Schnelle Polynomtransformationen und Vorkonditionierer für Toeplitz–Matrizen

Inaugural–Dissertation

zur Erlangung des akademischen Grades

Doktor eines Wissenschaftszweiges

doctor rerum naturalium

vorgelegt der

Mathematisch–Naturwissenschaftlichen Fakultät

der

Universität Rostock

von

Dipl.–Math. Daniel Potts

geboren am 11. Juli 1969

in Malchin

Rostock, im Dezember 1997

Dekan:	Prof. Dr. G. Wildenhain
Gutachter:	Prof. Dr. M. Tasche (Rostock)
Gutachter:	Prof. Dr. G. Steidl (Mannheim)
Gutachter:	Prof. Dr. B. Fischer (Lübeck)
Tag der öffentlichen Verteidigung:	24.03.98

Inhaltsverzeichnis

Einführung

Ein wesentliches Anliegen der numerischen Mathematik ist die Entwicklung schneller Algorithmen für häufig wiederkehrende Grundaufgaben. Zu den bekanntesten schnellen Algorithmen gehört der Radix–2–Algorithmus zur Berechnung der diskreten Fourier–Transformation (DFT). Durch seine Anwendung wird insbesondere die arithmetische Komplexität einer DFT der Länge $N = 2^t$ von $\mathcal{O}(N^2)$ auf $\mathcal{O}(N \log N)$ arithmetischen Operationen verringert. Diese Art der schnellen Realisierung der DFT wird *schnelle Fourier–Transformation* (Fast Fourier Transform, kurz: FFT) genannt. Viele Verfahren sind erst durch die Effizienz der FFT von praktischem Interesse, so z.B. Polynommultiplikation, Matrizenmultiplikation, Matrix–Vektor–Multiplikation, Invertierung großer strukturierter Matrizen oder trigonometrische Interpolation auf feinen Gittern. Die Untersuchung der FFT–Algorithmen im Zusammenhang mit der ständigen Verbesserung der Rechentechnik führte zu weiteren Anwendungsmöglichkeiten der DFT (siehe z.B. [28, 29, 32, 43, 44, 50, 57, 92]).

Die meisten Anwendungen der DFT basieren auf der Faltungseigenschaft. Durch die DFT wird die Faltung zweier Vektoren in das komponentenweise Produkt der Fourier–transformierten Vektoren überführt. Diese Faltungseigenschaft ist äquivalent zu der Aussage, daß sich eine zirkulante Matrix, die eine spezielle Toeplitz–Matrix ist, mittels der Fourier–Matrix diagonalisieren läßt. Mit anderen Worten, die zirkulanten Matrizen sind genau die diagonalisierbaren Matrizen bezüglich der Fourier–Matrix. Auf dieser wichtigen Eigenschaft beruhen viele Anwendungen zirkulanter Matrizen. Beispielsweise nutzte G. Strang [87] zirkulante Matrizen als Vorkonditionierer zur iterativen Lösung eines Toeplitz–Systems, d.h., eines großen linearen Gleichungssystems, deren Koeffizientenmatrix eine Toeplitz–Matrix ist. Seine Idee war einfach, aber sehr wirkungsvoll. Er „approximierte“ die gegebene Toeplitz–Matrix $\boldsymbol{A}$ in naheliegender Weise durch eine zirkulante Matrix $\boldsymbol{M}$ und verwendete deren Inverse $\boldsymbol{M}^{-1}$ als Vorkonditionierer bei einem PCG–Verfahren. Die Erfolge dieser Methode lösten seit 1986 eine wahre Flut von Arbeiten (siehe [7–10, 12–15, 17–38, 57–66, 71–76, 79, 83, 87–94]) aus.

Da einerseits die Eingabewerte bei den meisten Anwendungen reell sind und da andererseits die DFT eine komplexe Arithmetik benötigt, interessiert man sich auch für reelle Versionen der DFT. Dies führt zu verschiedenen diskreten trigonometrischen Transformationen, den *diskreten Kosinus–Transformationen* (DCT), den *diskreten Sinus–Transformationen* (DST) und der diskreten Hartley–Transformation. Eine herausragende Rolle spielen dabei die DCT, weil diese eine breite Anwendung in der numeri-

schen Analysis und digitalen Signalverarbeitung finden. Genannt seien die Chebyshev–Entwicklung, die Interpolation an Chebyshev–Knoten, die Clenshaw–Curtis–Quadratur und der JPEG–Standard in der Bilddatenkompression. Für eine diskrete trigonometrische Transformation der Länge N lassen sich reelle, schnelle, numerisch stabile Algorithmen, die nur $\mathcal{O}(N \log N)$ Rechenoperationen und einfache Datenumordnungen benötigen, herleiten (siehe [85, 84, 5]). Grundlage dieser schnellen Algorithmen bilden die Teile–und–Herrsche–Strategie und trigonometrische Additionstheoreme.
Die DCT kann man auch als diskrete Polynomtransformation (DPT) auffassen, bei der man die Chebyshev–Polynome erster Art benutzt. Bekanntlich sind die Chebyshev–Polynome orthogonal bezüglich des Chebyshev–Gewichtes und sie genügen einer Drei–Term–Rekursion. Deshalb ist es naheliegend, in Verallgemeinerung von diskreten trigonometrischen Transformationen auch diskrete Polynomtransformationen zu betrachten, dessen Polynome einer beliebigen Drei–Term–Rekursion genügen. Ein nichttriviales Beispiel ist die diskrete Legendre–Transformation, bei der man die Legendre–Polynome benutzt. Schnelle Algorithmen für die diskrete Legendre–Transformation wurden in [1, 43] angegeben.

Aus diesen Entwicklungsrichtungen leiten sich die Zielstellungen meiner Arbeit ab:

1. Die Eigenschaften diagonalisierbarer Matrizen bezüglich einer Kosinus– bzw. Sinus–Matrix sind herzuleiten und für eine schnelle Toeplitz–Matrix–Vektor–Multiplikation zu nutzen.
2. Auf Basis der schnellen DCT–Algorithmen und einer schnellen Polynom–Multiplikation soll ein schneller Algorithmus für eine diskrete Polynomtransformation hergeleitet werden, wobei die Polynome einer beliebigen Drei–Term–Rekursion genügen sollen.
3. Unter Verwendung diagonalisierbarer Matrizen bezüglich einer Kosinus– bzw. Sinus–Matrix soll eine einheitliche Theorie für Vorkonditionierer von reellen Toeplitz–Systemen entwickelt werden.
4. Die theoretischen Ergebnisse sind durch umfangreiche Testrechnungen zu belegen.

Die Arbeit ist in 3 Kapitel gegliedert, wobei Kapitel 1 die Grundlage unserer Betrachtungen in den Kapiteln 2 und 3 darstellt.

Ausgangspunkt in *Kapitel* 1 sind spezielle Matrizen, die in engem Zusammenhang mit den trigonometrischen Matrizen stehen. Motiviert wird dieses Vorgehen durch die große Bedeutung von zirkulanten Matrizen. Sie sind in der numerischen Analysis besonders wichtig, weil diese mit Hilfe der Fourier–Matrix diagonalisiert werden können (siehe z.B. [41]). In Abschnitt 1.2 werden wir deshalb diejenigen Matrizen untersuchen, die durch die trigonometrischen Matrizen diagonalisiert werden können. Dies führt uns auf spezielle Toeplitz–plus–Hankel–Matrizen. Folglich hat der Diagonalisierungssatz (siehe Satz 1.4) für die trigonometrischen Transformationen eine zentrale Bedeutung für diese Arbeit. So können wir in Abschnitt 1.4 eine neue Variante zur schnellen Multiplikation einer reellen Toeplitz–Matrix mit einem beliebigen Vektor angeben (siehe Lemma 1.7). Diese Ergebnisse sind neu (siehe [79]), wurden unabhängig aber auch in [52] hergeleitet.

