TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Mathematik
Professur für Angewandte Funktionalanalysis

Bachelorarbeit

Die Direkte Inverse NFFT

Melanie Kircheis

Chemnitz, den 9. März 2017

Prüfer: Prof. Dr. Daniel Potts
Dipl.-Math. Franziska Nestler

Kircheis, Melanie
Die Direkte Inverse NFFT
Bachelorarbeit, Fakultät für Mathematik
Technische Universität Chemnitz, März 2017

Inhaltsverzeichnis

1 Einleitung

Die NFFT, kurz für nicht-äquidistante schnelle Fourier-Transformation, ist ein schneller Algorithmus zur Auswertung eines trigonometrischen Polynoms

$$f(x) = \sum_{k=-\frac{M}{2}}^{\frac{M}{2}-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k x} \tag{1.1}$$

an nicht-äquidistant verteilten Knoten $y_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$, $j = 1, \ldots, N$. Für den Fall äquidistanter Knoten mit $M = N$ kann diese Auswertung mithilfe der FFT realisiert werden. Dieser schnelle Algorithmus ist invertierbar. Daher ist auch für die nicht-äquidistanten Daten eine Inversion gesucht, d. h. aus gegebenen Funktionswerten des trigonometrischen Polynoms (1.1) sollen nun die Fourierkoeffizienten $\hat{f}_k$ berechnet werden. Anders als bei der FFT ist für die NFFT die Anzahl N der Knoten jedoch unabhängig von der Anzahl M der Fourierkoeffizienten $\hat{f}_k$. Doch selbst im quadratischen Fall ist eine Invertierung nicht ohne Weiteres möglich.

Dennoch sind schon einige Ansätze zur Berechnung einer inversen NFFT (iNFFT) untersucht worden. Bereits in [6] wurde eine Methode erläutert, welche die Formel zur Lagrange'schen Interpolation sowie schnelle Multipol-Methoden (FMM) verwendet. Außerdem wurde in [13] das CG-Verfahren in Verbindung mit der NFFT genutzt, um einen iterativen Algorithmus zu entwerfen.

In dieser Arbeit soll nun für den Fall $M = N$ eine neue Methode zur direkten Berechnung einer iNFFT entwickelt werden. Dabei wird zunächst, analog zu [6], ein Ansatz mit Lagrange'scher Interpolation gewählt. Dazu kann ein Zusammenhang zwischen zwei Auswertungen des trigonometrischen Polynoms (1.1) an verschiedenen Punkten x_l und y_j genutzt werden. Dies liefert bereits die grundlegende Formel

$$g_l = c_l \sum_{j=1}^{N} f_j \, d_j \left(\frac{1}{\tan(\pi(x_l - y_j))} - \mathrm{i} \right), \quad l = 1, \ldots, N, \tag{1.2}$$

wobei

$$c_l = \prod_{n=1}^{N} \sin(\pi(x_l - y_n)) \;\text{ und }\; d_j = \prod_{\substack{n=1 \\ n \neq j}}^{N} \frac{1}{\sin(\pi(y_j - y_n))}, \quad l, j = 1, \ldots, N.$$

Werden die Punkte x_l nun äquidistant gewählt, so kann die Berechnung der iNFFT zurückgeführt werden auf die Berechnung der Koeffizienten c_l, d_j und g_l sowie anschließender Anwendung einer FFT. Anders als in [6] soll hier jedoch die schnelle

Summation zur Auswertung der Summen genutzt werden. Sind dabei die gewählten Knotenmengen disjunkt und die zugehörige nicht-äquidistante Fourier-Matrix

$$\boldsymbol{A} \coloneqq \left(\mathrm{e}^{2\pi \mathrm{i} k y_j}\right)_{j=1,\, k=-\frac{M}{2}}^{N,\, \frac{M}{2}-1} \in \mathbb{C}^{N \times M}$$

gut konditioniert, so ergibt sich ein Algorithmus mit Komplexität $\mathcal{O}(N \log N)$.

Die vorliegende Arbeit gliedert sich wie folgt. In Kapitel 2 wird zunächst auf einige wichtige Grundlagen aus dem Gebiet der Fourier-Analysis eingegangen. Das Hauptaugenmerk liegt dabei auf den bereits erwähnten Algorithmen, der NFFT und der schnellen Summation. In Abschnitt 3.1 folgt die Herleitung der Formel (1.2). Dies geschieht mithilfe der Lagrange'schen Interpolation, ähnlich wie in [6]. Daraus kann dann bereits der exakte Algorithmus 3.5 abgeleitet werden, gefolgt von ersten numerischen Experimenten. Dieser Algorithmus ist zwar theoretisch exakt, aber auch langsam. Darum soll im Anschluss nach einer effizienteren Methode gesucht werden, eine iNFFT zu berechnen. In dieser Arbeit wird dabei ein Ansatz mittels schneller Summation genutzt. Dieser wird in Abschnitt 3.2 zunächst nur auf die Koeffizienten g_l angewandt, in Abschnitt 3.3 wird dies jedoch auch auf die Koeffizienten c_l und d_j erweitert. Damit wird schließlich der effiziente Algorithmus 3.9 erhalten. Abschließend folgen noch einige weitere numerische Experimente.

2 Grundlagen

In diesem Kapitel sollen einige wichtige Begriffe aus dem Gebiet der Fourier-Analysis kurz erläutert werden. Neben dem Begriff der Fourier-Reihe werden zunächst zwei schnelle Algorithmen zur Berechnung einer diskreten Fourier-Transformation für verschieden verteilte Knoten eingeführt.

Auf Grundlage dessen kann im Anschluss die schnelle Summation definiert werden, welche im Weiteren eine wichtige Rolle spielen wird.

2.1 Fourier-Reihen

Betrachtet sei hier der Hilbert-Raum $L_2(\mathbb{T})$ aller 1-periodischen komplexwertigen Funktionen, wobei der sogenannte **Torus** gegeben ist durch $\mathbb{T} \coloneqq \mathbb{R}/\mathbb{Z} \simeq \left[-\frac{1}{2}, \frac{1}{2}\right)$.

Wichtig für die Theorie der Fourier-Reihen sind dabei Funktionen der Form

$$\left\{ \mathrm{e}^{2\pi \mathrm{i} k x} : k \in \mathbb{N} \right\}.$$

Diese bilden eine Orthonormalbasis von $L_2(\mathbb{T})$, für einen Beweis siehe z.B. [9, 20]. Damit ist jede Funktion $f \in L_2(\mathbb{T})$ eindeutig darstellbar in der Form

$$f(x) = \sum_{k \in \mathbb{Z}} c_k(f)\, \mathrm{e}^{2\pi \mathrm{i} k x}, \tag{2.1}$$

wobei die Summe in der $L_2(\mathbb{T})$-Norm gegen f konvergiert, vgl. [11, 21].

Definition 2.1 Eine Reihe der Gestalt (2.1) heißt **Fourier-Reihe**. Die Koeffizienten

$$c_k(f) \coloneqq \int_{-\frac{1}{2}}^{\frac{1}{2}} f(x)\, \mathrm{e}^{-2\pi \mathrm{i} k x}\, \mathrm{d}x, \quad k \in \mathbb{Z}, \tag{2.2}$$

werden **Fourierkoeffizienten** der Funktion f genannt. ◇

2.2 Schnelle Fourier-Transformation

Für Anwendungen werden die Koeffizienten (2.2) meist durch eine Quadraturformel angenähert. Dazu sei für $M \in 2\mathbb{N}$ die 1-periodische Funktion f an den M äquidistanten Punkten

$$x_j = \tfrac{j}{M} \in \left[-\tfrac{1}{2}, \tfrac{1}{2}\right), \quad j = -\tfrac{M}{2}, \ldots, \tfrac{M}{2} - 1,$$

betrachtet und die entsprechenden Funktionswerte seien mit $f_j \coloneqq f(x_j)$ bezeichnet. Durch Verwendung der Rechteckregel, welche im periodischen Fall der Trapezregel entspricht, können die Fourierkoeffizienten wie folgt approximiert werden.

$$c_k(f) \approx \hat{f}_k \coloneqq \frac{1}{M} \sum_{j=-\frac{M}{2}}^{\frac{M}{2}-1} f_j \, \mathrm{e}^{-2\pi \mathrm{i} jk/M}, \quad k \in \mathbb{Z}.$$

Dabei kann gezeigt werden, dass

$$\hat{f}_k \approx c_k(f) \quad \forall\, k = -\tfrac{M}{2}, \ldots, \tfrac{M}{2} - 1$$

als Näherung akzeptabel ist, siehe z.B. [11].

Definition 2.2 Die Abbildung

$$\mathbb{C}^M \to \mathbb{C}^M, \ \boldsymbol{f} \coloneqq \left(f_{-\frac{M}{2}}, \ldots, f_{\frac{M}{2}-1}\right) \mapsto \hat{\boldsymbol{f}} \coloneqq \left(\hat{f}_{-\frac{M}{2}}, \ldots, \hat{f}_{\frac{M}{2}-1}\right)$$

wird als **diskrete Fourier-Transformation (DFT)** der Ordnung M bezeichnet. In Matrix-Vektor-Notation kann die DFT auch notiert werden als $\hat{\boldsymbol{f}} = \boldsymbol{F}_M \boldsymbol{f}$ mit der M-ten **Fourier-Matrix**

$$\boldsymbol{F}_M = \left(\mathrm{e}^{-2\pi \mathrm{i} jk/M}\right)_{j,k=-\frac{M}{2}}^{\frac{M}{2}-1},$$

vgl. [20, 21]. Es kann gezeigt werden, dass diese invertierbar ist. Die **inverse diskrete Fourier-Transformation (iDFT)** ist gegeben durch

$$f_j = \sum_{k=-\frac{M}{2}}^{\frac{M}{2}-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} jk/M}, \quad j = -\tfrac{M}{2}, \ldots, \tfrac{M}{2} - 1.$$

◇

Soll nun tatsächlich eine DFT berechnet werden, so zeigt sich, dass dies schnell sehr aufwändig wird. So benötigt eine DFT der Länge M insgesamt M^2 Multiplikationen und $M(M-1)$ Additionen, also $\mathcal{O}(M^2)$ arithmetische Operationen. Dieser Aufwand ist für praktische Anwendungen jedoch indiskutabel.

Eine weitere Methode der Berechnung einer DFT ist die **schnelle Fourier-Transformation (FFT)**. Für $M \in 2^{\mathbb{N}}$ kann gezeigt werden, dass die Berechnung einer DFT der Länge M der Berechnung zweier DFT der Länge $\frac{M}{2}$ entspricht. Diese entsprechen wiederum der Berechnung von vier DFT der Länge $\frac{M}{4}$, usw. Mithilfe dieser Strategie ergibt sich also ein Aufwand von $\mathcal{O}(M \log M)$ arithmetischen Operationen, siehe auch [20, 19, 14].

2.3 Nicht-äquidistante schnelle Fourier-Transformation

Sind nun stattdessen nicht-äquidistante Daten gegeben, so wird ein weiterer schneller Algorithmus zur Berechnung der DFT an beliebigen Knoten benötigt. Dieser wird als **nicht-äquidistante schnelle Fourier-Transformation (NFFT)** bezeichnet und sei im Folgenden kurz erläutert, vgl. [5, 3, 19, 18, 17, 12].

Für gegebene Knoten $x_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$, $j = 1, \dots, N$, $M \in 2\mathbb{N}$, sowie beliebige Koeffizienten $\hat{f}_k \in \mathbb{C}$, $k = -\frac{M}{2}, \dots, \frac{M}{2}$, sei nun die Berechnung der Summen

$$f_j = \sum_{k=-\frac{M}{2}}^{\frac{M}{2}-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k x_j}, \quad j = 1, \dots, N, \tag{2.3}$$

bzw. das adjungierte Problem der Berechnung der Summen

$$h_k = \sum_{j=1}^{N} f_j \, \mathrm{e}^{-2\pi \mathrm{i} k x_j}, \quad k = -\frac{M}{2}, \dots, \frac{M}{2} - 1, \tag{2.4}$$

für gegebene Funktionswerte $f_j \coloneqq f(x_j) \in \mathbb{C}$ betrachtet.

Die Betrachtung sei zunächst nur auf das Problem (2.3) beschränkt. Dabei ist zu erkennen, dass dies äquivalent ist zur Auswertung eines trigonometrischen Polynoms

$$f(x) = \sum_{k=-\frac{M}{2}}^{\frac{M}{2}-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k x} \tag{2.5}$$

an gegebenen Punkten x_j, $j = 1, \dots, N$. Diese Funktion f kann nun approximiert werden durch eine Linearkombination von Translaten einer 1-periodischen Funktion $\tilde{\varphi}$, d. h.

$$f(x) \approx s_1(x) \coloneqq \sum_{l=-\frac{m}{2}}^{\frac{m}{2}-1} g_l \, \tilde{\varphi}\left(x - \frac{l}{m}\right),$$

wobei $m = \sigma M$ mit dem sogenannten **Oversampling-Faktor** $\sigma > 1$ gilt. Dies entspricht einer diskreten Faltung. Im einfachsten Fall entsteht $\tilde{\varphi}$ durch Periodisierung einer Funktion $\varphi\colon [-\frac{1}{2}, \frac{1}{2}) \to \mathbb{R}$. Diese sogenannte **Fensterfunktion** sei dabei so gewählt, dass die Fourier-Reihe ihrer periodisierten Funktion

$$\tilde{\varphi}(x) = \sum_{r \in \mathbb{Z}} \varphi(x + r)$$

gleichmäßig konvergiert. Mithilfe des Faltungssatzes und der Definition

$$\hat{g}_k \coloneqq \sum_{l=-\frac{m}{2}}^{\frac{m}{2}-1} g_l \, \mathrm{e}^{-2\pi \mathrm{i} k l / m}, \quad k \in \mathbb{Z},$$

kann s_1 im Fourier-Bereich nun geschrieben werden als

$$\begin{aligned} s_1(x) &= \sum_{k=-\infty}^{\infty} c_k(s_1)\, \mathrm{e}^{2\pi \mathrm{i} k x} \\ &= \sum_{k=-\infty}^{\infty} \hat{g}_k\; c_k(\tilde{\varphi})\, \mathrm{e}^{2\pi \mathrm{i} k x} \\ &= \sum_{k=-\frac{m}{2}}^{\frac{m}{2}-1} \hat{g}_k\; c_k(\tilde{\varphi})\, \mathrm{e}^{2\pi \mathrm{i} k x} + \sum_{\substack{r=-\infty \\ r\neq 0}}^{\infty} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}-1} \hat{g}_k\; c_{k+mr}(\tilde{\varphi})\, \mathrm{e}^{2\pi \mathrm{i}(k+mr)x}. \end{aligned} \tag{2.6}$$

Ein Vergleich zwischen (2.5) und (2.6) liefert die Intention für die folgende Definition.

$$\hat{g}_k \coloneqq \begin{cases} \dfrac{\hat{f}_k}{c_k(\tilde{\varphi})} & : k = -\frac{M}{2}, \ldots, \frac{M}{2} - 1, \\ 0 & : k = -\frac{m}{2}, \ldots, -\frac{M}{2} - 1, \frac{M}{2}, \ldots, \frac{m}{2} - 1. \end{cases}$$

Ist die Funktion φ außerdem klein außerhalb des Intervalls $\left[-\frac{n}{m}, \frac{n}{m}\right]$, $n \ll m$, so kann $\psi(x) = \chi_{\left[-\frac{n}{m}, \frac{n}{m}\right]} \cdot \varphi(x)$ gesetzt werden, wobei $\chi_{\left[-\frac{n}{m}, \frac{n}{m}\right]}$ die charakteristische Funktion des Intervalls $\left[-\frac{n}{m}, \frac{n}{m}\right]$ bezeichne. Damit kann $\tilde{\varphi}$ nun durch die 1-periodisierte Funktion $\tilde{\psi}$ mit

$$\tilde{\varphi}(x) \approx \tilde{\psi}(x) = \sum_{r \in \mathbb{Z}} \psi(x + r)$$

approximiert werden. Unter Beachtung der Definition der Funktion $\tilde{\psi}$ ergibt sich somit die folgende Approximation

$$f(x_j) \approx s_1(x_j) \approx s(x_j) \coloneqq \sum_{l=-\frac{m}{2}}^{\frac{m}{2}-1} g_l\, \tilde{\psi}\left(x_j - \tfrac{l}{m}\right) = \sum_{l=\lfloor m x_j \rfloor - n}^{\lceil m x_j \rceil + n} g_l\, \tilde{\psi}\left(x_j - \tfrac{l}{m}\right), \tag{2.7}$$

wobei sich die Vereinfachung dadurch ergibt, dass viele Summanden verschwinden.

