Universität zu Lübeck

Institut für Mathematik

Schnelle trigonometrische Transformationen an nicht-äquidistanten Knoten

Studienarbeit
im Rahmen des Diplomstudiengangs Informatik

Vorgelegt von
Steffen Klatt

Betreuer
Prof. Dr. Daniel Potts

Lübeck, Dezember 2005

Inhaltsverzeichnis

1 Einführung

Eines der zentralen Belange der angewandten Mathematik und gleichzeitig Motor für andere Naturwissenschaften, wie zum Beispiel der Physik, ist die Entwicklung effizienter Algorithmen. Mit Zunahme der Popularität des Rechners in diversesten Bereichen hat die Existenz schneller Algorithmen für fundamentale Aufgaben an Bedeutung gewonnen. Hierzu zählt unter anderem die sehr bekannte und auf zahlreichen Gebieten eingesetzte *Schnelle Fourier-Transformation* (Fast Fourier-Transform, FFT).

Bei der Fourier-Transformation handelt es sich um eine Integraltransformation, die einer Funktion eine andere, ihrer sogenannten Fourier-Transformierten, zuordnet. Durch Diskretisierung wird das Fourier-Integral in ein trigonometrisches Polynom vom Grade N überführt, das es an M Knoten auszuwerten gilt. Die Anzahl der arithmetischen Rechenoperationen zur Auswertung dieser Summe beträgt bei einer d-dimensionalen Transformation $\mathcal{O}(MN^d)$. Unter der Voraussetzung, dass die Knoten äquidistant sind, spricht man hierbei von der *Diskreten Fourier-Transformation* (Discrete Fourier-Transform, DFT). Unter Verwendung der FFT sinkt die arithmetische Komplexität für $M = N^d$ auf $\mathcal{O}(N \log N)$.

Im Jahr 1965 veröffentlichten die Mathematiker James W. Cooley und John W. Tukey einen FFT-Algorithmus der eine DFT der Länge N in nur $\mathcal{O}(N \log N)$ arithmetischen Operationen realisiert. Der Algorithmus zerbricht das Problem in kleinere DFTs. Es wurden später weitere, teils spezialisierte, Algorithmen vorgeschlagen. So zum Beispiel Bruun's FFT (1978), die nur mit reellen Eingabedaten arbeitet, für alle geraden Längen eingesetzt werden kann, jedoch gegenüber Cooley-Tukey an numerischer Ungenauigkeit leidet. Ein FFT-Algorithmus für die DFT von beliebiger Länge wurde 1968 von L.I. Bluestein publiziert. Rader und Brenner konzentrierten sich auf prime Längen. Diese können selbstverständlich mit Cooley-Tukey kombiniert werden. Somit existieren schnelle Algorithmen für verschiedenste Eingabelängen.

Falls die Eingabedaten reell und symmetrisch sind, kann man die DFT auf eine DCT (gerade, Discrete Cosine Transform) bzw. DST (ungerade, Discrete Sine Transform) zurückführen. Auch für diese Transformationen existieren schnelle Algorithmen mit einer arithmetischen Komplexität von $\mathcal{O}(N \log N)$.

Vorrangigen Einsatz dieser Transformationen finden wir in der digitalen Signalverarbeitung, wie zum Beispiel in Kompressionsverfahren für Bilddaten, oder in der numerischen Analysis. Dort werden beispielsweise spezielle Integralgleichungen mit Hilfe der genannten Transformationen gelöst. Jedoch haben alle im vorletzten Absatz genannten Algorithmen einen Nachteil gemeinsam: Das Gitter muss stets äquidistant sein. Dies ist in vielen Anwendungen, beispielsweise in der Magnet-Resonanz-Tomographie, ungünstig. Es ist daher wünschenswert die Stützstellen beziehungsweise Messpunkte beliebig wählen zu können. Hierfür wurde der NFFT-Algorithmus *(Nonequispaced Fast Fourier-Transform)*, der eine schnelle *Diskrete Fourier-Transformation für nicht-äquidistanten*

Knoten (NDFT) realisiert, vorgeschlagen [12, 4, 13]. Wir werden diesen Ansatz nutzen, um die diskreten trigonometrischen Transformationen für reelle, nicht-äquidistante Knoten zu entwickeln (NFCT: *Nonequispaced Fast Cosine-Transform* und NFST: *Nonequispaced Fast Sine-Transform*). Diese Algorithmen wurden erstmals in [5] vorgestellt. In dieser Arbeit übertragen wir die Ergebnisse erstmals in den multivariaten Fall und führen dazu verschiedene numerische Testrechnungen durch.

Die Arbeit ist wie folgt gegliedert. In Abschnitt 2 definieren wir zunächst die diskreten trigonometrischen Transformationen (DFT, DCT, DST) und vereinbaren grundlegende Notationen. Danach führen wir in Abschnitt 3 die nicht-äquidistanten trigonometrischen Transformationen ein. Im weiteren Verlauf dieses Abschnittes wird der Ansatz für den NFFT-Algorithmus erklärt, woraus wir anschließend die NFCT- und NFST-Algorithmen ableiten werden. Dazu gehören auch deren „transponierte“ Algorithmen auf die wir kurz eingehen. Als letzten Teil in diesem Abschnitt untersuchen wir erstmals die inversen trigonometrischen Transformationen. Wir werden sehen, dass diese durch iterative Lösungsverfahren für Gleichungssysteme (CGNR, CGNE) gelöst werden können.
Es folgt in Abschnitt 4 eine allgemeine Fehlerabschätzung für die Algorithmen NFCT und NFST. Nachdem wir einen kurzen Überblick von verschiedenen, vorgeschlagenen Fensterfunktionen φ gegeben haben, werden wir exemplarisch für einen dieser Fensterfunktionen eine Fehlerabschätzung herleiten.
Die Verwendung der Software-Bibliothek stellen wir in Abschnitt 5 vor. Wir erläutern die Schritte zur Durchführung einer Transformation und benennen Optionen und Parameter. Um den Vorteil der Spezialisierung des Algorithmus zu zeigen, werden Zeit- und Fehlermessungen durchgeführt und in Abschnitt 6 durch verschiedene Grafiken veranschaulicht. Abschließend betrachten wir einige numerische Beispiele. Wir werden mit Hilfe der in dieser Arbeit entwickelten Algorithmen gestreute Daten durch trigonometrische Polynome interpolieren. Diese Polynome können wir dann zum Beispiel auf unterschiedlichen Gittern auswerten. Der verwendete Datensatz entspricht Höhenangaben an einer endlichen Zahl von Gitterpunkten eines Gletschers. Ausserdem rekonstruieren wir eine vorgegebene zweidimensionale Funktion.

Ohne die bemerkenswerte Unterstützung und Geduld von Prof. Dr. Daniel Potts wäre es nicht zur Fertigstellung dieser Arbeit gekommen. Aus diesem Grund möchte ich ihm an dieser Stelle meinen besonderen Dank aussprechen.

2 Bezeichnungen und Definitionen

Wir führen für Vektoren $\boldsymbol{v} = (v_t)_{t=0}^{d-1}, \boldsymbol{w} = (w_t)_{t=0}^{d-1} \in \mathbb{C}^d$ die Operation $\boldsymbol{v} \odot \boldsymbol{w}$ ein und meinen damit die komponentenweise Multiplikation $(v_0 w_0, v_1 w_1, ..., v_{d-1}, w_{d-1})^{\mathrm{T}}$. Die Menge der komplexen, reellen, ganzen und natürlichen Zahlen kennzeichnen wir wie üblich mit den Symbolen $\mathbb{C}, \mathbb{R}, \mathbb{Z}$ und $\mathbb{N}$. In Tabelle 1 fassen wir weitere Bezeichnungen zusammen, die in dieser Arbeit benutzt werden.

$\boldsymbol{v} = \boldsymbol{v}_N := (v_j)_{l=0}^{N-1}$	Spaltenvektor der Länge N
$\boldsymbol{v}^{\mathrm{T}}$	transponierter Vektor von $\boldsymbol{v}$
$\mathbf{1}_d := (1, ..., 1)^T$	Eins-Vektor der Länge d
$\Pi(\boldsymbol{N}_d) := \prod_{t=0}^{d-1}(N_t)$	Produkte der Einträge des Vektors $\boldsymbol{N}_d$
$\|\|\boldsymbol{v}\|\|_1 := \sum_{l=0}^{N-1} \|v_l\|$	Summen-Norm des Vektors $\boldsymbol{v} := (v_l)_{l=0}^{N-1}$
$\|\|\boldsymbol{v}\|\|_\infty := \max_{l=0,...,N-1} \|v_l\|$	Maximum-Norm des Vektors $\boldsymbol{v} := (v_l)_{l=0}^{N-1}$
$<_c, \leq_c, \geq_c, >_c$	komponentenweise Vergleichsoperatoren auf Vektoren mit üblicher Bedeutung
$\mathbb{T}^d := [-\frac{1}{2}, \frac{1}{2})^d$	Torus der Dimension d
$I_{\boldsymbol{a}}^{\boldsymbol{b},d} := \{\boldsymbol{k} \in \mathbb{Z}^d : \boldsymbol{a} \leq_c \boldsymbol{k} <_c \boldsymbol{b} \wedge \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}^d\}$	Index-Bereich
$I_{\boldsymbol{a},m}^{\boldsymbol{b},d}(\boldsymbol{x}) := \{\boldsymbol{l} \in I_{\boldsymbol{a}}^{\boldsymbol{b},d} :$ $\boldsymbol{b} \odot \boldsymbol{x} - m\mathbf{1}_d \leq_c \boldsymbol{l} \leq_c \boldsymbol{b} \odot \boldsymbol{x} + m\mathbf{1}_d,$ $\boldsymbol{a}, \boldsymbol{b} \in \{\mathbb{N} \cup 0\}^d, \boldsymbol{x} \in \mathbb{R}^d, m \in \mathbb{N}\}$	Index-Bereich
$\varepsilon_{n,k} = \begin{cases} \frac{1}{2} & (k \in \{0, k\}) \\ 1 & (\text{sonst}) \end{cases}$, $\varepsilon_{\boldsymbol{a},\boldsymbol{k}} := \prod_{t=0}^{d-1} \varepsilon_{a_t,k_t}$	Vorfaktor
$\boldsymbol{A} := (a_{k,l})_{k,l=0}^{M-1,N-1}$	Matrix der Dimension (M, N)
$\mathrm{diag}(\boldsymbol{v}_M) := (a_{k,l})_{k,l=0}^{M-1} \in \mathbb{R}^{M \times M}$ mit $a_{k,l} = 0, (k \neq l), a_{k,k} = v_k$	Diagonalmatrix der Dimension (M, M)
$\boldsymbol{I}_M := \mathrm{diag}(\mathbf{1}_M) \in \mathbb{Z}^{M \times M}$	Einheitsmatrix der Dimension (M, M)
$\boldsymbol{A}^{\mathrm{T}}$	transponierte Matrix von $\boldsymbol{A}$

Tabelle 1: Mathematische Notationen

DFT, DCT-I und DST-I

An dieser Stelle definieren wir die diskrete Fourier-Transformation (DFT) und die diskreten Kosinus- und Sinus-Transformationen des Typs I (DCT-I, DST-I). Die eindimensionale DFT ist eine lineare Abbildung, die jedem Vektor $\hat{\boldsymbol{x}} := (\hat{x}_k)_{k=0}^{N-1} \in \mathbb{C}^N$ den Vektor $\boldsymbol{x} := (x_k)_{k=0}^{N-1} \in \mathbb{C}^N$ durch

$$x_k := \sum_{j=0}^{N-1} \hat{x}_j \mathrm{e}^{\frac{-2\pi \mathrm{i} jk}{N}}, \qquad (k = 0, ..., N-1), \tag{1}$$

zuordnet. Die DCT-I der Länge $N+1$ ist ebenfalls eine lineare Abbildungen, die jedem Vektor $\hat{\boldsymbol{x}} := (\hat{x}_k)_{k=0}^{N} \in \mathbb{R}^{N+1}$ den Vektor $\boldsymbol{x} := (x_k)_{k=0}^{N} \in \mathbb{R}^{N+1}$ zuordnet. Diese Abbildung ist definiert durch

$$x_k := \sum_{j=0}^{N} \varepsilon_{N,j} \hat{x}_j \cos\Big(\frac{jk\pi}{N}\Big), \qquad (k = 0, ..., N), \tag{2}$$

wobei $\varepsilon_{a,b}$ für $d = 1$ wie in Tabelle 1 eingeführt. Letztlich definieren wir die DST-I der Länge $N-1$. Auch hier handelt es sich um eine lineare Abbildungen, die jedem Vektor $\hat{\boldsymbol{x}} := (\hat{x}_k)_{k=1}^{N-1} \in \mathbb{R}^{N-1}$ den Vektor $\boldsymbol{x} := (x_k)_{k=1}^{N-1} \in \mathbb{R}^{N-1}$ durch die Vorschrift

$$x_k := \sum_{j=1}^{N-1} \hat{x}_j \sin\Big(\frac{jk\pi}{N}\Big), \qquad (k = 1, ..., N-1), \tag{3}$$

zuordnet. Die multivariaten Transformationen seien definiert als das separierbare Produkt der gegebenen 1-dimensionalen Transformationen entlang jeder Dimension. Führen wir den Vektor $\boldsymbol{N} := (N_0, N_1, ..., N_{d-1})^{\mathrm{T}} \in \mathbb{N}^d$ ein, so ist die d-dimensionale Kosinus-Transformation definiert durch

$$x_{\boldsymbol{k}} := \sum_{\boldsymbol{j} \in I_{\boldsymbol{0}}^{\boldsymbol{N}+\mathbf{1}_d,d}} \varepsilon_{\boldsymbol{N}+\mathbf{1}_d,\boldsymbol{j}} \hat{x}_{\boldsymbol{j}} \prod_{t=0}^{d-1} \cos\Big(\frac{j_t k_t \pi}{N_t}\Big), \qquad (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N}+\mathbf{1}_d,d}),$$

und analog die d-variate DST-I

$$x_{\boldsymbol{k}} := \sum_{\boldsymbol{j} \in I_{\mathbf{1}}^{\boldsymbol{N},d}} \hat{x}_{\boldsymbol{j}} \prod_{t=0}^{d-1} \sin\Big(\frac{j_t k_t \pi}{N_t}\Big), \qquad (\boldsymbol{k} \in I_{\mathbf{1}}^{\boldsymbol{N},d}).$$

Der Kosinus und Sinus eines Vektors $\boldsymbol{x} \in \mathbb{R}^d$ sind in dieser Arbeit als das Tensorprodukt $\cos(\boldsymbol{x}) := \prod_{t=0}^{d-1} \cos(x_t)$ und $\sin(\boldsymbol{x}) := \prod_{t=0}^{d-1} \sin(x_t)$ definiert. Im weiteren verwenden wir für die multivariate DCT-I die Schreibweise

$$x_{\boldsymbol{k}} := \sum_{\boldsymbol{j} \in I_{\boldsymbol{0}}^{\boldsymbol{N}+\mathbf{1}_d,d}} \varepsilon_{\boldsymbol{N}+\mathbf{1}_d,\boldsymbol{j}} \hat{x}_{\boldsymbol{j}} \cos\Big(\pi(\boldsymbol{j} \odot \boldsymbol{k}) \odot \boldsymbol{N}^{-1}\Big), \qquad (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N}+\mathbf{1}_d,d}), \tag{4}$$

beziehungsweise für die multivariate DST-I

$$x_{\boldsymbol{k}} := \sum_{\boldsymbol{j} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d}} \hat{x}_{\boldsymbol{j}} \sin\Big(\pi(\boldsymbol{j} \odot \boldsymbol{k}) \odot \boldsymbol{N}^{-1}\Big), \qquad (\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d}), \tag{5}$$

indem wir $\boldsymbol{N}^{-1}$ als den Vektor $(\frac{1}{N_0}, ..., \frac{1}{N_{d-1}})^{\mathrm{T}}$ verstehen und den Operator $\odot$ wie definiert als komponentenweises Produkt anwenden.

Bemerkung: Unter [7] findet man Algorithmen, mit denen die Auswertung der Summen (1), (2) und (3) jeweils $\mathcal{O}(N \log N)$ arithmetische Operationen benötigt. Die d-variaten Transformationen (4) und (5) beanspruchen jeweils $\mathcal{O}(\Pi(\boldsymbol{N}) \log \Pi(\boldsymbol{N}))$ arithmetische Rechenschritte. Auch für diese Fälle stellt [7] entsprechende Algorithmen zur Verfügung.

3 Diskrete Transformationen für nicht-äquidistante Knoten

Die NDFT (diskrete Fourier-Transformation für nicht-äquidistante Daten) ist bezüglich der Wahl der Knoten eine Verallgemeinerung der DFT. Während die Knoten $\boldsymbol{x}_{\boldsymbol{j}} := \left(\frac{j_t}{N_t}\right)_{t=0}^{d-1} \in \mathbb{C}^d$, $(\boldsymbol{j} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d})$, bei der DFT auf einem gleichmässig verteilten Gitter liegen, sind bei der NDFT beliebige Stützstellen $\boldsymbol{x}_j \in \mathbb{T}^d$, $(j \in I_0^{M,1})$, zugelassen. Sei $I_{-\boldsymbol{N}}^{\boldsymbol{N},d}$ wie in Tabelle 1 definiert und der d-dimensionale Vektor $\boldsymbol{x} = (x_0, ..., x_{d-1})^{\mathrm{T}} \in \mathbb{T}^d$ sowie die Koeffizienten $\hat{f}_{\boldsymbol{k}}^F \in \mathbb{C}^d$, $(\boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d})$, gegeben. Wir definieren das multivariate trigonometrische Polynom

$$f^F(\boldsymbol{x}) := \sum_{\boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}} \hat{f}_{\boldsymbol{k}}^F \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}}. \tag{6}$$

Für endlich viele diskrete Knoten $\boldsymbol{x}_j \in \mathbb{T}^d$, $(j \in I_0^{M,1})$, ist die NDFT somit als die lineare Abbildung, die jedem Vektor $\hat{\boldsymbol{f}}^F = (\hat{f}_{\boldsymbol{k}}^F)_{\boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}}$ den Vektor $\boldsymbol{f}^F = (f_j^F)_{j \in I_0^{M,1}}$ durch

$$f_j^F := f^F(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}} \hat{f}_{\boldsymbol{k}}^F \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}, \qquad (j \in I_0^{M,1}), \tag{7}$$

zuordnet, definiert. Wir werden mit (7) später den Zusammenhang zwischen der nicht-äquidistanten Fourier-Transformation und den nicht-äquidistanten Kosinus- und Sinus-Transformationen herstellen. Die Summe (7) kann als Matrix-Vektor-Multiplikation aufgefasst werden. Mit der nicht-äquidistanten Fourier-Matrix $\boldsymbol{A}_F := \left(\mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}\right)_{j \in I_0^{M,1}, \boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}}$ und dem Vektor $\hat{\boldsymbol{f}}^F := \left(\hat{f}_{\boldsymbol{k}}^F\right)_{\boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}}$ lässt sich (7) somit als das Matrix-Vektor-Produkt $\boldsymbol{A}_F \hat{\boldsymbol{f}}^F$ interpretieren. Des weiteren definieren wir die transponierte NDFT (kurz: $\mathrm{NDFT}^{\mathrm{T}}$) als die Abbildung

$$h_{\boldsymbol{k}}^F := h^F(\boldsymbol{k}) = \sum_{j \in I_0^{M,1}} f_j^F \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}, \qquad (\boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}), \tag{8}$$

wobei $\boldsymbol{x}_j \in \mathbb{T}^d$ und $f_j^F \in \mathbb{C}$, $(j \in I_0^{M,1})$, gegeben sind. Selbstverständlich können wir (8) wiederum als Matrix-Vektor-Multiplikation auffassen und betrachten daher das Matrix-Vektor-Produkt ${\boldsymbol{A}_F}^{\mathrm{T}} \boldsymbol{f}^F$. Nun verallgemeinern wir die DCT und die DST. Im Wesentlichen betrachten wir zwei Spezialfälle für die Summe (7). Auf diese Zusammenhänge werden spätere Abschnitte noch detailierter eingehen. Wir führen die diskreten Transformationen NDCT (nicht-äquidistante Kosinus-Transformation) und NDST (nicht-äquidistante Sinus-Transformation) ein. Dies sind die Abbildungen

$$f_j^C := f^C(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \hat{f}_{\boldsymbol{k}}^C \cos(2\pi(\boldsymbol{k} \odot \boldsymbol{x}_j)), \qquad (j \in I_0^{M,1}), \tag{9}$$

und

$$f_j^S := f^S(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d}} \hat{f}_{\boldsymbol{k}}^S \sin(2\pi(\boldsymbol{k} \odot \boldsymbol{x}_j)), \qquad (j \in I_0^{M,1}), \tag{10}$$

an beliebigen, nicht gleichmässig verteilten Knoten $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$, wobei $\hat{f}_{\boldsymbol{k}}^C \in \mathbb{R}$, $(\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$ und $\hat{f}_{\boldsymbol{k}}^S \in \mathbb{R}$, $(\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d})$ jeweils gegebene Koeffizienten sind.