Kapitel 2 stellt eine Weiterentwicklung des polynomialen Zugangs zur DCT und DST dar. Die bekannten Konzepte zur Herleitung schneller Algorithmen wie die Teile–und–Herrsche–Strategie werden dabei benutzt und klar herausgearbeitet. Zusammen mit einer schnellen Polynommultiplikation (siehe Algorithmus 2.1) erhalten wir einen Algorithmus für eine DPT der Länge N mit der Komplexität $\mathcal{O}(N \log^2 N)$. Diese Algorithmen werden zunächst für Zweierpotenzlängen entwickelt. Ohne Schwierigkeiten lassen sich diese auf beliebige Transformationslängen N ausdehnen. Das zentrale Ergebnis dieses Kapitels haben wir in Algorithmus 2.4 zusammengefaßt. Abschnitt 2.3 schließt mit numerischen Ergebnissen und weiteren Anwendungsmöglichkeiten.
Es sei erwähnt, daß wir die DPT verallgemeinert und zur schnellen Berechnung der Fourier–Transformation auf der Sphäre S^2 benutzt haben (siehe [78]). Ursprünglich wurden schnelle DPT–Algorithmen unter Verwendung der DFT von Driscoll und Healy [43] vorgeschlagen.

In *Kapitel* 3 werden wir Vorkonditionierer zur iterativen Lösung von reellen Toeplitz–Systemen mittels PCG–Verfahren entwickeln. Zunächst leiten wir grundlegende Eigenschaften (E1) – (E4) (siehe S. 43) her, die unsere Vorkonditionierer erfüllen sollen. Wir realisieren ein vorkonditioniertes CG–Verfahren für ein Toeplitz–System in reeller Arithmetik. In Abschnitt 3.2 werden wir ein *neues allgemeines Konzept* entwickeln, um optimale trigonometrische Vorkonditionierer einfach und schnell zu berechnen. Der Erfolg beruht abermals auf konsequenter Nutzung der einfachen Rekursionsformeln für Chebyshev–Polynome. Optimale Vorkonditionierer sind bei speziellen trigonometrischen Transformationen für symmetrische Toeplitz–Matrizen bereits bekannt (siehe [14, 12, 59, 31, 13, 62]). Ein offenes Problem stellten die superoptimalen trigonometrischen Vorkonditionierer [92] dar. Mit unserem allgemeinen Konzept sind wir in der Lage, diese schnell zu berechnen und numerisch zu testen. Andere trigonometrische Vorkonditionierer liefern aber in allen Tests bessere Resultate (siehe Bemerkung 3.28 und Abschnitt 3.3.3). Ein wichtiges Ergebnis stellen wir in Abschnitt 3.4 vor. Hier berechnen wir erstmals optimale trigonometrische Vorkonditionierer für die Normalgleichungen von unsymmetrischen und rechteckigen Toeplitz–Matrizen (siehe Satz 3.39). Es gibt bereits eine Arbeit [8], in der ein Vorkonditionierer für diese Matrizen bezüglich der DST–I berechnet wird. Wir werden diese Vorkonditionierer in numerischen Rechnungen vergleichen. In den Folgerungen 3.18, 3.26, 3.35 und 3.40 werden wir zeigen, daß die trigonometrischen Vorkonditionierer die Eigenschaften (E1) – (E4) erfüllen. Dies bestätigen auch unsere numerischen Ergebnisse (siehe S. 68).

Für schlecht konditionierte Toeplitz–Systeme sind die Strang–Typ–Vorkonditionierer im allgemeinen nicht mehr positiv definit. In diesem Fall liefern auch die optimalen trigonometrischen Vorkonditionierer keine guten numerischen Ergebnisse (siehe Abschnitt 3.3.3). Deshalb wurden Band–Toeplitz–Matrizen als Vorkonditionierer benutzt (siehe [10, 18, 26, 83]). Wir werden vereinfachte Strang–Typ–Vorkonditionierer einführen und erstmals zeigen, daß unter zusätzlichen Voraussetzungen die Eigenschaften (E1) – (E4) erfüllt sind (siehe Satz 3.21 und Satz 3.22). Umfangreiche numerische Testrechnungen mit Beispielen aus der Literatur zeigen, daß diese Vorkonditionierer die besten numerischen Ergebnisse liefern. Außerdem benötigen diese Vorkonditionierer keine zusätzlichen diskreten trigonometrischen Transformationen (siehe Bemerkung 3.29).

An dieser Stelle danke ich Herrn Prof. Dr. M. Tasche für die langjährige ausgezeichnete Betreuung und Frau Prof. Dr. G. Steidl für viele wertvolle Hinweise sowie die sehr gute Zusammenarbeit. Deren reges Interesse am Fortgang meiner Untersuchungen war ein ständiger Ansporn. Weiterer Dank gebührt Herrn Prof. Dr. G. Maeß für die Leitung des Promotionsvorhabens im Rahmen der Graduiertenförderung des Landes Mecklenburg–Vorpommern. Ferner danke ich Herrn Prof. Dr. B. Fischer sowie den Kollegen des Institutes für Mathematik der Medizinischen Universität zu Lübeck und den Mitarbeitern der Arbeitsgruppe „Approximationsmethoden und Wavelets“ am Fachbereich Mathematik der Universität Rostock für die angenehme Arbeitsatmosphäre. Für die vielseitige Unterstützung bedanke ich mich sehr herzlich bei meiner Frau Ina, besonders für die Zuversicht, die sie mir stets gab.

Rostock, im Dezember 1997 Dipl.–Math. Daniel Potts

Bezeichnungen

In dieser Arbeit verwenden wir Standardbezeichnungen. Beispielsweise sind $\mathbb{N}, \mathbb{N}_0, \mathbb{Z}, \mathbb{R}$ bzw. $\mathbb{C}$ die Mengen aller natürlichen, nichtnegativen ganzen, ganzen, reellen bzw. komplexen Zahlen.