Bemerkung 2.3 Als Fensterfunktionen geeignet sind z.B.

- zentrierte B-Splines

$$\varphi(x) \coloneqq B_{2n}(mx), \qquad \hat{\varphi}(k) = \frac{1}{m} \mathrm{sinc}^{2n}\left(\tfrac{\pi k}{m}\right),$$

- Gauß-Funktionen

$$\varphi(x) \coloneqq \frac{1}{\sqrt{\pi b}}\, \mathrm{e}^{-\frac{(mx)^2}{b}}, \qquad \hat{\varphi}(k) = \frac{1}{m}\, \mathrm{e}^{-b\left(\frac{\pi k}{m}\right)^2},$$

  wobei $b \coloneqq \frac{2\sigma}{2\sigma - 1} \frac{n}{\pi}$,

- sinc-Funktionen

$$\varphi(x) \coloneqq \frac{M(2\sigma-1)}{2n}\,\mathrm{sinc}^{2n}\left(\frac{\pi M x(2\sigma-1)}{2m}\right), \qquad \hat{\varphi}(k) = B_{2n}\left(\frac{2nk}{(2\sigma-1)M}\right),$$

- Kaiser-Bessel-Fenster

$$\varphi(x) \coloneqq \frac{1}{\pi}\begin{cases} \dfrac{\sinh(b\sqrt{n^2-m^2x^2})}{\sqrt{n^2-m^2x^2}} & : |x| \le \frac{n}{m}, \\ \dfrac{\sin(b\sqrt{m^2x^2-n^2})}{\sqrt{m^2x^2-n^2}} & : \text{sonst}, \end{cases}$$

$$\hat{\varphi}(k) = \begin{cases} \frac{1}{m}\,I_0\left(n\sqrt{b^2-\left(\frac{2\pi k}{m}\right)^2}\right) & : k = -m(1-\frac{1}{2\sigma}),\dots,m(1-\frac{1}{2\sigma}), \\ 0 & : \text{sonst}, \end{cases}$$

  wobei $b \coloneqq \pi(2-\frac{1}{\sigma})$ und I_0 die modifizierte Bessel-Funktion erster Gattung der Ordnung 0 bezeichnet.

Für weitere Details siehe z.B. [5, 3, 19, 4, 10, 8, 16, 12].

◇

Damit lässt sich der Algorithmus wie folgt formulieren.

Algorithmus 2.4 (NFFT)

Für $N \in \mathbb{N}$, $M \in 2\mathbb{N}$ seien Knoten $x_j \in \left[-\frac{1}{2},\frac{1}{2}\right)$, $j = 1,\dots,N$, sowie Koeffizienten $\hat{f}_k \in \mathbb{C}$ mit $k = -\frac{M}{2},\dots,\frac{M}{2}-1$ gegeben. Des Weiteren seien der Oversampling-Faktor $\sigma \ge 1$ und $m = \sigma M$.

1. Setze $\mathcal{O}(M)$

$$\hat{g}_k \coloneqq \begin{cases} \frac{\hat{f}_k}{\hat{\varphi}(k)} & : k \in \{-\frac{M}{2},\dots,\frac{M}{2}-1\}, \\ 0 & : k \in \{-\frac{m}{2},\dots,\frac{m}{2}-1\} \setminus \{-\frac{M}{2},\dots,\frac{M}{2}-1\}. \end{cases} \tag{2.8}$$

2. Berechne $\mathcal{O}(M \log M)$

$$g_l \coloneqq \frac{1}{m}\sum_{k=-\frac{m}{2}}^{\frac{m}{2}-1} \hat{g}_k\,\mathrm{e}^{2\pi i k l/m} \tag{2.9}$$

   für alle $l = -\frac{m}{2},\dots,\frac{m}{2}-1$ mittels der inversen FFT.

3. Berechne nun $\mathcal{O}(N)$

$$\tilde{f}_j \coloneqq \sum_{l=-\frac{m}{2}}^{\frac{m}{2}-1} g_l \, \tilde{\psi}\left(x_j - \tfrac{l}{m}\right), \quad j = 1, \ldots, N. \tag{2.10}$$

Ausgabe: $\tilde{f}_j \approx f_j$ aus (2.3), $j = 1, \ldots, N$.

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

Zu beachten ist hierbei, dass sich die Komplexität des zweiten Schrittes zunächst als $\mathcal{O}(m \log m)$ darstellt. Da aber $m = \sigma M$ mit einer Konstanten σ gilt, ist dies gleich $\mathcal{O}(M \log M)$. Ebenso ergibt sich im dritten Schritt zunächst ein Aufwand von $\mathcal{O}(m \cdot N)$. Mithilfe von (2.7) wird allerdings deutlich, dass pro Knoten höchstens $2n + 2$ Summanden ungleich null sein können. Da dies ebenfalls eine konstante Zahl ist, bleibt also ein Aufwand von $\mathcal{O}(N)$.

Als nächstes sei die Matrix-Vektor-Notation betrachtet. Analog zur DFT kann auch die NDFT aus (2.3) durch Definition der Vektoren

$$\boldsymbol{f} \coloneqq (f_j)_{j=1}^{N}, \qquad \hat{\boldsymbol{f}} \coloneqq (\hat{f}_k)_{k=-\frac{M}{2}}^{\frac{M}{2}-1}$$

sowie der **nicht-äquidistanten Fourier-Matrix**

$$\boldsymbol{A} \coloneqq \left(\mathrm{e}^{2\pi \mathrm{i} k x_j}\right)_{j=1,\, k=-\frac{M}{2}}^{N,\, \frac{M}{2}-1} \in \mathbb{C}^{N \times M}, \tag{2.11}$$

äquivalent geschrieben werden als $\boldsymbol{f} = \boldsymbol{A}\hat{\boldsymbol{f}}$. Die Betrachtung der einzelnen Schritte von Algorithmus 2.4 zeigt, dass auch diese durch Matrix-Vektor-Operationen darstellbar sind. Die Definition der zusätzlichen Vektoren

$$\boldsymbol{g} \coloneqq (g_l)_{l=-\frac{m}{2}}^{\frac{m}{2}-1}, \qquad \hat{\boldsymbol{g}} \coloneqq (\hat{g}_k)_{k=-\frac{m}{2}}^{\frac{m}{2}-1}$$

und

$$\tilde{\boldsymbol{f}} \coloneqq (\tilde{f}_j)_{j=1}^{N},$$

ergibt folgende Darstellungen.

(2.8)**:** $\hat{\boldsymbol{g}} = \boldsymbol{D}\hat{\boldsymbol{f}}$ mit der Diagonalmatrix

$$\boldsymbol{D} = \operatorname{diag}\left(\frac{1}{\hat{\varphi}(k)}\right)_{k=-\frac{M}{2}}^{\frac{M}{2}-1} \in \mathbb{C}^{M \times M},$$

(2.9): $\boldsymbol{g} = \boldsymbol{F}\hat{\boldsymbol{g}}$ mit der abgeschnittenen Fourier-Matrix

$$\boldsymbol{F} = \frac{1}{m}\left(\mathrm{e}^{2\pi \mathrm{i} k \frac{l}{m}}\right)_{l=-\frac{m}{2},\, k=-\frac{M}{2}}^{\frac{m}{2}-1,\ \frac{M}{2}-1} \in \mathbb{C}^{m\times M},$$

(2.10): $\tilde{\boldsymbol{f}} = \boldsymbol{B}\boldsymbol{g}$ mit der dünn besetzten Matrix

$$\boldsymbol{B} = \left(\tilde{\psi}(x_j - \tfrac{l}{m})\right)_{j=1,\, l=-\frac{m}{2}}^{N,\ \frac{m}{2}-1} \in \mathbb{R}^{N\times m}.$$

Damit ist durch die NFFT also die Approximation $\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{F}\boldsymbol{D}$ gegeben.

Nun sei auch wieder das adjungierte Problem (2.4) betrachtet. Dieses lässt sich ebenfalls mithilfe von

$$\boldsymbol{h} \coloneqq (h_k)_{k=-\frac{M}{2}}^{\frac{M}{2}-1}$$

darstellen als $\boldsymbol{h} = \boldsymbol{A}^*\boldsymbol{f}$, wobei $\boldsymbol{A}^*$ die adjungierte Matrix bezeichne, d. h. $\boldsymbol{A}^* = \overline{\boldsymbol{A}}^{\mathrm{T}}$. Somit ist durch $\boldsymbol{A}^* \approx \boldsymbol{D}^*\boldsymbol{F}^*\boldsymbol{B}^*$ auch eine Approximation der adjungierten NFFT gegeben. Es kann also folgender Algorithmus abgeleitet werden.

Algorithmus 2.5 (adjungierte NFFT)

Für $N \in \mathbb{N}$ seien Knoten $x_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$ sowie $f_j \in \mathbb{C}$, $j = 1, \ldots, N$, gegeben. Des Weiteren seien der Oversampling-Faktor $\sigma \geq 1$, $M \in 2\mathbb{N}$ und $m = \sigma M$.

1. Berechne
$$g_l \coloneqq \sum_{j=1}^{N} f_j\, \tilde{\psi}\left(x_j - \tfrac{l}{m}\right), \quad l = -\tfrac{m}{2}, \ldots, \tfrac{m}{2} - 1.$$
$\mathcal{O}(N)$

2. Berechne
$$\hat{g}_k \coloneqq \frac{1}{m} \sum_{l=-\frac{m}{2}}^{\frac{m}{2}-1} g_l\, \mathrm{e}^{-2\pi \mathrm{i} k l/m}$$
für alle $k = -\frac{m}{2}, \ldots, \frac{m}{2} - 1$ mithilfe der FFT. $\mathcal{O}(M \log M)$

3. Setze
$$\tilde{h}_k \coloneqq \frac{\hat{g}_k}{\hat{\varphi}(-k)}, \quad k \in \{-\tfrac{M}{2}, \ldots, \tfrac{M}{2} - 1\}.$$
$\mathcal{O}(M)$

Ausgabe: $\tilde{h}_k \approx h_k$ aus (2.4), $k = -\frac{M}{2}, \ldots, \frac{M}{2} - 1$.

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

2.4 Schnelle Summation

Betrachtet sei nun das Problem der schnellen Berechnung von Summen der Gestalt

$$f(y_j) = \sum_{k=1}^{N_x} \alpha_k \, K(y_j - x_k), \quad j = 1, \ldots, N_y, \tag{2.12}$$

wobei $x_k, y_j \in \mathbb{R}$ und K ein Kern aus $C^\infty(\mathbb{R})$ außer im Nullpunkt.

Für den Spezialfall äquidistant verteilter Knoten x_k, y_j ist eine Herleitung des schnellen Algorithmus' beschrieben in [17]. Betrachtet seien hier beliebig verteilte Knoten $x_k, y_j,$ vgl. [17, 7, 15]. Ohne Einschränkung der Allgemeinheit gelte

$$|x_k|, |y_j| < \tfrac{1}{4} - \tfrac{1}{2}\,\varepsilon_\mathrm{B}, \quad 0 \le \varepsilon_\mathrm{B} < \tfrac{1}{4}.$$

Damit folgt

$$|y_j - x_k| \le |y_j| + |x_k| \le \tfrac{1}{2} - \varepsilon_\mathrm{B},$$

wobei für 1-periodische Kerne $\varepsilon_\mathrm{B} = 0$ gewählt werden kann.

Sind stattdessen Punkte $x_k, y_j \in [-L, L)$ gegeben, so können diese mittels

$$\tilde{x}_k \;\coloneqq\; \left(\tfrac{1}{4L} - \tfrac{1}{2L}\,\varepsilon_\mathrm{B}\right) \cdot x_k =: c\, x_k \tag{2.13}$$

auf das Intervall $\left[-\frac{1}{4} + \frac{1}{2}\,\varepsilon_\mathrm{B},\ \frac{1}{4} - \frac{1}{2}\,\varepsilon_\mathrm{B}\right)$ skaliert werden. In diesem Fall muss die Skalierung auch in der Summation beachtet werden, d. h. es wird dann statt (2.12) die Summation

$$f(y_j) = \sum_{k=1}^{N_x} \alpha_k \, K\left(\frac{1}{c}(\tilde{y}_j - \tilde{x}_k)\right), \quad j = 1, \ldots, N_y, \tag{2.14}$$

betrachtet.

Um aus dem beliebigen, eventuell singulären Kern einen glatten, 1-periodischen Kern zu erhalten, sei zunächst die Regularisierung

$$K_\mathrm{R}(x) \coloneqq \begin{cases} K_\mathrm{I}(x) & : x \in [-\varepsilon_\mathrm{I}, \varepsilon_\mathrm{I}] \\ K_\mathrm{B}(x) & : x \in \left[\frac{1}{2} - \varepsilon_\mathrm{B}, \frac{1}{2} + \varepsilon_\mathrm{B}\right] \\ K(x) & : x \in \left(-\frac{1}{2} + \varepsilon_\mathrm{B}, -\varepsilon_\mathrm{I}\right) \cup \left(\varepsilon_\mathrm{I}, \frac{1}{2} - \varepsilon_\mathrm{B}\right) \end{cases}$$

gewählt, wobei $0 < \varepsilon_\mathrm{I} < \frac{1}{2} - \varepsilon_\mathrm{B} < \frac{1}{2}$. Dabei sollen die folgenden $4(p+1)$ Bedingungen für einen Glattheitsparameter $p \in \mathbb{N}$ erfüllt sein.

$$\begin{aligned} \frac{\mathrm{d}^s}{\mathrm{d}x^s} K_\mathrm{I}(x)\bigg|_{x=\pm\varepsilon_\mathrm{I}} &= \frac{\mathrm{d}^s}{\mathrm{d}x^s} K(x)\bigg|_{x=\pm\varepsilon_\mathrm{I}}, \quad s = 0, \ldots, p, \\ \frac{\mathrm{d}^s}{\mathrm{d}x^s} K_\mathrm{B}(x)\bigg|_{x=\frac{1}{2}\pm\varepsilon_\mathrm{B}} &= \frac{\mathrm{d}^s}{\mathrm{d}x^s} K(x)\bigg|_{x=\frac{1}{2}\pm\varepsilon_\mathrm{B}}, \quad s = 0, \ldots, p. \end{aligned}$$

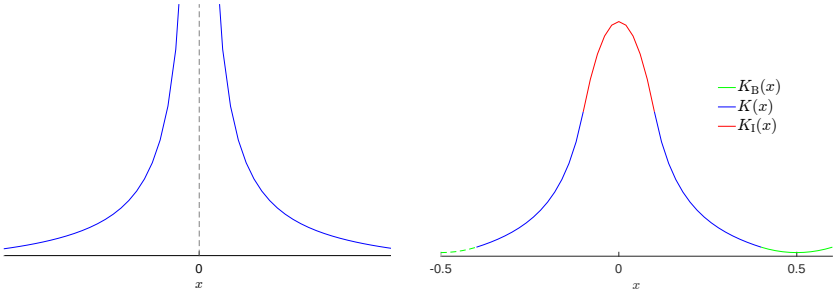


Abbildung 2.1: Beispiel einer Regularisierung für die Funktion $K(x) = \frac{1}{|x|}$, ursprüngliche Kernfunktion (links) und regularisierter Kern (rechts).