Die transponierte NDCT ($\mathrm{NDCT}^{\mathrm{T}}$) und die transponierte NDST ($\mathrm{NDST}^{\mathrm{T}}$) sind mit den gegebenen Koeffizienten $f_j^C, f_j^S \in \mathbb{R}$, $(j \in I_0^{M,1})$, und den beliebigen Stützstellen $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$, als die Abbildungen

$$h_{\boldsymbol{k}}^C := h^C(\boldsymbol{k}) = \sum_{j \in I_0^{M,1}} f_j^C \cos(2\pi(\boldsymbol{k} \odot \boldsymbol{x}_j)), \qquad (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}), \tag{11}$$

und

$$h_{\boldsymbol{k}}^S := h^S(\boldsymbol{k}) = \sum_{j \in I_0^{M,1}} f_j^S \sin(2\pi(\boldsymbol{k} \odot \boldsymbol{x}_j)), \qquad (\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d}), \tag{12}$$

definiert. Beachten wir auch hier wieder, dass die Berechnung der Summen (9) bis (12) äquivalent zur Berechnung der Matrix-Vektor-Produkte $\boldsymbol{A}_C \hat{\boldsymbol{f}}^C$, ${\boldsymbol{A}_C}^{\mathrm{T}} \boldsymbol{f}^C$, $\boldsymbol{A}_S \hat{\boldsymbol{f}}^S$ und ${\boldsymbol{A}_S}^{\mathrm{T}} \boldsymbol{f}^S$ ist, wobei

$$\hat{\boldsymbol{f}}^C := (\hat{f}_{\boldsymbol{k}}^C)_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \in \mathbb{R}^{\Pi(\boldsymbol{N})}, \qquad \hat{\boldsymbol{f}}^S := (\hat{f}_{\boldsymbol{k}}^S)_{\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d}} \in \mathbb{R}^{\Pi(\boldsymbol{N}-\boldsymbol{1}_d)}$$

und

$$\begin{aligned} \boldsymbol{A}_C := \boldsymbol{A}_C^d &= \big(\prod_{t=0}^{d-1} \cos(2\pi k_t(x_t)_j)\big)_{j\in I_0^{M,1}, \boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \\ &= \big(\cos(2\pi(\boldsymbol{k}\odot\boldsymbol{x}_j))\big)_{j\in I_0^{M,1}, \boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \in \mathbb{R}^{M\times\Pi(\boldsymbol{N})} \end{aligned}$$

die nicht-äquidistante Kosinus-Matrix ist. Analog stellt $\boldsymbol{A}_S$ die nicht-äquidistante Sinus-Matrix, definiert durch

$$\begin{aligned} \boldsymbol{A}_S := \boldsymbol{A}_S^d &= \big(\prod_{t=0}^{d-1} \sin(2\pi k_t(x_t)_j)\big)_{j\in I_0^{M,1}, \boldsymbol{k}\in I_{\boldsymbol{1}}^{\boldsymbol{N},d}} \\ &= \big(\sin(2\pi(\boldsymbol{k}\odot\boldsymbol{x}_j))\big)_{j\in I_0^{M,1}, \boldsymbol{k}\in I_{\boldsymbol{1}}^{\boldsymbol{N},d}} \in \mathbb{R}^{M\times\Pi(\boldsymbol{N}-\boldsymbol{1}_d)}, \end{aligned}$$

dar. In dieser Arbeit werden wir Algorithmen (NFFT, NFFT$^{\mathrm{T}}$) zur schnellen Berechnung der Summen (7) und (8) kennenlernen und daraus Algorithmen (NFCT, NFST, NFCT$^{\mathrm{T}}$ und NFST$^{\mathrm{T}}$) zur schnellen Auswertung der d-variaten Summen (9), (10), (11) und (12) entwickeln. Wir werden die Zusammenhänge zwischen der NFFT und deren Spezialfälle verdeutlichen, auf Laufzeitvorteile eingehen und Fehlerabschätzungen durchführen, da es sich bei allen Typen um approximative Algorithmen handelt.

3.1 Schnelle nicht-äquidistante Fourier-Transformation

Die NFCT- und NFST-Algorithmen bauen auf den Ideen zur schnellen Fourier-Transformation für nicht-äquidistante Knoten (NFFT) auf. Aus diesem Grund werden wir hier die Herleitung des NFFT-Algorithmus zusammenfassen, um die NFCT- und NFST-Algorithmen daraus herzuleiten. Wir sind nun daran interessiert die Stützwerte f_j^F, $(j \in I_0^{M,1})$, und $h_{\boldsymbol{k}}^F$, $(\boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d})$ der Summen (7) und (8) an den Stützstellen $\boldsymbol{x}_j \in \mathbb{T}^d$, $(j \in I_0^{M,1})$, effizient zu berechnen. Der grundlegende Gedanke der NFFT ist die Approximation des trigonometrischen Polynoms f^F durch eine Linearkombination von Translaten einer stetigen 1-periodischen Funktion $\tilde{\varphi}$. Wir nennen φ in diesem Artikel *Fensterfunktion*. Es wurden unter anderen in [1],[4] und [8] mehrere Fensterfunktionen vorgeschlagen. Sei $\varphi \in L^2(\mathbb{R}^d) \cap L^1(\mathbb{R}^d)$ derart, dass ihre 1-periodisierte Funktion $\tilde{\varphi}$, definiert durch

$$\tilde{\varphi}(\boldsymbol{x}) := \sum_{\boldsymbol{r}\in\mathbb{Z}^d} \varphi(\boldsymbol{x}+\boldsymbol{r}), \tag{13}$$

eine gleichmäßig konvergente Fourier-Reihe hat. Dann können wir $\tilde{\varphi}$ als Fourier-Reihe

$$\tilde{\varphi}(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^d} c_{\boldsymbol{k}}(\tilde{\varphi})\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} \tag{14}$$

mit den Fourier-Koeffizienten

$$c_{\boldsymbol{k}}(\tilde{\varphi}) = \int\limits_{\mathbb{T}^d} \tilde{\varphi}(\boldsymbol{x}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} \mathrm{d}\boldsymbol{x}, \qquad (\boldsymbol{k} \in \mathbb{Z}^d) \tag{15}$$

ausdrücken. Die mehrdimensionale Funktion $\varphi(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ interpretieren wir als Tensorprodukt der univariaten Funktionen $\varphi(x_t)$, $(t = 0, ..., d-1)$. So gelten für φ sowie im weiteren für $\hat{\varphi}$ und ψ die Definitionen

$$\varphi(\boldsymbol{x}) := \prod_{t=0}^{d-1} \varphi(x_t), \quad \hat{\varphi}(\boldsymbol{x}) := \prod_{t=0}^{d-1} \hat{\varphi}(x_t) \quad \text{und} \quad \psi(\boldsymbol{x}) := \prod_{t=0}^{d-1} \psi(x_t).$$

Mit den Definitionen (13) und (15) ergibt sich der Zusammenhang

$$\hat{\varphi}(\boldsymbol{k}) := \int\limits_{\mathbb{R}^d} \varphi(\boldsymbol{x}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} \mathrm{d}\boldsymbol{x} = c_{\boldsymbol{k}}(\tilde{\varphi}), \tag{16}$$

zwischen der Fourier-Transformierten $\hat{\varphi}$, der Funktion φ und den Fourier-Koeffizienten $c_{\boldsymbol{k}}(\tilde{\varphi})$ der Funktion $\tilde{\varphi}$, denn es gilt

$$\hat{\varphi}(\boldsymbol{k}) = \int\limits_{\mathbb{T}^d} \sum_{\boldsymbol{r} \in \mathbb{R}^d} \varphi(\boldsymbol{x} + \boldsymbol{r}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} \mathrm{d}\boldsymbol{x} = \int\limits_{\mathbb{R}^d} \varphi(\boldsymbol{x}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} \mathrm{d}\boldsymbol{x} = c_{\boldsymbol{k}}(\tilde{\varphi}). \tag{17}$$

Folglich sind die Koeffizienten $c_{\boldsymbol{k}}(\tilde{\varphi})$ an den Punkten $\boldsymbol{k} \in \mathbb{Z}^d$ abgetastete Funktionswerte der Fourier-Transformierten $\hat{\varphi}$. Als nächstes betrachten wir die Linearkombination

$$s_1(\boldsymbol{x}) := \sum_{\boldsymbol{l} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} g^F_{\boldsymbol{l}} \tilde{\varphi}\Big(\boldsymbol{x} - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)\Big). \tag{18}$$

deren Koeffizienten $g^F_{\boldsymbol{l}}$ möglichst so zu bestimmen sind, dass $s_1(\boldsymbol{x})$ das Polynom f^F gut annähert. Die konstanten Komponenten des Vektors $\boldsymbol{\sigma} > \mathbf{1}_d$ werden Oversampling-Faktoren genannt und sind dergestalt, dass $\boldsymbol{\sigma} \odot \boldsymbol{N}$ aus $\mathbb{N}^d$ ist. Dies ist im weiteren Verlauf dieser Arbeit insbesondere die Länge einer d-variaten DFT. Bei günstiger Wahl von $\boldsymbol{\sigma}$ (z.B. $\sigma_0 N_0, ..., \sigma_{d-1} N_{d-1}$ sind Zweierpotenzen) kann die FFT zur Berechnung der diskreten Fourier-Transformation sehr effizient angewendet werden. Wir entwickeln s_1 in eine Fourier-Reihe und erhalten

$$s_1(\boldsymbol{x}) = \sum_{\boldsymbol{r} \in \mathbb{Z}^d} c_{\boldsymbol{k}}(s_1) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}}$$

mit den Fourier-Koeffizienten $c_{\boldsymbol{k}}(s_1)$, die wir wie üblich durch

$$\begin{aligned} c_{\boldsymbol{k}}(s_1) &= \int\limits_{\mathbb{T}^d} s_1(\boldsymbol{x}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} \mathrm{d}\boldsymbol{x} \\ &= \int\limits_{\mathbb{T}^d} \sum_{\boldsymbol{l} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} g^F_{\boldsymbol{l}} \tilde{\varphi}\Big(\boldsymbol{x} - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)\Big) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} \mathrm{d}\boldsymbol{x} \end{aligned}$$

berechnen. Substituieren wir nun $\boldsymbol{x} - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)$ mit $\boldsymbol{y}$ dann folgt

$$c_{\boldsymbol{k}}(s_1) = \int\limits_{\mathbb{T}^d} \sum_{\boldsymbol{l} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} g^F_{\boldsymbol{l}} \tilde{\varphi}(\boldsymbol{y}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}(\boldsymbol{y} + \frac{1}{2}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}))} \mathrm{d}\boldsymbol{y}.$$

Zuletzt schreiben wir die von $\boldsymbol{y}$ unabhängigen Summanden vor das Integral. Somit ergibt sich für die Fourier-Koeffizienten von s_1 die Identität

$$c_{\boldsymbol{k}}(s_1) = \Big(\sum_{\boldsymbol{l} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} g^F_{\boldsymbol{l}} \mathrm{e}^{-\pi \mathrm{i} \boldsymbol{k}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})} \Big) \underbrace{\Big(\int\limits_{\mathbb{T}^d} \tilde{\varphi}(\boldsymbol{y}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{y}} \mathrm{d}\boldsymbol{y} \Big)}_{c_{\boldsymbol{k}}(\tilde{\varphi})}.$$

Mit (15) und der Vereinbarung

$$\hat{g}^F_{\boldsymbol{k}} = \sum_{\boldsymbol{l} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} g^F_{\boldsymbol{l}} \mathrm{e}^{-\pi \mathrm{i} \boldsymbol{k}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})} \tag{19}$$

schreiben wir die Fourier-Reihe von s_1 als

$$\begin{aligned} s_1(\boldsymbol{x}) &= \sum_{\boldsymbol{r} \in \mathbb{Z}^d} \hat{g}^F_{\boldsymbol{k}} c_{\boldsymbol{k}}(\tilde{\varphi}) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} \\ = \sum_{\boldsymbol{k} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} \hat{g}^F_{\boldsymbol{k}} c_{\boldsymbol{k}}(\tilde{\varphi}) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}} &+ \sum_{\boldsymbol{r} \in \mathbb{Z}^d \setminus \{\boldsymbol{0}\}} \sum_{\boldsymbol{k} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} \hat{g}^F_{\boldsymbol{k}} c_{\boldsymbol{k}+(2\boldsymbol{\sigma}\odot\boldsymbol{N})\odot\boldsymbol{r}}(\tilde{\varphi}) \mathrm{e}^{2\pi \mathrm{i}(\boldsymbol{k}+(2\boldsymbol{\sigma}\odot\boldsymbol{N})\odot\boldsymbol{r})\boldsymbol{x}}. \end{aligned} \tag{20}$$

Da die Fourier-Matrix $\boldsymbol{F}^d_{\boldsymbol{\sigma}\odot\boldsymbol{N}} := \big(\mathrm{e}^{-\pi \mathrm{i} \boldsymbol{k}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})}\big)_{\boldsymbol{l} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}, \boldsymbol{k} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}}$ bis auf Normierung unitär ist, können wir die Koeffizienten $g^F_{\boldsymbol{l}}$ mittels

$$g^F_{\boldsymbol{l}} := (2^{-d} \Pi(\boldsymbol{\sigma} \odot \boldsymbol{N}))^{-1} \sum_{\boldsymbol{k} \in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} \hat{g}^F_{\boldsymbol{k}} \mathrm{e}^{\pi \mathrm{i} \boldsymbol{k}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})} \tag{21}$$

bestimmen. Dabei handelt es sich um eine d-dimensionale inverse Fourier-Transformation der Länge $2\sigma_t N_t$ für jede Dimension $t = 0, ..., d-1$. Durch einen Vergleich von f^F_j und der Fourier-Reihe von s_1 zeigt sich, dass mit $c_{\boldsymbol{k}}(\tilde{\varphi}) \neq 0$, $(\boldsymbol{k} \in I^{\boldsymbol{N},d}_{-\boldsymbol{N}})$, als Forderung an die Fensterfunktion φ die Koeffizienten $\hat{g}^F_{\boldsymbol{k}}$ durch

$$\hat{g}_{\boldsymbol{k}}^F = \begin{cases} \frac{\hat{f}_{\boldsymbol{k}}^F}{c_{\boldsymbol{k}}(\tilde{\varphi})} & (\boldsymbol{k} \in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}) \\ 0 & (\boldsymbol{k} \in I_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d} \setminus I_{-\boldsymbol{N}}^{\boldsymbol{N},d}) \end{cases} \tag{22}$$

berechnet werden können. Wir stellen fest, dass sich die Ergebnisse verbessern, wenn wir weiterhin fordern, dass die Werte von $c_{\boldsymbol{k}}(\tilde{\varphi})$ für $\boldsymbol{k} \geq_c 2\boldsymbol{\sigma} \odot \boldsymbol{N} - \boldsymbol{N}$ klein sind. Die so bestimmten $\hat{g}_{\boldsymbol{k}}^F$ verwenden wir für die Berechnung der $g_{\boldsymbol{l}}^F$ in (21). Wir führen die Funktion

$$\psi(\boldsymbol{x}) := \begin{cases} \varphi(\boldsymbol{x}) & (x_t \in \big[-\frac{1}{2}m(\sigma_t N_t)^{-1}, \frac{1}{2}m(\sigma_t N_t)^{-1}\big], \forall t = 0, ..., d-1) \\ 0 & (\text{sonst}) \end{cases} \tag{23}$$

ein und definieren ihre 1-periodisierte Version $\tilde{\psi}(\boldsymbol{x}) := \sum_{\boldsymbol{r}\in\mathbb{Z}^d} \psi(\boldsymbol{x}+\boldsymbol{r})$. Sie soll eine Annäherung an die Fensterfunktion φ sein. Die Voraussetzung dafür ist eine gute Lokalität der gewählten Fensterfunktion. Da φ unter Beachtung dieser Bedingung gewählt worden ist, können wir s_1 nun durch

$$\begin{aligned} s(\boldsymbol{x}) &:= \sum_{\boldsymbol{l}\in I_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} g_{\boldsymbol{l}}^F \tilde{\psi}\Big(\boldsymbol{x} - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1}\big)\Big) \\ &= \sum_{\boldsymbol{l}\in I_{-\boldsymbol{\sigma}\odot\boldsymbol{N},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}(\boldsymbol{x})} g_{\boldsymbol{l}}^F \tilde{\psi}\Big(\boldsymbol{x} - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1}\big)\Big) \end{aligned} \tag{24}$$

annähern. Für jeden Knoten $\boldsymbol{x}_j$, $(j \in I_0^{M,1})$, hat die Summe $s(\boldsymbol{x}_j)$ in (24) $(2m+1)^d$ oder weniger nichtverschwindende Summanden. Wir vermerken abschließend das $s(\boldsymbol{x})$ die Summe $f^F(\boldsymbol{x})$ annähert und verweisen auf die Fehlerabschätzungen und praktischen Testbeispiele in anschließenden Abschnitten. Zunächst sollen jedoch die vorangegangenen Rechenschritte in Matrix-Vektor-Darstellung veranschaulicht werden.

Matrix-Vektor-Notation

Es ist sinnvoll die Algorithmen in Matrix-Vektor-Notation zu schreiben, da es im weiteren Verlauf die Darstellung der Algorithmen (NFFT, NFCT, NFST) und insbesondere der Algorithmen NFFT$^{\mathrm{T}}$, NFCT$^{\mathrm{T}}$, NFST$^{\mathrm{T}}$ zur Berechnung der Summen (8), (11) und (12) erleichtert. Wir definieren die Diagonal-Matrix $\boldsymbol{D}$

$$\boldsymbol{D} := \operatorname{diag}\Big(\frac{1}{c_{\boldsymbol{k}}(\tilde{\varphi})}\Big)_{\boldsymbol{k}\in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}} \in \mathbb{C}^{2^d\Pi(\boldsymbol{N})\times 2^d\Pi(\boldsymbol{N})},$$

die abgeschnittene, d-dimensionale Fourier-Matrix $\boldsymbol{F}_{2\boldsymbol{\sigma}\odot\boldsymbol{N},2\boldsymbol{N}}^d \in \mathbb{C}^{2^d\Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})\times 2^d\Pi(\boldsymbol{N})}$

$$\begin{aligned} \boldsymbol{F}_{2\boldsymbol{\sigma}\odot\boldsymbol{N},2\boldsymbol{N}}^d &:= 2^{-d}\Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1}\big(\mathrm{e}^{\pi\mathrm{i}\boldsymbol{k}(\boldsymbol{l}\odot(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1})}\big)_{\boldsymbol{l}\in I_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}, \boldsymbol{k}\in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}} \\ &= \underbrace{\boldsymbol{F}_{2\sigma_0 N_0, 2N_0}^1 \otimes \cdots \otimes \boldsymbol{F}_{2\sigma_{d-1}N_{d-1},2N_{d-1}}^1}_{d\text{-mal}} \end{aligned}$$

mit den eindimensionalen Fourier-Matrizen $\boldsymbol{F}^1_{2\sigma_t N_t,2N_t} := (2\sigma_t N_t)^{-1}\big(\mathrm{e}^{\pi \mathrm{i} kl/(\sigma_t N_t)}\big)_{l=-\sigma_t N_t,k=-N_t}^{\sigma_t N_t-1,N_t-1}$ für $t = 0, ..., d-1$ und letztlich die Multilevel-Bandmatrix $\boldsymbol{B}$

$$\boldsymbol{B} := \Big(\tilde{\psi}\Big(\boldsymbol{x}_j - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)\Big)\Big)_{j\in I_0^{M,1},\boldsymbol{l}\in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} \in \mathbb{R}^{M\times 2^d\Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})}.$$

Folglich kann der NFFT-Algorithmus in Matrix-Vektor-Darstellung als

$$(f_j^F)_{j\in I_0^{M,1}} \approx \boldsymbol{B}(\boldsymbol{F}^d_{2\boldsymbol{\sigma}\odot\boldsymbol{N},2\boldsymbol{N}})\boldsymbol{D}\hat{\boldsymbol{f}}^F$$

formuliert werden. Somit ergeben sich die Funktionswerte von $h^F(\boldsymbol{k})$, $(\boldsymbol{k} \in I^{\boldsymbol{N},d}_{-\boldsymbol{N}})$, durch Auswertung von

$$\begin{aligned}(h(\boldsymbol{k}))_{\boldsymbol{k}\in I^{\boldsymbol{N},d}_{-\boldsymbol{N}}} &\approx (\boldsymbol{B}(\boldsymbol{F}^d_{2\boldsymbol{\sigma}\odot\boldsymbol{N},2\boldsymbol{N}})\boldsymbol{D})^{\mathrm{T}}\boldsymbol{f}^F\\ &= \boldsymbol{D}(\boldsymbol{F}^d_{2\boldsymbol{\sigma}\odot\boldsymbol{N},2\boldsymbol{N}})^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}(f_j^F)_{j\in I_0^{M,1}}.\end{aligned}$$

3.2 Schnelle nicht-äquidistante Kosinus-Transformation

Mit Hilfe der Ergebnisse des vorangegangenen Abschnitts entwickeln wir nun die schnellen Algorithmen NFCT und NFCT$^{\mathrm{T}}$ zur Berechnung der Summen (9) und (11). Seien die Funktionen φ und $\tilde{\varphi}$, der Vektor $\boldsymbol{\sigma}$ mit den Oversampling-Faktoren σ_t, $(t = 0, ..., d-1)$, wie gehabt definiert und $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$, gegeben. Wir betrachten die Fourier-Reihe in (20), vernachlässigen den rechten Summanden und stellen fest

$$\begin{aligned}f^F(\boldsymbol{x}_j) &= \sum_{\boldsymbol{k}\in I^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}} \hat{g}^F_{\boldsymbol{k}}\, c_{\boldsymbol{k}}(\tilde{\varphi})\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j} \qquad (25)\\ &= \sum_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} \hat{g}^C_{\boldsymbol{k}}\, c_{\boldsymbol{k}}(\tilde{\varphi})\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\\ &\approx s_1(\boldsymbol{x}_j).\end{aligned}$$

