts, G. Steidl, and M. Tasche. Fast Fourier transforms for nonequispaced data: A tutorial. In J. J. Benedetto and P. J. S. G. Ferreira, editors, *Modern Sampling Theory: Mathematics and Applications*, pages 247 – 270, Boston, 2001. Birkhäuser.

[105] D. Potts, G. Steidl, and M. Tasche. Numerical stability of fast trigonometric transforms - a worst case study. *J. Concrete Appl. Math.*, 1:1 – 36, 2003.

[106] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1992.

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

[108] L. Reichel, G. Ammar, and W. Gragg. Discrete least squares approximation by trigonometric polynomials. *Math. Comput.*, 57:273 – 289, 1991.

[109] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publ., Boston, 1996.

[110] R. Schaback. *Reconstruction of Multivariate Functions from Scattered Data.* Buchmanuskript.

[111] H. Schomberg and J. Timmer. The gridding method for image reconstruction by Fourier transformation. IEEE *Trans. Med. Imag.*, MI–14:596 – 607, 1995.

[112] U. Schreiber. Numerische Stabilität von schnellen trigonometrischen Transformationen. Dissertation, Universität Rostock, 2000.

[113] J. Schulte. Fourier–Rekonstruktionen in der Computer–Tomographie. Diplomarbeit, Universität Münster, 1994.
http://wwwmath.uni-muenster.de/inst/num/Abschlussarbeiten/Schulte/.

[114] R. A. Scramek and F. R. Schwab. Imaging. In R. Perley, F. R. Schwab, and A. Bridle, editors, *Astronomical Society of the Pacific Conference*, volume 6, pages 117 – 138, 1988.

[115] G. Steidl. Fast radix–p discrete cosine transform. *Appl. Algebra Eng. Commun. Comput.*, 3:39 – 46, 1992.

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

[117] G. Steidl and M. Tasche. A polynomial approach to fast algorithms for discrete Fourier–cosine and Fourier–sine transforms. *Math. Comput.*, 56:281 – 296, 1991.

[118] G. Steidl and M. Tasche. *Schnelle Fourier–Transformation – Theorie und Anwendungen.* 8 Lehrbriefe der FernUniv. Hagen, 1997.

[119] R. Suda and M. Takami. A fast spherical harmonics transform algorithm. *Math. Comput.*, 71:703 – 715, 2001.

[120] X. Sun and Y. Bao. A kronecker product representation of the fast Gauss transform. SIAM *J. Matrix Anal. Appl.*, 24:768 – 786, 2003.

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

[122] M. Tasche and H. Zeuner. Roundoff error analysis for fast trigonometric transforms. In *Handbook of Analytic-Computational Methods in Applied Mathematics*, pages 357 – 406, CRC Press, Boca Raton, 2000.

[123] B. Tian and Q. H. Liu. Nonuniform fast cosine transform and Chebyshev PSTD algorithm. *J. Electromagnet. Waves Appl*, 14:797 – 798, 2000.

[124] P. Toft. *The Radon Transform - Theory and Implementation.* PhD thesis, Technical University of Denmark, 1996. `http://www.sslug.dk/~pto`.

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

[126] J. Walden. Analysis of the direct Fourier method for computer tomography. IEEE *Trans. Med. Imag.*, 9:211 – 222, 2000.

[127] Z. Wang. Fast algorithms for the discrete W transform and for the discrete Fourier transform. IEEE *Trans. Acoust. Speech Signal Process*, 32:803 – 816, 1984.

[128] A. F. Ware. Fast approximate Fourier transforms for irregularly spaced data. SIAM *Rev.*, 40:838 – 856, 1998.

[129] P. Y. Yalamov. Improvements of some bounds on the stability of fast Helmholtz solvers. *Numer. Algorithms*, 26:11 – 20, 2001.

[130] N. Yarvin and V. Rokhlin. A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics. *J. Comput. Phys.*, 147:549 – 609, 1998.

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