Somit ist $K_{\mathrm{R}}(x)$ p-mal stetig differenzierbar. Sei nun noch

$$\tilde{K}_{\mathrm{R}}(x) \coloneqq K_{\mathrm{R}}(x+r) \text{ mit } r \in \mathbb{Z} : \; x + r \in \left(-\tfrac{1}{2} + \varepsilon_{\mathrm{B}}, \tfrac{1}{2} + \varepsilon_{\mathrm{B}}\right]$$

gesetzt, so ergibt sich ein glatter, 1-periodischer Kern. Dieser wird nun durch eine Fourier-Reihe

$$\tilde{K}_{\mathrm{R}}(x) \approx K_{\mathrm{F}}(x) \coloneqq \sum_{l=-\frac{M}{2}}^{\frac{M}{2}-1} b_l \, \mathrm{e}^{2\pi \mathrm{i} l x} \tag{2.15}$$

approximiert, wobei

$$b_l \coloneqq \frac{1}{M} \sum_{j=-\frac{M}{2}}^{\frac{M}{2}-1} \tilde{K}_{\mathrm{R}}\left(\tfrac{j}{M}\right) \mathrm{e}^{2\pi \mathrm{i} j l / M}, \quad l = -\tfrac{M}{2}, \ldots, \tfrac{M}{2} - 1, \tag{2.16}$$

gewählt ist, d. h. die Koeffizienten b_l entstehen durch eine FFT angewendet auf den Vektor

$$\left(\tilde{K}_{\mathrm{R}}\left(\tfrac{j}{M}\right)\right)_{j=-\frac{M}{2}}^{\frac{M}{2}-1}.$$

Es sei nun folgende Schreibweise betrachtet

$$K = \underbrace{K - \tilde{K}_{\mathrm{R}}}_{=:K_{\mathrm{NE}}} + \underbrace{\tilde{K}_{\mathrm{R}} - K_{\mathrm{F}}}_{=:K_{\mathrm{ERR}}} + K_{\mathrm{F}},$$

wobei mit K_{ERR} der Fehler der Approximation $\tilde{K}_{\mathrm{R}} \approx K_{\mathrm{F}}$ und mit K_{NE} das sogenannte Nahfeld bezeichnet werden. Wird nun angenommen, der Fehler K_{ERR} werde hinreichend klein, so ergibt sich die Approximation

$$K \approx K_{\mathrm{NE}} + K_{\mathrm{F}}$$

und damit

$$f(y) \approx \underbrace{\sum_{k=1}^{N_x} \alpha_k \, K_{\mathrm{NE}}(y - x_k)}_{=: f_{\mathrm{NE}}(y)} + \underbrace{\sum_{k=1}^{N_x} \alpha_k \, K_{\mathrm{F}}(y - x_k)}_{=: f_{\mathrm{F}}(y)} =: \tilde{f}(y).$$

Weiterhin wird vorausgesetzt, dass jedes Intervall der Länge $2\varepsilon_{\mathrm{I}}$ höchstens ν Knoten x_k bzw. ν Knoten y_j enthält, d. h. die Knoten seien gleichmäßig verteilt. Da

$$|y_j - x_k| < \tfrac{1}{2} - \varepsilon_{\mathrm{B}}$$

und

$$\operatorname{supp}(K - \tilde{K}_{\mathrm{R}}) \cap \left[-\tfrac{1}{2} + \varepsilon_{\mathrm{B}}, \tfrac{1}{2} - \varepsilon_{\mathrm{B}}\right] = [-\varepsilon_{\mathrm{I}}, \varepsilon_{\mathrm{I}}],$$

gilt in vielen Fällen $(K - \tilde{K}_{\mathrm{R}})(y_j - x_k) = 0$. Durch die eben getroffene Voraussetzung kann die Summe

$$f_{\mathrm{NE}}(y_j) = \sum_{k=1}^{N_x} \alpha_k \, (K - \tilde{K}_{\mathrm{R}})(y_j - x_k), \quad j = 1, \ldots, N_y,$$

also in nur $\nu \cdot N_y = \mathcal{O}(N_y)$ Operationen berechnet werden. Für die zweite Summe $f_{\mathrm{F}}(y_j)$ wird (2.15) genutzt und durch Tauschen der Summationsreihenfolge schließlich Folgendes erhalten.

$$\begin{aligned} f_{\mathrm{F}}(y_j) &= \sum_{k=1}^{N_x} \alpha_k \, K_{\mathrm{F}}(y_j - x_k) = \sum_{k=1}^{N_x} \alpha_k \sum_{l=-\frac{M}{2}}^{\frac{M}{2}-1} b_l \, \mathrm{e}^{2\pi \mathrm{i} l (y_j - x_k)} \\ &= \sum_{l=-\frac{M}{2}}^{\frac{M}{2}-1} b_l \left(\sum_{k=1}^{N_x} \alpha_k \, \mathrm{e}^{-2\pi \mathrm{i} l x_k} \right) \mathrm{e}^{2\pi \mathrm{i} l y_j}. \end{aligned}$$

Dabei können die Terme in Klammern zunächst mittels einer adjungierten NFFT berechnet werden. Danach folgt eine simple Multiplikation mit b_l und schließlich kann die verbleibende Summation durch eine NFFT durchgeführt werden.

Ein Algorithmus kann demnach wie folgt formuliert werden.

Algorithmus 2.6 (Schnelle Summation)

Für $N_x, N_y \in \mathbb{N}$, $\varepsilon_\mathrm{B} < 1/4$, $\varepsilon_\mathrm{I} < 1/2 - \varepsilon_\mathrm{B}$ seien Knoten x_k, y_j gegeben mit $|x_k|, |y_j| < 1/4 - 1/2\,\varepsilon_\mathrm{B}$, $k = 1, \ldots, N_x$, $j = 1, \ldots, N_y$. Weiterhin seien $\alpha_k \in \mathbb{C}$, $M \in 2\mathbb{N}$ und K ein Kern, welcher mindestens p-mal stetig differenzierbar ist.

Vorberechnung: (i) b_l nach (2.16)
(ii) $K_\mathrm{NE}(y_j - x_k) \quad \forall j, k : |y_j - x_k| \le \varepsilon_\mathrm{I}$

1. Berechne
$$\tilde{\alpha}_l \coloneqq \sum_{k=1}^{N_x} \alpha_k \, \mathrm{e}^{-2\pi \mathrm{i} l x_k}$$
für alle $l = -\frac{M}{2}, \ldots, \frac{M}{2}$ mithilfe einer adjungierten NFFT.
$\mathcal{O}(M \log M + N_x)$

2. Setze
$$\beta_l \coloneqq b_l \, \tilde{\alpha}_l, \quad l = -\tfrac{M}{2}, \ldots, \tfrac{M}{2}.$$
$\mathcal{O}(M)$

3. Berechne
$$f_\mathrm{F}(y_j) = \sum_{l=-\frac{M}{2}}^{\frac{M}{2}-1} \beta_l \, \mathrm{e}^{2\pi \mathrm{i} l y_j}$$
für alle $j = 1, \ldots, N_y$ mithilfe einer NFFT.
$\mathcal{O}(M \log M + N_y)$

4. Berechne
$$f_\mathrm{NE}(y_j) = \sum_{k=1}^{N_x} \alpha_k \, K_\mathrm{NE}(y_j - x_k), \quad j = 1, \ldots, N_y.$$
$\mathcal{O}(\nu \cdot N_y)$

5. Setze $\tilde{f}(y_j) \coloneqq f_\mathrm{F}(y_j) + f_\mathrm{NE}(y_j)$. $\mathcal{O}(N_y)$

Ausgabe: $\tilde{f}(y_j) \approx f(y_j)$ aus (2.12), $j = 1, \ldots, N_y$.

Komplexität: $\mathcal{O}(M \log M + N_x + N_y)$

3 Die Direkte Inverse NFFT

Gesucht sei nun eine Inversion der NFFT, d. h. eine Möglichkeit der Berechnung der Fourierkoeffizienten $\hat{f}_k \in \mathbb{C}$, $k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1$, aus gegebenen Funktionswerten eines trigonometrischen Polynoms

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

Dies soll analog zu [6] mithilfe der Lagrange-Interpolation realisiert werden. In [6] wurde dabei eine schnelle inverse NFFT mithilfe von Multipol-Methoden entwickelt. Hier soll hingegen ein Ansatz mit schneller Summation gewählt werden. Dabei kann ein Zusammenhang zwischen

$$f_j \coloneqq f(y_j) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k y_j}, \quad j = 1, \ldots, N, \tag{3.1}$$

und

$$g_l \coloneqq f(x_l) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k x_l}, \quad l = 1, \ldots, N,$$

für verschiedene Knoten x_l und y_j genutzt werden, welcher im nachfolgenden Abschnitt hergeleitet wird. Im Anschluss können zur Reduzierung des Aufwands noch weitere Modifikationen vorgenommen werden.

3.1 Herleitung des Algorithmus'

Bemerkung 3.1 Für gegebene Punkte $z_1, \ldots, z_N \in \mathbb{C}$ mit $z_j \neq z_k$ für alle $j \neq k$ sind die Lagrange'schen Grundpolynome gegeben durch

$$l_j(z) \coloneqq \prod_{\substack{n=1 \\ n \neq j}}^{N} \frac{z - z_n}{z_j - z_n}.$$

Nach dieser Definition gilt $l_j(z_k) = \delta_{jk}$. Damit kann das zugehörige Interpolationspolynom N-ten Grades eindeutig angegeben werden als

$$p(z) = \sum_{j=1}^{N} f(z_j)\ l_j(z). \tag{3.2}$$

Diese Darstellung sowie die beiden nachfolgenden Lemmata können genutzt werden, um die in Satz 3.4 behauptete Formel zu beweisen.

◇

Lemma 3.2 *Es seien $x_l, y_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$ sowie $w_l = \mathrm{e}^{2\pi \mathrm{i} x_l}$ und $z_j = \mathrm{e}^{2\pi \mathrm{i} y_j}$ für $l, j = 1, \ldots, N$. Dann gilt für $j \in \{1, \ldots, N\}$ beliebig*

$$\prod_{\substack{n=1 \\ n\neq j}}^{N} (w_l - z_n) = w_l^{(N-1)/2} \cdot \prod_{\substack{n=1 \\ n\neq j}}^{N} z_n^{1/2} \cdot 2\mathrm{i}\, \sin(\pi(x_l - y_n)), \quad l = 1, \ldots, N, \tag{3.3}$$

sowie

$$\prod_{\substack{n=1 \\ n\neq j}}^{N} (z_j - z_n) = z_j^{(N-1)/2} \cdot \prod_{\substack{n=1 \\ n\neq j}}^{N} z_n^{1/2} \cdot 2\mathrm{i}\, \sin(\pi(y_j - y_n)), \quad j = 1, \ldots, N. \tag{3.4}$$

Beweis.

$$\begin{aligned}
\prod_{\substack{n=1 \\ n\neq j}}^{N} (w_l - z_n) &= \prod_{\substack{n=1 \\ n\neq j}}^{N} (\mathrm{e}^{2\pi \mathrm{i} x_l} - \mathrm{e}^{2\pi \mathrm{i} y_n}) = \prod_{\substack{n=1 \\ n\neq j}}^{N} \mathrm{e}^{\pi \mathrm{i}(x_l + y_n)} \cdot \left(\mathrm{e}^{\pi \mathrm{i}(x_l - y_n)} - \mathrm{e}^{-\pi \mathrm{i}(x_l - y_n)}\right) \\
&= \mathrm{e}^{(N-1)\pi \mathrm{i}\, x_l} \cdot \prod_{\substack{n=1 \\ n\neq j}}^{N} \mathrm{e}^{\pi \mathrm{i} y_n} \cdot 2\mathrm{i}\, \sin(\pi(x_l - y_n)) \\
&= w_l^{(N-1)/2} \cdot \prod_{\substack{n=1 \\ n\neq j}}^{N} z_n^{1/2} \cdot 2\mathrm{i}\, \sin(\pi(x_l - y_n)).
\end{aligned} \tag{3.5}$$

Analog ergibt sich (3.4) indem in (3.5) w_l durch z_j sowie x_l durch y_j substituiert wird.

□

Lemma 3.3 *Es seien $x_l, y_j \in \left[-\frac{1}{2}, \frac{1}{2}\right), f_j \in \mathbb{C}$ sowie $w_l = \mathrm{e}^{2\pi \mathrm{i} x_l}$ und $z_j = \mathrm{e}^{2\pi \mathrm{i} y_j}$ für $l, j = 1, \ldots, N$. Dann gilt für $l = 1, \ldots, N$ die Beziehung*

$$\sum_{j=1}^{N} f_j \cdot \prod_{\substack{n=1 \\ n\neq j}}^{N} \frac{w_l - z_n}{z_j - z_n} = w_l^{N/2} \cdot c_l \sum_{j=1}^{N} f_j \cdot z_j^{-N/2}\ d_j \cdot \left(\frac{1}{\tan(\pi(x_l - y_j))} - \mathrm{i}\right), \tag{3.6}$$

wobei c_l und d_j wie folgt definiert sind

$$c_l = \prod_{n=1}^{N} \sin(\pi(x_l - y_n)), \quad l = 1, \dots, N, \tag{3.7}$$

$$d_j = \prod_{\substack{n=1 \\ n \neq j}}^{N} \frac{1}{\sin(\pi(y_j - y_n))}, \quad j = 1, \dots, N. \tag{3.8}$$

Beweis. Division von (3.3) durch (3.4) liefert

$$\prod_{\substack{n=1 \\ n \neq j}}^{N} \frac{w_l - z_n}{z_j - z_n} = \frac{w_l^{(N-1)/2}}{z_j^{(N-1)/2}} \prod_{\substack{n=1 \\ n \neq j}}^{N} \frac{z_n^{1/2} \cdot 2\mathrm{i} \sin(\pi(x_l - y_n))}{z_n^{1/2} \cdot 2\mathrm{i} \sin(\pi(y_j - y_n))} \tag{3.9}$$

$$= \frac{w_l^{-1/2}}{z_j^{-1/2}} \cdot \frac{1}{\sin(\pi(x_l - y_j))} \cdot \underbrace{\frac{w_l^{N/2}}{z_j^{N/2}}}_{= w_l^{N/2} z_j^{-N/2}} \underbrace{\frac{\prod_{n=1}^{N} \sin(\pi(x_l - y_n))}{\prod_{n \neq j} \sin(\pi(y_j - y_n))}}_{= c_l \cdot d_j}. \tag{3.10}$$

Nach Definition gilt folgende Umformung.

$$\frac{w_l^{-1/2}}{z_j^{-1/2}} = \frac{\left(\mathrm{e}^{2\pi \mathrm{i} x_l}\right)^{-1/2}}{\left(\mathrm{e}^{2\pi \mathrm{i} y_j}\right)^{-1/2}} = \frac{\mathrm{e}^{-\pi \mathrm{i} x_l}}{\mathrm{e}^{-\pi \mathrm{i} y_j}} = \mathrm{e}^{-\pi \mathrm{i}(x_l - y_j)}.$$

Mithilfe der Euler'schen Formel ergibt sich damit

$$\frac{w_l^{-1/2}}{z_j^{-1/2}} \cdot \frac{1}{\sin(\pi(x_l - y_j))} = \frac{\mathrm{e}^{-\pi \mathrm{i}(x_l - y_j)}}{\sin(\pi(x_l - y_j))} = \frac{\cos(\pi(x_l - y_j)) - \mathrm{i} \sin(\pi(x_l - y_j))}{\sin(\pi(x_l - y_j))}$$

$$= \frac{1}{\tan(\pi(x_l - y_j))} - \mathrm{i}. \tag{3.11}$$

Kombination von (3.10) und (3.11) liefert schließlich die gesuchte Formel (3.6). □

Aus folgendem Satz ergibt sich nun der Zusammenhang zwischen den Werten einer Fourier-Reihe für verschiedene Mengen von Punkten.