Wir werden später noch auf die Approximationsfehler eingehen. Wählen wir in (7) die Koeffizienten $\hat{f}^F_{-\boldsymbol{k}} = \hat{f}^F_{\boldsymbol{k}}$, $\hat{f}^F_{-\boldsymbol{N}} = 0$, $(\hat{f}^F_{\boldsymbol{k}} \in \mathbb{R}, \boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$, und $\hat{f}^C_{\boldsymbol{k}} = 2\varepsilon_{\boldsymbol{N},\boldsymbol{k}}\hat{f}^F_{\boldsymbol{k}}$ ergibt sich die Identität

$$\begin{aligned}f^F(\boldsymbol{x}_j) &= \sum_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} 2\varepsilon_{\boldsymbol{N},\boldsymbol{k}}\hat{f}^F_{\boldsymbol{k}}\cos\big(2\pi(\boldsymbol{k}\odot\boldsymbol{x}_j)\big) \qquad (26)\\ &= \sum_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \hat{f}^C_{\boldsymbol{k}}\cos\big(2\pi(\boldsymbol{k}\odot\boldsymbol{x}_j)\big) \qquad (27)\\ &= f^C(\boldsymbol{x}_j).\end{aligned}$$

Durch einen Vergleich von (25), (26) und (27) ergibt sich für die Berechnung der $\hat{g}_{\boldsymbol{k}}^C$ die Beziehung

$$\hat{g}_{-\boldsymbol{k}}^C = \hat{g}_{\boldsymbol{k}}^C = \begin{cases} \frac{\hat{f}_{\boldsymbol{k}}^C}{2\varepsilon_{\boldsymbol{N},\boldsymbol{k}} c_{\boldsymbol{k}}(\tilde{\varphi})} & (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}) \\ 0 & (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\mathbf{1}_d,d} \setminus I_{\boldsymbol{0}}^{\boldsymbol{N},d}). \end{cases} \tag{28}$$

Es sind auch hier die Forderungen an die Eigenschaften der $c_{\boldsymbol{k}}(\tilde{\varphi})$ wichtig. Aufgrund der Symmetrien $\hat{f}_{\boldsymbol{k}}^C = \hat{f}_{-\boldsymbol{k}}^C$ und $c_{\boldsymbol{k}}(\tilde{\varphi}) = c_{-\boldsymbol{k}}(\tilde{\varphi})$, $(\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$ gilt offenbar auch die Symmetrie $\hat{g}_{\boldsymbol{k}}^C = \hat{g}_{-\boldsymbol{k}}^C$. Die Koeffizienten $g_{\boldsymbol{l}}^C$ werden anhand einer d-variaten, diskreten Kosinus-Transformation vom Typ-I der Länge $\boldsymbol{\sigma} \odot \boldsymbol{N} + \mathbf{1}_d$ bestimmt, die sich durch Umformen von Formel (21) ergibt. Wir hatten gezeigt, dass sich die $g_{\boldsymbol{l}}^F$,$(\boldsymbol{l} \in I_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d})$, durch die d-variate Fourier-Transformation

$$g_{\boldsymbol{l}}^F = 2^{-d}\Pi(\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1} \sum_{\boldsymbol{k} \in I_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} \hat{g}_{\boldsymbol{k}}^F \mathrm{e}^{\pi \mathrm{i} \boldsymbol{k}(\boldsymbol{l} \odot (\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1})}$$

berechnen lassen. Dies schreiben wir unter Verwendung der Symmetrie $\hat{g}_{\boldsymbol{k}}^F = \hat{g}_{-\boldsymbol{k}}^F$ zunächst noch ausführlicher durch

$$\begin{aligned} g_{\boldsymbol{l}}^F &= \frac{\Pi(\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}}{2^d} \sum_{\boldsymbol{k} \in I_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} \hat{g}_{\boldsymbol{k}}^F \mathrm{e}^{\pi \mathrm{i}\left(\frac{k_0 l_0}{\sigma_0 N_0} + \frac{k_1 l_1}{\sigma_1 N_1} + \cdots + \frac{k_{d-1} l_{d-1}}{\sigma_{d-1} N_{d-1}}\right)} \\ &= \frac{\Pi(\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}}{2^d} \sum_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\mathbf{1}_{\boldsymbol{d}},d}} \hat{g}_{\boldsymbol{k}}^C \left(\mathrm{e}^{\pi \mathrm{i}\left(\frac{k_0 l_0}{\sigma_0 N_0} + \frac{k_1 l_1}{\sigma_1 N_1} + \cdots + \frac{k_{d-1} l_{d-1}}{\sigma_{d-1} N_{d-1}}\right)} \mathrm{e}^{-\pi \mathrm{i}\left(\frac{k_0 l_0}{\sigma_0 N_0} + \frac{k_1 l_1}{\sigma_1 N_1} + \cdots + \frac{k_{d-1} l_{d-1}}{\sigma_{d-1} N_{d-1}}\right)} \right) \\ &= \frac{\Pi(\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}}{2^d} \sum_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\mathbf{1}_{\boldsymbol{d}},d}} \hat{g}_{\boldsymbol{k}}^C \prod_{t=0}^{d-1} \left(2\cos\left(\pi \frac{k_t l_t}{\sigma_t N_t}\right) \right). \end{aligned} \tag{29}$$

Diese Identität lässt sich durch die bekannte (eindimensionale) Äquivalenz $\mathrm{e}^{-ix} + e^{ix} = 2\cos(x)$ für $x \in \mathbb{R}$ ableiten. Wir hatten (29) eingangs als d-variate Kosinus-Transformation vom Typ-I der Länge $\boldsymbol{\sigma} \odot \boldsymbol{N} + \mathbf{1}_d$ mit der Vektor-Schreibweise

$$g_{\boldsymbol{l}}^C = \Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1} \sum_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\mathbf{1}_d,d}} \hat{g}_{\boldsymbol{k}}^C \cos\left(\pi(\boldsymbol{k}\odot\boldsymbol{l}) \odot (\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1})\right), \qquad (\boldsymbol{l} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\mathbf{1}_d,d}) \tag{30}$$

definiert. Sei $\tilde{\psi}$ wie in (23) und deren 1-periodisierte Version wieder durch $\tilde{\psi}(\boldsymbol{x}) := \sum_{\boldsymbol{r} \in \mathbb{Z}^d} \psi(\boldsymbol{x}+\boldsymbol{r})$ gegeben. Beachten wir die Periodizität $g_{\boldsymbol{l}}^C = g_{2\boldsymbol{\sigma}\odot\boldsymbol{N}\odot\boldsymbol{r}+\boldsymbol{l}}^C$ und die Symmetrie $g_{\boldsymbol{l}}^C = g_{2\boldsymbol{\sigma}\odot\boldsymbol{N}\odot\boldsymbol{r}-\boldsymbol{l}}^C$,$(\boldsymbol{r} \in \mathbb{Z}^d)$, der $g_{\boldsymbol{l}}^C$, so können wir schließlich für alle $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$ die Summe

$$f^C(\boldsymbol{x}_j) \approx s(\boldsymbol{x}_j) := \sum_{\boldsymbol{l} \in I_{\boldsymbol{0},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^C \tilde{\psi}\Big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot (\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1})\Big) \tag{31}$$

ausrechnen. Der NFCT-Algorithmus in Matrix-Vektor-Notation ist der des NFFT-Algorithmus sehr ähnlich. Wie wir gesehen haben, werden die $\hat{g}_{\boldsymbol{k}}^C$ anders ausgerechnet. Dazu führen wir die Matrix $\hat{\boldsymbol{D}} \in \mathbb{R}^{\Pi(\boldsymbol{N})\times\Pi(\boldsymbol{N})}$ ein. Die angenäherte Lösung von (9) ist dann

$$\boldsymbol{f}^C \approx \boldsymbol{B}(\boldsymbol{C}_{\boldsymbol{\sigma}\odot\boldsymbol{N},\boldsymbol{N}})\hat{\boldsymbol{D}}\boldsymbol{D}\hat{\boldsymbol{f}}^C,$$

wobei $\boldsymbol{D}$ die Diagonalmatrix

$$\boldsymbol{D} := \operatorname{diag}\Big(\frac{1}{c_{\boldsymbol{k}}(\tilde{\varphi})}\Big)_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \in \mathbb{R}^{\Pi(\boldsymbol{N})\times\Pi(\boldsymbol{N})},$$

$\hat{\boldsymbol{D}}$ die Diagonalmatrix

$$\hat{\boldsymbol{D}} := \operatorname{diag}\Big(\frac{1}{2\varepsilon_{\boldsymbol{N},\boldsymbol{k}}}\Big)_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \in \mathbb{R}^{\Pi(\boldsymbol{N})\times\Pi(\boldsymbol{N})},$$

und $\boldsymbol{C}^d_{\boldsymbol{\sigma}\odot\boldsymbol{N},\boldsymbol{N}} \in \mathbb{R}^{\Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})\times\Pi(\boldsymbol{N})}$ die abgeschnittene, d-dimensionale Kosinus-Matrix

$$\begin{aligned}\boldsymbol{C}^d_{\boldsymbol{\sigma}\odot\boldsymbol{N},\boldsymbol{N}} &:= \Big(\prod_{t=0}^{d-1}\frac{\varepsilon_{N_t,k_t}}{\sigma_t N_t}\cos\Big(\frac{\pi k_t l_t}{\sigma_t N_t}\Big)\Big)_{\boldsymbol{l}\in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d},\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}}\\ &= \Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-d}\big(\varepsilon_{\boldsymbol{N},\boldsymbol{k}}\cos\big(\pi((\boldsymbol{k}\odot\boldsymbol{l})\odot(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1})\big)\big)_{\boldsymbol{l}\in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d},\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}}\end{aligned}$$

darstellen. Schließlich ist $\boldsymbol{B}$ die Multilevel-Bandmatrix

$$\boldsymbol{B} := \Big(\tilde{\psi}\Big(\boldsymbol{x}_j - \frac{1}{2}\big(\boldsymbol{l}\odot(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1}\big)\Big)\Big)_{j\in I_0^{M,1},\boldsymbol{l}\in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} \in \mathbb{R}^{M\times\Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})}.$$

Um (11) zu berechnen, werden wir wie im NFFT$^{\mathrm{T}}$-Algorithmus gesehen das transponierte Matrix-Vektor-Produkt angeben. Da $\boldsymbol{D}$ und $\hat{\boldsymbol{D}}$ quadratische Diagonalmatrizen sind, ergibt sich der NFCT$^{\mathrm{T}}$-Algorithmus zu

$$(h^C(\boldsymbol{k}))_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \approx \boldsymbol{D}\hat{\boldsymbol{D}}(\boldsymbol{C}^d_{\boldsymbol{\sigma}\odot\boldsymbol{N},\boldsymbol{N}})^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}(f_j^C)_{j\in I_0^{M,1}}.$$

Der NFCT-Algorithmus

1. Eingaben:

   - $d, M, m \in \mathbb{N}$
   - $\boldsymbol{N} \in \mathbb{N}^d$
   - $\boldsymbol{\sigma} >_c \boldsymbol{1}_d$
   - $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$
   - $\hat{f}_{\boldsymbol{k}}^C \in \mathbb{R}$, $(\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$

2. Vorberechnungen:

   - $c_{\boldsymbol{k}}(\tilde{\varphi})$, $\forall \boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}$ und $c_{\boldsymbol{k}}(\tilde{\varphi}) \neq \boldsymbol{0}$
   - $\tilde{\varphi}\big(\boldsymbol{x}_j - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)\big)$, $\forall j \in I_0^{M,1}$ und $\forall \boldsymbol{l} \in I_{\boldsymbol{0},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}(\boldsymbol{x}_j)$

3. Berechne $\hat{g}_{\boldsymbol{k}}^C$:

$$\hat{g}_{\boldsymbol{k}}^C = \begin{cases} \frac{\hat{f}_{\boldsymbol{k}}^C}{2\varepsilon_{\boldsymbol{\sigma}\odot\boldsymbol{N},\boldsymbol{k}} c_{\boldsymbol{k}}(\tilde{\varphi})} & , (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}) \\ 0 & , (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\boldsymbol{1}_d,d} \setminus I_{\boldsymbol{0}}^{\boldsymbol{N},d}). \end{cases}$$

4. Berechne $g_{\boldsymbol{l}}^C$ durch eine d-variate DCT-I der Länge $\boldsymbol{\sigma} \odot \boldsymbol{N} + \boldsymbol{1}_d$:

$$g_{\boldsymbol{l}}^C = \Pi(\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1} \sum_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\boldsymbol{1}_d,d}} \hat{g}_{\boldsymbol{k}}^C \cos\Big(\pi(\boldsymbol{k} \odot \boldsymbol{l}) \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})\Big), \qquad (\boldsymbol{l} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\boldsymbol{1}_d,d})$$

5. Berechne $s(\boldsymbol{x}_j)$, $(j \in I_0^{M,1})$:

$$f^C(\boldsymbol{x}_j) \approx s(\boldsymbol{x}_j) := \sum_{\boldsymbol{l} \in I_{\boldsymbol{0},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^C \tilde{\psi}\Big(\boldsymbol{x}_j - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)\Big)$$

6. Ausgabe $s(\boldsymbol{x}_j)$, $(j \in I_0^{M,1})$:

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

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

Der NFCT$^{\mathrm{T}}$-Algorithmus

1. Eingaben:

   - $d, M, m \in \mathbb{N}$
   - $\boldsymbol{N} \in \mathbb{N}^d$
   - $\boldsymbol{\sigma} >_c \boldsymbol{1}_d$
   - $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$
   - $f_j^C \in \mathbb{R}$, $(\jmath \in I_0^{M,1})$

2. Vorberechnungen

   - $c_{\boldsymbol{k}}(\tilde{\varphi})$, $\forall \boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}$ und $c_{\boldsymbol{k}}(\tilde{\varphi}) \neq \boldsymbol{0}$

- $\tilde{\varphi}\big(\boldsymbol{x}_j - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)\big)$, $\forall \boldsymbol{l} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}$ und $\forall j \in \hat{I}_{\boldsymbol{0},m}^{\boldsymbol{M},d}(\boldsymbol{l}) := \{j \in I_0^{M,1} : \boldsymbol{l} - m\boldsymbol{1}_d \leq_c \boldsymbol{\sigma} \odot \boldsymbol{N} \odot \boldsymbol{x_j} \leq_c \boldsymbol{l} + m\boldsymbol{1}_d\}$, wobei die Ungleichungen auch hier komponentenweise gelten.

3. Berechne $g_{\boldsymbol{l}}^C$, $(\boldsymbol{l} \in I_{\boldsymbol{0},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\boldsymbol{1}_d,d}(\boldsymbol{x}_j))$:

$$g_{\boldsymbol{l}}^C := \sum_{j \in \hat{I}_{\boldsymbol{0},m}^{\boldsymbol{M},d}(\boldsymbol{l})} f_j^C \tilde{\psi}\Big(\boldsymbol{x}_j - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)\Big)$$

4. Berechne $\hat{g}_{\boldsymbol{k}}^C$ durch eine d-variate DCT-I der Länge $\boldsymbol{\sigma} \odot \boldsymbol{N} + \boldsymbol{1}_d$:

$$\hat{g}_{\boldsymbol{k}}^C = \Pi(\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1} \sum_{\boldsymbol{l} \in I_{\boldsymbol{0}}^{\boldsymbol{\sigma}\odot\boldsymbol{N}+\boldsymbol{1}_d,d}} g_{\boldsymbol{l}}^C \cos\Big(\pi(\boldsymbol{k} \odot \boldsymbol{l}) \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\Big), \qquad (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$$

5. Berechne $\tilde{h}^C(\boldsymbol{k})$, $(\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$:

$$\tilde{h}^C(\boldsymbol{k}) := \frac{\hat{g}_{\boldsymbol{k}}^C}{2\varepsilon_{\boldsymbol{\sigma}\odot\boldsymbol{N},\boldsymbol{k}} c_{\boldsymbol{k}}(\tilde{\varphi})}, \qquad (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$$

6. Ausgabe:

$$\tilde{h}^C(\boldsymbol{k}) \approx h^C(\boldsymbol{k}), \qquad (\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$$

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

3.3 Schnelle nicht-äquidistante Sinus-Transformation

In diesem Abschnitt werden wir ähnlich wie im Vorigen die Algorithmen NFST und NFST^{T} für die schnelle Auswertung der Summen (10) und (12) herleiten. Seien die Notationen und Symbole definiert wie in vorangegangenen Abschnitten und $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$, gegeben. Wählen wir diesmal $\hat{f}_{-\boldsymbol{k}}^F = -\hat{f}_{\boldsymbol{k}}^F$ und $\hat{f}_{\boldsymbol{0}_d}^F = \hat{f}_{-\boldsymbol{N}}^F = 0$, $(\hat{f}_{\boldsymbol{k}}^F \in \mathbb{R}, \boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d})$, so gilt mit $\hat{f}_{\boldsymbol{k}}^S = 2\hat{f}_{\boldsymbol{k}}^F$ und (25) für $j \in I_0^{M,1}$ die Identität

$$\begin{aligned} f^F(\boldsymbol{x}_j) &= \sum_{\boldsymbol{k}\in I_{-\boldsymbol{N}}^{\boldsymbol{N},d}} \hat{f}_{\boldsymbol{k}}^F \mathrm{e}^{2\pi \mathrm{i}\boldsymbol{k}\boldsymbol{x}_j} \\ &= \sum_{\boldsymbol{k}\in I_{-\boldsymbol{\sigma}\odot\boldsymbol{N}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} \hat{g}_{\boldsymbol{k}}^F c_{\boldsymbol{k}}(\tilde{\varphi}) \mathrm{e}^{2\pi \mathrm{i}\boldsymbol{k}\boldsymbol{x}_j} \\ &= \mathrm{i} \sum_{\boldsymbol{k}\in I_{\boldsymbol{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} 2\hat{g}_{\boldsymbol{k}}^S c_{\boldsymbol{k}}(\tilde{\varphi}) \sin(2\pi \boldsymbol{k}\odot\boldsymbol{x}_j). \qquad (32) \\ &= \mathrm{i} \sum_{\boldsymbol{k}\in I_{\boldsymbol{1}}^{\boldsymbol{N},d}} \hat{f}_{\boldsymbol{k}}^S \sin(2\pi \boldsymbol{k}\odot\boldsymbol{x}_j) \qquad (33) \\ &= \mathrm{i} f^S(\boldsymbol{x}_j). \end{aligned}$$

Vergleichen wir (32) und (33) so können wir wieder die Berechnung der $\hat{g}_{\boldsymbol{k}}^S$ ableiten. Analog zu vorangegangenen Abschnitten halten wir zur Bestimmung der Koeffizienten $\hat{g}_{\boldsymbol{k}}^S$ die Beziehung

$$\hat{g}_{\boldsymbol{k}}^S = -\hat{g}_{-\boldsymbol{k}}^S \begin{cases} \frac{\hat{f}_{\boldsymbol{k}}^S}{2c_{\boldsymbol{k}}(\tilde{\varphi})} & (\boldsymbol{k}\in I_{\boldsymbol{1}}^{\boldsymbol{N},d}) \\ 0 & (\boldsymbol{k}\in I_{\boldsymbol{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d} \setminus I_{\boldsymbol{1}}^{\boldsymbol{N},d}). \end{cases}$$

fest. Es ergibt sich durch umstellen von (21) und unter Beachtung der Symmetrie $\hat{g}_{\boldsymbol{k}}^S = -\hat{g}_{-\boldsymbol{k}}^S$ zur Berechnung der Koeffizienten $g_{\boldsymbol{l}}^S$ die Gleichung

$$g_{\boldsymbol{l}}^S = \Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1} \sum_{\boldsymbol{k}\in I_{\boldsymbol{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} \hat{g}_{\boldsymbol{k}}^S \sin\big(\pi\big((\boldsymbol{k}\odot\boldsymbol{l})\odot(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1}\big)\big), \qquad (\boldsymbol{l}\in I_{\boldsymbol{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}).$$

Dies ist eine d-dimensionale, diskrete Sinus-Transformation der Länge $\boldsymbol{\sigma}\odot\boldsymbol{N}-\boldsymbol{1}_d$. Unter Berücksichtigung der Eigenschaft $g_{\boldsymbol{l}}^S = -g_{2(\boldsymbol{\sigma}\odot\boldsymbol{N}\odot\boldsymbol{r})-\boldsymbol{l}}^S$, $(\boldsymbol{r}\in\mathbb{Z}^d)$ werten wir zuletzt die Summen

$$f^S(\boldsymbol{x}_j) \approx \tilde{f}^S = s(\boldsymbol{x}_j) := \sum_{\boldsymbol{l}\in I_{\boldsymbol{1},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^S \tilde{\psi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l}\odot(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1})\big) \qquad (34)$$

für alle $j\in I_0^{M,1}$ aus, wobei $\tilde{\psi}(\boldsymbol{x})$ wieder die 1-periodisierte Version der Abschneidefunktion ψ ist. Analog zum NFCT-Algorithmus schreiben wir die Matrix-Vektor-Darstellung als

$$\boldsymbol{f}^S \approx \tilde{\boldsymbol{f}}^S = \boldsymbol{B}(\boldsymbol{S}^d_{(\boldsymbol{\sigma}\odot\boldsymbol{N}-\boldsymbol{1}_d),(\boldsymbol{N}-\boldsymbol{1}_d)})\hat{\boldsymbol{D}}\boldsymbol{D}\hat{\boldsymbol{f}}^S.$$

Hier ist Matrix $\boldsymbol{D}\in\mathbb{R}^{\Pi(\boldsymbol{N}-\boldsymbol{1}_d)-1\times\Pi(\boldsymbol{N}-\boldsymbol{1}_d)-1}$ analog wie zuvor bei der NFCT eingeführt, $\boldsymbol{S}^d_{(\boldsymbol{\sigma}\odot\boldsymbol{N}-\boldsymbol{1}_d),(\boldsymbol{N}-\boldsymbol{1}_d)}$ die abgeschnittene Sinus-Matrix

$$\begin{aligned} \boldsymbol{S}^d_{(\boldsymbol{\sigma}\odot\boldsymbol{N}-\mathbf{1}_d),(\boldsymbol{N}-\mathbf{1}_d)} &:= \Big(\prod_{t=0}^{d-1}\frac{2}{\sigma_t N_t}\sin\Big(\frac{\pi k_t l_t}{\sigma_t N_t}\Big)\Big)_{\boldsymbol{l}\in I_{\mathbf{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d},\boldsymbol{k}\in I_{\mathbf{1}}^{\boldsymbol{N},d}} \\ &= 2^d\Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1}\big(\sin\big(\pi((\boldsymbol{k}\odot\boldsymbol{l})\odot(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1})\big)\big)_{\boldsymbol{l}\in I_{\mathbf{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d},\boldsymbol{k}\in I_{\mathbf{1}}^{\boldsymbol{N},d}} \end{aligned}$$

der Dimension $\Pi(\boldsymbol{\sigma}\odot\boldsymbol{N}-\mathbf{1}_d)\times\Pi(\boldsymbol{N}-\mathbf{1}_d)$ und $\hat{\boldsymbol{D}} := \operatorname{diag}(\frac{1}{2}) \in \mathbb{R}^{\Pi(\boldsymbol{N}-\mathbf{1}_d)\times\Pi(\boldsymbol{N}-\mathbf{1}_d)}$. Der Ausdruck

$$(h^S(\boldsymbol{k}))_{\boldsymbol{k}\in I_{\mathbf{1}}^{\boldsymbol{N},d}} \approx \tilde{h}^S(\boldsymbol{k}))_{\boldsymbol{k}\in I_{\mathbf{1}}^{\boldsymbol{N},d}} := \boldsymbol{D}\hat{\boldsymbol{D}}(\boldsymbol{S}^d_{(\boldsymbol{\sigma}\odot\boldsymbol{N}-\mathbf{1}_d),(\boldsymbol{N}-\mathbf{1}_d)})^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}(f_j^S)_{j\in I_0^{M,1}}$$

approximiert somit die Summe (12). Im Folgenden fassen wir die Algorithmen nochmals in einer Übersicht zusammen.