$\delta_{j,k}$	Kronecker–Symbol
$\bar{a}$	konjugiert komplexe Zahl von a
$\boldsymbol{a}_N := (a_j)_{j=0}^{N-1} \in \mathbb{R}^N$	Spaltenvektor der Länge N
$\boldsymbol{o}_N := (0)_{j=0}^{N-1} \in \mathbb{R}^N$	Nullvektor der Länge N
$\boldsymbol{e}_k := (\delta_{j,k})_{j=0}^{N-1} \in \mathbb{R}^N$	k–ter Einheitsvektor
$\boldsymbol{A} := (a_{j,k})_{j,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N}$	Matrix vom Format (M, N)
$\boldsymbol{I}_N := (\delta_{j,k})_{j,k=0}^{N-1} \in \mathbb{R}^{N,N}$	Einheitsmatrix
$\boldsymbol{A}'$	transponierte Matrix von $\boldsymbol{A}$
$\boldsymbol{a}'_N$	transponierter Vektor von $\boldsymbol{a}_N$
$\operatorname{toep}(\boldsymbol{a}'_N, \boldsymbol{b}'_M) := \begin{pmatrix} a_0 & a_1 & \dots & a_{N-1} \\ b_1 & a_0 & \dots & a_{N-2} \\ \vdots & \vdots & & \vdots \\ b_{M-2} & b_{M-3} & \dots & \\ b_{M-1} & b_{M-2} & \dots & b_{M-N} \end{pmatrix}$	Toeplitz–Matrix mit $a_0 = b_0$ und $M \geq N$
$\operatorname{stoep} \boldsymbol{a}'_N := \operatorname{toep}(\boldsymbol{a}'_N, \boldsymbol{a}'_N)$	symmetrische Toeplitz–Matrix
$\operatorname{atoep} \boldsymbol{a}'_N := \operatorname{toep}(\boldsymbol{a}'_N, -\boldsymbol{a}'_N)$	antisymmetrische Toeplitz–Matrix mit $a_0 = 0$
$\operatorname{hank}(\boldsymbol{a}'_N, \boldsymbol{b}'_N) := \begin{pmatrix} a_0 & \dots & a_{N-2} & a_{N-1} \\ a_1 & \dots & a_{N-1} & b_{N-2} \\ \vdots & & \vdots & \vdots \\ a_{N-2} & \dots & b_2 & b_1 \\ a_{N-1} & \dots & b_1 & b_0 \end{pmatrix}$	Hankel–Matrix mit $a_{N-1} = b_{N-1}$
$\operatorname{shank} \boldsymbol{a}'_N := \operatorname{hank}(\boldsymbol{a}'_N, \boldsymbol{a}'_N)$	persymmetrische Hankel–Matrix
$\operatorname{ahank} \boldsymbol{a}'_N := \operatorname{hank}(\boldsymbol{a}'_N, -\boldsymbol{a}'_N)$	antipersymmetrische Hankel–Matrix mit $a_{N-1} = 0$

$\boldsymbol{Z}'_{N,1} := \begin{pmatrix} 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \in \mathbb{R}^{N,N+1}$	Abschneide–Matrix
$\boldsymbol{Z}'_{N,2} := \begin{pmatrix} 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 1 & 0 \end{pmatrix} \in \mathbb{R}^{N,N+1}$	Abschneide–Matrix
$\boldsymbol{R}'_{N} := \begin{pmatrix} 0 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} \in \mathbb{R}^{N-1,N+1}$	Abschneide–Matrix
$\operatorname{diag} \boldsymbol{a}_N := (a_j\, \delta_{j,k})_{j,k=0}^{N-1} \in \mathbb{R}^{N,N}$	Diagonalmatrix mit $\boldsymbol{a}_N = (a_j)_{j=0}^{N-1}$
$\delta(\boldsymbol{A}) := \operatorname{diag}(a_{j,j})_{j=0}^{N-1}$	Diagonalmatrix von $\boldsymbol{A} = (a_{j,k})_{j,k=0}^{N-1}$
$\operatorname{tr} \boldsymbol{A} := \sum_{j=0}^{N-1} a_{j,j}$	Spur von $\boldsymbol{A} = (a_{j,k})_{j,k=0}^{N-1} \in \mathbb{R}^{N,N}$
$\boldsymbol{J}_N := \operatorname{shank} \boldsymbol{e}'_{N-1} \in \mathbb{R}^{N,N}$	Gegenidentität
$\operatorname{circ} \boldsymbol{a}'_N := \operatorname{toep}(\boldsymbol{a}'_N, (a_0, a_{N-1}, a_{N-2}, \dots, a_1)') \in \mathbb{R}^{N,N}$	zirkulante Matrix mit $\boldsymbol{a}_N = (a_j)_{j=0}^{N-1} \in \mathbb{R}^N$

Kapitel 1

Diskrete trigonometrische Transformationen

In diesem Kapitel führen wir trigonometrische Matrizen ein. Unter einer diskreten trigonometrischen Transformation werden wir eine lineare Abbildung von $\mathbb{R}^N$ in sich verstehen, die durch eine trigonometrische Matrix erzeugt wird. Oft werden die diskreten trigonometrischen Transformationen mit einer schnellen Fourier–Transformation realisiert (siehe [67], S. 229 ff). Der Nachteil dieser Methode besteht darin, daß eine diskrete trigonometrische Transformation, die nach Definition reell ist, mittels *komplexer* Arithmetik berechnet wird. Für unsere Methoden werden wir schnelle reelle Algorithmen für die diskreten trigonometrischen Transformationen nutzen (siehe [85, 84, 5]). Der Vorteil besteht insbesondere in der geringeren arithmetischen Komplexität der Algorithmen im Vergleich zur schnellen Fourier–Transformation für reelle Eingabedaten.

In Abschnitt 1.1 werden wir Kosinus–Matrizen mittels der Chebyshev–Polynome erster Art einführen. Mit Hilfe von Christoffel–Darboux–Formeln weisen wir nach, daß diese Matrizen orthogonal sind. Analoge Ergebnisse erhalten wir für Sinus–Matrizen, wenn wir diese mit Hilfe der Chebyshev–Polynome zweiter Art darstellen. Zirkulante Matrizen sind spezielle Toeplitz–Matrizen. Sie stehen in engem Zusammenhang mit der DFT und lassen sich mit Hilfe der Fourier–Matrix diagonalisieren. Die Konsequenz dieses Diagonalisierungssatzes ist, daß man zirkulante Matrizen vom Format (N, N) mit einem Vektor in $\mathcal{O}(N \log N)$ Operationen multiplizieren kann. Außerdem kann die Inverse einer regulären zirkulanten Matrix durch Invertierung einer Diagonalmatrix gefunden werden (siehe z.B. [41]). Deshalb bildet der Diagonalisierungssatz 1.4 eine Grundlage für die weiteren Betrachtungen. So werden wir in Abschnitt 1.4 eine neue Variante angeben, um eine reelle Toeplitz–Matrix mit einem reellen Vektor schnell zu multiplizieren (siehe Lemma 1.7).

Ähnliche Ergebnisse wurden ebenfalls in [52] dargestellt. Mit dieser Methode benötigt man nur 4 diskrete trigonometrische Transformationen der Länge N, um eine beliebige reelle Toeplitz–Matrix vom Format (N, N) mit einem Vektor schnell zu multiplizieren.

1.1 Trigonometrische Matrizen

Mit Π_n bezeichnen wir die Menge aller Polynome vom Grade höchstens n. Wir führen die *Chebyshev–Polynome erster Art* durch

$$T_n(x) := \cos(n \arccos x) \quad (x \in [-1,1],\, n \in \mathbb{N}_0) \tag{1.1}$$

ein. Somit gilt $T_0(x) = 1$ und $T_1(x) = x$ $(x \in [-1,1])$. Das Additionstheorem

$$\cos(k+1)t \;+\; \cos(k-1)t \;=\; 2\cos t\, \cos(kt) \quad (k \in \mathbb{N}, t \in \mathbb{R})$$

liefert uns dann die Drei–Term–Rekursion

$$2\,x\,T_k(x) \;=\; T_{k-1}(x) \;+\; T_{k+1}(x) \quad (k \in \mathbb{N})\;. \tag{1.2}$$

Die *Chebyshev–Polynome zweiter Art* lassen sich durch die gleiche Rekursionsformel

$$2\,x\,U_k(x) \;=\; U_{k-1}(x) \;+\; U_{k+1}(x) \quad (k \in \mathbb{N})$$

mit $U_0(x) = 1$ und $U_1(x) = 2x$ erklären. Für $x \in (-1,1)$ gilt dann die Darstellung

$$U_n(x) = \frac{\sin((n+1)\arccos x)}{\sin(\arccos x)} \quad (n \in \mathbb{N}_0)\;. \tag{1.3}$$

Die Chebyshev–Polynome sind spezielle *Jacobi–Polynome*, die bezüglich des Skalarproduktes

$$\int_{-1}^{1} (1-x)^{\alpha}(1+x)^{\beta}\, f(x)\, g(x)\, \mathrm{d}x \quad (\alpha, \beta > -1)$$

orthogonal sind. Dabei sind die Chebyshev–Polynome erster Art orthogonal bezüglich $\alpha = \beta = -1/2$. Die Chebyshev–Polynome zweiter Art sind orthogonal bezüglich $\alpha = \beta = 1/2$. Ähnlich lassen sich Chebyshev–Polynome dritter und vierter Art erklären. Eine bemerkenswerte Eigenschaft der Chebyshev–Polynome ist die einfache Drei–Term–Rekursion (1.2).