Satz 3.4 *Es seien $x_l, y_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$, $w_l = \mathrm{e}^{2\pi \mathrm{i} x_l}$ und $z_j = \mathrm{e}^{2\pi \mathrm{i} y_j}$ für $l, j = 1, \dots, N$, sowie $\hat{f}_k \in \mathbb{C}$, $k = -\frac{N}{2}, \dots, \frac{N}{2} - 1$. Weiterhin seien f_j und g_l wie folgt definiert.*

$$f_j \coloneqq \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k y_j},$$

$$g_l \coloneqq \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k x_l}.$$

Dann gilt die folgende Beziehung, wobei c_l und d_j wie in (3.7) *und* (3.8) *definiert sind.*

$$g_l = c_l \sum_{j=1}^{N} f_j \, d_j \left(\frac{1}{\tan(\pi(x_l - y_j))} - \mathrm{i} \right), \quad l = 1, \ldots, N. \tag{3.12}$$

Beweis. Gegeben sei das Polynom

$$P(z) = \sum_{k=0}^{N-1} \hat{f}_{k-\frac{N}{2}} \cdot z^k.$$

Mithilfe der Formel zur Lagrange'schen Interpolation (3.2) und unter Verwendung von Lemma 3.3 wird die folgende Identität erhalten.

$$\begin{aligned} P(w_l) &\overset{(3.2)}{=} \sum_{j=1}^{N} P(z_j) \cdot \prod_{\substack{n=1 \\ n \neq j}}^{N} \frac{w_l - z_n}{z_j - z_n} \\ &\overset{(3.6)}{=} w_l^{N/2} \cdot c_l \sum_{n=1}^{N} P(z_j) \cdot z_j^{-N/2} \, d_j \cdot \left(\frac{1}{\tan(\pi(x_l - y_j))} - \mathrm{i} \right) \end{aligned} \tag{3.13}$$

Außerdem gilt nach Definition

$$P(z_j) = \sum_{k=0}^{N-1} \hat{f}_{k-\frac{N}{2}} \cdot z_j^k = z_j^{N/2} \cdot \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k \, z_j^k = z_j^{N/2} \cdot f_j \tag{3.14}$$

und

$$P(w_l) = \sum_{k=0}^{N-1} \hat{f}_{k-\frac{N}{2}} \cdot w_l^k = w_l^{N/2} \cdot \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k \, w_l^k = w_l^{N/2} \cdot g_l. \tag{3.15}$$

Einsetzen von (3.14) und (3.15) in (3.13) liefert schließlich (3.12).

□

Soll nun zu gegebenen nicht-äquidistanten Punkten y_j eine inverse NFFT berechnet werden, so kann dies realisiert werden durch die Wahl zusätzlicher Punkte x_l und Anwendung der eben erhaltenen Formel (3.12). Werden die Knoten x_l dabei äquidistant gewählt, so können die gesuchten Fourierkoeffizienten $\hat{f}_k$ leicht mithilfe einer FFT aus den Daten g_l berechnet werden.

Obige Überlegungen führen somit zu folgendem Algorithmus.

Algorithmus 3.5 (Exakte Inverse NFFT)

Gegeben seien $N \in 2\mathbb{N}$, äquidistante Knoten $x_l \in \left[-\frac{1}{2}, \frac{1}{2}\right), l = 1, \ldots, N$, nicht-äquidistant verteilte Knoten $y_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$ sowie $f_j = f(y_j) \in \mathbb{C}$, $j = 1, \ldots, N$.

1. Berechne g_l nach (3.12). $\mathcal{O}(N^2)$
2. Berechne
$$\check{f}_k = \frac{1}{N} \sum_{l=1}^{N} g_l \, \mathrm{e}^{-2\pi \mathrm{i} k x_l}, \quad k = -\tfrac{N}{2}, \ldots, \tfrac{N}{2},$$
mithilfe einer FFT. $\mathcal{O}(N \log N)$

Ausgabe: $\check{f}_k \approx \hat{f}_k$ aus (3.1).

Komplexität: $\mathcal{O}(N^2)$

Dieser Algorithmus ist durch die FFT zwar bereits schnell in Schritt 2, die Berechnungen des ersten Schrittes werden jedoch langsam durchgeführt, weshalb insgesamt nur ein langsamer Algorithmus entsteht.

Dabei ist allerdings zu beachten, dass die Mengen der x_l und der y_j keine gemeinsamen Punkte enthalten dürfen. Ist dies verletzt, so entstehen bei der Berechnung der c_l und d_j Faktoren gleich Null, was insbesondere im Falle der d_j nach (3.8) nicht erlaubt ist, da dies Division durch Null bedeuten würde. Der hier untersuchte Ansatz zur Invertierung ist also nur möglich für disjunkte Knotenmengen.

Durch weitere Betrachtung von Algorithmus 3.5 ist erkennbar, dass dieser auch in Matrix-Vektor-Notation dargestellt werden kann. Betrachtet sei zunächst der erste Schritt. Dieser ist mithilfe der Matrizen

$$\boldsymbol{T} = \left(\frac{1}{\tan(\pi(x_l - y_j))} - \mathrm{i} \right)_{l,j=1}^{N} \quad \text{sowie}$$

$$\boldsymbol{D}_c = \operatorname{diag}(c_1, \ldots, c_N) \quad \text{und} \quad \boldsymbol{D}_d = \operatorname{diag}(d_1, \ldots, d_N)$$

darstellbar als $\boldsymbol{g} = \boldsymbol{D}_c \boldsymbol{T} \boldsymbol{D}_d \boldsymbol{f}$. Der zweite Schritt wiederum entspricht $\check{\boldsymbol{f}} = \boldsymbol{F}_N \boldsymbol{g}$, wobei $\boldsymbol{F}_N$ die bereits eingeführte N-te Fourier-Matrix bezeichnet. Damit ergibt sich also insgesamt die Darstellung $\check{\boldsymbol{f}} = \boldsymbol{F}_N \boldsymbol{D}_c \boldsymbol{T} \boldsymbol{D}_d \boldsymbol{f}$. Wie schon zuvor in Abschnitt 2.3 erläutert, kann die NDFT aus (3.1) mit der nicht-äquidistanten Fourier-Matrix aus (2.11) äquivalent geschrieben werden als $\boldsymbol{f} = \boldsymbol{A}\hat{\boldsymbol{f}}$. Ist $\boldsymbol{A}$ invertierbar, würde dies $\boldsymbol{A}^{-1}\boldsymbol{f} = \hat{\boldsymbol{f}}$ bedeuten. Mittels der eben gefundenen Darstellung des Algorithmus' ist somit die Faktorisierung $\boldsymbol{A}^{-1} = \boldsymbol{F}_N \boldsymbol{D}_c \boldsymbol{T} \boldsymbol{D}_d$ gegeben. Damit ist Algorithmus 3.5 also theoretisch exakt. Der approximative Charakter entsteht erst numerisch bedingt.

Beispiel 3.6 Zu gegebenen äquidistanten Knoten

$$x_l = \frac{l-1}{N} - \frac{1}{2} \in \left[-\tfrac{1}{2}, \tfrac{1}{2}\right), \quad l = 1, \ldots, N, \tag{3.16}$$

soll nun überprüft werden, für welche Knoten $y_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$ mit dieser Strategie eine gute Näherung der Fourierkoeffizienten erreicht werden kann. Dazu seien hier verwackelte äquidistante Knoten

$$y_j = -\frac{1}{2} + \frac{j-1}{N} + \frac{1}{4N}\,\theta, \quad j = 1, \ldots, N \text{ mit } \theta \sim U(0,1), \tag{3.17}$$

Tschebyscheff-Knoten

$$y_j = \frac{1}{2}\cos\left(\frac{2(N-j)+1}{2N}\,\pi\right), \quad j = 1, \ldots, N, \tag{3.18}$$

und logarithmisch verteilte Knoten

$$y_j = \left(\frac{6}{5}\right)^{j-N} - \frac{1}{2}, \quad j = 1, \ldots, N, \tag{3.19}$$

betrachtet, wobei U(0,1) die Gleichverteilung auf dem Intervall $(0, 1)$ bezeichne. Im hier dargestellten Experiment wurden zunächst beliebige Fourierkoeffizienten $\hat{f}_k \in [1, 100]$, $k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1$, gewählt und damit die Funktionsauswertungen f_j des trigonometrischen Polynoms nach (3.1) berechnet. Mithilfe dieser Daten können nun durch Algorithmus 3.5 die Näherungswerte $\check{f}_k$ berechnet werden, welche mit den vorgegebenen Werten für $\hat{f}_k$ verglichen werden sollen.

Im Folgenden sind für verschiedene $N \in 2\mathbb{N}$ die jeweiligen Fehler aufgelistet. Dabei wurden sowohl maximale absolute und mittlere absolute Fehler, d. h.

$$\max_{-\frac{N}{2} \le k \le \frac{N}{2}-1} |\hat{f}_k - \check{f}_k| \quad \text{und} \quad \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |\hat{f}_k - \check{f}_k|,$$

als auch maximale relative und mittlere relative Fehler betrachtet, also

$$\max_{-\frac{N}{2} \le k \le \frac{N}{2}-1} \frac{|\hat{f}_k - \check{f}_k|}{|\hat{f}_k|} \quad \text{und} \quad \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \frac{|\hat{f}_k - \check{f}_k|}{|\hat{f}_k|}.$$

Zusätzlich sind auch die Konditionszahlen $\mathrm{cond}_2(\boldsymbol{A})$ der nicht-äquidistanten Fourier-Matrix $\boldsymbol{A}$ angegeben, welche wie folgt definiert sind

$$\mathrm{cond}_2(\boldsymbol{A}) = \|\boldsymbol{A}\|_2 \, \|\boldsymbol{A}^{-1}\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}}.$$

Dabei bezeichnet $\|\boldsymbol{A}\|_2$ die Spektralnorm von $\boldsymbol{A}$ sowie $\sigma_{\max}, \sigma_{\min}$ den größten bzw. kleinsten Singulärwert der Matrix $\boldsymbol{A}$.

N	max. abs. Fehler	max. rel. Fehler	mittl. abs. Fehler	mittl. rel. Fehler	$\mathrm{cond}_2(\boldsymbol{A})$
16	2.41e-13	2.53e-14	8.09e-14	5.55e-15	1.49e+00
256	7.18e-12	3.94e-13	1.14e-12	3.40e-14	1.49e+00
512	1.56e-11	1.66e-12	2.45e-12	7.66e-14	1.50e+00
1018	3.05e-11	2.52e-11	6.30e-12	2.46e-13	1.56e+00
1020	NaN	NaN	NaN	NaN	1.56e+00

Tabelle 3.1: Fehler bei der Inversion sowie Konditionszahl der nicht-äquidistanten Fourier-Matrix $\boldsymbol{A}$ für zufällig verwackelte äquidistante Knoten, siehe (3.17), bei verschiedenen Größen N.

N	max. abs. Fehler	max. rel. Fehler	mittl. abs. Fehler	mittl. rel. Fehler	$\mathrm{cond}_2(\boldsymbol{A})$
8	3.10e-13	2.68e-13	1.61e-13	3.65e-14	8.51e+01
16	1.10e-09	3.05e-11	4.57e-10	9.75e-12	1.22e+05
24	4.86e-06	2.38e-06	1.65e-06	1.28e-07	2.80e+08
32	1.00e-02	9.14e-03	2.91e-03	3.60e-04	7.69e+11
36	1.64e-01	1.33e-02	4.69e-02	1.93e-03	4.18e+13
40	4.01e+01	4.80e+00	1.07e+01	4.35e-01	2.87e+15

Tabelle 3.2: Fehler bei der Inversion sowie Konditionszahl der nicht-äquidistanten Fourier-Matrix $\boldsymbol{A}$ für Tschebyscheff-Knoten, siehe (3.18), bei verschiedenen Größen N.

N	max. abs. Fehler	max. rel. Fehler	mittl. abs. Fehler	mittl. rel. Fehler	$\mathrm{cond}_2(\boldsymbol{A})$
8	2.21e-13	1.51e-14	1.69e-13	5.36e-15	2.47e+01
16	7.40e-10	2.20e-10	3.87e-10	2.41e-11	4.08e+05
20	6.28e-05	2.99e-06	2.99e-05	8.57e-07	3.27e+09
22	3.83e-03	5.16e-04	1.72e-03	8.19e-05	7.69e+11
24	2.80e+00	6.67e-01	1.22e+00	6.27e-02	3.52e+14

Tabelle 3.3: Fehler bei der Inversion sowie Konditionszahl der nicht-äquidistanten Fourier-Matrix $\boldsymbol{A}$ für logarithmische Knoten, siehe (3.19), bei verschiedenen Größen N.

Es ist zu erkennen, dass es dabei einen großen Unterschied macht, welche Verteilung der Punkte zugrunde liegt. Für verwackelte Knoten zeigt sich, dass die Fehler auch für große Knotenanzahlen nahezu gleichbleibend klein sind, siehe Tabelle 3.1. Unter Beachtung der Konditionszahlen wird dies leicht erklärbar, denn diese sind für alle betrachteten Problemgrößen klein und nahezu gleich. Bei Tschebyscheff-Knoten, siehe Tabelle 3.2, oder logarithmisch verteilten Knoten, siehe Tabelle 3.3, ist hingegen leicht zu sehen, dass in diesen Fällen nur für kleine Anzahlen von Punkten eine gute Approximation erzielt werden kann. Mit zunehmender Problemgröße wachsen die Fehler hier sehr schnell an. Werden wieder die Konditionszahlen betrachtet, so zeigt sich, dass diese beiden Probleme schlichtweg schlecht konditioniert sind. Bessere Ergebnisse können also nicht erwartet werden.

Doch selbst bei dem gut konditionierten Problem der verwackelten Knoten können ab einer gewissen Größe keine sinnvollen Resultate mehr erzielt werden. In Tabelle 3.1 ist deutlich zu erkennen, dass ab $N = 1020$ keine Berechnungen mehr möglich sind, obwohl sich die Kondition nicht verändert hat. Dies ist darauf zurückzuführen, dass die Koeffizienten c_l dann so klein bzw. die Koeffizienten d_j so groß werden, dass bei der Berechnung von (3.12) nahezu der unbestimmte Ausdruck $0 \cdot \infty$ erhalten wird. Diese Art der Berechnung ist also numerisch instabil. Später zeigt sich jedoch, dass dies verbessert werden kann und auch für größere Anzahlen noch gute Ergebnisse erzielt werden können, siehe Beispiel 3.12.