Der NFST-Algorithmus

1. Eingaben:
   - $d, M, m \in \mathbb{N}$
   - $\boldsymbol{N} \in \mathbb{N}^d$
   - $\boldsymbol{\sigma} >_c \mathbf{1}_d$
   - $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$
   - $\hat{f}_{\boldsymbol{k}}^S \in \mathbb{R}$, $(\boldsymbol{k} \in I_{\mathbf{1}}^{\boldsymbol{N},d})$
2. Vorberechnungen:
   - $c_{\boldsymbol{k}}(\tilde{\varphi})$, $\forall \boldsymbol{k} \in I_{\mathbf{1}}^{\boldsymbol{N},d}$ und $c_{\boldsymbol{k}}(\tilde{\varphi}) \neq \mathbf{0}_d$
   - $\tilde{\varphi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l}\odot(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1})\big)$, $\forall j \in I_0^{M,1}$ und $\forall \boldsymbol{l} \in I_{\mathbf{1},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}(\boldsymbol{x}_j)$.
3. Berechne $\hat{g}_{\boldsymbol{k}}^S$:

$$\hat{g}_{\boldsymbol{k}}^S = -\hat{g}_{-\boldsymbol{k}}^S \begin{cases} \frac{\hat{f}_{\boldsymbol{k}}^S}{2c_{\boldsymbol{k}}(\tilde{\varphi})} & (\boldsymbol{k}\in I_{\mathbf{1}}^{\boldsymbol{N},d}) \\ 0 & (\boldsymbol{k}\in I_{\mathbf{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}\setminus I_{\mathbf{1}}^{\boldsymbol{N},d}). \end{cases}$$

4. Berechne $g_{\boldsymbol{l}}^S$ unter Verwendung einer d-variaten DST-I der Länge $\boldsymbol{\sigma}\odot\boldsymbol{N}-\mathbf{1}_d$:

$$g_{\boldsymbol{l}}^S = \Pi(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1}\sum_{\boldsymbol{k}\in I_{\mathbf{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}}\hat{g}_{\boldsymbol{k}}^S\sin\Big(\pi(\boldsymbol{k}\odot\boldsymbol{l})\odot(\boldsymbol{\sigma}\odot\boldsymbol{N})^{-1}\Big), \qquad (\boldsymbol{l}\in I_{\mathbf{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}).$$

5. Berechne $s(\boldsymbol{x}_j)$, $(j \in I_0^{M,1})$:

$$f^S(\boldsymbol{x}_j) \approx s(\boldsymbol{x}_j) := \sum_{\boldsymbol{l} \in I_{\boldsymbol{1},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^S \tilde{\psi}\Big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})\Big).$$

6. Ausgabe $s(\boldsymbol{x}_j)$, $(j \in I_0^{M,1})$:

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

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

Der NFST$^{\mathrm{T}}$-Algorithmus

1. Eingaben:

   - $d, M, m \in \mathbb{N}$
   - $\boldsymbol{N} \in \mathbb{N}^d$
   - $\boldsymbol{\sigma} >_c \boldsymbol{1}_d$
   - $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$
   - $f_{\boldsymbol{k}}^S \in \mathbb{R}$, $(\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d})$

2. Vorberechnungen:

   - $c_{\boldsymbol{k}}(\tilde{\varphi})$, $\forall \boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d}$ und $c_{\boldsymbol{k}}(\tilde{\varphi}) \neq \boldsymbol{0}$
   - $\tilde{\varphi}\big(\boldsymbol{x}_j - \frac{1}{2}\big(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1}\big)\big)$, $\forall \boldsymbol{l} \in I_{\boldsymbol{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}$ und
     $\forall j \in \hat{I}_{\boldsymbol{0},m}^{\boldsymbol{M},d}(\boldsymbol{l}) := \{j \in I_0^{M,1} : \boldsymbol{l} - m\boldsymbol{1}_d \leq_c \boldsymbol{\sigma} \odot \boldsymbol{N} \odot \boldsymbol{x_j} \leq_c \boldsymbol{l} + m\boldsymbol{1}_d\}$.

3. Berechne $g_{\boldsymbol{l}}^S$, $(\boldsymbol{l} \in I_{\boldsymbol{1},m}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}(\boldsymbol{x}_j))$:

$$g_{\boldsymbol{l}}^S := \sum_{j \in \hat{I}_{\boldsymbol{0},m}^{\boldsymbol{M},d}(\boldsymbol{l})} f_j^S \tilde{\psi}\Big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})\Big)$$

4. Berechne die $\hat{g}_{\boldsymbol{k}}^S$ mittels einer d-variaten DST-I der Länge $\boldsymbol{\sigma} \odot \boldsymbol{N} - \boldsymbol{1}_d$:

$$\hat{g}_{\boldsymbol{k}}^S = \Pi(\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1} \sum_{\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}} g_{\boldsymbol{l}}^S \sin\Big(\pi(\boldsymbol{k} \odot \boldsymbol{l}) \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})\Big), \qquad (\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d})$$

5. Berechne $\tilde{h}^S(\boldsymbol{k})$, $(\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d})$:

$$\tilde{h}^S(\boldsymbol{k}) := \frac{\hat{g}_{\boldsymbol{k}}^S}{2c_{\boldsymbol{k}}(\tilde{\varphi})}$$

6. Ausgabe:

$$\tilde{h}^S(\boldsymbol{k}) \approx h^S(\boldsymbol{k}), \qquad (\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d})$$

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

3.4 Inverse nicht-äquidistante trigonometrische Transformationen

In der Praxis ist es häufig der Fall, dass beispielsweise nach Messungen eine endliche Menge an Daten vorliegt. Zu diesen Werten ist es dann von Interesse deren Fourier-Koeffizienten zu kennen. Da im Allgemeinen keine triviale Inversion existiert, versuchen wir ein trigonometrisches Polynom, das die aufgenommenen Daten optimal approximiert, durch iterative Lösungsverfahren zu rekonstruieren. Das bedeutet in unserer Notation, die Koeffizienten $\hat{f}_{\boldsymbol{k}}^C$, $(\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d})$, bzw. $\hat{f}_{\boldsymbol{k}}^S$, $(\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d})$, zu vorgegebenen Daten y_j^C, y_j^S, $(j \in I_0^{M,1})$, an nicht-äquidistanten Knoten $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $(j \in I_0^{M,1})$, derart zu bestimmen, dass

$$f^C(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \hat{f}_{\boldsymbol{k}}^C \cos(2\pi(\boldsymbol{k} \odot \boldsymbol{x}_j)) \approx y_j^C, \tag{35}$$

beziehungsweise

$$f^S(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in I_{\boldsymbol{1}}^{\boldsymbol{N},d}} \hat{f}_{\boldsymbol{k}}^S \sin(2\pi(\boldsymbol{k} \odot \boldsymbol{x}_j)) \approx y_j^S \tag{36}$$

gilt. Wir nennen dieses Problem inverse diskrete Kosinus- bzw. Sinus-Transformation (iNDCT, iNDST). Mit Hilfe der Ergebnisse vorangegangener Abschnitte werden nun schnelle iterative Algorithmen (iNFCT, iNFST) zur Berechnung dieser inversen Transformationen entwickelt. In diesem Abschnitt unterscheiden wir nicht zwischen Kosinus- und Sinus-Transformation, da sich keine wesentlichen Unterschiede ergeben. Aus diesem Grund beziehen wir uns mit y_j sowohl auf y_j^C als auch y_j^S. Ebenso beschreibt die Matrix $\boldsymbol{A}$ je nach Kontext die nicht-äquidistante Kosinus-Matrix $\boldsymbol{A}_C$ oder die nicht-äquidistante Sinus-Matrix $\boldsymbol{A}_S$. Demzufolge lässt sich für die beiden obigen Gleichungen (35) und (36) vereinfacht

$$\boldsymbol{A}\hat{\boldsymbol{f}} \approx \boldsymbol{y} \tag{37}$$

schreiben wobei $\hat{\boldsymbol{f}} := (\hat{f}_{\boldsymbol{k}})_{\boldsymbol{k} \in I_{\boldsymbol{a}}^{\boldsymbol{N},d}}$, $(a = 0, 1)$, und $\boldsymbol{y} := (y_j)_{j \in I_0^{M,1}}$. Wir werden im folgenden den Vektor $\hat{\boldsymbol{f}}$ durch die Kleinste-Quadrate-Methode bestimmen, also das Minimierungsproblem

$$||\boldsymbol{y} - A\hat{\boldsymbol{f}}||_2 \xrightarrow{\hat{\boldsymbol{f}}} \min$$

lösen. Für $\Pi(\boldsymbol{N}) < M$ ist das lineare Gleichungssystem $\boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{y}$ überbestimmt. Somit können die y_j, $(j \in I_0^{M,1})$, im Allgemeinen nur angenähert werden. Um einer ungleichmäßigen Verteilung der Stützstellen im Intervall $[0, \frac{1}{2}]^d$ entgegenzuwirken, betrachten wir das gewichtete Minimierungsproblem

$$||\boldsymbol{W}^{\frac{1}{2}}(\boldsymbol{y} - A\hat{\boldsymbol{f}})||_2 = \Big(\sum_{j=0}^{M-1} w_j |y_j - f(\boldsymbol{x}_j)|^2 \Big)^{\frac{1}{2}} \xrightarrow{\hat{\boldsymbol{f}}} \min \tag{38}$$

Die Koeffizienten $w_j > 0$, $(j \in I_0^{M,1})$, sind dabei die Gewichte. Definieren wir die Matrix $\boldsymbol{W} \in \mathbb{R}^{M \times M}$ als Diagonalmatrix mit den Einträgen w_j, $(j \in I_0^{M,1})$, auf der Hauptdiagonale so ist mit [2, Seiten 5-6] der Vektor $\hat{\boldsymbol{f}}$ genau dann ein Minimierer von 38, wenn er die gewichtete Normalengleichung erster Art

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{y}$$

erfüllt. Die Methode der Gewichtung soll verhindern, dass Gebiete in denen verhältnismäßig viele Stützstellen sehr dicht beieinander liegen stärker berücksichtigt werden als weniger repräsentierte Gebiete.
Anstatt nur die Norm $||\boldsymbol{W}^{\frac{1}{2}}(\boldsymbol{y} - A\hat{\boldsymbol{f}})||_2$ zu minimieren, können wir zusätzlich fordern, dass die Norm der Koeffizienten $||\hat{\boldsymbol{f}}||_2$ minimal wird. Sei $\lambda > 0$ ein Regularisierungsparameter. Wir betrachten das gewichtete Minimierungsproblem mit Regularisierung

$$||\boldsymbol{W}^{\frac{1}{2}}(\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{f}})||_2^2 + \lambda||\hat{\boldsymbol{f}}||_2^2 \xrightarrow{\hat{\boldsymbol{f}}} \min. \tag{39}$$

Durch das Quadrieren der Summanden in (39) kann das Minimierungsproblems auf das Lösen eines linearen Gleichungssystem zurückgeführt werden. Sei $\boldsymbol{E} := \mathrm{diag}(\boldsymbol{1}_{\Pi(\boldsymbol{N})})$ eine Einheitsmatrix, dann ist mit [9, Lemma 2.2] der Vektor $\hat{\boldsymbol{f}}$ genau dann eine Lösung von (39) wenn er der Gleichung

$$(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{A} + \lambda\boldsymbol{E})\hat{\boldsymbol{f}} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{y}$$

genügt.

Ist dagegen der Polynomgrad relativ hoch, d.h. $\Pi(\boldsymbol{N})$ ist größer als M, dann ist $\boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{y}$ unterbestimmt und wir können die gegebenen Daten y_j, $(j \in I_0^{M,1})$ exakt interpolieren. Seien $\hat{w}_{\boldsymbol{k}} > 0$ Gewichtungsfaktoren und $\hat{\boldsymbol{W}} := \mathrm{diag}(\hat{w}_{\boldsymbol{k}})_{\boldsymbol{k} \in I_a^{\boldsymbol{N},d}}$, $(a \in \{0, 1\})$, um den Abfall der diskreten Fourier-Koeffizienten $\hat{\boldsymbol{f}}$ zu steuern. Wir betrachten das gedämpfte Minimierungsproblem

$$||\hat{\boldsymbol{W}}^{-\frac{1}{2}}\hat{\boldsymbol{f}}||_2 = \Big(\sum_{\boldsymbol{k} \in I_a^{\boldsymbol{N},d}} \hat{w}_{\boldsymbol{k}}^{-1} |\hat{f}_{\boldsymbol{k}}|^2 \Big)^{\frac{1}{2}} \xrightarrow{\hat{\boldsymbol{f}}} \min$$

mit der Forderung $\boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{y}$. Dieses Problem ist auch bekannt als gedämpfte Normalengleichung zweiter Art

$$\boldsymbol{A}\hat{\boldsymbol{W}}\boldsymbol{A}^{\mathrm{T}}\tilde{\boldsymbol{f}} = \boldsymbol{y}, \qquad \hat{\boldsymbol{f}} = \hat{\boldsymbol{W}}\boldsymbol{A}^{\mathrm{T}}\tilde{\boldsymbol{f}}. \tag{40}$$

Es existieren mehrere Verfahren um die Minimierungsprobleme (38), (39) und (40) zu lösen. Dazu gehören unter anderem die iterativen Verfahren CGNE, CGNR, Landweber und die Methode des steilsten Abstiegs. In [11, Seiten 10-12] sind diese Verfahren in Pseudo-Code zu finden. Die Matrix-Vektor-Multiplikationen in diesen Verfahren werden mit Hilfe der NFCT beziehungsweise der NFST durchgeführt. Algorithmen, die auf diese Weise (35) respektive (36) lösen, bezeichnen wir als inverse NFCT (iNFCT) respektive inverse NFST (iNFST). In Abschnitt 6.3 betrachten wir anhand eines numerischen Beispiels den Verlauf des Approximationsfehlers in Abhängigkeit der Iterationsschritte.

4 Fehlerabschätzung

Da die NFFT, NFCT und NFST approximative Algorithmen sind, untersuchen wir den Zusammenhang zwischen Exaktheit und arithmetischer Komplexität der Algorithmen. Zunächst betrachten wir eine allgemeine Fehlerabschätzung und führen danach ein Beispiel für eine konkrete Fensterfunktion φ mit B-Splines an. Diese Abschätzungen werden exemplarisch für den Spezialfall $d = 1$ (univariat) durchgeführt. Im folgenden wird darauf verzichtet beide Algorithmen zu untersuchen. Stattdessen wird die NFCT als Beispiel dienen und darauf verwiesen, dass die Fehlerabschätzung für die NFST analog verläuft. Da die Algorithmen $\mathrm{NFCT}^{\mathrm{T}}$ und $\mathrm{NFST}^{\mathrm{T}}$ die selben Approximationsfehler liefern wird auf sie an dieser Stelle nicht weiter eingegangen.

4.1 Allgemeine Approximationsfehler

Die folgenden Fehlerabschätzungen sind aus [12, Seiten 19-21] übernommen, jedoch unseren Notationsvereinbarungen angepaßt. In diesem Abschnitt betrachten wir keine konkrete Fensterfunktion, somit können diese Abschätzungen als Ansatz für spezielle Fehlerbetrachtungen dienen.
Offenbar ergibt sich der Approximationsfehler aus der Differenz von (9) und (31)

$$E(\boldsymbol{x}_j) := |f^C(\boldsymbol{x}_j) - s(\boldsymbol{x}_j)|.$$

Nach Zerlegung in *Abschneidefehler* (truncation error)

$$E_t(\boldsymbol{x}_j) := |s_1(\boldsymbol{x}_j) - s(\boldsymbol{x}_j)|$$

und *Überlappungsfehler* (aliasing error)

$$E_a(\boldsymbol{x}_j) := |f^C(\boldsymbol{x}_j) - s_1(\boldsymbol{x}_j)|$$

gilt somit $E(\boldsymbol{x}_j) \leq |E_t(\boldsymbol{x}_j)| + |E_a(\boldsymbol{x}_j)|$. Wir definieren den maximalen Approximationsfehler $E_\infty(\boldsymbol{x}_j)$ als Maximum über alle $\boldsymbol{x}_j$, $(j \in I_0^{M,1})$ von $E(\boldsymbol{x}_j)$ durch die Vorschrift

$$E_\infty := \max_{j \in I_0^{M,1}} E(\boldsymbol{x}_j).$$

Durch das Abschneiden von φ im Zeitbereich kommt es zum Abschneidefehler E_t. Dieser ist gegeben als Betrag der Differenz von (18) und (31). Wir schätzen ihn durch die Ungleichung

$$E_t(\boldsymbol{x}_j) \leq \frac{||\hat{\boldsymbol{f}}^C||_1}{\Pi(\boldsymbol{n})} \max_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{1}{\left|\hat{\varphi}(\boldsymbol{k})\right|} \sum_{\boldsymbol{x}_j + \frac{1}{2}\boldsymbol{r} \odot \boldsymbol{n}^{-1} \geq_c \frac{m}{2}\boldsymbol{n}^{-1}} \left|\varphi\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{r} \odot \boldsymbol{n}^{-1})\big)\right| \qquad (41)$$

ab, wobei $\boldsymbol{n}$ zugunsten der Übersichtlichkeit in diesem Abschnitt als $\boldsymbol{\sigma} \odot \boldsymbol{N}$ definiert ist.