Lemma 1.1 *Die Polynome $P_k,\ Q_k \in \Pi_k$ $(k \in \mathbb{N})$ mögen den Drei–Term–Rekursionen*

$$2\,x\,P_k(x) \;=\; P_{k-1}(x) \;+\; P_{k+1}(x) \quad (x \in [-1,1])\;, \tag{1.4}$$

$$2\,y\,Q_k(y) \;=\; Q_{k-1}(y) \;+\; Q_{k+1}(y) \quad (y \in [-1,1]) \tag{1.5}$$

genügen. Dann gelten die Christoffel–Darboux–Formeln

$$\begin{aligned} 2\,(x-y) \sum_{k=1}^{N-1} \;& P_k(x)Q_k(y) \;=\; P_0(x)Q_1(y) - P_1(x)Q_0(y) \\ + \;& P_N(x)Q_{N-1}(y) - P_{N-1}(x)Q_N(y)\;, \end{aligned} \tag{1.6}$$

$$4\,(x^2-y^2)\sum_{k=1}^{N-1} P_{2k+1}(x)Q_{2k+1}(y) \;=\; P_1(x)Q_3(y)-P_3(x)Q_1(y) + P_{2N+1}(x)Q_{2N-1}(y)-P_{2N-1}(x)Q_{2N+1}(y)\,, \tag{1.7}$$

$$4\,(x^2-y^2)\sum_{k=1}^{N-1} P_{2k}(x)Q_{2k}(y) \;=\; P_0(x)Q_2(y)-P_2(x)Q_0(y) + P_{2N}(x)Q_{2N-2}(y)-P_{2N-2}(x)Q_{2N}(y)\,. \tag{1.8}$$

Beweis: 1. Wir multiplizieren (1.4) mit $Q_k(y)$ und (1.5) mit $(-P_k(x))$. Nach Addition dieser Gleichungen summieren wir von $k=1$ bis $N-1$. Dann ergibt sich (1.6).
2. Wir multiplizieren die Gleichung (1.4) mit x und wenden auf der rechten Seite erneut die Drei–Term–Rekursion (1.4) an. Für alle $k \in \mathbb{N}$ folgern wir

$$4\,x^2\,P_{2k+1}(x) \;=\; P_{2k-1}(x) \;+\; 2\,P_{2k+1}(x) \;+\; P_{2k+3}(x) \quad (x \in [-1,1]) \tag{1.9}$$

und analog

$$4\,y^2\,Q_{2k+1}(y) \;=\; Q_{2k-1}(y) \;+\; 2\,Q_{2k+1}(y) \;+\; Q_{2k+3}(y) \quad (y \in [-1,1])\,. \tag{1.10}$$

Die Formel (1.7) erhalten wir, indem wir (1.9) mit $Q_{2k+1}(y)$ und (1.10) mit $(-P_{2k+1}(y))$ multiplizieren, anschließend diese Gleichungen addieren und danach von $k=1$ bis $N-1$ summieren.
3. Analog leiten wir die Formel (1.8) her. ■

Aus (1.3) schließen wir, daß die Nullstellen des Polynoms $(1-x^2)U_{N-1}(x)$ durch

$$c_k^N := \cos\frac{k\pi}{N} \quad (k=0,\dots,N)$$

gegeben sind. Die Nullstellen des Polynoms $T_N(x)$ lauten c_{2k+1}^{2N} $(k=0,\dots,N-1)$ (siehe (1.1)).
Wir führen nun vier *Kosinus–Matrizen* (siehe [95], [80], S. 11)

$$\begin{aligned}
\boldsymbol{C}_{N+1}^{I} &:= \left(\frac{2}{N}\right)^{1/2} \left(\varepsilon_j^N \varepsilon_k^N\, T_j(c_k^N)\right)_{j,k=0}^{N} \in \mathbb{R}^{N+1,N+1},\\
\boldsymbol{C}_{N}^{II} &:= \left(\frac{2}{N}\right)^{1/2} \left(\varepsilon_j^N\, T_j(c_{2k+1}^{2N})\right)_{j,k=0}^{N-1} \in \mathbb{R}^{N,N},\\
\boldsymbol{C}_{N}^{III} &:= \left(\boldsymbol{C}_{N}^{II}\right)' \in \mathbb{R}^{N,N},\\
\boldsymbol{C}_{N}^{IV} &:= \left(\frac{2}{N}\right)^{1/2} \left(T_{2j+1}(c_{2k+1}^{4N})\right)_{j,k=0}^{N-1} \in \mathbb{R}^{N,N}
\end{aligned}$$

mit $\varepsilon_k^N := 1/\sqrt{2}$ $(k=0,N)$ und $\varepsilon_k^N := 1$ $(k=1,\dots,N-1)$ ein.

Lemma 1.2 *Die Kosinus–Matrizen* $\boldsymbol{C}^{I}_{N+1}, \boldsymbol{C}^{II}_{N}, \boldsymbol{C}^{III}_{N}$ *und* $\boldsymbol{C}^{IV}_{N}$ *sind orthogonal.*

Beweis: Wir beweisen nur $(\boldsymbol{C}^{I}_{N+1})' \, \boldsymbol{C}^{I}_{N+1} = \boldsymbol{I}_{N+1}$. Dazu zeigen wir für $j, n = 0, \ldots, N$

$$\frac{2}{N}\, \varepsilon_j^N \; \varepsilon_n^N \; \sum_{k=0}^{N} (\varepsilon_k^N)^2 \, T_k(c_j^N) \, T_k(c_n^N) \;\; = \;\; \delta_{j,n} \, . \tag{1.11}$$

Für $j \neq n$ folgt aus der Gleichung (1.6) für $P_k = Q_k := T_k$

$$\sum_{k=1}^{N-1} T_k(c_j^N) \, T_k(c_n^N) \;\; = \;\; \begin{cases} -1, & \text{falls} \quad j+n \quad \text{ungerade}, \\ \;\;\, 0, & \text{falls} \quad j+n \quad \text{gerade}. \end{cases}$$

Wegen $T_0(c_j^N)T_0(c_n^N) + T_N(c_j^N)T_N(c_n^N) = 1 + (-1)^{j+n}$ folgt für $j \neq n$ die Behauptung (1.11). Für $j = n = 0$ und für $j = n = N$ ist die Behauptung offensichtlich. Setzen wir $P_k = Q_k := T_k$ und $y = 1$ in (1.8), so erhalten wir unter Beachtung von $T_{2N-2}(c_j^N) = T_2(c_j^N)$ und $T_{2N}(c_j^N) = 1$ für $j = 1, \ldots, N-1$ die Formel

$$\sum_{k=1}^{N-1} T_{2k}(c_j^N) \;\; = -1 \qquad (j = 1, \ldots, N-1).$$

Aus der Beziehung $1 + \cos(2t) = 2\,(\cos t)^2$ folgt $1 + T_{2k}(x) = 2\,(T_k(x))^2$ und somit

$$2 \sum_{k=1}^{N-1} T_k(c_j^N) \; T_k(c_j^N) \;\; = \;\; N - 2 \qquad (j = 1, \ldots, N-1)\, .$$