◇

Beispiel 3.7 Bisher wurde für f stets ein trigonometrisches Polynom verwendet. Nun seien für f auch andere Funktionen erlaubt und es soll überprüft werden, ob damit ebenfalls gute Ergebnisse erzielt werden können. In diesem Fall seien die B-Splines gewählt, welche wie folgt definiert sind. Als zentrierter B-Spline der Ordnung 1 wird die charakteristische Funktion des Intervalls $[-\frac{1}{2}, \frac{1}{2})$ bezeichnet, d.h.

$$B_1(x) := \begin{cases} 1 : x \in [-\frac{1}{2}, \frac{1}{2}), \\ 0 : \text{sonst.} \end{cases}$$

Für $b > 1$ wird die Funktion $B_b(\cdot)$ zentrierter B-Spline der Ordnung b genannt, welche durch Faltung von B_1 mit B_{b-1} entsteht, d.h. also

$$B_b(x) := \int_{-\infty}^{\infty} B_1(y)\ B_{b-1}(x-y)\ \mathrm{d}y = \int_{-\frac{1}{2}}^{\frac{1}{2}} B_{b-1}(x-y)\ \mathrm{d}y, \quad b > 1.$$

Nun gilt für diese Funktionen $\operatorname{supp}(B_b) = [-\frac{b}{2}, \frac{b}{2}]$. In vorliegendem Beispiel seien jedoch alle auf das Intervall $[-\frac{1}{2}, \frac{1}{2}]$ skaliert. Damit kann nun eine inverse NFFT mittels Algorithmus 3.5 durchgeführt werden.

a) Für verschiedene Größen $N = 2^a$ mit $a = 1, \ldots, 9$ soll überprüft werden, wie gut die Funktion f durch ein trigonometrisches Polynom mit den berechneten Fourierkoeffizienten approximiert werden kann. Dazu seien zunächst die zentrierten äquidistanten Knoten

$$\tilde{y}_j = \frac{2j - N - 1}{2N}, \quad j = 1, \ldots, N,$$

gewählt. Diese werden in beide Richtungen verwackelt, d. h. es entstehen Knoten

$$y_j = \tilde{y}_j + \alpha \cdot \frac{2\theta - 1}{N}, \quad j = 1, \ldots, N,$$

wobei $\alpha < \frac{1}{2}$ und $\theta \sim U(0, 1)$ gleichverteilt. An diesen verwackelten Knoten werde nun der gewählte B-Spline ausgewertet, dies ergibt die Funktionswerte f_j. Werden zusätzlich die Knoten x_l wie in Beispiel 3.6 gewählt, so sind damit alle Daten gegeben und es können mittels Algorithmus 3.5 die Koeffizienten $\check{f}_k$ berechnet werden. Daraus kann nun das trigonometrische Polynom

$$t_{\mathrm{N}}(x) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \check{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k x}$$

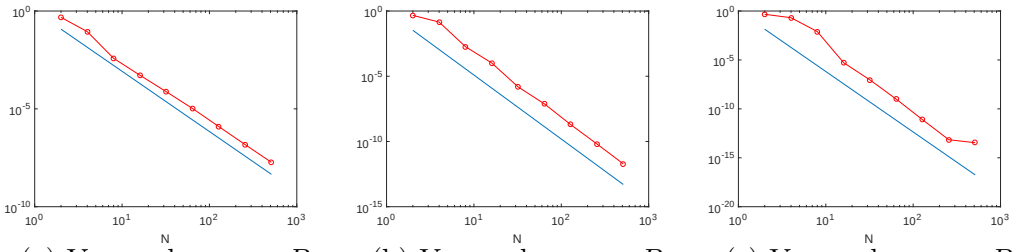
aufgestellt werden. Für dieses soll der maximale Fehler

$$\|f - t_N\|_\infty = \max |f(x) - t_{\mathrm{N}}(x)|$$

berechnet werden, welcher hier approximativ durch Wahl $20N + 1$ vieler äquidistant verteilter Knoten aus dem Intervall $[-\frac{1}{2}, \frac{1}{2}]$ bestimmt sei.

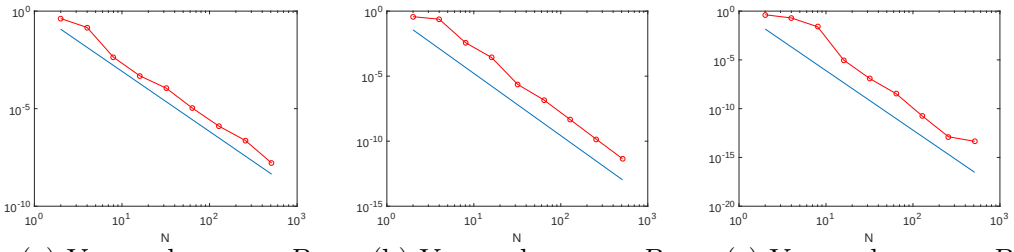
Wird dieses Experiment für alle gegebenen Größen N durchgeführt, so kann aus den Daten eine Fehlerkurve erzeugt werden. Mittels linearer Regression kann nun die Steigung o dieser Kurve bestimmt werden. Für die B-Splines der Ordnung 4, 6 und 8 sind in den Abbildungen 3.1 und 3.2 sowohl die Fehlerkurven als auch die entsprechenden Kurven N^o für $\alpha = 0.05$ bzw. $\alpha = 0.49$ abgebildet. Zusätzlich sind in Tabelle 3.4 die Anstiege der Ausgleichsgeraden für B-Splines verschiedener Ordnung und für verschiedene Wahl von α zu sehen.

In [2] ist dafür die Abschätzung $\|f - t_N\|_\infty = \mathcal{O}(N^{4\alpha-(b-1)})$ gegeben, falls f zweimal stetig differenzierbar ist. Es wird sogar vermutet, dass statt 4α auch 2α möglich ist. Im hier betrachteten Beispiel ist jedoch zu erkennen, dass die Größe von $\alpha < \frac{1}{2}$ so gut wie keine Rolle zu spielen scheint. Die Anstiege bleiben nahezu gleich. Die Glattheit scheint jedoch trotzdem den entsprechenden Einfluss zu haben. Es ist zu sehen, dass die Ordnung des Abfalls ca. bei $b-1$ liegt. Für sehr hohe B-Spline-Grade ist dies nach Tabelle 3.4 nicht erfüllt. Wird aber auch die



(a) Verwendung von B_4 (b) Verwendung von B_6 (c) Verwendung von B_8

Abbildung 3.1: Fehlerkurven bzgl. $\|\,.\,\|_\infty$ und entsprechende Ausgleichsgerade für die Approximation von B-Splines durch das trigonometrische Polynom t_N mit $\alpha = 0.05$, geplottet für steigendes N.



(a) Verwendung von B_4 (b) Verwendung von B_6 (c) Verwendung von B_8

Abbildung 3.2: Fehlerkurven bzgl. $\|\,.\,\|_\infty$ und entsprechende Ausgleichsgerade für die Approximation von B-Splines durch das trigonometrische Polynom t_N mit $\alpha = 0.49$, geplottet für steigendes N.

zugehörige Fehlerkurve wie in Abbildung 3.1(c) bzw. 3.2(c) betrachtet, so fällt auf, dass dort bereits die höchste Genauigkeit im Fehler erreicht wird und sich somit kein weiterer Abfall einstellen kann.

b) Analog zu Teil a) seien wieder die Knoten $\tilde{y}_j$ gewählt. Diese sollen jedoch nun nicht mehr zufällig, sondern wie auch in [1] nach einem bestimmten Muster verwackelt werden. Dabei sei mit `1` eine Verschiebung nach rechts und mit `-1` eine Verschiebung nach links bezeichnet. Wird also z.B. das Muster `[1 -1]` gewählt, so bedeutet dies $y_j = \tilde{y}_j + \frac{\alpha}{N}$ für ungerade j sowie $y_j = \tilde{y}_j - \frac{\alpha}{N}$ für gerade j. Abhängig von der Länge des gewählten Musters seien dabei stets 25 verschiedene Anzahlen N der Knoten getestet, um die Asymptotik zu untersuchen.

Bei Betrachtung des Musters `[1 -1]` ergeben sich die Anstiege der Ausgleichsgeraden wie in Tabelle 3.5 zu sehen. Für den B-Spline der Ordnung 4 zeigt Tabelle 3.6 zusätzlich für einige Muster die Anstiege der ermittelten Ausgleichsgeraden. Dabei ist auch hier zu erkennen, dass weder das Muster noch der Grad

α der Verwacklung eine Rolle zu spielen scheinen. Der beobachtete Abfall ist weiterhin ausschließlich abhängig vom Grad des gewählten B-Splines.

Zusätzlich zeigt Tabelle 3.7 die Konditionszahlen der nicht-äquidistanten Fourier-Matrix A bei Verwacklung nach verschiedenen Mustern. Hierbei ist es jedoch ausreichend, für jedes Muster einen Wert anzugeben, da die Konditionszahl für alle getesteten Größen N gleich bleibt. ◇

Grad b des B-Splines	Anstieg $\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.5$
2	-1.0361	-1.0262	-0.9818
3	-2.1264	-2.1727	-2.1033
4	-3.0894	-3.0527	-3.1387
5	-4.0598	-4.0368	-4.0071
6	-4.9370	-4.9094	-4.8451
7	-5.7962	-5.7655	-5.6891
8	-6.2008	-6.1820	-6.0085

Tabelle 3.4: Anstieg der Ausgleichsgeraden bei Approximation von B-Splines durch das mittels Algorithmus 3.5 berechnete trigonometrische Polynom t_{N} an zufällig verwackelten Knoten.

3.2 Effiziente Berechnung der Koeffizienten g_l

In sämtlichen bisherigen Betrachtungen wurden stets alle Berechnungen, welche die nicht-äquidistanten Daten betreffen, direkt und damit auch langsam durchgeführt. Der nächste Schritt ist also die Approximation der hier benötigten Koeffizienten. Dazu soll zunächst die schnelle Summation, siehe Abschnitt 2.4, genutzt werden, um die Berechnung der g_l aus (3.12) effizienter zu gestalten.

Betrachtet sei nochmals die Berechnungsvorschrift dieser Koeffizienten. Durch einfache Umformungen ergibt sich folgende Darstellung, wobei in (3.20) $K(x) = \cot(\pi x)$ gesetzt wurde.

$$\begin{aligned} g_l &= c_l \sum_{j=1}^{N} f_j\, d_j \left(\frac{1}{\tan(\pi(x_l - y_j))} - \mathrm{i} \right) \\ &= c_l \left(\sum_{j=1}^{N} \underbrace{f_j\, d_j}_{=:\alpha_j} \cdot \cot(\pi(x_l - y_j)) - \mathrm{i} \cdot \sum_{j=1}^{N} f_j\, d_j \right) \end{aligned}$$

Grad b des B-Splines	Anstieg $\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.4999$
2	-1.0361	-1.0553	-1.0772
3	-2.1264	-2.1947	-2.1742
4	-3.0895	-3.0516	-3.0749
5	-4.0598	-3.9738	-4.0003
6	-4.9370	-4.8512	-4.7703
7	-5.7963	-5.7257	-5.0168
8	-6.2009	-6.1190	-4.9693

Tabelle 3.5: Anstieg der Ausgleichsgeraden bei Approximation von B-Splines durch das mittels Algorithmus 3.5 berechnete trigonometrische Polynom t_{N} für Knoten verwackelt nach dem Muster `[1 -1]`.

Muster	Anstieg $\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.4999$
`[1 1 -1]`	-2.9499	-2.9333	-2.9339
`[1 1 -1 -1]`	-2.9590	-2.9126	-2.8246
`[1 1 1 -1 -1]`	-3.0274	-3.0406	-3.0501
`[1 -1 1 -1 -1]`	-3.0274	-3.0335	-3.0392
`[1 1 1 -1 -1 -1]`	-3.0324	-3.0577	-3.0936

Tabelle 3.6: Anstieg der Ausgleichsgeraden bei Approximation des B-Splines der Ordnung 4 durch das mittels Algorithmus 3.5 berechnete trigonometrische Polynom t_{N} für verschiedene Muster.

Muster	Konditionszahl $\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.4999$	$\alpha = 0.5$
`[1 1 -1]`	1.00e+00	2.62e+00	8.27e+03	3.49e+31
`[1 1 -1 -1]`	1.00e+00	2.95e+00	1.27e+04	9.89e+16
`[1 1 1 -1 -1]`	1.00e+00	3.08e+00	1.51e+04	1.13e+16
`[1 -1 1 -1 -1]`	1.00e+00	2.65e+00	8.54e+03	1.94e+16
`[1 1 1 -1 -1 -1]`	1.00e+00	3.27e+00	1.91e+04	4.62e+60

Tabelle 3.7: Konditionszahlen der nicht-äquidistanten Fourier-Matrix $\boldsymbol{A}$ für Knoten verwackelt nach verschiedenen Mustern.

$$= c_l \left(\sum_{j=1}^{N} \alpha_j \, K(x_l - y_j) - \mathrm{i} \cdot \sum_{j=1}^{N} f_j \, d_j \right), \quad l = 1, \ldots, N. \tag{3.20}$$

Die erste Summe kann nun mittels schneller Summation berechnet werden. Die zweite Summe stellt ein einfaches Skalarprodukt zweier Vektoren dar, welches ebenfalls ohne viel Aufwand berechnet werden kann. Nachfolgend sei dementsprechend ausschließlich die Berechnung der Summe

$$\tilde{g}_l \coloneqq \sum_{j=1}^{N} \alpha_j \, \cot(\pi(x_l - y_j)), \quad l = 1, \ldots, N, \tag{3.21}$$

betrachtet, woraus mittels

$$g_l = c_l \left(\tilde{g}_l - \mathrm{i} \cdot \sum_{j=1}^{N} f_j \, d_j \right), \quad l = 1, \ldots, N, \tag{3.22}$$

die gesuchten Koeffizienten ermittelt werden können.

Um nun Algorithmus 2.6 verwenden zu können, sei zunächst das damit zu lösende Problem etwas näher betrachtet. Gegeben sind Knoten $x_l, y_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$, damit liegen die Differenzen $x_l - y_j$ also im Intervall $(-1, 1)$. Durch Betrachtung der Funktion $f(x) = \cot(\pi x)$ auf diesem Intervall, siehe Abbildung 3.3, ist jedoch zu erkennen, dass diese dort nicht nur die Singularität im Ursprung besitzt, sondern noch Weitere an den Rändern des Intervalls. Die schnelle Summation ist jedoch nur für eine Singularität ausgelegt. Somit ist die Berechnung von $\tilde{g}_l$ also nicht ohne Weiteres möglich.

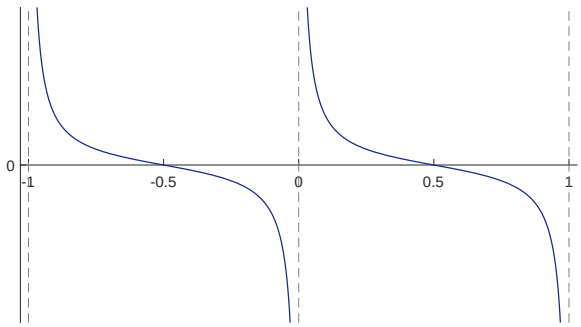


Abbildung 3.3: Funktion $f(x) = \cot(\pi x)$ im Intervall $[-1, 1]$.

Zusätzlich werden für die schnelle Summation Punkte innerhalb des kleineren Intervalls $(-\frac{1}{4}+\frac{1}{2}\,\varepsilon_\mathrm{B}, \frac{1}{4}-\frac{1}{2}\,\varepsilon_\mathrm{B})$ benötigt, sodass die Differenzen im Intervall $[-\frac{1}{2}+\varepsilon_\mathrm{B}, \frac{1}{2}-\varepsilon_\mathrm{B}]$

liegen. Dies lässt sich im vorliegenden Fall jedoch nicht durch bloße Skalierung mittels (2.13) und (2.14) beheben, da dies wiederum die Funktion skaliert und damit die äußeren Singularitäten nur näher an den Ursprung geschoben werden.

Es muss also eine Möglichkeit gefunden werden, die Knoten so aufzuteilen, dass die Differenzen im richtigen Intervall landen und die Singularitäten am Rand für die Berechnung keine Rolle mehr spielen. Dazu sei zunächst noch bemerkt, dass der Cotangens bereits eine 1-periodische Funktion ist. Somit kann, wie bereits erläutert, in der Regularisierung für die schnelle Summation auf die Approximation der Funktion am Rand verzichtet werden, d. h. es kann $\varepsilon_\mathrm{B} = 0$ gesetzt werden.