Beweis: Mit den Formeln (18) und (24) ergibt sich mit der Definition von E_t die Beziehung

$$E_t(\boldsymbol{x}_j) = \left| \sum_{\boldsymbol{l} \in I_{-\boldsymbol{n}}^{\boldsymbol{n},d}} g_{\boldsymbol{l}}^C \tilde{\varphi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot \boldsymbol{n}^{-1})\big) - \sum_{\boldsymbol{l} \in I_{-\boldsymbol{n},m}^{\boldsymbol{n},d}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^C \tilde{\psi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot \boldsymbol{n}^{-1})\big) \right|.$$

Beachten wir die Symmetrien $g_{\boldsymbol{l}}^C = g_{-\boldsymbol{l}}^C$ und der 1-periodisierten Funktionen $\tilde{\varphi}$ und $\tilde{\psi}$ so kann E_t auch als

$$E_t(\boldsymbol{x}_j) = 2^d \left| \sum_{\boldsymbol{l} \in I_{\boldsymbol{0}}^{\boldsymbol{n},d}} g_{\boldsymbol{l}}^C \tilde{\varphi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot \boldsymbol{n}^{-1})\big) - \sum_{\boldsymbol{l} \in I_{\boldsymbol{0},m}^{\boldsymbol{n},d}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^C \tilde{\psi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot \boldsymbol{n}^{-1})\big) \right|.$$

geschrieben werden. Unter Ausnutzung der Berechnungsvorschrift der $\hat{g}_{\boldsymbol{k}}^C$ in (28), der $g_{\boldsymbol{l}}^C$ in (30) sowie der daraus folgenden Beziehung

$$g_{\boldsymbol{l}}^C = \Pi(\boldsymbol{n})^{-1} \sum_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{n},d}} \frac{\hat{f}_{\boldsymbol{k}}^C}{2\varepsilon_{\boldsymbol{n},\boldsymbol{k}}\hat{\varphi}(\boldsymbol{k})} \cos\big(\pi(\boldsymbol{k} \odot \boldsymbol{l}) \odot \boldsymbol{n}^{-1}\big), \qquad (\boldsymbol{l} \in I_{\boldsymbol{0}}^{\boldsymbol{n},d})$$

ergibt sich mit $\zeta(\boldsymbol{y}) := \tilde{\varphi}\big(\boldsymbol{y}\big) - \tilde{\psi}\big(\boldsymbol{y}\big)$ die Identität

$$E_t(\boldsymbol{x}_j) = \frac{1}{\Pi(\boldsymbol{n})} \left| \sum_{\boldsymbol{l} \in I_{\boldsymbol{0}}^{\boldsymbol{n},d}} \sum_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{\hat{f}_{\boldsymbol{k}}^C}{\varepsilon_{\boldsymbol{n},\boldsymbol{k}}\hat{\varphi}(\boldsymbol{k})} \cos\big(\pi((\boldsymbol{k} \odot \boldsymbol{l}) \odot \boldsymbol{n}^{-1})\big)\zeta(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot \boldsymbol{n}^{-1})) \right|.$$

Wir schätzen E_t durch

$$
\begin{aligned}
E_t(\boldsymbol{x}_j) &\leq \frac{1}{\Pi(\boldsymbol{n})}\left|\sum_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{\hat{f}_{\boldsymbol{k}}^C}{\hat{\varphi}(\boldsymbol{k})} \sum_{\boldsymbol{l}\in I_{\boldsymbol{0}}^{\boldsymbol{n},d}} \zeta\big(\boldsymbol{x}_j - \frac{1}{2}\boldsymbol{l}\odot\boldsymbol{n}^{-1}\big)\cos\big(\pi(\boldsymbol{k}\odot\boldsymbol{l})\odot\boldsymbol{n}^{-1}\big)\right| \\
&\leq \frac{||\hat{\boldsymbol{f}}^C||_1}{\Pi(\boldsymbol{n})} \max_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|}\left|\sum_{\boldsymbol{l}\in I_{\boldsymbol{0}}^{\boldsymbol{n},d}} \zeta(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l}\odot\boldsymbol{n}^{-1}))\cos\big(\pi(\boldsymbol{k}\odot\boldsymbol{l})\odot\boldsymbol{n}^{-1}\big)\right|
\end{aligned}
$$

ab und formen die verbleibende Summe um. Mit Hilfe der Definitionen von $\tilde{\varphi}$ (13), ψ (23) und deren 1-periodisierten Version $\tilde{\psi}$ können wir E_t zu

$$
\begin{aligned}
E_t(\boldsymbol{x}_j) &\leq \frac{||\hat{\boldsymbol{f}}^C||_1}{\Pi(\boldsymbol{n})} \max_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|}\left|\sum_{\boldsymbol{l}\in I_{\boldsymbol{0}}^{\boldsymbol{n},d}} \Big(\sum_{\boldsymbol{r}\in\mathbb{Z}^d} \zeta(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l}\odot\boldsymbol{n}^{-1}) + \boldsymbol{r})\Big)\cos\big(\pi(\boldsymbol{k}\odot\boldsymbol{l})\odot\boldsymbol{n}^{-1}\big)\right| \\
&= \frac{||\hat{\boldsymbol{f}}^C||_1}{\Pi(\boldsymbol{n})} \max_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|}\left|\sum_{\boldsymbol{r}\in\mathbb{Z}^d} \zeta(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{r}\odot\boldsymbol{n}^{-1}))\cos\big(\pi(\boldsymbol{k}\odot\boldsymbol{l})\odot\boldsymbol{n}^{-1}\big)\right| \\
&= \frac{||\hat{\boldsymbol{f}}^C||_1}{\Pi(\boldsymbol{n})} \max_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|}\left|\sum_{\boldsymbol{x}_j+\frac{1}{2}\boldsymbol{r}\odot\boldsymbol{n}^{-1}\geq_c \frac{m}{2}\boldsymbol{n}^{-1}} \varphi\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{r}\odot\boldsymbol{n}^{-1})\big)\cos\big(\pi(\boldsymbol{k}\odot\boldsymbol{l})\odot\boldsymbol{n}^{-1}\big)\right|
\end{aligned}
$$

vereinfachen. Unter Berücksichtigung der Minkowski-Ungleichung und unserer Definition der Summennorm können wir noch weiter umformen. Letztlich schätzen wir den Kosinusterm mit 1 ab und erhalten

$$
\begin{aligned}
E_t(\boldsymbol{x}_j) &\leq \frac{||\hat{\boldsymbol{f}}^C||_1}{\Pi(\boldsymbol{n})} \max_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|} \sum_{\boldsymbol{x}_j+\frac{1}{2}\boldsymbol{r}\odot\boldsymbol{n}^{-1}\geq_c \frac{m}{2}\boldsymbol{n}^{-1}} \left|\varphi\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{r}\odot\boldsymbol{n}^{-1})\big)\cos\big(\pi(\boldsymbol{k}\odot\boldsymbol{l})\odot\boldsymbol{n}^{-1}\big)\right| \\
&\leq \frac{||\hat{\boldsymbol{f}}^C||_1}{\Pi(\boldsymbol{n})} \max_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \frac{1}{|\hat{\varphi}(\boldsymbol{k})|} \sum_{\boldsymbol{x}_j+\frac{1}{2}\boldsymbol{r}\odot\boldsymbol{n}^{-1}\geq_c \frac{m}{2}\boldsymbol{n}^{-1}} \left|\varphi\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{r}\odot\boldsymbol{n}^{-1})\big)\right|.
\end{aligned}
$$

□

Wir wollen auch eine Abschätzung für den Überlappungsfehler angeben. Dieser Fehler entsteht als Folge des Abschneidens im Frequenzbereich. In dieser Arbeit geben wir ihn durch die Ungleichung

$$
E_a(\boldsymbol{x}_j) \leq \big|\big|\hat{\boldsymbol{f}}^C\big|\big|_1 \max_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{\boldsymbol{0}\}} \left|\frac{\hat{\varphi}(\boldsymbol{k}+2(\boldsymbol{n}\odot\boldsymbol{r}))}{\hat{\varphi}(\boldsymbol{k})}\right| \tag{42}
$$

an. Der folgende Beweis zeigt wie wir zu dieser Abschätzungen gelangen.

Beweis: Es kommt durch den Zusammenhang zwischen (7), (27), der Abschätzung in (19) und der Definition von E_a zu dem Überlappungsfehler

$$E_a(\boldsymbol{x}_j) = \left| \sum_{\boldsymbol{k}\in I_{-\boldsymbol{n}}^{\boldsymbol{n},d}} \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{\boldsymbol{0}\}} \hat{g}_{\boldsymbol{k}}^C \hat{\varphi}(\boldsymbol{k}+2(\boldsymbol{n}\odot\boldsymbol{r}))\mathrm{e}^{2\pi\mathrm{i}(\boldsymbol{k}+2(\boldsymbol{n}\odot\boldsymbol{r}))\boldsymbol{x}_j} \right|,$$

den wir aufgrund der Beziehung (28) und der Symmetrie $\hat{g}_{\boldsymbol{k}}^C = \hat{g}_{-\boldsymbol{k}}^C$, ($\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{n},d}$ wie gewohnt zu

$$E_a(\boldsymbol{x}_j) = \left| \sum_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{n},d}} \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{\boldsymbol{0}\}} 2\varepsilon_{\boldsymbol{n},\boldsymbol{k}} \hat{g}_{\boldsymbol{k}}^C \hat{\varphi}(\boldsymbol{k}+2(\boldsymbol{n}\odot\boldsymbol{r}))\cos(2\pi(\boldsymbol{k}+2(\boldsymbol{n}\odot\boldsymbol{r}))\boldsymbol{x}_j) \right|$$

umformen können. Wir verwenden wieder die Minkowski-Ungleichung und beachten die Beziehung zwischen den $\hat{f}_{\boldsymbol{k}}$, $\hat{g}_{\boldsymbol{k}}^C$ und den $\hat{\varphi}(\boldsymbol{k}) = c_{\boldsymbol{k}}(\tilde{\varphi})$. Dann sind wir in der Lage E_a durch

$$\begin{aligned} E_a(\boldsymbol{x}_j) &\leq \sum_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \left|\hat{f}_{\boldsymbol{k}}^C\right| \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{\boldsymbol{0}\}} \left| \frac{\hat{\varphi}(\boldsymbol{k}+2(\boldsymbol{n}\odot\boldsymbol{r}))}{\hat{\varphi}(\boldsymbol{k})} \right| \\ &\leq \left|\left|\hat{\boldsymbol{f}}^C\right|\right|_1 \max_{\boldsymbol{k}\in I_{\boldsymbol{0}}^{\boldsymbol{N},d}} \sum_{\boldsymbol{r}\in\mathbb{Z}^d\setminus\{\boldsymbol{0}\}} \left| \frac{\hat{\varphi}(\boldsymbol{k}+2(\boldsymbol{n}\odot\boldsymbol{r}))}{\hat{\varphi}(\boldsymbol{k})} \right| \end{aligned}$$

nach oben abzuschätzen.

□

Fensterfunktionen

Es wurden unter anderen in [12] vier Fensterfunktionen für den NFFT-Algorithmus vorgeschlagen und implementiert. Diese wurden ohne weitere Änderung in den NFCT- und NFST-Algorithmus übernommen. Damit führen wir konsequent den modularen Aufbau weiter. Einmal programmiert, können diese Fensterfunktionen in allen Algorithmen (NFFT, NFCT und NFST) der Software-Bibliothek verwendet werden. Zu diesen Funktionen gehören *B*-Splines [1, 14]), Gauß- [4, 14, 13], Kaiser-Bessel- [6] und die Sinc-Funktionen [12]. Die Referenzen geben jeweils Quellen an, in denen die Funktionen in Zusammenhang mit der NFFT näher untersucht wurden. Die Wahl der Fensterfunktion bestimmt nicht zuletzt maßgeblich die Genauigkeit des Ergebnisses (Rundungsfehler, Abklingverhalten) und die Ausführungsgeschwindigkeit des Algorithmus (Auswertungszeit der Fensterfunktion).
Es sei hier darauf hingewiesen, dass die verschiedenen Fensterfunktionen eine Auswahl bilden, die wie folgt eingeteilt werden kann: keinen Abschneidefehler E_t (*B*-Splines), keinen Überlappungsfehler E_a (Sinc- und Kaiser-Bessel-Funktionen). Dies wird durch kompakte Träger im Zeitbereich bzw. im Frequenzbereich erreicht. Parametrisierte Funktionen (Gauß-Funktionen) erlauben die Beeinflussung beider Fehler, wobei diese jedoch einander bedingen. Exemplarisch wird im anschließenden Abschnitt auf die Approximationsfehler mit *B*-Splines als Fensterfunktion eingegangen.

4.2 Fehlerabschätzung am Beispiel von B-Splines als Fensterfunktion φ

Wie in [1] erstmals untersucht, gibt es Funktionen bei denen der eingeführte Fehler E_t verschwindet. Aufgrund ihres beschränkten Trägers sind B-Splines Funktionen dieser Klasse.
Sei $0 \leq m \leq N$ eine positive ganze Zahl, so definieren wir die zentrierten B-Splines mittels der Rekursionsgleichung:

$$\begin{aligned} M_1(x) &:= \chi_{[-\frac{1}{2},\frac{1}{2}]}(x) \\ M_{m+1}(x) &:= \int_{-\frac{1}{2}}^{\frac{1}{2}} M_m(x-t)\mathrm{d}t, \qquad (m = 1, 2, ...). \end{aligned}$$

Daraus ergeben sich einige Eigenschaften, die für die weiteren Betrachtungen von Interesse sind. Der Träger von M_{2m} ist das Intervall $[-m, m]$ und die Fourier-Transformierte $\hat{M}_1$ von M_1 wird unter Verwendung der Sinc-Funktion

$$\mathrm{sinc}(x) = \begin{cases} 1 & , (x = 0) \\ \frac{\sin(x)}{x} & , (sonst) \end{cases} \tag{43}$$

zu dem kurzem Ausdruck

$$\hat{M}_1(w) = \int_{-1/2}^{1/2} \mathrm{e}^{-2\pi \mathrm{i} w x}\mathrm{d}x = \mathrm{sinc}(\pi w).$$

Mit Hilfe des Faltungssatzes lässt sich die Fourier-Transformation für die Funktion M_m bestimmen. Mit Definition (43) gilt

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

und die Faltungseigenschaft der Fourier-Transformation erlaubt die bequeme Formulierung der Fourier-Transformierten von M_m als

$$\hat{M}_m(k) = (\mathrm{sinc}(\pi k))^m. \tag{44}$$

Wir können nun konkrete Beispiele für die Fensterfunktionen φ und ψ aus den vorangegangenen Abschnitten angeben. Hierzu definieren wir φ als skalierten, zentrierten, kardinalen B-Spline der Ordnung $2m$ durch

$$\varphi(x) := M_{2m}(2\sigma N x) \tag{45}$$

mit $x \in \mathbb{R}$. Da φ eine stetige Funktion mit dem beschränkten Träger $\mathrm{supp}(\varphi) \subseteq [\frac{-m}{2\sigma N}, \frac{m}{2\sigma N}]$ ist, ist es im weiteren sinnvoll $\psi \equiv \varphi$ zu wählen, da somit der Abschneidefehler E_t (siehe (41)) entfällt.

Wir wenden uns nun dem Überlappungsfehler E_a zu. Für dessen Abschätzung beweisen wir zuerst folgendes Lemma.

Lemma 1: Sei $u \in (0,1)$ und $m \in \mathbb{N}$, dann gilt

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

Beweis: Für $r \geq 0$ gilt wegen

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

die Ungleichung

$$\sum_{r \in \mathbb{Z}} \Big(\frac{u}{u+r}\Big)^{2m} \leq 1 + 2\Big(\frac{u}{u-1}\Big)^{2m} + 2\sum_{r=2}^{\infty}\Big(\frac{u}{u-r}\Big)^{2m}.$$

Statt der Summe auf der rechten Seite verwenden wir ein bestimmtes Integral und schätzen erneut durch

$$\sum_{r \in \mathbb{Z}} \Big(\frac{u}{u+r}\Big)^{2m} \leq 1 + 2\Big(\frac{u}{u-1}\Big)^{2m} + 2\int_{1}^{\infty}\Big(\frac{u}{u-x}\Big)^{2m}\mathrm{d}x$$

nach oben ab. Wir können das Integral berechnen und erhalten somit nach den Umformungen

$$\begin{aligned}
\sum_{r \in \mathbb{Z}} \Big(\frac{u}{u+r}\Big)^{2m} &= 1 + 2\Big(\frac{u}{u-1}\Big)^{2m}\Big[1 + \frac{1-u}{2m-1}\Big] \\
&< 1 + 2\Big(\frac{u}{u-1}\Big)^{2m}\Big[1 + \frac{1}{2m-1}\Big] \\
&= 1 + \Big(\frac{u}{u-1}\Big)^{2m} 2\Big[\frac{2m-1+1}{2m-1}\Big] \\
&= 1 + \Big(\frac{u}{u-1}\Big)^{2m}\Big(\frac{4m}{2m-1}\Big).
\end{aligned}$$

die Behauptung.

□

Satz 1: *Zur Berechnung der Summe* (9) *durch den NFCT-Algorithmus sei die Fensterfunktion φ wie in* (45) *als B-Spline der Ordnung 2m gegeben. Dann kann der Überlappungsfehler E_a für alle Vektoren $\hat{\boldsymbol{f}}^C := (\hat{f}_k)_{\boldsymbol{k} \in I_{\boldsymbol{0}}^{\boldsymbol{N},d}}$ im univariaten Fall $d = 1$ durch*

$$E_a(x_j) \leq ||\hat{\boldsymbol{f}}^C||_1 \frac{4m}{2m-1}\Big(\frac{1}{2\sigma-1}\Big)^{2m}$$

abgeschätzt werden.

Beweis: Wir haben schon gesehen, dass aufgrund der Definition $\psi := \varphi$ und des kompakten Trägers von φ, der Abschneidefehler E_t für alle $x \in [-0.5, 0.5]$ gleich Null ist. Folglich konzentrieren wir uns auf die Abschätzung des Überlappungsfehlers E_a. Dazu betrachten wir die $c_k(\tilde{\varphi})$ unter Berücksichtigung von (44) und (45) und formulieren die Identität

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

Demzufolge gilt offensichtlich auch

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

und nach Verwendung von Definition (43) gilt weiterhin

$$\begin{aligned}
2\sigma N\hat{\varphi}(k+2\sigma N r) &= \left(\mathrm{sinc}\Big(\frac{\pi k}{2\sigma N}+r\pi\Big)\right)^{2m} \\
&= \left(\frac{\sin\left(\frac{k\pi}{2\sigma N}+r\pi\right)}{\frac{k\pi}{2\sigma N}+r\pi}\right)^{2m} \\
&= \left(\frac{\sin\left(\frac{k\pi}{2\sigma N}\right)}{\frac{k\pi}{2\sigma N}+r\pi}\right)^{2m} \\
&= \underbrace{\left(\frac{\sin\left(\frac{k\pi}{2\sigma N}\right)}{\frac{k\pi}{2\sigma N}}\right)^{2m}}_{\mathrm{sinc}\left(\frac{k\pi}{2\sigma N}\right)^{2m}=2\sigma N\hat{\varphi}(k)} \left(\frac{\frac{k\pi}{2\sigma N}}{\frac{k\pi}{2\sigma N}+r\pi}\right)^{2m} \\
&= 2\sigma N\hat{\varphi}(k)\left(\frac{\frac{k}{2\sigma N}}{\frac{k}{2\sigma N}+r}\right)^{2m}.
\end{aligned}$$

Schließlich erhalten wir die Beziehung

$$\frac{\hat{\varphi}(k+2\sigma N r)}{\hat{\varphi}(k)} = \left(\frac{\frac{k}{2\sigma N}}{\frac{k}{2\sigma N}+r} \right)^{2m}.$$

Aus der Definition von E_a (42) ergibt sich mit dieser Identität die Abschätzung

$$\begin{aligned} E_a &\leq \left|\left|\hat{\boldsymbol{f}}^C\right|\right|_1 \max_{k=0,\ldots,N-1} \sum_{r\in\mathbb{Z}\setminus\{0\}} \left(\frac{\frac{k}{2\sigma N}}{\frac{k}{2\sigma N}+r} \right)^{2m} \\ &\leq ||\hat{\boldsymbol{f}}^C||_1 \max_{k=0,\ldots,N-1} \frac{4m}{2m-1} \left(\frac{\frac{k}{2\sigma N}}{\frac{k}{2\sigma N}-1} \right)^{2m}. \end{aligned}$$

Da der Ausdruck $\frac{u^{2m}}{(u-1)^{2m}}$ monoton wächst für $\frac{k}{\sigma N} =: u \in [0, \frac{1}{2}]$ wird folglich das Maximum bei $k = N-1$ angenommen. Wir schwächen diese Abschätzung ab indem wir $k = N$ wählen. Dann ergibt sich der vereinfachte Fehlerterm

$$\max_{k=0,\ldots,N} \left(\frac{\frac{k}{2\sigma N}}{\frac{k}{2\sigma N}-1} \right)^{2m} = \left(\frac{1}{1-2\sigma} \right)^{2m}$$

und wegen dem geraden Exponenten folgt mit

$$E_a \leq ||\hat{\boldsymbol{f}}^C||_1 \frac{4m}{2m-1} \left(\frac{1}{2\sigma-1} \right)^{2m}.$$

die Behauptung.

□

Wir stellen fest, dass wegen $E_t \equiv 0$ der maximale Approximationsfehler E_∞ äquivalent zum maximalen Überlappungsfehler E_a ist und somit für B-Splines als Fensterfunktionen

$$E_\infty = \max_{j=0,\ldots,M} E_a(x_j)$$

gilt. Fehlerabschätzungen für die restlichen vorgeschlagenen Fensterfunktionen sind in [12] zu finden. Die Anpassungen für die NFCT/NFST verlaufen analog. Der Fehler E_∞ fällt exponentiell mit wachsendem m. Dies wird in Abschnitt 6.2 durch numerische Beispiele graphisch anhand von Fehlerkurven illustriert.