Die Relation $T_0(c_j^N)T_0(c_j^N) + T_N(c_j^N)T_N(c_j^N) = 2$ vervollständigt den Beweis.
Mittels der Gleichungen (1.6) – (1.8) können wir in ähnlicher Weise die Orthogonalität der anderen Kosinus–Matrizen beweisen. ∎

Weiterhin führen wir vier *Sinus–Matrizen* (siehe [95], [80], S. 11) ein:

$$\begin{aligned}
\boldsymbol{S}^{I}_{N-1} \;\; &:= \;\; \left(\frac{2}{N}\right)^{1/2} \left(\sin \, \frac{(j+1)(k+1)\pi}{N}\right)_{j,k=0}^{N-2} \;\in \mathbb{R}^{N-1,N-1}, \\
\boldsymbol{S}^{II}_{N} \;\; &:= \;\; \left(\frac{2}{N}\right)^{1/2} \left(\varepsilon_{j+1}^N \; \sin \; \frac{(j+1)(2k+1)\pi}{2N}\right)_{j,k=0}^{N-1} \;\in \mathbb{R}^{N,N}, \\
\boldsymbol{S}^{III}_{N} \;\; &:= \;\; \left(\boldsymbol{S}^{II}_{N}\right)' \;\in \mathbb{R}^{N,N}, \\
\boldsymbol{S}^{IV}_{N} \;\; &:= \;\; \left(\frac{2}{N}\right)^{1/2} \left(\sin \, \frac{(2j+1)(2k+1)\pi}{4N}\right)_{j,k=0}^{N-1} \;\in \mathbb{R}^{N,N} \; .
\end{aligned}$$

Lemma 1.3 *Die Sinus–Matrizen* $\boldsymbol{S}^{I}_{N-1}, \boldsymbol{S}^{II}_{N}, \boldsymbol{S}^{III}_{N}$ *und* $\boldsymbol{S}^{IV}_{N}$ *sind orthogonal.*

Beweis: Wir schreiben die Sinus–Matrix $\boldsymbol{S}_{N-1}^{I}$ mit Hilfe der Chebyshev–Polynome zweiter Art in der Form

$$\boldsymbol{S}_{N-1}^{I} = \left(\frac{2}{N}\right)^{1/2} \left(\sin \frac{(k+1)\pi}{N}\, U_j(c_{k+1}^{N})\right)_{j,k=0}^{N-2} \in \mathbb{R}^{N-1,N-1}.$$

Mit ähnlichen Überlegungen wie in Lemma 1.2 können wir nun die Orthogonalität der Sinus–Matrizen aus den Gleichungen (1.6) – (1.8) herleiten. ■

Oft werden einige Matrizen in einer etwas modifizierten Form benutzt. Deshalb führen wir noch folgende Matrizen ein:

$$\tilde{\boldsymbol{C}}_{N+1}^{I} := \left((\varepsilon_k^N)^2 \cos \frac{jk\pi}{N}\right)_{j,k=0}^{N}, \quad \tilde{\boldsymbol{S}}_{N-1}^{I} := \left(\sin \frac{jk\pi}{N}\right)_{j,k=1}^{N-1},$$

$$\tilde{\boldsymbol{C}}_{N}^{II} := \left(\cos \frac{j(2k+1)\pi}{2N}\right)_{j,k=0}^{N-1}, \quad \tilde{\boldsymbol{S}}_{N}^{II} := \left(\sin \frac{(j+1)(2k+1)\pi}{2N}\right)_{j,k=0}^{N-1},$$

$$\tilde{\boldsymbol{C}}_{N}^{III} := \left((\varepsilon_k^N)^2 \cos \frac{(2j+1)k\pi}{2N}\right)_{j,k=0}^{N-1}, \quad \tilde{\boldsymbol{S}}_{N}^{III} := \left((\varepsilon_{k+1}^N)^2 \sin \frac{(2j+1)(k+1)\pi}{2N}\right)_{j,k=0}^{N-1}.$$

Aus (1.6), (1.7) und (1.8) folgt

$$\begin{aligned} \tilde{\boldsymbol{C}}_{N+1}^{I}\, \tilde{\boldsymbol{C}}_{N+1}^{I} &= \frac{N}{2}\, \boldsymbol{I}_{N+1}, \qquad \tilde{\boldsymbol{S}}_{N-1}^{I}\, \tilde{\boldsymbol{S}}_{N-1}^{I} = \frac{N}{2}\, \boldsymbol{I}_{N-1} \\ \tilde{\boldsymbol{C}}_{N}^{II}\, \tilde{\boldsymbol{C}}_{N}^{III} = \tilde{\boldsymbol{C}}_{N}^{III}\, \tilde{\boldsymbol{C}}_{N}^{II} &= \frac{N}{2}\, \boldsymbol{I}_{N}, \quad \tilde{\boldsymbol{S}}_{N}^{II}\, \tilde{\boldsymbol{S}}_{N}^{III} = \tilde{\boldsymbol{S}}_{N}^{III}\, \tilde{\boldsymbol{S}}_{N}^{II} = \frac{N}{2}\, \boldsymbol{I}_{N}. \end{aligned} \tag{1.12}$$

All diese Matrizen werden wir als *trigonometrische Matrizen* bezeichnen. Einen ähnlichen Beweis für die Orthogonalitätsbeziehungen findet man zum Beispiel in [80], S. 60 oder in [86].

1.2 Diagonalisierbare Matrizen

Zirkulante Matrizen $\boldsymbol{C}_N = \operatorname{circ} \boldsymbol{c}_N' \in \mathbb{C}^{N,N}$ können durch die Fourier–Matrix $\boldsymbol{F}_N := N^{-1/2}\left(\mathrm{e}^{-2\pi\mathrm{i}jk/N}\right)_{j,k=0}^{N-1} \in \mathbb{C}^{N,N}$ diagonalisiert werden, d.h.

$$\boldsymbol{C}_N = \bar{\boldsymbol{F}}_N (\operatorname{diag} \hat{\boldsymbol{c}}_N) \boldsymbol{F}_N$$

mit $\hat{\boldsymbol{c}}_N := \boldsymbol{F}_N \boldsymbol{c}_N \ \in \mathbb{C}^N$ (siehe z.B. [41], S. 73). Die Anwendung der schnellen Fourier–Transformation erlaubt somit eine schnelle Matrix–Vektor–Multiplikation, falls die Matrix eine Zirkulante ist.

Im folgenden wollen wir Matrizen untersuchen, welche durch trigonometrischen Matrizen diagonalisiert werden können. Wir beschränken uns hier auf die orthogonalen trigonometrischen Matrizen von Abschnitt 1.1.

Das folgende Ergebnis ist Grundlage der schnellen Toeplitz–Matrix–Vektor–Multiplikation und wird in Kapitel 3 angewandt.