Durch Wahl der folgenden Aufteilung ergeben sich die in Tabelle 3.8 abgebildeten Differenzen.

$$x_l, y_j \in \begin{cases} \left[-\frac{1}{2}, -\frac{1}{4}\right) & \text{(I)} \\ \left[-\frac{1}{4}, \frac{1}{4}\right) & \text{(II)} \\ \left[\frac{1}{4}, \frac{1}{2}\right) & \text{(III)} \end{cases}$$

Dabei ist zu erkennen, dass nur in zwei (rot markierten) Fällen eine der äußeren Singularitäten eine Rolle spielt. Diese beiden Fälle müssen gesondert behandelt werden, da auch trotz der Periodisierung diese Differenzen vom Algorithmus nicht verarbeitet werden können. In den restlichen (grün markierten) Fällen kann die schnelle Summation wie gewohnt benutzt werden. Zu beachten ist hierbei, dass für gewöhnlich $|x_l - y_j| < \frac{1}{2} - \varepsilon_\mathrm{B} = \frac{1}{2}$ gefordert wird. Dennoch stellen auch die Kombinationen auf den „Nebendiagonalen“ von Tabelle 3.8 kein Problem dar, da in diesem Fall eine periodische Funktion vorausgesetzt wurde.

Die einzige Einschränkung, die sich hieraus ergibt, ist die Bedingung $\varepsilon_\mathrm{I} \leq \frac{1}{4}$. Ist dies verletzt, so rutschen Differenzen ins Nahfeld einer der äußeren Singularitäten, was bei der Nahfeldkorrektur durch die schnelle Summation nicht beachtet werden würde, da diese nur ein Nahfeld im Nullpunkt berücksichtigt. Dies ist jedoch nicht wirklich eine Beschränkung, da der Parameter ε_I prinzipiell möglichst klein gewählt sein sollte, um die direkte Berechnung des Nahfelds so gering wie möglich zu halten.

x_l \ y_j	(I)	(II)	(III)
(I)	$\left(-\frac{1}{4}, \frac{1}{4}\right)$	$\left(-\frac{3}{4}, 0\right)$	$\left(-1, -\frac{1}{2}\right)$
(II)	$\left(0, \frac{3}{4}\right)$	$\left(-\frac{1}{2}, \frac{1}{2}\right)$	$\left(-\frac{3}{4}, 0\right)$
(III)	$\left(\frac{1}{2}, 1\right)$	$\left(0, \frac{3}{4}\right)$	$\left(-\frac{1}{4}, \frac{1}{4}\right)$

Tabelle 3.8: Differenzen $x_l - y_j$ der Punkte x_l und y_j.

Laut Tabelle 3.8 sind also nun 9 Fälle gegeben. Diese können allerdings noch zu fünf Fällen zusammengefasst werden, siehe Tabelle 3.9. Ist x_l aus (II), so treten keinerlei Probleme auf, d. h. es kann für alle y_j die schnelle Summation durchgeführt werden (Fall 3). Für x_l aus (I) bzw. (III) gibt es jeweils einen Sonderfall, der nicht ohne Weiteres behandelt werden kann. Bei diesen beiden Fällen (Fall 2 und Fall 4) muss eine andere Möglichkeit gefunden werden, während der Rest (Fall 1 und 5) wie gewohnt berechnet werden kann.

x_l \ y_j	(I)	(II)	(III)
(I)	Fall 1		Fall 2
(II)	Fall 3		
(III)	Fall 4	Fall 5	

Tabelle 3.9: Einteilung der Fälle.

Es soll nun also die Berechnung für die einzelnen Fälle näher betrachtet werden. Dazu sei zunächst x_l aus (I). Werden hierfür die Mengen

$$L_1 := \left\{ l \in \{1, \dots, N\} \,:\, x_l \in \left[-\tfrac{1}{2}, -\tfrac{1}{4}\right) \right\}, \tag{3.23}$$

$$J_1 := \left\{ j \in \{1, \dots, N\} \,:\, y_j \in \left[-\tfrac{1}{2}, \tfrac{1}{4}\right) \right\} \text{ und} \tag{3.24}$$

$$J_2 := \{1, \dots, N\} \setminus J_1 \tag{3.25}$$

definiert, so kann für diese x_l die Summation folgendermaßen durchgeführt werden.

$$\tilde{g}_l = \sum_{j \in J_1} \alpha_j \cot(\pi(x_l - y_j)) + \sum_{j \in J_2} \alpha_j \cot(\pi((x_l + 1) - y_j)), \quad l \in L_1. \tag{3.26}$$

Die Verschiebung der Knoten im zweiten Teil ist dabei erlaubt, da der Cotangens eine π-periodische Funktion ist und hier exakt um eine Periode verschoben wird. Auf diese Weise kann erreicht werden, dass die Differenzen nun statt in $\left(-1, -\frac{1}{2}\right)$ im Intervall $\left(0, \frac{1}{2}\right)$ liegen, womit wieder die Singularität im Nullpunkt betrachtet wird.

Sind x_l aus (II), so wird nur die Menge

$$L_2 := \left\{ l \in \{1, \dots, N\} \,:\, x_l \in \left[-\tfrac{1}{4}, \tfrac{1}{4}\right) \right\} \tag{3.27}$$

benötigt, sodass

$$\tilde{g}_l \coloneqq \sum_{j=1}^{N} \alpha_j \cot(\pi(x_l - y_j)), \quad l \in L_2. \tag{3.28}$$

Für x_l aus (III) werden die Mengen

$$L_3 \coloneqq \left\{ l \in \{1, \dots, N\} \,:\, x_l \in \left[\tfrac{1}{4}, \tfrac{1}{2}\right) \right\}, \tag{3.29}$$

$$J_3 \coloneqq \left\{ j \in \{1, \dots, N\} \,:\, y_j \in \left[\tfrac{1}{4}, \tfrac{1}{2}\right) \right\} \text{ und} \tag{3.30}$$

$$J_4 \coloneqq \{1, \dots, N\} \setminus J_3 \tag{3.31}$$

eingeführt, sodass die Summation hier analog zu (I) geschrieben werden kann als

$$\tilde{g}_l \coloneqq \sum_{j \in J_3} \alpha_j \cot(\pi(x_l - y_j)) + \sum_{j \in J_4} \alpha_j \cot(\pi((x_l - 1) - y_j)), \quad l \in L_3. \tag{3.32}$$

Beispiel 3.8 Analog zu Beispiel 3.6 soll nun ebenfalls die Güte der Approximation für äquidistante Knoten x_l und verwackelte Knoten y_j überprüft werden. Gegeben sind dabei verschiedene Parameter, für die nun einige Experimente betrachtet seien. Zum einen gibt es die Parameter der schnellen Summation, d. h. die Anzahl der Knoten $N \in 2\mathbb{N}$, den Glattheitsparameter $p \in \mathbb{N}$ sowie $\varepsilon_{\mathrm{I}} \leq \frac{1}{4}$; zum anderen weitere Größen der NFFT, nämlich die Fensterbreite n und die Gittergröße m. Zusätzlich ist auch der Parameter ε_{B} gegeben, dieser ist jedoch wie zuvor bereits erläutert für dieses Problem auf Null gesetzt.

Bei allen dargestellten Experimenten wurden aufgrund größerer Schwankungen bei wiederholter Durchführung die Versuche jeweils 10-mal wiederholt und danach das Maximum der maximalen relativen bzw. absoluten Fehler gebildet.

Zunächst sei die Auswirkung einer Veränderung des Nahfeldparameters ε_{I} untersucht. Dafür wurden $n = p = 4$ und $m = 80$ fest gewählt und Werte von $\varepsilon_{\mathrm{I}} = 0.01$ bis $\varepsilon_{\mathrm{I}} = 0.25$ getestet. Wie in Abbildung 3.4 zu erkennen, können für alle möglichen Knotenanzahlen die relativen Fehler hinreichend klein gemacht werden. Außerdem ist gut zu sehen, dass für sehr kleine ε_{I} der Fehler recht groß wird, während die Genauigkeit bei größer werdendem ε_{I} immer weiter zunimmt. Dies ist auch nicht weiter überraschend, da durch ein kleines Nahfeld die Singularität mehr Einfluss gewinnt und damit größere Sprünge überbrückt werden müssen. Ist dagegen ein großes Nahfeld gegeben, so muss für viele Knoten die Nahfeldkorrektur direkt berechnet werden. Dies erhöht natürlich die Genauigkeit, ist jedoch auch aufwändiger. Außerdem sollte die schnelle Summation gerade die direkte Berechnung ersetzen.

In diesem Versuch wurden die minimalen Fehler bei $\varepsilon_{\mathrm{I}} \approx 0.2$ angenommen. Dies entspricht auch dem theoretischen Ergebnis in [17], dass eine gute Wahl des Nahfeldparameters von der Form $\varepsilon_{\mathrm{I}} = c \cdot \frac{2p}{m}$ ist. In diesem Fall wäre also $c = 2$ zu setzen.

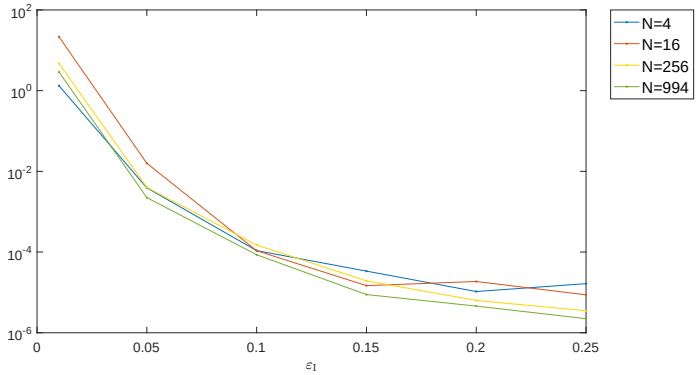


Abbildung 3.4: Maximale relative Fehler für verschiedene Größen von $0 < \varepsilon_I \leq \frac{1}{4}$ bei unterschiedlicher Anzahl N verwackelter äquidistanter Knoten mit $n = p = 4$ und $m = 80$.

Als nächstes sei die Veränderung bei verschiedenen Anzahlen m, der durch die NFFT berechneten Fourierkoeffizienten, untersucht. Dabei wurden hier Größen von $m = 2^a$ mit $a = 6, \ldots, 10$ betrachtet und jeweils das passende $\varepsilon_I = \frac{4p}{m}$ gewählt. Nach wie vor gelte auch hier $n = p = 4$. Abbildung 3.5 zeigt nun sowohl die damit berechneten relativen als auch die absoluten Fehler. Es ist zu sehen, dass die Fehler mit steigender Gittergröße leicht zunehmen, aber weiterhin ungefähr in der gleichen Größenordnung bleiben. Dies ist unter anderem darauf zurückzuführen, dass sich bei größerem Gitter das Nahfeld verkleinert und damit die Berechnung, wie bereits gesehen, schwieriger wird.

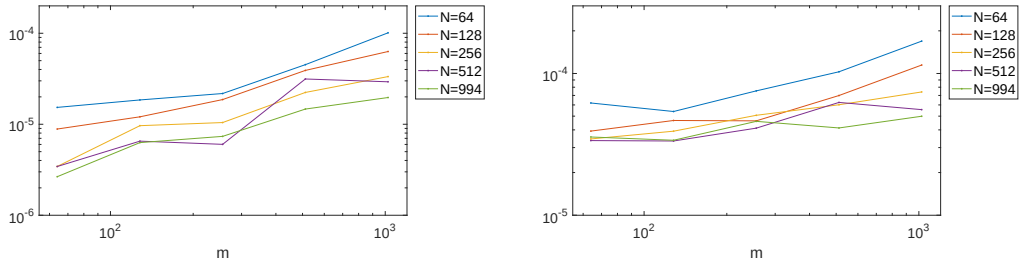


Abbildung 3.5: Maximale relative (links) und maximale absolute Fehler (rechts) für verschiedene Größen des Parameters m bei unterschiedlicher Anzahl N verwackelter äquidistanter Knoten mit $n = p = 4$ und $\varepsilon_I = \frac{4p}{m}$.

Es sei nun noch der Einfluss des Glattheitsparameters p untersucht. Werden für diesen die Werte $1, \ldots, 10$ sowie $n = 4, m = 256$ und $\varepsilon_I = \frac{4p}{m}$ gewählt, so ergeben sich Fehler wie in Abbildung 3.6 zu sehen. Dabei ist klar zu erkennen, dass durch einen höheren Wert noch deutlich bessere Ergebnisse erzielt werden können.

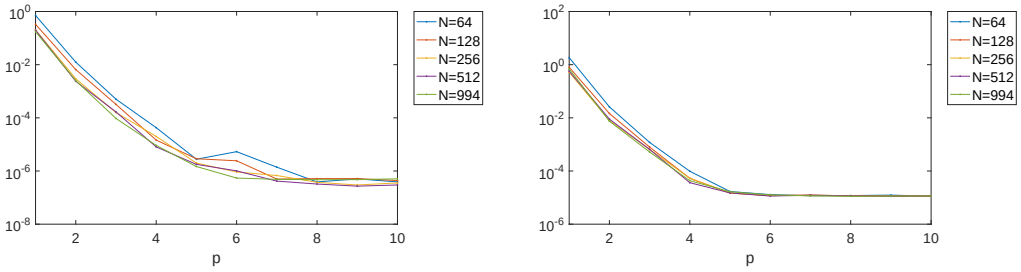


Abbildung 3.6: Maximale relative (links) und maximale absolute Fehler (rechts) für verschiedene Größen des Parameters p bei unterschiedlicher Anzahl N verwackelter äquidistanter Knoten mit $n = 4$, $m = 256$ und $\varepsilon_{\mathrm{I}} = \frac{4p}{m}$.

Bei Veränderung der Fenstergröße n können auch bei höheren Werten kaum Verbesserungen im Fehler erzielt werden. Daher sei darauf hier nicht näher eingegangen. ◇

3.3 Effiziente Berechnung der Koeffizienten c_l und d_j

Nun sollen auch die Koeffizienten c_l und d_j mithilfe der schnellen Summation effizienter berechnet werden. Bei der Betrachtung von (3.7) und (3.8) fällt jedoch auf, dass diese Berechnungsvorschriften gar keine Summen enthalten, somit also nicht von vornherein mittels Algorithmus 2.6 berechnet werden können. Durch Logarithmieren und Anwendung der Logarithmengesetze ergibt sich jedoch

$$\begin{aligned}
\ln c_l &= \ln\left(\prod_{n=1}^{N} \sin(\pi(x_l - y_n))\right) \quad \text{und} \quad \ln d_j = \ln\left(\prod_{\substack{n=1\\n\neq j}}^{N} \frac{1}{\sin(\pi(y_j - y_n))}\right) \qquad (3.33)\\
&= \sum_{n=1}^{N} \ln\left(\sin(\pi(x_l - y_n))\right) \qquad = \sum_{\substack{n=1\\n\neq j}}^{N} \ln\left(\frac{1}{\sin(\pi(y_j - y_n))}\right)\\
&\qquad = \sum_{\substack{n=1\\n\neq j}}^{N} \underbrace{\ln 1}_{=0} - \ln\left(\sin(\pi(y_j - y_n))\right)\\
&\qquad = -\sum_{\substack{n=1\\n\neq j}}^{N} \ln\left(\sin(\pi(y_j - y_n))\right),
\end{aligned}$$

d. h. die Logarithmen der gesuchten Koeffizienten können mithilfe einer Summation mit dem Kern $K(x) = \ln(\sin(\pi x))$ berechnet werden. Allerdings ist hierbei zu beachten, dass möglicherweise Logarithmen negativer Zahlen berechnet werden, womit

komplexe Zahlen erhalten werden würden. Dies ist für die schnelle Summation jedoch nicht vorgesehen. Wird statt c_l nun aber $|c_l|$ betrachtet, so lässt sich dies nach (3.7) darstellen als

$$|c_l| = \left| \prod_{n=1}^{N} \sin(\pi(x_l - y_n)) \right| = \prod_{n=1}^{N} |\sin(\pi(x_l - y_n))| .$$

Genau wie oben ergibt sich durch Logarithmieren und Anwendung der Logarithmengesetze

$$\ln |c_l| = \ln \prod_{n=1}^{N} |\sin(\pi(x_l - y_n))| = \sum_{n=1}^{N} \ln |\sin(\pi(x_l - y_n))|$$

und Analoges für d_j. Soll also nur der Betrag der Koeffizienten berechnet werden, so kann dafür der Kern $K(x) = \ln(|\sin(\pi x)|)$ genutzt werden. Zusätzlich wird dafür eine Vorzeichenkorrektur benötigt, um wieder die vorzeichenbehafteten Koeffizienten c_l und d_j zu erhalten.