5 Die Software-Bibliotheken

Die in dieser Arbeit beschriebenen Algorithmen wurden in C implementiert. Sie greifen zur Berechnung der diskreten Fourier-Transformation beziehungsweise der diskreten Kosinus- und Sinus-Transformation auf die FFTW-Bibliothek [7] zurück. Die Programmbibliotheken der Algorithmen NFFT, NFFT$^{\mathrm{T}}$ und iNFFT wurden von S. Kunis und D. Potts implementiert. In [11] wird unter anderem näher auf sie eingegangen, während sich dieser Abschnitt mit der NFCT, NFST und deren transponierten und inversen Versionen befasst. Deren Code ist zu großen Teilen aus den NFFT-Implementierungen übernommen worden.
Im Folgenden geben wir einen Überblick wie die Software zu verwenden ist und benennen verfügbare Parameter. Da die Bibliotheken sich sehr ähnlich sind, ist ihre Anwendung dementsprechend vergleichbar.

5.1 Anwendung der Software-Bibliothek

Die üblichen Schritte bei der Verwendung dieser Programmbibliotheken sind die Initialisierung eines Plans, die Ausführung der Transformation und die abschließende Freigabe des allozierten Speichers bei der Initialisierung, genannt Finalisierung. Der Plan (`nfct_plan` bzw. `nfst_plan`) enthält alle nötigen Informationen für eine Transformation beinhaltet. Wenn nicht besonders unterschieden werden muss, so verwenden wir anstelle von `nfct` und `nfst` die Bezeichnung `nfRt` als Bezug auf beide Transformationstypen. Es existieren jeweils vier Initialisierungs-Schnittstellen. Drei einfache, namentlich `nfRt_init_`d`d( &ein_plan, N`$_1, ..., $`N`$_d$`,M_total)`, ($d = 1, 2, 3$), sind für das schnelle Einrichten einer ein-, zwei- oder dreidimensionalen nicht-äquidistanten Kosinus- beziehungsweise Sinus-Transformation mit Bandbreite N_t (`N`$_t$) für jede Dimension $t = 1, ..., d$ und Knotenzahl M (`M_total`) vorhanden. Nach dem Initialisieren ist sämtlicher vom Plan benötigter Speicher alloziert. Es können nun die Knoten $\boldsymbol{x} \in [0, \frac{1}{2}]^d$ gesetzt werden, wobei der Index der i-ten Koordinate des j-ten Knoten $jd + i$ entspricht. Weiterhin können entweder die $\hat{f}^C_{\boldsymbol{k}} \in \mathbb{R}$, $(\boldsymbol{k} = 0, ..., \Pi\big((N_1, ..., N_d)^{\mathrm{T}}\big) - 1)$, $\hat{f}^S_{\boldsymbol{k}} \in \mathbb{R}$, $(\boldsymbol{k} = 0, ..., \Pi\big((N_1 - 1, ..., N_d - 1)^{\mathrm{T}}\big) - 1)$, oder die $f_j \in \mathbb{R}$, $(j = 0, ..., M - 1)$, mit Werten belegt werden. Zu beachten ist, dass für die NDST die Koeffizienten $\hat{f}^S_{\boldsymbol{k}}$ mit $\Pi(\boldsymbol{k}) = 0$ nicht existieren. Zur Ausführung der Transformationen genügt nun der Aufruf der Routinen `nfRt_trafo( &ein_plan)` oder `nfRt_transposed( &ein_plan)`. Wird der Plan nicht mehr gebraucht so muss der Speicher durch `nfRt_finalize( &ein_plan)` freigegeben werden.
Die Prozeduren `nfRt_init_guru( &ein_plan, d, &N, M_total, &n, m, nfRt_flags, fftw_flags)` dienen zum initialisieren spezifischer Pläne mit Einfluss auf verschiedene Parameter. So bietet dieses Interface zusätzlich die direkte Angabe der Dimension d und des Abschneide-Parameters m. Des weiteren ermöglicht es die (indirekte) Angabe von $\boldsymbol{\sigma}$ ($\boldsymbol{\sigma} = \boldsymbol{n} \odot \boldsymbol{N}^{-1}$) und verschiedener Flags, auf die wir noch eingehen werden. Die weitere Vorgehensweise bleibt identisch. Wir listen in Tabelle 2 die wichtigsten Strukturen der Pläne mit ihrer Bedeutung in den Algorithmen NFCT, NFCT$^{\mathrm{T}}$, NFST und NFST$^{\mathrm{T}}$ auf.

Datentyp	Name	Bedeutung	Größe
`int`	`d`	d	1
`int`	`M_total`	M	1
`int*`	`N`	$\boldsymbol{N}$	d
`int*`	`n`	$\boldsymbol{n} = \boldsymbol{\sigma} \odot \boldsymbol{N}$	d
`int`	`N_total`	$\Pi\big(\boldsymbol{N}\big)$ bzw. $\Pi\big(\boldsymbol{N} - \boldsymbol{1}_d\big)$	1
`int`	`m`	m	1
`double*`	`x`	$\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$	dM
`double*`	`f`	f^C bzw. f^S	M
`double*`	`f_hat`	$\hat{f}^C_{\boldsymbol{k}}$ bzw. $\hat{f}^S_{\boldsymbol{k}}$	$\Pi(\boldsymbol{N})$ bzw. $\Pi(\boldsymbol{N} - \boldsymbol{1}_d)$
`double*`	`g`	$g^C_{\boldsymbol{l}}$ bzw $g^S_{\boldsymbol{l}}$	$\Pi(\boldsymbol{n})$ bzw $\Pi(\boldsymbol{n} - \boldsymbol{1}_d)$
`double*`	`g_hat`	$\hat{g}^C_{\boldsymbol{k}}$ bzw $\hat{g}^S_{\boldsymbol{k}}$	$\Pi(\boldsymbol{n})$ bzw $\Pi(\boldsymbol{n} - \boldsymbol{1}_d)$
`double**`	`c_phi_inv`	$c_{\boldsymbol{k}}(\tilde{\varphi})^{-1}$	$\sum_{t=0}^{d-1} N_t$
`double*`	`psi`	$\tilde{\psi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})\big)$	$2d(m+1)M$

Tabelle 2: Die wichtigsten Komponenten der Datenstrukturen `nfct_plan` und `nfst_plan`

5.2 Optionen und Parameter

Zur Anpassung der Bibliothek an eigene Wünsche gibt es zwei grundlegende Methoden. Neben den Optionen in der Datei `options.h` sind das die Flags, die man mit dem Parameter `nfRt_flags` an die Initialisierungs-Routine übergibt. In dieser Header-Datei trifft man ebenfalls die Wahl der Fensterfunktion indem man sie als Compilerflag definiert. Es stehen im Moment die vier Möglichkeiten `B_SPLINE`, `GAUSSIAN`, `KAISER_BESSEL` und `SINC_POWER` zur Verfügung. Mit dieser Wahl legt man automatisch einen spezifischen Abschneide-Parameter m fest , der so gewählt ist, dass der Fehler E_∞ kleiner als 10^{-12} ist. Dort kann auch die CPU-Zeitmessung durch `define MEASURE_TIME` aktiviert werden. Die derzeitige Liste der Flags für die NFCT, NFCT$^{\mathrm{T}}$, NFST und NFST$^{\mathrm{T}}$ Algorithmen einschließlich ihrer Bedeutung ist in Tabelle 3 aufgeführt. Diese werden ODER-verknüpft an die `init`-Routinen gereicht. Ebenso die `fftw_flags`, deren Default-Wert zu diesem Zeitpunkt aus den Flags `FFTW_ESTIMATE` und `FFTW_DESTROY_INPUT` besteht. Mehr Informationen zu FFTW-Flags findet man unter [7].

Ist das Flag `PRE_PSI` gesetzt, so muss nach dem Initialisieren der Knoten `ein_plan.x` die Routine `nfRt_precompute_psi` aufgerufen werden. Das Flag `PRE_FULL_PSI` kann nur zusätzlich zu `PRE_PSI` gesetzt werden. Ist dies der Fall so kann `nfRt_full_psi_eps` mit

Name	Beschreibung
`PRE_PSI`†	Speicherallokation für $\tilde{\psi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})\big)$
`PRE_PHI_HUT`†	Speicherallokation und Vorberechnung der $c_{\boldsymbol{k}}(\tilde{\varphi})$
`MALLOC_X`†	Speicherallokation für $\boldsymbol{x}_j \in [0, \frac{1}{2}]^d$, $j \in I_0^{M,1}$
`MALLOC_F_HAT`†	Speicherallokation für $\hat{f}^C_{\boldsymbol{k}}$ bzw. $\hat{f}^S_{\boldsymbol{k}}$
`MALLOC_F`†	Speicherallokation für f_j
`FFTW_INIT`†	Initialisierung der FFTW-Pläne für die diskrete Kosinus- bzw. Sinus-Transformation
`FFT_OUT_OF_PLACE`†	Eingabe- und Ausgabevektoren der FCT bzw. FST haben eigene Speicherbereiche
`PRE_FULL_PSI`	Speicherallokation und Vorberechnung der $\tilde{\psi}\big(\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})\big)$ mit Eliminierung der Einträge kleiner $\lvert$`nfRt_full_psi_eps`$*\tilde{\psi}\big(-\frac{1}{2}(\boldsymbol{l} \odot (\boldsymbol{\sigma} \odot \boldsymbol{N})^{-1})\big)\rvert$, (schneller)

Tabelle 3: Die Flags der Algorithmen. Sie sind in allen sechs Algorithmen verwendbar.
† Flag ist standardmäßig für die einfachen Initialisierungsroutinen gesetzt.

einem Wert aus $\mathbb{R}$ belegt werden. Mit diesem Wert lässt sich steuern welche Einträge der Matrix $\boldsymbol{B}$ vernachlässigt werden um dadurch die Laufzeit zu verringern. Standardmässig ist dieser Wert auf 10^{-10} gesetzt.

Listing 1 ist ein Beispiel für eine zweidimensionale nicht-äquidistante Kosinus- oder Sinus-Transformation. Es soll anschaulich die Verwendung der Bibliothek demonstrieren.

5.3 iNFCT und iNFST

Um eine inverse NFCT/NFST durchzuführen, initialisiert man wie gesehen einen Plan für eine direkte NFCT/NFST, den man, wie folgt, an die Init-Routine `infRt_init( &ein_inv_plan, &ein_plan)` oder `infRt_init_specific( &ein_inv_plan, &ein_plan, infRt_flags)` übergibt. Die Variable `ein_inv_plan` ist dabei vom Typ `infRt_plan`. In letzterer Initialisierungsroutine ist es mittels des Flags `infRt_flags` möglich das Verfahren zur Berechnung der inversen Transformation zu wählen. Im Moment ist durch die Angabe von einem der Werte `CGNE, CGNR, LANDWEBER, STEEPEST_DESCENT` als `infRt_flags` jeweils eines der Verfahren CGNE, CGNR, Landweber oder die Methode des steilsten Abstiegs auswählbar. Der Standardwert ist `CGNR`.
Nachdem alle gewünschten Variablen (Knoten `ein_plan.x`, Samples `ein_inv_plan.y`,

```
void eine_2d_nfRt_trafo( ); {
  int j,k;
  nfRt_plan ein_plan;

   // Initialisierung einer 2 dim. Transformation
   // mit Bandbreiten 32, 32 und 1024 Knoten
  nfRt_init_2d( &ein_plan, 32, 32, 1024);

   // Initialisieren der Knoten
  for( j = 0;  j < ein_plan.d * ein_plan.M_total;  j++)
    ein_plan.x[j]  = 0.5 * ((double)rand()) / RAND_MAX;

   // wenn das Flag PRE_PSI gesetzt ist, dann fuehre Vorberechnung aus
  if( ein_plan.nfRt_flags  &  PRE_PSI)
    nfRt_precompute_psi( &ein_plan);

   // Initialisieren der Koeffizienten
  for( k = 0;  k < ein_plan.N_total;  k++)
    ein_plan.f_hat[k] = ((double)rand()) / RAND_MAX;

   // Ausfuehrung der Transformation
  nfRt_trafo( &ein_plan);

   // weitere Verarbeitung der Daten (z.B Ausgabe)
  for( j = 0;  j < ein_plan.M_total;  j++)
    printf( "f[%d] = %e\n", j, ein_plan.f[j]);

   // Freigabe des allozierten Speichers
  nfRt_finalize( &ein_plan);
}
```

Listing 1: Beispiel-Routine für eine 2-dimensionale NFCT bzw. NFST

Koeffizienten (Lösung) `ein_inv_plan.f_hat_iter`) initialisiert sind, muss die Funktion `infRt_before_loop( &ein_inv_plan)` aufgerufen werden, um (verfahrensabhängige) Vorberechnungen auszuführen. Danach kann mit `infRt_loop_one_step( &ein_inv_plan)` jeweils ein Schritt des iterativen Lösungsverfahren berechnet werden.
Auch hier muss der Speicher der Pläne `ein_inv_plan` und `ein_plan` durch die Prozeduren `infRt_finalize( &ein_inv_plan)` gefolgt von `nfRt_finalize( &ein_plan)` freigegeben werden. Ein Beispiel für eine 2-dimensionale inverse nicht-äquidistante Kosinus/Sinus-Transformation zeigt Listing 2. Weitere Informationen zu Flags und Plänen der inversen Fourier-Transformationen für nicht-äquidistanten Daten sind in [10, Seiten 17-19] zu finden. Alle dort eingeführten Flags sind auf gleiche Weise auch in der iNFCT und iNFST

verwendbar.

```
void eine_2d_infRt_trafo( ); {
  int j,k, iter, iter_end = 5;
  nfRt_plan  ein_plan;
  infRt_plan ein_inv_plan;

   // Initialisierung einer 2 dim. Transformation
   // mit Bandbreiten 32, 32 und 1024 Knoten
  nfRt_init_2d( &ein_plan, 32, 32, 1024);

   // Initialisierung der inversen Transformation
  infRt_init( &ein_inv_plan, &ein_plan);

   // Initialisieren der Knoten
  for( j = 0;  j < ein_plan.d * ein_plan.M_total;  j++)
    ein_plan.x[j] = 0.5 * ((double)rand()) / RAND_MAX;

   // wenn das Flag PRE_PSI gesetzt ist, dann fuehre Vorberechnung aus
  if( ein_plan.nfRt_flags  &  PRE_PSI)
    nfRt_precompute_psi( &ein_plan);

   // Initialisieren der Samples
  for( j = 0;  j < ein_plan.M_total;  j++)
    ein_inv_plan.y[j] = ((double)rand()) / RAND_MAX;

   // Initialisieren der Koeffizienten
  for( k = 0;  k < ein_plan.N_total;  k++)
    ein_inv_plan.f_hat_iter[k] = 0.0;

   // Vorberechnungen
  infRt_before_loop( &ein_inv_plan);

   // iterative Loesung
  for( iter = 0;  iter < iter_end;  iter++)
    infRt_loop_one_step( &ein_inv_plan);

   // Freigabe des allozierten Speichers
  infRt_finalize( &ein_inv_plan);
  nfRt_finalize( &ein_plan);
}
```

Listing 2: Beispiel-Routine für eine 2-dimensionale inverse NFCT bzw. inverse NFST

6 Eigenschaften der Algorithmen

In diesem Abschnitt wird kurz auf die Ergebnisse von Laufzeitvergleichen eingegangen, die durchgeführt wurden um zu zeigen, ob es Geschwindigkeitssteigerung gibt, wenn für reelle symmetrische Eingabedaten die spezialisierten Algorithmen verwendet werden. Zu erwarten wäre der Faktor vier zugunsten der trigonometrischen Transformationen, da die Bandbreite N_t in jeder Dimension t, $(t = 0, ...d-1)$, nur halb so groß ist, wie die einer äquivalenten Fourier-Transformation und wegen der ausschließlich reellen Daten nur halb so viel Werte zugrundeliegen. Ein erheblicher Teil der Laufzeit entfällt jedoch auf die Berechnung der Koeffizienten $g_{\boldsymbol{l}}$ ($\boldsymbol{l} \in I_{\boldsymbol{a}}^{\boldsymbol{\sigma}\odot\boldsymbol{N},d}$, $a \in \{0,1\}$) die von der eingebundenen FFTW-Bibliothek übernommen wird. Da die FFTW-Bibliothek die diskreten trigonometrischen Transformationen jedoch durch eine FFT berechnet, für die die Eingabedaten angepasst („gespiegelt“) werden, kann man also mit einer Halbierung der Laufzeit rechnen.
Danach schauen wir uns numerisch bestimmte Approximationsfehler der direkten (NFCT, NFST) und inversen Transformationen (iNFCT, iNFST) an. Während wir den Fehler für erstere in Abhängigkeit des Abschneideparameters m betrachten, messen wir den Fehler der inversen Transformationen pro Iterationsschritt.

Es sind in allen numerischen Beispielen dieses Abschnitts die (Default-)Flags `PRE_PHI_HUT`, `PRE_PSI`, `MALLOC_X`, `MALLOC_F_HAT`, `MALLOC_F`, `FFTW_INIT`, `FFT_OUT_OF_PLACE` gesetzt. An die FFTW-Bibliothek wurden jeweils die Flags `FFTW_ESTIMATE` und `FFTW_DESTROY_INPUT` übergeben.
Wenn nicht anders angegeben, wurden die Beispielprogramme auf einer Maschine mit einem 2000MHz getakteten AMD Athlon$^{\textcircled{T}}$ XP 2600+ Prozessor ausgeführt. Der Rechner war ausgestattet mit 2GB RAM. Kompiliert wurden die Testbeispiele mit dem GCC in der Version 3.3 auf einem SUSE GNU/Linux System mit Kernel 2.4.20-4GB-athlon.

6.1 Laufzeiten im Vergleich

Es sind vor allem zwei Fragestellungen interessant. Zum einen wollen wir wissen welche Geschwindigkeitsvorteile die schnellen Algorithmen gegenüber den langsamen mit sich bringen und zum anderen wie groß der Unterschied zwischen den Laufzeiten der NFFT und den Algorithmen NFCT und NFST in der Praxis ist. In allen Testfällen wurde die Zeit ab dem Aufruf der Funktion, die die Transformation startet (`ndct_trafo`, `ndst_trafo`, `nfct_trafo`, usw.), bis zur Rückkehr zum aufrufenden Programm gemessen. Beide Probleme wurden jeweils für die Dimensionen $d = 1$ und $d = 2$ untersucht. Die Koeffizienten $\hat{\boldsymbol{f}}$ und die Knoten $\boldsymbol{x}$ wurden mit zufallsgenerierten Werten initialisiert.

Die Ergebnisse der ersten Tests sind in Abbildung 1 zu sehen. Dort werden die Laufzeiten der langsamen Algorithmen (NDCT, NDST) mit denen der schnellen Algorithmen (NFCT, NFST) verglichen. Die Knotenzahl M ist in allen Beispielen gleich der Bandbreiten $N_{(t-1)}$ für jede Dimension $t = 1, 2$. Verwendet wurde die Gauß-Funktion als Fensterfunktion mit dem Abschneideparameter $m = 8$. Die Grafiken bestätigen nicht

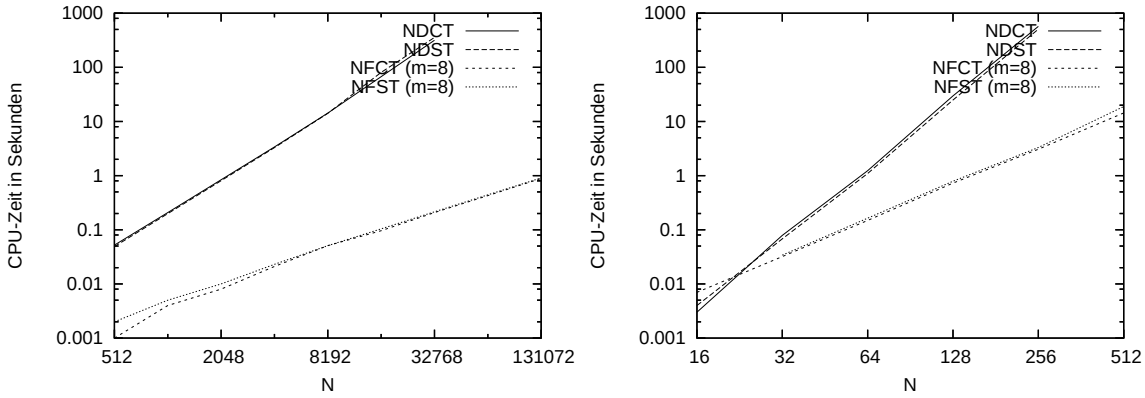

Abbildung 1: Vergleich der Laufzeiten von NDCT/NDST und NFCT/NFST für die Dimensionen $d = 1$ (links) und $d = 2$ (rechts). Die Gauß-Funktion diente als Fensterfunktion φ. Es gilt jeweils $\boldsymbol{N} = M\mathbf{1}_d$. (Flags: `PRE_PHI_HUT`, `PRE_PSI`, `MALLOC_X`, `MALLOC_F_HAT`, `MALLOC_F`, `FFTW_INIT`, `FFT_OUT_OF_PLACE`)

nur die offensichtlich kürzeren Laufzeiten, sondern zeigen ebenso, dass die Laufzeiten der schnellen Algorithmen (NFCT und NFST) sehr eng beieinander liegen. Aufgrund dessen beschränken wir uns im Folgenden auf die NFCT.
Der Veranschaulichung der Laufzeitunterschiede zwischen den schnellen reellen Transformationen (NFCT, NFST) und der NFFT widmet sich Abbildung 2. Hier ist zu beachten, dass die NFFT mit entsprechend größerer Bandbreite ausgeführt wurde, d.h. im wesentlichen $2N_t$ je Dimension $t = 0, ..., d-1$. Die Anzahl der Knoten blieb dabei jedoch konstant. Die Zeiten wurden mit unveränderten Eingaben und gleicher Messweise ermittelt. Zum Vergleich wurden die Laufzeiten von Testbeispielen mit kleinerem Abschneideparameter m und die des langsamen Algorithmus mit aufgenommen. Den erwarteten Faktor 2 bezüglich der Laufzeiten zwischen (hier exemplarisch) der NFCT und der NFFT bestätigt Abbildung 3.