Satz 1.4 *Es besteht folgender Zusammenhang zwischen den trigonometrischen Matrizen und Toeplitz–plus–Hankel–Matrizen:*

(i)

$$\begin{aligned}
\boldsymbol{R}_N' \boldsymbol{C}_{N+1}^I \ \boldsymbol{D} \ \boldsymbol{C}_{N+1}^I \boldsymbol{R}_N &= \frac{1}{2} \ \mathrm{stoep}(a_0, \ldots, a_{N-2}) \ + \ \frac{1}{2} \ \mathrm{shank}(a_2, \ldots, a_{N-2}, 0, 0) \ , \\
\boldsymbol{S}_{N-1}^I \ \boldsymbol{R}_N' \ \boldsymbol{D} \ \boldsymbol{R}_N \boldsymbol{S}_{N-1}^I &= \frac{1}{2} \ \mathrm{stoep}(a_0, \ldots, a_{N-2}) \ - \ \frac{1}{2} \ \mathrm{shank}(a_2, \ldots, a_{N-2}, 0, 0) \ , \\
\boldsymbol{R}_N' \boldsymbol{C}_{N+1}^I \ \tilde{\boldsymbol{D}} \ \boldsymbol{R}_N \boldsymbol{S}_{N-1}^I &= \frac{1}{2} \ \mathrm{atoep}(0, a_1, \ldots, a_{N-2}) \ + \ \frac{1}{2} \ \mathrm{ahank}(a_2, \ldots, a_{N-1}, 0) \ , \\
\boldsymbol{S}_{N-1}^I \ \boldsymbol{R}_N' \ \tilde{\boldsymbol{D}} \ \boldsymbol{C}_{N+1}^I \boldsymbol{R}_N &= -\frac{1}{2} \ \mathrm{atoep}(0, a_1, \ldots, a_{N-2}) \ + \ \frac{1}{2} \ \mathrm{ahank}(a_2, \ldots, a_{N-1}, 0)
\end{aligned}$$

mit

$$\begin{aligned}
\boldsymbol{D} &:= \mathrm{diag}(d_0, \ldots, d_N)' \ , \ \tilde{\boldsymbol{D}} := \mathrm{diag}(0, \tilde{d}_1, \ldots, \tilde{d}_{N-1}, 0)' \ , \\
(d_0, \ldots, d_N)' &:= \tilde{\boldsymbol{C}}_{N+1}^I \ (a_0, \ldots, a_{N-2}, 0, 0)' \ , \\
(\tilde{d}_1, \ldots, \tilde{d}_{N-1})' &:= \tilde{\boldsymbol{S}}_{N-1}^I \ (a_1, \ldots, a_{N-1})' \ .
\end{aligned}$$

(ii)

$$\begin{aligned}
\left(\boldsymbol{C}_N^{II}\right)' \ \boldsymbol{Z}_{N,2}' \ \boldsymbol{D} \ \boldsymbol{Z}_{N,2} \boldsymbol{C}_N^{II} &= \frac{1}{2} \ \mathrm{stoep}(a_0, \ldots, a_{N-1}) \ + \ \frac{1}{2} \ \mathrm{shank}(a_1, \ldots, a_{N-1}, 0) \ , \\
\left(\boldsymbol{S}_N^{II}\right)' \ \boldsymbol{Z}_{N,1}' \ \boldsymbol{D} \ \boldsymbol{Z}_{N,1} \boldsymbol{S}_N^{II} &= \frac{1}{2} \ \mathrm{stoep}(a_0, \ldots, a_{N-1}) \ - \ \frac{1}{2} \ \mathrm{shank}(a_1, \ldots, a_{N-1}, 0) \ , \\
\left(\boldsymbol{C}_N^{II}\right)' \ \boldsymbol{Z}_{N,2}' \ \tilde{\boldsymbol{D}} \ \boldsymbol{Z}_{N,1} \boldsymbol{S}_N^{II} &= \frac{1}{2} \ \mathrm{atoep}(0, a_1, \ldots, a_{N-1}) \ + \ \frac{1}{2} \ \mathrm{ahank}(a_1, \ldots, a_{N-1}, 0) \ , \\
\left(\boldsymbol{S}_N^{II}\right)' \ \boldsymbol{Z}_{N,1}' \ \tilde{\boldsymbol{D}} \ \boldsymbol{Z}_{N,2} \boldsymbol{C}_N^{II} &= -\frac{1}{2} \ \mathrm{atoep}(0, a_1, \ldots, a_{N-1}) \ + \ \frac{1}{2} \ \mathrm{ahank}(a_1, \ldots, a_{N-1}, 0)
\end{aligned}$$

mit

$$\begin{aligned}
\boldsymbol{D} &:= \mathrm{diag}(d_0, \ldots, d_N)' \ , \ \tilde{\boldsymbol{D}} := \mathrm{diag}(0, \tilde{d}_1, \ldots, \tilde{d}_{N-1}, 0)' \ , \\
(d_0, \ldots, d_N)' &:= \tilde{\boldsymbol{C}}_{N+1}^I \ (a_0, \ldots, a_{N-1}, 0)' \ , \\
(\tilde{d}_1, \ldots, \tilde{d}_{N-1})' &:= \tilde{\boldsymbol{S}}_{N-1}^I \ (a_1, \ldots, a_{N-1})' \ .
\end{aligned}$$

(iii)

$$
\begin{aligned}
\boldsymbol{C}_N^{IV}\ \boldsymbol{D}\ \boldsymbol{C}_N^{IV} &= \frac{1}{2}\ \mathrm{stoep}(a_0,\dots,a_{N-1})\ +\ \frac{1}{2}\ \mathrm{ahank}(a_1,\dots,a_{N-1},0)\ ,\\
\boldsymbol{S}_N^{IV}\ \boldsymbol{D}\ \boldsymbol{S}_N^{IV} &= \frac{1}{2}\ \mathrm{stoep}(a_0,\dots,a_{N-1})\ -\ \frac{1}{2}\ \mathrm{ahank}(a_1,\dots,a_{N-1},0)\ ,\\
\boldsymbol{C}_N^{IV}\ \tilde{\boldsymbol{D}}\ \boldsymbol{S}_N^{IV} &= \frac{1}{2}\ \mathrm{atoep}(0,a_1,\dots,a_{N-1})\ +\ \frac{1}{2}\ \mathrm{shank}(a_1,\dots,a_{N-1},0)\ ,\\
\boldsymbol{S}_N^{IV}\ \tilde{\boldsymbol{D}}\ \boldsymbol{C}_N^{IV} &= -\frac{1}{2}\ \mathrm{atoep}(0,a_1,\dots,a_{N-1})\ +\ \frac{1}{2}\ \mathrm{shank}(a_1,\dots,a_{N-1},0)
\end{aligned}
$$

mit

$$
\begin{aligned}
\boldsymbol{D} &:= \mathrm{diag}(d_0,\dots,d_{N-1})'\ ,\ \tilde{\boldsymbol{D}} := \mathrm{diag}(\tilde{d}_0,\dots,\tilde{d}_{N-1})'\\
(d_0,\dots,d_{N-1})' &:= \tilde{\boldsymbol{C}}_N^{III}\ (a_0,\dots,a_{N-1})'\ ,\\
(\tilde{d}_0,\dots,\tilde{d}_{N-1})' &:= \tilde{\boldsymbol{S}}_N^{III}\ (a_1,\dots,a_{N-1},0)'\ .
\end{aligned}
$$