Zur Bestimmung der gesuchten Vorzeichen sei nun der Sinus näher betrachtet. Wird dieser für ein Paar von Punkten negativ, so entsteht bei der Berechnung der Koeffizienten nach (3.7) bzw. (3.8) ein negativer Faktor. Ist also bekannt, wie oft dies auftritt, so kann das gesamte Vorzeichen damit bestimmt werden. In diesem Fall reicht es aus, den Sinus im Intervall $[-1, 1]$ zu betrachten, da nur in diesem die Differenzen $x_l - y_n$ bzw. $y_j - y_n$ liegen. Wie in Abbildung 3.7 zu sehen, ist $\sin(\pi x)$ dort aber genau dann negativ wenn $x < 0$, d. h. für $x_l < y_n$ bzw. $y_j < y_n$.

Werden also beide Knotenmengen aufsteigend sortiert, so ist es durch sukzessives Durchgehen der Listen möglich für jedes l die Anzahl r_l der Knoten y_j kleiner als x_l effizient zu bestimmen. Da $N \in 2\mathbb{N}$ gilt, kann das Vorzeichen der c_l dann durch $\operatorname{sgn}(c_l) = (-1)^{r_l}$ bestimmt werden. Für die Koeffizienten d_j werden nur Knoten y_j benötigt, d. h. in diesem Fall gilt sogar $\operatorname{sgn}(d_j) = (-1)^{N-j}$, falls die Knotenmenge der y_j bereits sortiert ist.

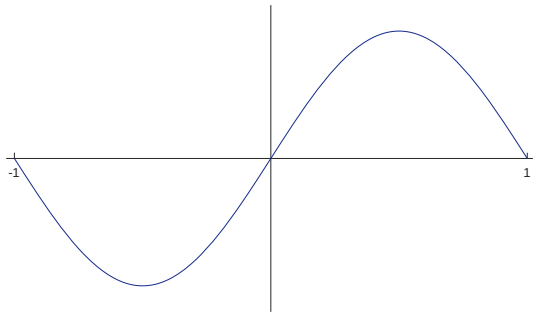


Abbildung 3.7: Funktion $f(x) = \sin(\pi x)$ im Intervall $[-1, 1]$.

Durch Sortierung der Knoten können also für alle Verteilungen der Punkte y_j die Vorzeichen der Koeffizienten effizient bestimmt werden. Hier sei aber nur der Spezialfall von Knoten x_l, y_j wie in (3.16) und (3.17) betrachtet. Aus den beiden Vorschriften ist zu erkennen, dass für diese Punkte stets $x_j < y_j < x_{j+1}$, $j = 1, \ldots, N$, gilt. Damit entstehen also immer wechselnde Vorzeichen, wobei $c_1 > 0$ und $d_1 < 0$ ist.

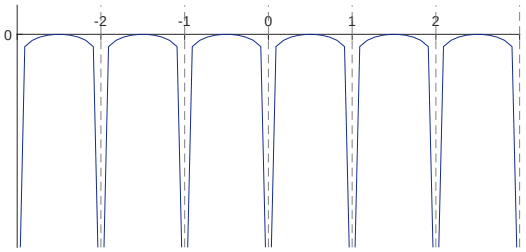


Abbildung 3.8: Funktion $f(x) = \ln(|\sin(\pi x)|)$ im Intervall $[-3, 3]$.

Wird nun der Kern $K(x) = \ln(|\sin(\pi x)|)$ genauer betrachtet, siehe Abbildung 3.8, so fällt auf, dass dieser, genau wie der Cotangens, 1-periodisch ist und an den gleichen Stellen Singularitäten aufweist. Damit kann die Berechnung hier also analog zu Abschnitt 3.2 durchgeführt werden.

Dies ist allerdings nach wie vor numerisch recht instabil. Wie bereits erwähnt, werden die Koeffizienten c_l mit zunehmender Knotenanzahl extrem klein, die d_j hingegen extrem groß. Beide sind aber stets etwa in der gleichen Größenordnung. Wird nun $\tilde{c}_l \coloneqq \ln|c_l|$ und $\tilde{d}_j \coloneqq \ln|d_j|$ gesetzt, so gilt für beliebiges $s \in \mathbb{R}$

$$\begin{aligned} |c_l| \cdot |d_j| &= \mathrm{e}^{\ln|c_l|}\,\mathrm{e}^{\ln|d_j|} = \mathrm{e}^{\tilde{c}_l}\,\mathrm{e}^{\tilde{d}_j} = \mathrm{e}^{\tilde{c}_l+\tilde{d}_j} \\ &= \mathrm{e}^{\tilde{c}_l+s+\tilde{d}_j-s} = \mathrm{e}^{\tilde{c}_l+s}\,\mathrm{e}^{\tilde{d}_j-s}. \end{aligned} \tag{3.34}$$

Da zur Berechnung der g_l nur Produkte der (vorzeichenbehafteten) Koeffizienten benötigt werden, kann also ein Faktor e^s hinzugenommen und wieder abdividiert werden. Dadurch werden die Koeffizienten näher zusammengebracht, was numerische Fehler vermindert. Dieser Faktor wird jedoch, wie in (3.34) angedeutet, als Summand behandelt und erst danach werden die logarithmierten Koeffizienten wieder potenziert, da dies numerisch stabiler ist. Um die Koeffizienten möglichst gleich weit zu verschieben, sei hier s wie folgt gewählt.

$$s = \min\left\{\min_j\{\tilde{d}_j\}, \min_l\{\tilde{c}_l\}\right\}. \tag{3.35}$$

Dies liefert schließlich den folgenden Algorithmus.

Algorithmus 3.9 (Schnelle Inverse NFFT)

Gegeben seien $N \in 2\mathbb{N}$, äquidistante Knoten $x_l \in \left[-\frac{1}{2}, \frac{1}{2}\right), l = 1, \ldots, N$, nichtäquidistant verteilte Knoten $y_j \in \left[-\frac{1}{2}, \frac{1}{2}\right)$ sowie $f_j \in \mathbb{C}, j = 1, \ldots, N$. Weiterhin seien $m \in 2\mathbb{N}$, $p, n \in \mathbb{N}$ und $\varepsilon_{\mathrm{I}} < \frac{1}{4}$ gegeben.

1. Berechne $\tilde{c}_l$ mittels schneller Summation mit $K(x) = \ln(|\sin(\pi x)|)$.

   - $\tilde{c}_l = \sum_{j \in J_1} \ln|\sin(\pi(x_l - y_j))|$ (L_1, J_1 siehe (3.23), (3.24))
     $\quad + \sum_{j \in J_2} \ln|\sin(\pi((x_l + 1) - y_j))|, \quad l \in L_1,$ (J_2 siehe (3.25))
   - $\tilde{c}_l = \sum_{j=1}^{N} \ln|\sin(\pi(x_l - y_j))|, \quad l \in L_2,$ (L_2 siehe (3.27))
   - $\tilde{c}_l = \sum_{j \in J_3} \ln|\sin(\pi(x_l - y_j))|$ (L_3, J_3 siehe (3.29), (3.30))
     $\quad + \sum_{j \in J_4} \ln|\sin(\pi((x_l - 1) - y_j))|, \quad l \in L_3.$ (J_4 siehe (3.31))

   $\tilde{d}_j$ analog $\mathcal{O}(m \log m + N)$

2. Ermittle s nach (3.35). $\mathcal{O}(N)$

3. Setze $c_l = \mathrm{e}^{\tilde{c}_l + s}$ und $d_j = \mathrm{e}^{\tilde{d}_j - s}$. $\mathcal{O}(N)$

4. Führe eine Vorzeichenkorrektur der c_l und d_j durch. $\mathcal{O}(N)$
   (Müssen die Knoten noch sortiert werden, so ergibt sich $\mathcal{O}(N \log N)$)

5. Berechne $\tilde{g}_l$ (analog zu $\tilde{c}_l$) mittels schneller Summation mit $K(x) = \cot(\pi x)$, vgl. (3.21), (3.26), (3.28) und (3.32).

   $\mathcal{O}(m \log m + N)$

6. Berechne g_l nach (3.22). $\mathcal{O}(N)$

7. Berechne
   $$\check{f}_k = \frac{1}{N} \sum_{l=1}^{N} g_l \, \mathrm{e}^{-2\pi \mathrm{i} k x_l}, \quad k = -\tfrac{N}{2}, \ldots, \tfrac{N}{2},$$
   mithilfe einer FFT. $\mathcal{O}(N \log N)$

Ausgabe: $\check{f}_k \approx \hat{f}_k$ aus (3.1).

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

Beispiel 3.10 Es sei nun nochmals eine Versuchsdurchführung wie in Beispiel 3.6 betrachtet. Für wachsende Anzahl von Knoten $N = 2^a$ mit $a = 4, \dots, 17$ soll hierbei die Gittergröße m linear mitskaliert werden. In diesem Falle seien $m = 2N$, $3N$ und $m = 4N$ gewählt. Dementsprechend werde auch $\varepsilon_{\mathrm{I}} = \frac{2p}{m}$ angepasst und es sei $n = p = 4$. Für äquidistante Knoten x_l und zufällig verwackelte Knoten y_j nach (3.17) soll nun eine inverse NFFT durchgeführt werden. Diese sei zum einen exakt und zum anderen effizient mittels Algorithmus 3.9 berechnet und die jeweiligen Zeiten gemessen. Bei exakter Rechnung kann hierbei der gleiche Trick benutzt werden wie in Algorithmus 3.9, es werde aber dennoch weiterhin Algorithmus 3.5 verwendet. Für Details dazu siehe Beispiel 3.12.

Zudem wurden für die durch Algorithmus 3.9 berechneten Näherungen die maximalen absoluten sowie maximalen relativen Fehler, also

$$e_{\mathrm{abs}} \coloneqq \max_{-\frac{N}{2} \le k \le \frac{N}{2}-1} |\hat{f}_k - \check{f}_k| \quad \text{und} \quad e_{\mathrm{rel}} \coloneqq \max_{-\frac{N}{2} \le k \le \frac{N}{2}-1} \frac{|\hat{f}_k - \check{f}_k|}{|\hat{f}_k|}$$

gemessen. Zusätzlich wurden hier auch die Fehler pro Knoten, d. h.

$$\frac{e_{\mathrm{abs}}}{N} \quad \text{und} \quad \frac{e_{\mathrm{rel}}}{N}$$

betrachtet. In den Abbildungen 3.9, 3.10 und 3.11 sind nun für $m = 2N$, $m = 3N$ und $m = 4N$ diese Fehler sowie die ermittelten Berechnungszeiten dargestellt. Dabei ist zu erkennen, dass alle drei Abbildungen nahezu gleich aussehen. Durch eine höhere Gittergröße wird die zu berechnende Fourier-Reihe zwar länger und damit aufwändiger, allerdings wird dafür das entsprechende Nahfeld kleiner, sodass der Skalierungsfaktor also nur einen geringen Einfluss hat.

Werden zunächst die Fehler betrachtet, so fällt auf, dass bis zu einer Größe von ca. $N = 2^{11} = 2048$ die absoluten Fehler nahezu in der gleichen Größenordnung liegen. Vergrößert sich die Anzahl der Knoten dann noch weiter, so wächst der Fehler immer schneller an. Für die relativen Fehler ergibt sich ein ähnliches Bild. Wird allerdings der Fehler bezogen auf die Anzahl der Knoten betrachtet, so bleibt dieser gerade bei sehr vielen Knoten nahezu konstant. Ein Großteil der Fehler bei hohen Knotenanzahlen entsteht also erst numerisch bedingt.

◇

Beispiel 3.11 Analog zu Beispiel 3.7 b) sei nochmals die Approximation des B-Splines der Ordnung 4 durch ein trigonometrisches Polynom betrachtet. Ebenso seien die Knoten wieder nach bestimmtem Muster verwackelt, jedoch soll nun die Inversion schnell mittels Algorithmus 3.9 berechnet werden.

Die Abbildungen 3.12 und 3.13 zeigen dafür die Kurven der maximalen absoluten Fehler für die Muster `[1 1 -1]` und `[1 1 1 -1 -1 -1]`. Dabei ist neben der schnellen Berechnung mittels Algorithmus 3.9 auch die exakte Berechnung, d. h. die

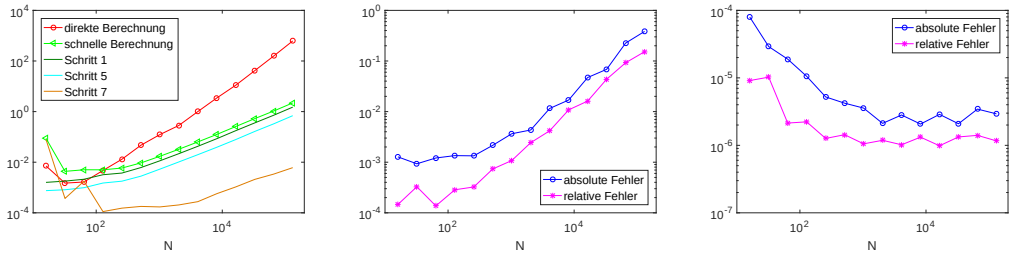


Abbildung 3.9: Berechnungszeiten in Sekunden (links) sowie maximale Fehler (Mitte) und maximale Fehler pro Knoten (rechts) der mithilfe von Algorithmus 3.9 berechneten iNFFT für N zufällig verwackelte äquidistante Knoten mit $n = p = 4$, $m = 2N$ und $\varepsilon_{\mathrm{I}} = \frac{2p}{m}$.

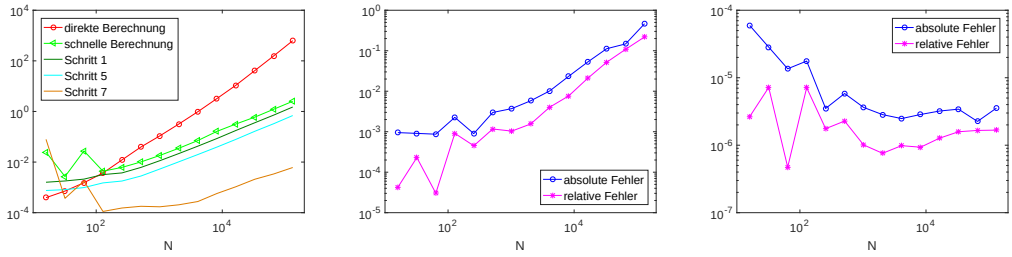


Abbildung 3.10: Berechnungszeiten in Sekunden (links) sowie maximale Fehler (Mitte) und maximale Fehler pro Knoten (rechts) der mithilfe von Algorithmus 3.9 berechneten iNFFT für N zufällig verwackelte äquidistante Knoten mit $n = p = 4$, $m = 3N$ und $\varepsilon_{\mathrm{I}} = \frac{2p}{m}$.