6.2 Approximationsfehler der NFCT und NFST

Die Grafiken in diesem Abschnitt zeigen den Approximationsfehler in Abhängigkeit vom Abschneideparameter m. Wir hatten in der Einleitung zu diesem Abschnitt behauptet, dass der Fehler E_∞ mit linear steigendem m exponentiell abfällt. Dies wird am einfachsten deutlich, wenn wir den Fehler logarithmisch auftragen. Die numerischen Testbeispiele sind jeweils mit dem NFCT- und dem NFST-Algorithmus für die Dimensionen $d = 1, ..., 3$ ausgeführt worden.
Der Abschneideparameter m lief jeweils von 0 bis 14. Die Eingabedaten ($\boldsymbol{x}$, $\hat{\boldsymbol{f}}^C$ bzw. $\hat{\boldsymbol{f}}^S$) wurden vom Zufallsgenerator erzeugt. Zum Einsatz kam ein AMD Athlon(T) XP 2700+ mit 2GB RAM, der zu diesem Zeitpunkt unter einem GNU/Linux System (Kernel 2.6.5) lief. Auf der linken Seite der Abbildungen 4, 5 und 6 ist das Verhalten des Fehlers für

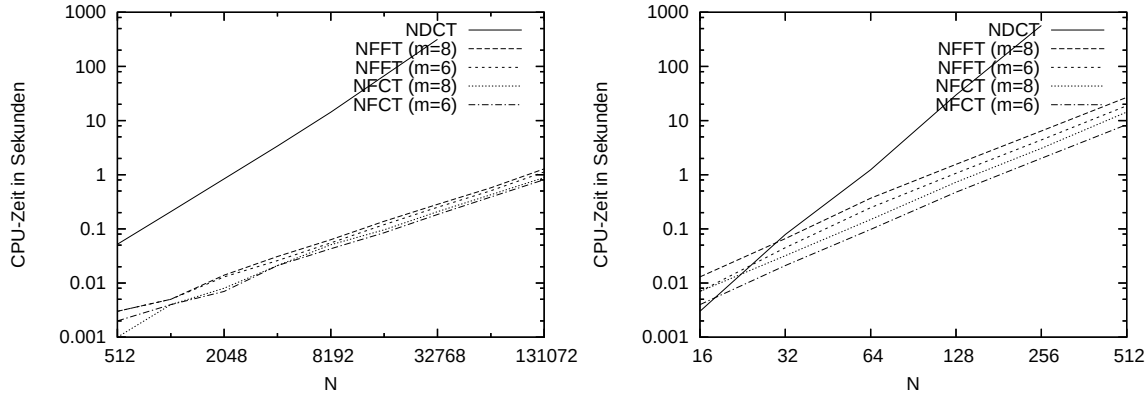

Abbildung 2: Laufzeiten der direkten Summation (NDCT) und der schnellen Algorithmen NFFT und NFCT für jeweils zwei verschiedene m und den Flags `PRE_PHI_HUT`, `PRE_PSI`, `MALLOC_X`, `MALLOC_F_HAT`, `MALLOC_F`, `FFTW_INIT`, `FFT_OUT_OF_PLACE`. Als Fensterfunktion φ haben wir jeweils die Gauß-Funktion verwendet. Links sehen wir die Ergebnisse für die Dimension $d = 1$ und $N_0 = M$, während die rechte Abbildung die Zeiten für $d = 2$ und $N_0 = N_1 = M$ illustriert. Die NFFT arbeitet intern mit entsprechender äquivalenter Problemgröße, d.h. $\hat{\boldsymbol{f}}^F \in \mathbb{R}^{(2N_0)^d}$.

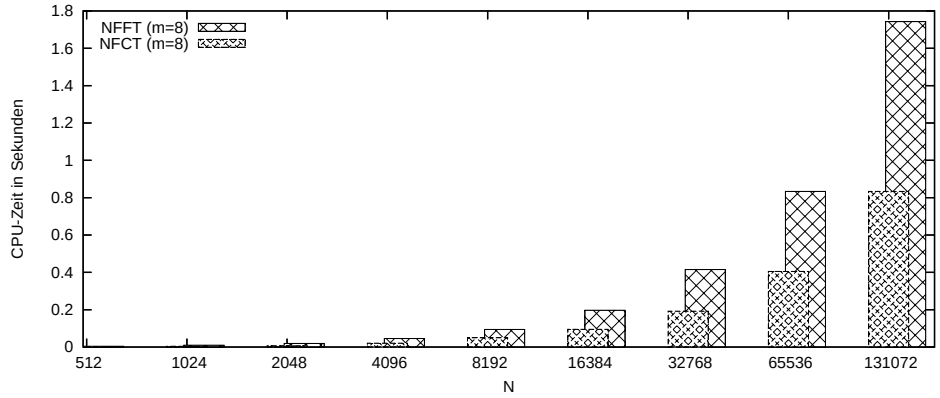

Abbildung 3: Laufzeitunterschiede der NFCT und NFFT. Die Zeiten sind identisch mit denen aus Abbildung 2 (links).

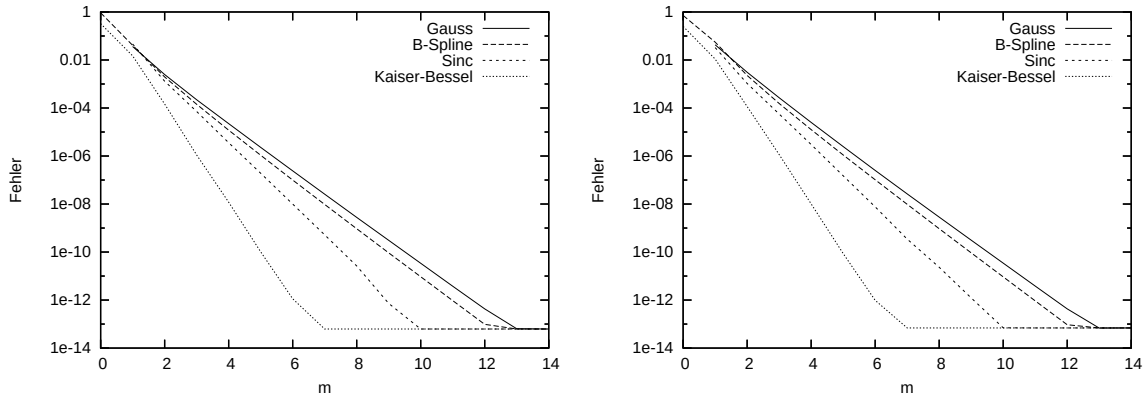

Abbildung 4: Der Approximationsfehler E_∞ der NFCT (links) und der NFST (rechts) in Abhängigkeit von m für die Dimension $d = 1$. ($N_0 = 4096, M = 10000$, mit den Flags `PRE_PHI_HUT`, `PRE_PSI`)

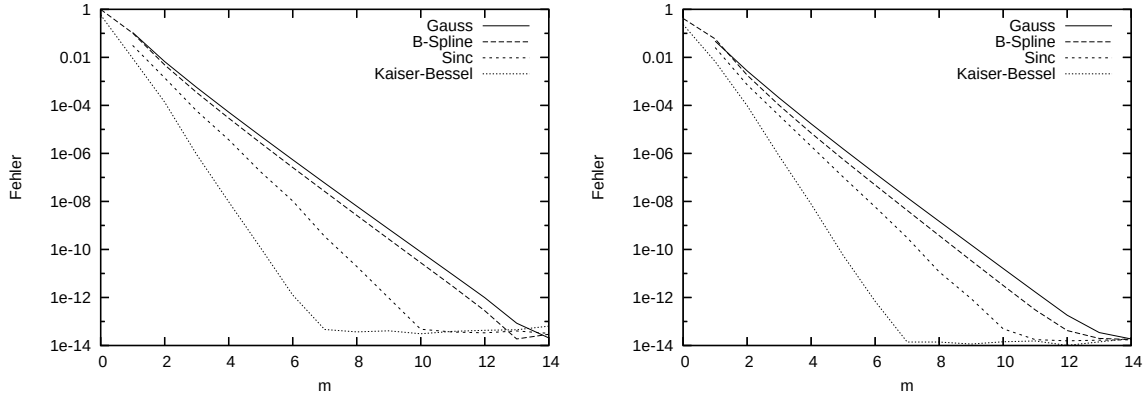

Abbildung 5: Der Approximationsfehler E_∞ der NFCT (links) und der NFST (rechts) in Abhängigkeit von m für die Dimension $d = 2$. ($N_0 = 64, N_1 = 64, M = 10000$, mit den Flags `PRE_PHI_HUT`, `PRE_PSI`,)

die NFCT zu sehen, während die rechte Seite die Resultate der NFST zeigt. Es sind die ersten drei Dimensionen abgebildet mit den jeweils vier verschiedenen Fensterfunktionen (Gauß, B-Splines, Sinc und Kaiser-Bessel).
Seien $\boldsymbol{f}_{f,m}^R$ die Ergebnisvektoren der schnellen Algorithmen (NFCT resp. NFST) mit Abschneideparameter m und $\boldsymbol{f}_d^R$ die (exakten) Ergebnisvektoren der langsamen Algorithmen (NDCT resp. NDST). Der Fehler E_∞ ist dann die 2-Norm der Differenz der Vektoren $\boldsymbol{f}_{f,m}^R$ und $\boldsymbol{f}_d^R$ gewichtet mit dem inversen der 2-Norm von $\boldsymbol{f}_{f,m}^R$.

Wie die Abbildungen 4 bis 6 beweisen, fällt der Fehler bei wachsendem m mit der Kaiser-Bessel-Funktion als Fensterfunktion für alle Dimensionen am schnellsten. Die langsamste Konvergenz an die korrekte Lösung wurde bei der Gauß-Funktion festgestellt.

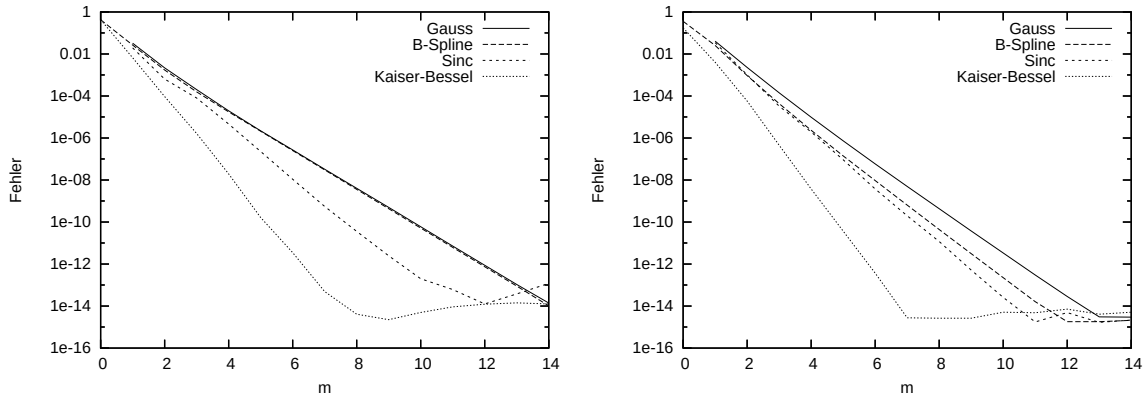

Abbildung 6: Der Approximationsfehler E_∞ der NFCT(links) und der NFST (rechts) in Abhängigkeit von m für die Dimension $d = 3$. ($N_0 = 16, N_1 = 16, N_2 = 16, M = 10000$, mit den Flags `PRE_PHI_HUT`, `PRE_PSI`)

Aus Interesse an kurzen Laufzeiten bei einem möglichst genauen Ergebnis ist die Kaiser-Bessel-Funktion somit die Funktion, die wir in weiteren Betrachtungen vorzugsweise nutzen werden.

6.3 Approximationsfehler der iNFCT und iNFST

Wie in Abschnitt 3.4 erwähnt, realisieren wir die inversen Transformationen mit iterativen Gleichungssystemlösern. Es stehen dafür in unserer Software-Bibliothek unter anderen die Verfahren CGNE und CGNR bereits zur Verfügung. Es soll R wie in anderen Teilen wieder stellvertretend für C bzw. S stehen. Es sind die Koeffizienten $\hat{\boldsymbol{f}}^R$ und deren kosinus- bzw. sinustransformierte Vektoren $\boldsymbol{f}^R$ wie bisher gegeben. Dann seien $\tilde{\hat{\boldsymbol{f}}}_l^R$ die Fourier-Koeffizienten, die durch die inversen Transformationen nach l Iterationen bei Eingabe des Vektor $\boldsymbol{f}^R$ (in Abschnitt 3.4 als $\boldsymbol{y}$ bezeichnet) ermittelt wurden. Zuletzt soll der Vektor $\tilde{\boldsymbol{f}}_l^R$ wiederum die Kosinus- bzw. Sinustransformierte des Vektors $\tilde{\hat{\boldsymbol{f}}}_l^R$ sein. Wir werden an jeweils zwei Testprogrammen für die iNFCT und iNFST den Fehler $E_{\max} := \frac{||\boldsymbol{f}^R - \tilde{\boldsymbol{f}}_l^R||_\infty}{||\boldsymbol{f}^R||_\infty}$, als Maß für die Annäherung an die ursprünglichen Daten, bestimmen.

In den Testroutinen geben wir uns zufällig gewählte Knoten $\boldsymbol{x} \in [0, \frac{1}{2}]^d$ und Samples $\boldsymbol{f}^R \in \mathbb{R}^M$ vor. Dann initialisieren wir die Stützwerte (`infRt_plan.y`) mit den Werten $\boldsymbol{f}^R$. Nachdem die Koeffizienten $\tilde{\hat{\boldsymbol{f}}}_0^R$ (`infRt_plan.f_hat_iter`) mit dem Eins-Vektor $\mathbf{1}_d$ initialisiert und die Vorberechnungen abgeschlossen sind, wird pro Iterationsschritt der Fehler $E_{\max}$ berechnet.

Abbildung 7 zeigt den Verlauf des Fehlers für das Verfahren CGNR mit den Eingabegrößen $N = 128$, Knotenzahl $M = 1000$ und Abschneideparameter $m = 12$. Für das

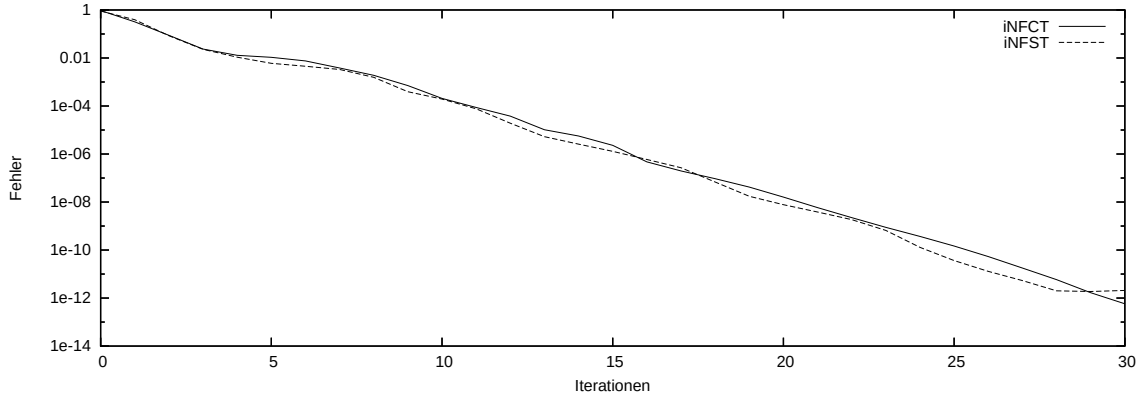

Abbildung 7: Der Approximationsfehler $E_{\max}$ der iNFCT und iNFST mit dem CGNR Verfahren in Abhängigkeit der Iterationsschritte l für die Dimension $d = 1$. ($N_0 = 128, M = 1000, m = 10, \tilde{\boldsymbol{f}}_0^R = \mathbf{1}_d$, mit den Flags `PRE_PHI_HUT`, `PRE_PSI`)

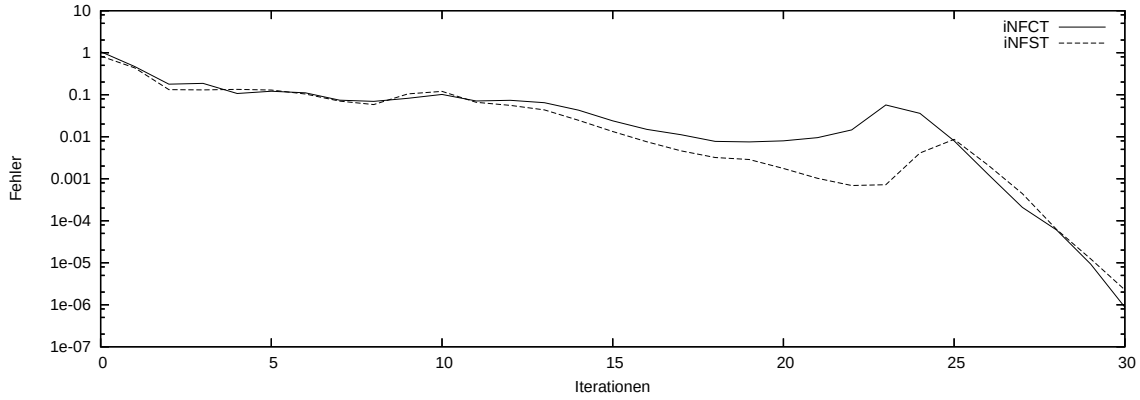

Abbildung 8: Der Approximationsfehler $E_{\max}$ der iNFCT und iNFST mit dem CGNE Verfahren in Abhängigkeit der Iterationsschritte l für die Dimension $d = 1$. ($N_0 = 1024, M = 100, m = 10, \tilde{\boldsymbol{f}}_0^R = \mathbf{1}_d$, mit den Flags `PRE_PHI_HUT`, `PRE_PSI`)

Verfahren CGNE wurde der Polynomgrad $N = 1024$ und Knotenzahl $M = 100$ bei gleichem Abschneideparameter m gewählt. Gewichte wurden nicht verwendet. Die Fehlerkurve ist in Abbildung 8 zu sehen. Die Testroutinen durchliefen je 30 Iterationsschritte.

7 Numerische Beispiele

Wir wollen uns zum Abschluß exemplarisch zwei praktische Anwendungsfälle ansehen. Das sind zum einen Scattered Data Interpolationen und Funktionsrekonstruktionen. Es wird versucht auf Basis von gegebenen Knoten $\boldsymbol{x}$ und Stützwerten $f(\boldsymbol{x})$ ein trigonometrisches Polynom zu bestimmen, dass die gestreuten Daten beziehungsweise die gegebene Funktion annähert.

7.1 Scattered Data Interpolation

Dieser Abschnitt befaßt sich mit einem Beispiel zur Scattered Data Interpolation. Die verwendeten Daten stammen von [3]. Hier werden wir den Datensatz des Gletschers (`glacier`) benutzen, in dem Höhenangaben an verschiedenen Punkten $\boldsymbol{x} \in \mathbb{R}^2$ hinterlegt sind.

Abbildungen 9 bis 12 zeigen Rekonstruktionen des Gletscher-Datensatzes, wobei zum Vergleich die iNFCT (links) und die iNFST (rechts) angewendet wurden. Der Gletscher ist an $M = 8345$ verschiedenen Koordinaten $\boldsymbol{x} = (x, y)^{\mathrm{T}} \in \mathbb{R}^2$ gegeben. Um Artefakte bei der Berechnung mit der NFCT/iNFCT beziehungsweise NFST/iNFST zu verhindern, wurde auf das Intervall $I^R := [\frac{1}{5}, \frac{4}{5}]^2$ skaliert. Zur Bestimmung der (in diesem Beispiel $\Pi((256, 256)^{\mathrm{T}}) = 65536$) Fourier-Koeffizienten $\hat{f}_{\boldsymbol{k}}$ wurde das CGNR Verfahren mit inversen multi-quadratischen Dämpfungsfaktoren

$$\hat{w}_{\boldsymbol{k}} := \sum_{p=0}^{1} \left[\left(\lfloor ||\boldsymbol{k}||_2 \rfloor + p \right)^2 + c^2 \right]^{-\mu}$$

verwendet, wobei für $\mu = 1.2$ und $c = 0.8$ gewählt wurden. Die Kaiser-Bessel-Funktion war mit Abschneideparameter $m = 10$ die Fensterfunktion. Nach Berechnung der Fourier-Koeffizienten $\hat{f}^C_{\boldsymbol{k}}$ bzw. $\hat{f}^S_{\boldsymbol{k}}$ mit $10, 20, 30$ bzw. 40 Iterationsschritten wurde der Gletscher in allen Fällen auf einem äquidistanten, quadratischen Gitter mit 10000 Punkten in I^R rekonstruiert. Dabei war die Fensterfunktion wieder die Kaiser-Bessel-Funktion. Während das Ergebnis nach 10 Iterationen noch sehr ungenau ist, werden nach 20 Schritten schon Details sichtbar.