Beweis: Hier wird nur die erste Formel von (ii) bewiesen. Zur Vereinfachung setzen wir $\boldsymbol{C} := \boldsymbol{Z}_{N,2}\boldsymbol{C}_N^{II}$ und $\boldsymbol{D} := \mathrm{diag}(d_0,\dots,d_N)'$. Wir benutzen das Additionstheorem

$$\cos\alpha\ \cos\beta\ =\ \frac{1}{2}\ \cos(\alpha-\beta)\ +\ \frac{1}{2}\ \cos(\alpha+\beta)\ ,$$

um den (u,v)–ten Eintrag der Matrix $\boldsymbol{C}'\boldsymbol{D}\boldsymbol{C}$ in der Form

$$(\boldsymbol{C}'\boldsymbol{D}\boldsymbol{C})_{u,v}\ =\ \frac{1}{2}\ \frac{2}{N}\sum_{k=0}^{N-1}\ (\varepsilon_k^N)^2\ d_k\cos\frac{(u-v)k\pi}{N} + \frac{1}{2}\ \frac{2}{N}\sum_{k=0}^{N-1}\ (\varepsilon_k^N)^2\ d_k\cos\frac{(u+v+1)k\pi}{N}$$

zu schreiben. Wegen $-(-1)^{u-v}d_N = (-1)^{u+v+1}d_N$ für beliebiges $d_N\in\mathbb{R}$ erhalten wir

$$(\boldsymbol{C}'\boldsymbol{D}\boldsymbol{C})_{u,v}\ =\ \frac{1}{2}\ \frac{2}{N}\sum_{k=0}^{N}\ (\varepsilon_k^N)^2\ d_k\cos\frac{(u-v)k\pi}{N} + \frac{1}{2}\ \frac{2}{N}\sum_{k=0}^{N}\ (\varepsilon_k^N)^2\ d_k\cos\frac{(u+v+1)k\pi}{N}\ .$$

Wir wählen nun $d_N\in\mathbb{R}$ derart, daß

$$\sum_{k=0}^{N}(\varepsilon_k^N)^2\ d_k(-1)^k\ =\ 0\ ,$$

um unter Nutzung der Symmetrieeigenschaften der Kosinus–Funktion

$$\boldsymbol{C}'\boldsymbol{D}\ \boldsymbol{C}\ =\ \frac{1}{2}\ \mathrm{stoep}(a_0,\dots,a_{N-1})\ +\ \frac{1}{2}\ \mathrm{shank}(a_1,\dots,a_{N-1},0)$$

zu erhalten. Dabei ist

$$(a_0, \ldots, a_{N-1}, 0)' = \frac{2}{N}\, \tilde{\boldsymbol{C}}^I_{N+1}\, (d_0, \ldots, d_N)' \,,$$

d.h. mit (1.12)

$$(d_0, \ldots, d_N)' = \tilde{\boldsymbol{C}}^I_{N+1}\, (a_0, \ldots, a_{N-1}, 0)' \,.$$

Die anderen Zerlegungsformeln folgen auf ähnliche Weise unter Nutzung von

$$\sin\alpha\ \sin\beta \;=\; \frac{1}{2}\ \cos(\alpha-\beta) \;-\; \frac{1}{2}\ \cos(\alpha+\beta)\ ,$$

$$\sin\alpha\ \cos\beta \;=\; \frac{1}{2}\ \sin(\alpha-\beta) \;+\; \frac{1}{2}\ \sin(\alpha+\beta)\ . \qquad \blacksquare$$

Bemerkung 1.5 Es gibt natürlich noch eine Vielzahl weiterer Beziehungen zwischen den trigonometrischen Matrizen und Toeplitz–plus–Hankel–Matrizen, so gilt z.B.

$$\begin{aligned}\boldsymbol{R}'_N \boldsymbol{C}^I_{N+1}\, \boldsymbol{D}\, \boldsymbol{Z}_{N,2} \boldsymbol{C}^{II}_N \boldsymbol{Z}_{N,1} \;=\;& \frac{1}{2}\ \mathrm{toep}((a_0, a_1, \ldots, a_{N-2})', (a_0, a_0, a_1, \ldots, a_{N-3})')\\ &+ \frac{1}{2}\ \mathrm{hank}((a_2, a_3, \ldots, a_{N-1}, a_{N-1})', (a_1, a_2, a_3, \ldots, a_{N-1})')\end{aligned}$$

mit

$$\boldsymbol{D} := \mathrm{diag}(d_0, \ldots, d_{N-1}, 0)' \,, \quad (d_0, \ldots, d_{N-1})' \;:= \tilde{\boldsymbol{C}}^{II}_N\, (a_0, a_1, \ldots, a_{N-1})' \,.$$

Weitere orthogonale trigonometrische Matrizen kann man z.B. mit Hilfe der Chebyshev–Polynome dritter und vierter Art bilden (siehe [68]). Aus diesen ergeben sich ähnliche Zerlegungen wie in Satz 1.4. Ebenfalls kann man Zerlegungen mit der diskreten Hartley–Transformation angeben, die in [52] untersucht wurde. Im weiteren werden wir uns auf die oben eingeführten Matrizen beschränken. $\square$

1.3 Schnelle trigonometrische Transformationen

Unter der *diskreten Fourier–Transformation der Länge N* (DFT(N)) versteht man diejenige lineare Abbildung von $\mathbb{C}^N$ in sich, die jedem Vektor $\boldsymbol{x} := (x_j)_{j=0}^{N-1} \in \mathbb{C}^N$ den Vektor $\hat{\boldsymbol{x}} := (\hat{x}_j)_{j=0}^{N} \in \mathbb{C}^N$ mit

$$\hat{\boldsymbol{x}} \;:=\; \boldsymbol{F}_N \boldsymbol{x}$$

zuordnet. Analog werden wir die *diskreten trigonometrischen Transformationen* einführen. Wir betrachten hier die *diskreten Kosinustransformationen* (DCT) und die *diskreten*

Sinustransformationen (DST). Die wichtigsten DCT sind die folgenden linearen Abbildungen:

Unter der DCT *vom Typ* I *der Länge* $(N+1)$ (DCT–I$(N+1)$) verstehen wir diejenige lineare Abbildung von $\mathbb{R}^{N+1}$ in sich, die jedem Vektor $\boldsymbol{x} := (x_j)_{j=0}^{N} \in \mathbb{R}^{N+1}$ den Vektor $\hat{\boldsymbol{x}} := (\hat{x}_j)_{j=0}^{N} \in \mathbb{R}^{N+1}$ mit

$$\hat{\boldsymbol{x}} \ := \ \tilde{\boldsymbol{C}}_{N+1}^{I} \boldsymbol{x}$$

zuordnet. Analog definieren wir die DCT *vom Typ* II – IV *der Länge* N:

$$\text{DCT–II}(N) \ : \ \mathbb{R}^N \to \mathbb{R}^N \text{ mit } \hat{\boldsymbol{y}} := \tilde{\boldsymbol{C}}_N^{II} \boldsymbol{y},$$

$$\text{DCT–III}(N) \ : \ \mathbb{R}^N \to \mathbb{R}^N \text{ mit } \hat{\boldsymbol{y}} := \tilde{\boldsymbol{C}}_N^{III} \boldsymbol{y},$$

$$\text{DCT–IV}(N) \ : \ \mathbb{R}^N \to \mathbb{R}^N \text{ mit } \hat{\boldsymbol{y}} := \boldsymbol{C}_N^{IV} \boldsymbol{y},$$

wobei stets $\boldsymbol{y} := (y_j)_{j=0}^{N-1} \in \mathbb{R}^N$ und $\hat{\boldsymbol{y}} := (\hat{y}_j)_{j=0}^{N-1} \in \mathbb{R}^N$ sind.