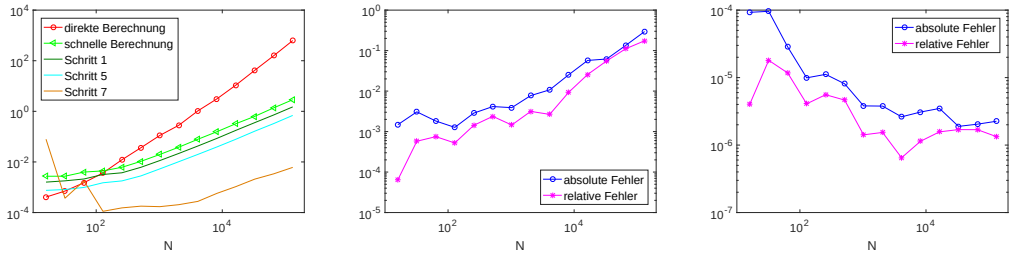


Abbildung 3.11: Berechnungszeiten in Sekunden (links) sowie maximale Fehler (Mitte) und maximale Fehler pro Knoten (rechts) der mithilfe von Algorithmus 3.9 berechneten iNFFT für N zufällig verwackelte äquidistante Knoten mit $n = p = 4$, $m = 4N$ und $\varepsilon_{\mathrm{I}} = \frac{2p}{m}$.

tatsächliche Invertierung der Matrix $\boldsymbol{A}$ abgebildet. Es ist zu erkennen, dass die Kurven von exakter und approximativer Berechnung für unverwackelte Knoten nahezu gleich sind. Bei exakter Rechnung bleibt der Abfall des Fehlers auch für verwackelte Knoten gleich. Bei Nutzung des schnellen Algorithmus 3.9 hingegen ist zu sehen, dass für geringere Knotenanzahlen die Anstiege noch ungefähr in der Größenordnung liegen, ab einem bestimmten Wert die Genauigkeit jedoch nicht weiter verbessert werden kann. Der Fehler bleibt ab diesem Punkt konstant. Dieses Verhalten lässt sich genauso auch bei geringerem Wert der Verwacklung feststellen. Abbildung 3.14 zeigt dazu beispielsweise die gleichen Kurven wie Abbildung 3.12 nur für kleinere Werte des Parameters α.

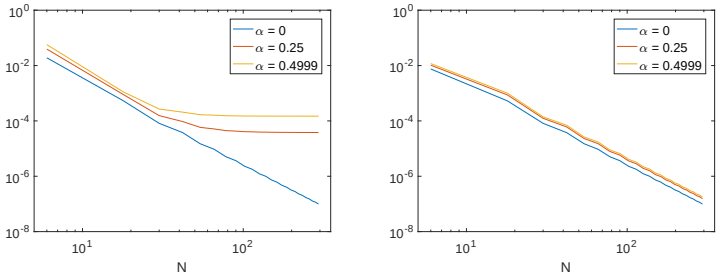


Abbildung 3.12: Maximale absolute Fehler bei Approximation des B-Splines der Ordnung 4 durch das mittels Algorithmus 3.9 (links) bzw. exakt (rechts) berechnete trigonometrische Polynom t_N für Knoten verwackelt nach dem Muster `[1 1 -1]`.

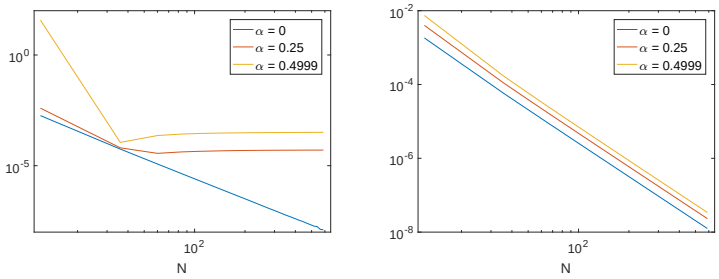


Abbildung 3.13: Maximale absolute Fehler bei Approximation des B-Splines der Ordnung 4 durch das mittels Algorithmus 3.9 (links) bzw. exakt (rechts) berechnete trigonometrische Polynom t_N für Knoten verwackelt nach dem Muster `[1 1 1 -1 -1 -1]`.

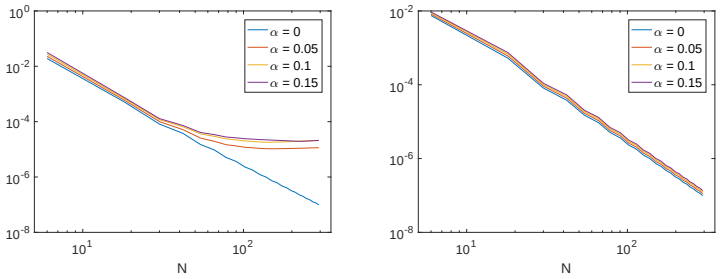


Abbildung 3.14: Maximale absolute Fehler bei Approximation des B-Splines der Ordnung 4 durch das mittels Algorithmus 3.9 (links) bzw. exakt (rechts) berechnete trigonometrische Polynom t_N für Knoten verwackelt nach dem Muster `[1 1 -1]` für kleinere Werte von α.

◇

Beispiel 3.12 Abschließend sei nun nochmals Beispiel 3.6 betrachtet. Für die dort durchgeführte exakte Berechnung soll, wie bereits angekündigt, ebenfalls eine stabilere Methode erläutert werden.

Statt die Koeffizienten c_l und d_j nach (3.7) bzw. (3.8) zu berechnen, sollen ähnlich wie bei schneller Rechnung die Logarithmen $\ln c_l$ und $\ln d_j$ berechnet werden. Dies ist ebenfalls exakt möglich, indem einfach die Summen in (3.33) exakt ausgewertet werden. Anschließend sind die Schritte 2 und 3 aus Algorithmus 3.9 ebenfalls durchführbar. Eine Vorzeichenkorrektur wird nicht benötigt, da hier gleich die Logarithmen der vorzeichenbehafteten Koeffizienten berechnet werden können.

Wird dieser Ansatz in Algorithmus 3.5 verwendet, so können für den gut konditionierten Fall der zufällig verwackelten Knoten nun auch höhere Anzahlen betrachtet werden. Während zuvor schon bei einer Größe von $N = 1020$ keine Berechnung mehr durchführbar war, siehe Tabelle 3.1, so sind jetzt sehr viel höhere Anzahlen von Knoten möglich. Tabelle 3.10 zeigt für $N = 2^a$ mit $a = 4, \ldots, 17$ die nun erhaltenen Fehler. Zusätzlich sind ebenfalls die Konditionszahlen angegeben. Dies ist jedoch nur möglich bis $N = 2^{14} = 16384$, danach kann die Matrix $\boldsymbol{A}$ schlichtweg nicht mehr gespeichert werden. Die auftretenden Unterschiede zwischen Tabelle 3.1 und 3.10 sind dabei dadurch zu erklären, dass bei der Wahl der Knoten der Zufall eine Rolle spielt.

◇

N	max. abs. Fehler	max. rel. Fehler	mittl. abs. Fehler	mittl. rel. Fehler	$\text{cond}_2(\boldsymbol{A})$
16	4.95e-13	3.81e-14	1.63e-13	5.92e-15	1.43e+00
32	1.14e-12	8.53e-14	5.51e-13	1.96e-14	1.45e+00
64	2.43e-12	4.33e-13	7.53e-13	3.06e-14	1.49e+00
128	9.55e-12	2.24e-12	1.98e-12	9.77e-14	1.49e+00
256	2.95e-11	4.68e-12	4.86e-12	2.15e-13	1.52e+00
512	7.92e-11	3.21e-11	1.99e-11	1.07e-12	1.51e+00
1024	4.85e-10	2.05e-10	1.73e-10	8.27e-12	1.56e+00
2048	8.43e-10	3.41e-10	2.13e-10	1.16e-11	1.54e+00
4096	1.08e-08	9.88e-10	5.66e-10	2.91e-11	1.58e+00
8192	2.27e-08	2.48e-09	1.00e-09	5.13e-11	1.83e+00
16384	1.56e-07	7.68e-09	4.02e-09	2.07e-10	3.89e+00
32768	3.67e-07	3.21e-08	1.68e-08	8.92e-10	-
65536	2.60e-06	1.17e-07	5.03e-08	2.61e-09	-
131072	3.45e-06	6.77e-07	3.27e-08	1.68e-09	-

Tabelle 3.10: Fehler bei exakter Berechnung der Inversion sowie Konditionszahlen der nicht-äquidistanten Fourier-Matrix $\boldsymbol{A}$ für zufällig verwackelte äquidistante Knoten, siehe (3.17), bei verschiedenen Größen N.

Zusammenfassung

In dieser Arbeit wurde eine neue Methode zur direkten Berechnung einer iNFFT, d. h. zur Berechnung der Fourierkoeffizienten aus gegebenen Funktionswerten eines trigonometrischen Polynoms, entwickelt. Dabei entspricht die hier vorgestellte iNFFT einer Summation von Termen der Form $\cot(\pi(x_l - y_j))$. Hierbei wird die Idee genutzt, eine Gruppe der Knoten äquidistant zu wählen und die Berechnung der Fourierkoeffizienten auf diesen Fall zurückzuführen, um so eine FFT benutzen zu können. Daher wird in der vorliegenden Arbeit der Fall betrachtet, dass die Anzahl an gegebenen Knoten mit der Anzahl der zu berechnenden Fourierkoeffizienten übereinstimmt.

Zunächst wurde dafür der exakte Algorithmus mittels Lagrange'scher Interpolation hergeleitet. In Experimenten war zu sehen, dass dieser vor allem im Fall verwackelter Knoten anwendbar ist. Für Tschebyscheff-Knoten oder logarithmisch verteilte Knoten hingegen sind die Probleme viel zu schlecht konditioniert. Weiterhin ist der erhaltene Algorithmus recht instabil, da es stets zu einem Über- bzw. Unterlauf kommt. Dieses Problem kann aber behoben werden durch den später entwickelten Trick, zunächst die Logarithmen der gesuchten Koeffizienten zu berechnen und in dieser Darstellung eine Konstante abzudividieren.

Dennoch ist dieser Algorithmus sehr langsam, da sich durch die exakt berechneten Summationen eine Komplexität von $\mathcal{O}(N^2)$ ergibt. Dies sollte jedoch behoben werden durch Anwendung der schnellen Summation. Zunächst wurden dabei nur die Koeffizienten g_l betrachtet. Deren Berechnung ist möglich mittels schneller Summation mit der Kernfunktion $\cot(\pi x)$, die nicht dem typischen Kern entspricht, da diese auch Singularitäten am Rand des Intervalls aufweist. Durch Aufteilen der Knoten in je 3 Gruppen konnte die Berechnung jedoch auf den Standardfall zurückgeführt werden. Damit können diese Koeffizienten nun durch 5 schnelle Summationen berechnet werden.

Um einen schnellen Algorithmus zu erhalten, müssen aber noch weitere, für die Summation benötigte Koeffizienten, effizient berechnet werden. Diese ergeben sich als Produkte von Termen der Gestalt $\sin(\pi(x_l - y_n))$ bzw. $\sin(\pi(y_j - y_n))^{-1}$. Durch Logarithmieren und einfache Umformungen ist auch hier die Berechnung mithilfe der schnellen Summation möglich. Der dabei genutzte Kern hat ganz ähnliche Eigenschaften wie der bereits oben Verwendete. Daher erfolgt auch die Berechnung analog. Mithilfe dieser Kernfunktion können allerdings nur die Beträge der gesuchten Koeffizienten berechnet werden, d. h. es wird eine zusätzliche Vorzeichenkorrektur benötigt. Es zeigt sich, dass dies durch Sortierung der Knoten effizient möglich ist.

Nach wie vor ist der Algorithmus allerdings recht instabil durch den Unter- bzw.

Überlauf. Die numerischen Schwierigkeiten bei der Berechnung können jedoch einfach durch Abdividieren einer großen Konstante verbessert werden. Dies liefert schließlich einen schnellen Algorithmus mit Komplexität $\mathcal{O}(m \log m + N \log N)$. Wird nun davon ausgegangen, dass die Gittergröße mit N mitskaliert, d h. $m = cN$, so ergibt sich ein Aufwand von $\mathcal{O}(N \log N)$.

Abschließend wurde der entwickelte Algorithmus nochmals getestet und mit der exakten Berechnung verglichen. Dabei konnte, wie vermutet, eine deutliche Verbesserung hinsichtlich der Rechenzeiten beobachtet werden. Für die absoluten Fehler zeigte sich, dass diese auch für steigende Knotenanzahl nahezu in der gleichen Größenordnung bleiben. Werden die Fehler bezogen auf die Anzahl der Knoten betrachtet, so ist zu erkennen, dass diese gerade bei sehr vielen Knoten nahezu konstant bleiben, d. h. ein Großteil der Fehler entsteht erst numerisch bedingt.

Literaturverzeichnis

[1] A. P. Austin. *Some New Results on and Applications of Interpolation in Numerical Computation.* Ph. D. Thesis, University of Oxford Mathematical Institute, 2016.

[2] A. P. Austin and L. N. Trefethen. Trigonometric interpolation and quadrature in perturbed points. *SIAM J. Numer. Anal.*, 55:2113–2122, 2017.

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

[4] A. J. W. Duijndam and M. A. Schonewille. Nonuniform fast Fourier transform. *Geophysics*, 64:539–551, 1999.

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

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

[7] M. Fenn. Fast Fourier transform at nonequispaced nodes and applications. Dissertation, Universität Mannheim, 2006.

[8] J. A. Fessler and B. P. Sutton. Nonuniform fast Fourier transforms using min-max interpolation. *IEEE Trans. Signal Process.*, 51:560–574, 2003.

[9] G. B. Folland. *Fourier Analysis and its Applications.* Brooks/Cole, Albany, 1992.

[10] K. Fourmont. Non equispaced fast Fourier transforms with applications to tomography. *J. Fourier Anal. Appl.*, 9:431–450, 2003.

[11] C. Gasquet and P. Witomski. *Fourier Analysis and Applications: Filtering, Numerical Computation, Wavelets.* Springer, 1999.

[12] J. Keiner, S. Kunis, and D. Potts. Using NFFT3 - a software library for various nonequispaced fast Fourier transforms. *ACM Trans. Math. Software*, 36:Article 19, 1–30, 2009.

[13] S. Kunis and D. Potts. Stability results for scattered data interpolation by trigonometric polynomials. *SIAM J. Sci. Comput.*, 29:1403–1419, 2007.

[14] F. Nestler. Approximationsverfahren zur schnellen Energieberechnung in Partikelsystemen. Diplomarbeit, Fakultät für Mathematik, Technische Universität Chemnitz, 2012.

[15] M. Pippig. Parallele nichtäquidistante schnelle Fourier-Transformation und schnelle Summation. Diplomarbeit, Faculty of Mathematics, Chemnitz University of Technology, 2009.

[16] D. Potts. Schnelle Fourier-Transformationen für nichtäquidistante Daten und Anwendungen. Habilitation, Universität zu Lübeck, `http://www.tu-chemnitz.de/~potts`, 2003.

[17] D. Potts and G. Steidl. Fast summation at nonequispaced knots by NFFTs. *SIAM J. Sci. Comput.*, 24:2013–2037, 2003.

[18] 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, MA, USA, 2001. Birkhäuser.

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

[20] G. Steidl and M. Tasche. *Schnelle Fourier–Transformation–Theorie und Anwendungen.* 8 Lehrbriefe der FernUniv. Hagen, 1997.

[21] M. W. Wong. *Discrete Fourier Analysis.* Birkhäuser Basel, 2011.