In Abbildung 13 ist der Fehler $E_2 := \frac{||\boldsymbol{f}^R - \tilde{\boldsymbol{f}}^R_l||_2}{||\boldsymbol{f}^R||_2}$ dargestellt, wobei $\boldsymbol{f}^R$ den Höhenangaben der Originaldaten des Gletscher-Datensatzes entsprechen und $\tilde{\boldsymbol{f}}^R_l$, $(l \in \{0, ..., 40\})$, die rekonstruierten Werte nach l Iterationen an den vorgegebenen $M = 8345$ Punkten sind.

7.2 Funktionsrekonstruktion

Im zweiten Anwendungsbeispiel, das wir wie bisher für beide Transformationstypen betrachten, wollen wir eine gegebene Funktion rekonstruieren. Wir verwenden dazu die

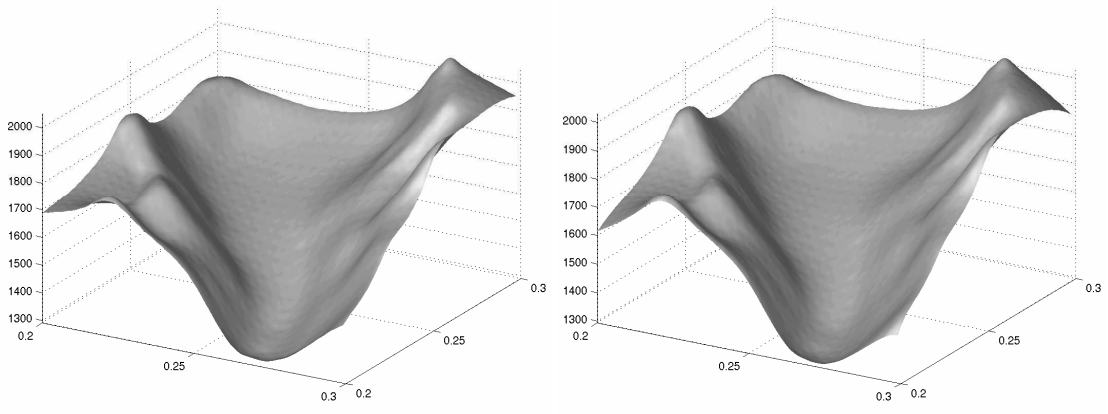

Abbildung 9: Rekonstruktion des Gletscher-Datensatzes `glacier` von [3] mit Kosinus-Polynomen (links) bzw. Sinus-Polynomen (rechts) und den Dämpfungsfaktoren $\hat{w}_{\boldsymbol{k}} = \sum_{p=0}^{1} \left[\left(\lfloor ||\boldsymbol{k}||_2 \rfloor + p \right)^2 + c^2 \right]^{-\mu}$, $(\mu = 1.2, c = 0.8)$, nach 10 Iterationen. Der Datensatz enthält 8345 Punkte. Interpoliert wurde an $M = 10000$ Punkten auf einem quadratischen äquidistanten Gitter mit Polynomgrad $N = 256$ in beide Dimensionen.

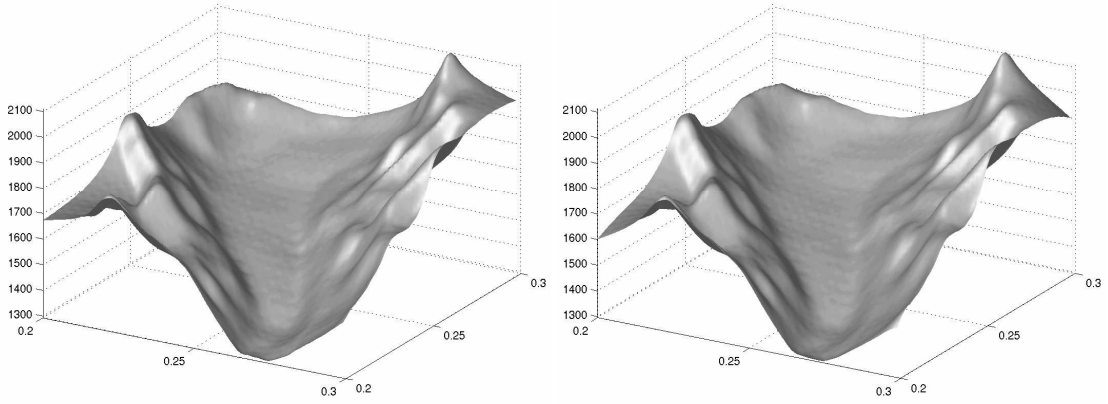

Abbildung 10: Rekonstruktion des Gletscher-Datensatzes `glacier` von [3] nach 20 Iterationen (Parameter wie in Abbildung 9).

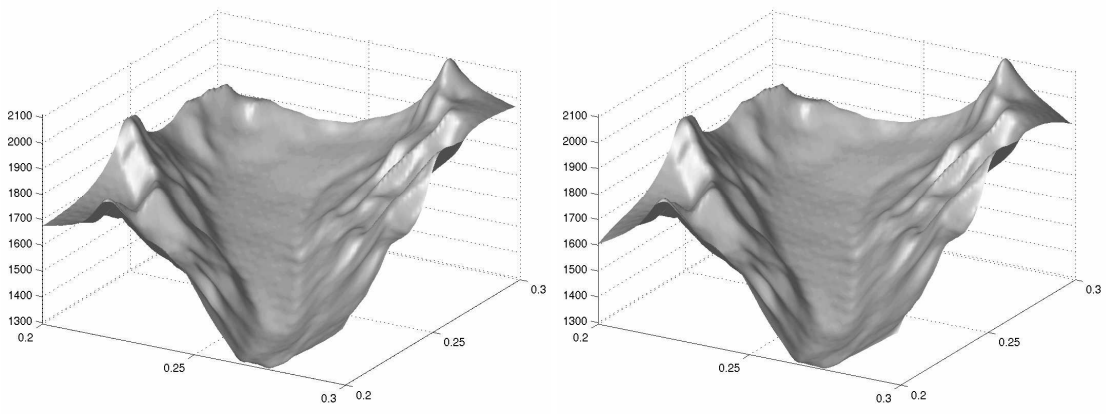

Abbildung 11: Rekonstruktion des Gletscher-Datensatzes `glacier` von [3] nach 30 Iterationen (Parameter wie in Abbildung 9).

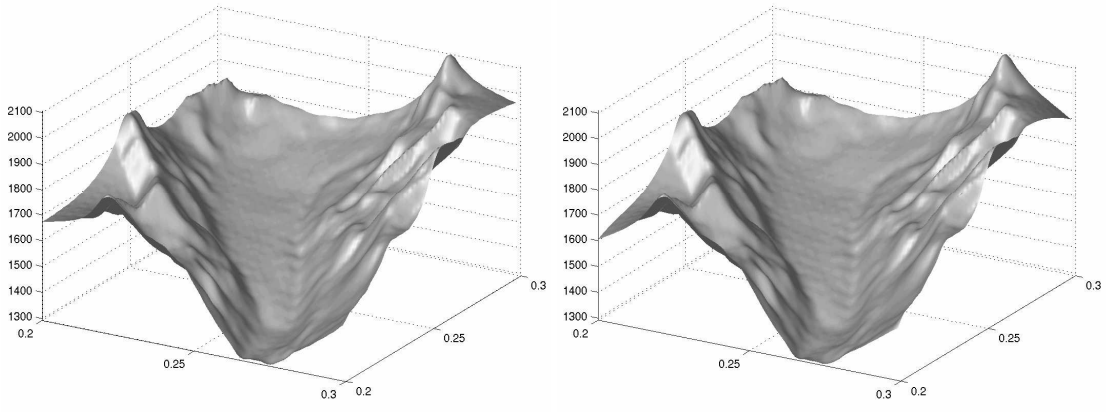

Abbildung 12: Rekonstruktion des Gletscher-Datensatzes `glacier` von [3] nach 40 Iterationen (Parameter wie in Abbildung 9).

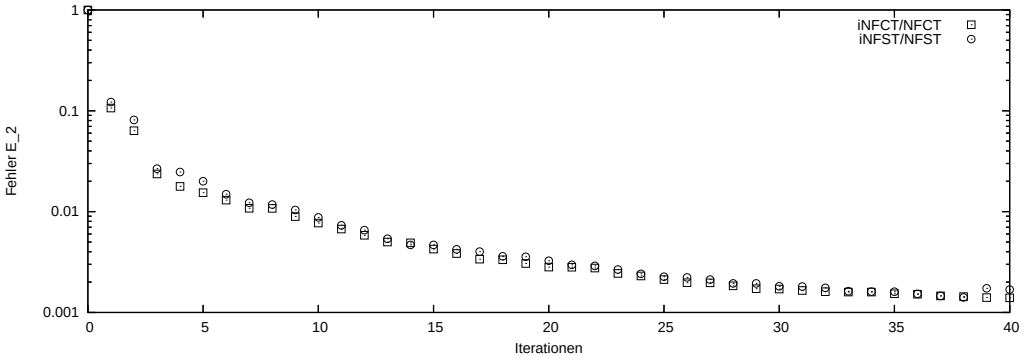

Abbildung 13: Fehler E_2 bei Rekonstruktion des Gletscher-Datensatzes an den gegebenen Knoten mit dem CGNR Verfahren. Verwendete Fensterfunktion: Kaiser-Bessel-Funktion (Abschneideparameter $m = 10$). Anzahl der Knoten ist $M = 8345$ und Polynomgrad $N_t = 256$, $(t = 0, 1)$. Die inversen multi-quadratischen Gewichte mit $\mu = 1.2$ und $c = 0.8$ wurden auch hier verwendet.

zweidimensionale Sinc-Funktion $\mathrm{sinc}((x-2)^2 + (y-2)^2)$ im Intervall $[0, 4]^2$. Das Vorgehen ist dem vorigen Beispiel sehr ähnlich, jedoch können wir die Funktion nun an beliebigen Punkten abtasten, um einen Datensatz zu erzeugen. Dies wurde an $M = 8000$ Punkten $\boldsymbol{x} \in [0, 4]^2$ getan, sodass die Problemgrößen der zwei Beispiele vergleichbar sind. Der Grad des Interpolations-Polynoms $\Pi(\boldsymbol{N})$ ist wie gehabt 65536 mit $N_0 = 256$ und $N_1 = 256$. Die zu berechnenden Komponenten des Vektors $\hat{\boldsymbol{f}}$ wurden anfänglich mit 1.0 initialisiert und die Knoten $\boldsymbol{x}$ auf das Intervall $[0, \frac{1}{2}]^2$ skaliert. Unter Verwendung des CGNR Verfahrens und den multi-quadratischen Dämpfungsfaktoren $\hat{w}_{\boldsymbol{k}} = (c^2 + k^2)^{-\mu} + (c^2 + (k+1)^2)^{-\mu}$ mit $\mu = 1.2$ und $c = 0.8$ wurden dann 40 Iterationsschritte durchgeführt. Mit den so ermittelten Fourier-Koeffizienten $\hat{\boldsymbol{f}}_{40}$, der Index deutet die Iterationsschritte an, wurde das Interpolations-Polynom auf einem 100 mal 100 Gitter in $[0, \frac{1}{2}]^2$ mit den schnellen Algorithmen NFCT beziehungsweise NFST ausgewertet.

Abbildungen 14 und 15 zeigen jeweils die Resultate der Rekonstruktionen neben dem Original. Der Verlauf des Fehlers E_2 ist in Abbildung 16 abgetragen.

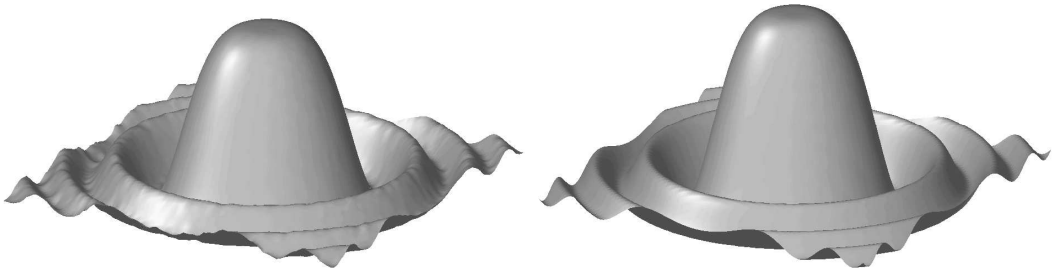

Abbildung 14: Rekonstruktion der 2-dimensionalen Sinc-Funktion $\mathrm{sinc}((x-2)^2+(y-2)^2)$ mit Kosinus-Polynomen (links) nach 40 Iterationsschritten mit dem CGNE Verfahren. Rechts ist das Original im Intervall $[0,4]^2$ zu sehen. Für die NFCT wurden die gegebenen Knoten $\boldsymbol{x}$ auf das Intervall $[0,\frac{1}{2}]^2$ skaliert. Um den Abfall der Fourier-Koeffizienten zu steuern, wurden die multi-quadratischen Dämpfungsfaktoren $\hat{w}_{\boldsymbol{k}} = \sum_{p=0}^{1} \left[\left(\lfloor ||\boldsymbol{k}||_2 \rfloor + p\right)^2 + c^2\right]^{-\mu}$, $(\mu = 1.2, c = 0.8)$ verwendet. Die Funktion wurde an 8000 zufällig gewählten Punkte abgetastet. Interpoliert wurde an $M = 10000$ Punkten auf einem quadratischen, äquidistanten Gitter mit Polynomgrad $N = 256$ in beide Dimensionen. Die verwendete Fensterfunktion φ war die Kaiser-Bessel-Funktion mit Abschneideparameter $m = 6$.

8 Zusammenfassung

In dieser Arbeit haben wir die schnellen trigonometrischen Transformationen für nicht-äquidistante Knoten kennengelernt. Wir haben gesehen, dass die schnelle Kosinus- und Sinus-Transformation für nicht-äquidistante Knoten aus der NFFT entwickelt werden kann. Danach sind wir auf die inversen trigonometrischen Transformation an nicht-äquidistanten Knoten eingegangen. Diese Probleme konnten wir sowohl auf die gewichtete als auch gedämpfte Normalengleichung erster und zweiter Art zurückführen, die wir mit iterativen Gleichungssystemlösern, wie zum Beispiel CGNE oder CGNR, unter Verwendung der schnellen trigonometrischen Transformationen für die Matrix-Vektor-Multiplikation, gelöst haben.

Da die besprochenen Algorithmen, NFCT und NFST, Verfahren zur näherungsweisen Berechnung der Transformatierten sind, haben wir uns in Abschnitt 4 der Fehlerabschätzung gewidmet. Im darauffolgenden Abschnitt wurde die Anwendung der Software-Bibliothek besprochen.

Numerische Tests in Abschnitt 6 haben die Effizienz der entwickelten Verfahren gezeigt. Hier wurde nachgewiesen, dass die reellen trigonometrischen Transformationen NFCT und NFST im Falle reeller Eingabedaten gegenüber der NFFT kürzere Laufzeiten haben. Es wäre weiterhin interessant, ob mit spezialisierten Algorithmen zur Ausführung der diskreten Kosinus- bzw. Sinus-Transformation eine zusätzliche Effizienzsteigerung

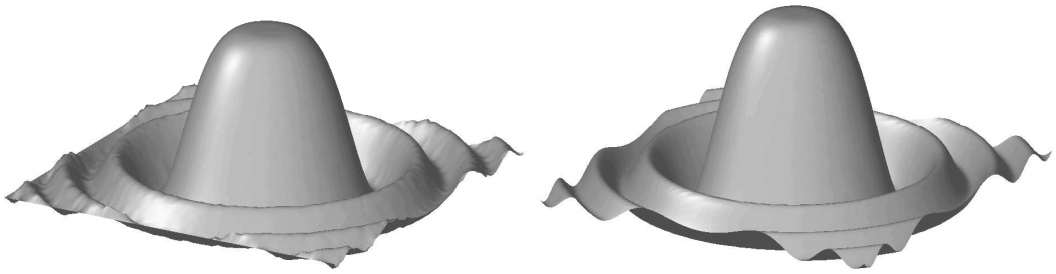

Abbildung 15: Rekonstruktion der 2-dimensionalen Sinc-Funktion $\mathrm{sinc}((x-2)^2+(y-2)^2)$ mit Sinus-Polynomen (links) nach 40 Iterationsschritten mit dem CGNE Verfahren. Rechts ist das Original im Intervall $[0,4]^2$ zu sehen. Für die NFST wurden die gegebenen Knoten $\boldsymbol{x}$ auf das Intervall $[0,\frac{1}{2}]^2$ skaliert. Um den Abfall der Fourier-Koeffizienten zu steuern, wurden die multi-quadratischen Dämpfungsfaktoren $\hat{w}_{\boldsymbol{k}} = \sum_{p=0}^{1} \left[\left(\lfloor ||\boldsymbol{k}||_2 \rfloor + p \right)^2 + c^2 \right]^{-\mu}$, $(\mu = 1.2, c = 0.8)$ verwendet. Die Funktion wurde an 8000 zufällig gewählten Punkten abgetastet. Interpoliert wurde an $M = 10000$ Punkten auf einem quadratischen, äquidistanten Gitter mit Polynomgrad $N = 256$ in beide Dimensionen. Die verwendete Fensterfunktion φ war die Kaiser-Bessel-Funktion mit Abschneideparameter $m = 6$.

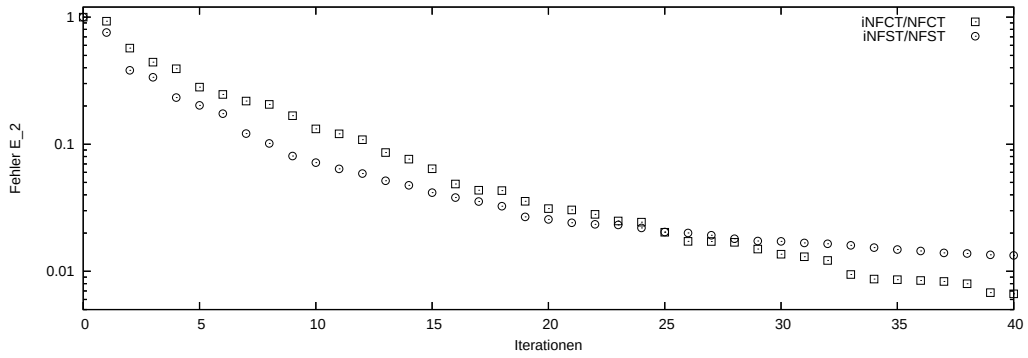

Abbildung 16: Fehler E_2 bei Rekonstruktion der Sinc-Funktion $\mathrm{sinc}((x-2)^2+(y-2)^2)$ an zufällig gewählten Knoten mit dem CGNR Verfahren. Verwendete Fensterfunktion: Kaiser-Bessel-Funktion (Abschneideparameter $m = 6$). Anzahl der Knoten ist $M = 8000$ und Polynomgrad $N_t = 256$, $(t = 0, 1)$. Die verwendeten inversen multi-quadratischen Dämpfungsfaktoren wurden mit den Parametern $\mu = 1.2$ und $c = 0.8$ berechnet.

möglich ist. Des Weiteren sind in diesem Abschnitt Grafiken zu Approximationsfehlern zu finden.

Im letzten Abschnitt haben wir zwei Anwendungsfälle betrachtet: zum einen Scattered Data Interpolation und zum anderen Funktionsrekonstruktion.

Literatur

[1] G.Beyklin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2:363 - 381, 1995.

[2] A. Björck. Numerical methods for least squares problems. SIAM Society for Industrial & Applied Mathematics, 1996.

[3] O. Davydov. http://www.maths.strath.ac.uk/∼aas04108/tsfit/index.html.

[4] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. SIAM *J. Sci. Stat. Comput.*, 14:1368 - 1393, 1993.

[5] M. Fenn and D. Potts. Fast summation based on trigonometric transforms at nonequispaced nodes. *Numer. Linear Algebra Appl.*, 2002.

[6] K. Fourmont. Non equispaced fast Fourier transforms with applications to tomography. Preprint Universität Münster, 2002.

[7] M.Frigo and S.G.Johnson. FFTW a C subroutine library. http://www.fftw.org.

[8] J. A. Fessler and B. P. Sutton. Nonuniform fast Fourier transforms using min-max interpolation. *IEEE Trans. Signal Process.*, 2002.

[9] T. Knopp. Rekonstruktion von Magnet-Resonanz-Tomographie Bildern. Studienarbeit, Preprint B-05-02, 2005.

[10] S. Kunis und D. Potts. NFFT 2.0. Preprint B-04-07, 2004.

[11] S. Kunis and D. Potts. Stability Results for Scattered Data Interpolation by Trigonometric Polynomials. Preprint, 2005.

[12] D. Potts. Schnelle Fourier Transformationen für nichtäquidistante Daten und Anwendungen. Habilitationsschrift, 2003.

[13] 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.

[14] G. Steidl. A note on fast Fourier transforms for nonequispaced grids. *Adv. Comput. Math.*, 9:337 - 353, 1998.