Außerdem führen wir die DST *vom Typ* I – IV wie folgt ein:

$$\text{DST–I}(N-1) \ : \ \mathbb{R}^{N-1} \to \mathbb{R}^{N-1} \text{ mit } \hat{\boldsymbol{z}} := \tilde{\boldsymbol{S}}_{N-1}^{I} \boldsymbol{z},$$

wobei $\boldsymbol{z} := (z_j)_{j=0}^{N-2} \in \mathbb{R}^{N-1}$ und $\hat{\boldsymbol{z}} := (\hat{z}_j)_{j=0}^{N-2} \in \mathbb{R}^{N-1}$ sind;

$$\text{DST–II}(N) \ : \ \mathbb{R}^N \to \mathbb{R}^N \text{ mit } \hat{\boldsymbol{y}} := \tilde{\boldsymbol{S}}_N^{II} \boldsymbol{y},$$

$$\text{DST–III}(N) \ : \ \mathbb{R}^N \to \mathbb{R}^N \text{ mit } \hat{\boldsymbol{y}} := \tilde{\boldsymbol{S}}_N^{III} \boldsymbol{y},$$

$$\text{DST–IV}(N) \ : \ \mathbb{R}^N \to \mathbb{R}^N \text{ mit } \hat{\boldsymbol{y}} := \boldsymbol{S}_N^{IV} \boldsymbol{y},$$

wobei stets $\boldsymbol{y} := (y_j)_{j=0}^{N-1} \in \mathbb{R}^N$ und $\hat{\boldsymbol{y}} := (\hat{y}_j)_{j=0}^{N-1} \in \mathbb{R}^N$ sind.

Es existieren verschiedene Möglichkeiten einer schnellen Realisierung dieser Transformationen in $\mathcal{O}(N \log N)$ arithmetischen Rechenoperationen . Mit anderen Worten, die Multiplikation einer trigonometrischen Matrix mit einem Vektor kann in $\mathcal{O}(N \log N)$ arithmetischen Rechenoperationen durchgeführt werden. Hintergrund dieser schnellen Algorithmen sind die Additionstheoreme, die eine Teile–und–Herrsche–Strategie ermöglichen (siehe hierzu [85] und [5]). Für die numerischen Testrechnungen benutzen wir ein Programmpaket [4] von G. Baszenski. Ausgangspunkt dieser C–Implementierung sind die Algorithmen aus [85] und [5]. Diese benötigen nur reelle Additionen und Multiplikationen. Die Tabelle 1.1 gibt die Anzahl der reellen Rechenoperationen an, um eine diskrete trigonometrische Transformation mittels [4] zu berechnen.

Bemerkung 1.6 Eine schnelle Realisierung der DFT(N) in $\mathcal{O}(N \log N)$ Operationen wird *schnelle Fourier–Transformation* FFT(N) (Fast Fourier Transform) genannt. Wir bemerken, daß die FFT(2^t) mit 2^t reellen Eingabedaten in $2.5 \cdot t \cdot 2^t + \mathcal{O}(2^t)$ Operationen realisiert werden kann (siehe [67], S. 215 ff). Die diskreten trigonometrischen Transformationen benötigen nur $2 \cdot t \cdot 2^t + \mathcal{O}(2^t)$ Operationen. Diese geringere arithmetische Komplexität ist ein *wesentliches Motiv* für die folgenden Untersuchungen, bei der anstelle der DFT diskrete trigonometrische Transformationen verwendet werden. □

$a\,t2^t \;+\; b\,2^t \;+\; ct \;+\; d$								
	Anzahl der reellen Multiplikationen				Anzahl der reellen Additionen			
$N = 2^t$	a	b	c	d	a	b	c	d
$\boldsymbol{C}_{N+1}^{I}$	1/2	0	−2	3	3/2	−2	1	4
$\boldsymbol{C}_{N}^{II}$	1/2	1	0	−2	3/2	−1	0	1
$\boldsymbol{C}_{N}^{III}$	1/2	5/2	0	−3	3/2	−1/2	0	0
$\boldsymbol{C}_{N}^{IV}$	1/2	2	0	−2	3/2	0	0	0
$\boldsymbol{S}_{N-1}^{I}$	1/2	0	−2	1	3/2	−2	−1	2
$\boldsymbol{S}_{N}^{II}$	1/2	1	0	−2	3/2	−1	0	1
$\boldsymbol{S}_{N}^{III}$	1/2	5/2	0	−3	3/2	−1/2	0	0
$\boldsymbol{S}_{N}^{IV}$	1/2	2	0	−2	3/2	0	0	0

Tabelle 1.1: Anzahl der arithmetischen Operationen bei dem Programm [4] zur Berechnung einer diskreten trigonometrischen Transformation im Fall $N = 2^t$.

1.4 Toeplitz–Matrizen

In diesem Abschnitt geben wir eine neue Variante an, um eine reelle Toeplitz–Matrix mit einem Vektor schnell zu multiplizieren. In vielen Arbeiten [28, 37, 44, 92] wird vorgeschlagen, die Toeplitz–Matrix zu einer zirkulanten Matrix zu erweitern und diese mit Hilfe einer Fourier–Matrix zu diagonalisieren. Um die komplexe Arithmetik mit der DFT zu vermeiden, wird in [14] erstmals eine reelle Variante vorgeschlagen. Dazu wird die reelle Toeplitz–Matrix in eine Matrix eingebettet, die von einer Sinus–Matrix vom Typ I diagonalisiert wird. Wir werden hier verschiedene Diagonalisierungsaussagen (siehe Satz 1.4) verwenden, um eine reelle Toeplitz–Matrix darzustellen. Wir zerlegen eine beliebige Toeplitz–Matrix $\boldsymbol{T} \in \mathbb{R}^{N,N}$ in die Summe einer symmetrischen Toeplitz–Matrix $\frac{1}{2}(\boldsymbol{T} + \boldsymbol{T}')$ und einer antisymmetrischen Toeplitz–Matrix $\frac{1}{2}(\boldsymbol{T} - \boldsymbol{T}')$.

Lemma 1.7 *Es sei $N = 2^t$ $(t \in \mathbb{N})$. Die Multiplikation einer Toeplitz–Matrix*

$$\boldsymbol{T} = (t_{j-k})_{j,k=0}^{N-1} \;=\; \mathrm{toep}((t_0, t_{-1}, \ldots, t_{-(N-1)}), (t_0, t_1, \ldots, t_{N-1})) \in \mathbb{R}^{N,N}$$

mit einem beliebigen Vektor $\boldsymbol{x} \in \mathbb{R}^N$ kann mit $t \cdot 2^{t+1} + \mathcal{O}(2^t)$ reellen Multiplikationen und $3t \cdot 2^{t+1} + \mathcal{O}(2^t)$ reellen Additionen durchgeführt werden. Dazu ist die Vorberechnung einer DCT–I$(N+1)$ *und einer* DST–I$(N-1)$ *nötig.*

Beweis: Wir benutzen hier die Transformationen DCT–II und DST–II. Zur Vereinfachung setzen wir $\boldsymbol{C} := \boldsymbol{Z}_{N,2}\boldsymbol{C}_N^{II}$ und $\boldsymbol{S} := \boldsymbol{Z}_{N,1}\boldsymbol{S}_N^{II}$. Dann gilt nach Satz 1.4

$$\boldsymbol{T} \;=\; \frac{1}{2}\,(\boldsymbol{T}+\boldsymbol{T}') \;+\; \frac{1}{2}\,(\boldsymbol{T}-\boldsymbol{T}') = \boldsymbol{C}'\boldsymbol{D}\boldsymbol{C} \;+\; \boldsymbol{S}'\boldsymbol{D}\boldsymbol{S} \;+\; \boldsymbol{C}'\tilde{\boldsymbol{D}}\boldsymbol{S} \;-\; \boldsymbol{S}'\tilde{\boldsymbol{D}}\boldsymbol{C} \tag{1.13}$$

mit

$$\boldsymbol{D} := \operatorname{diag}(d_0,\dots,d_N)'\,,\;\; \tilde{\boldsymbol{D}} := \operatorname{diag}(0,\tilde{d}_1,\dots,\tilde{d}_{N-1},0)'\,,$$

$$\begin{aligned}
(d_0,\dots,d_N)' \quad &:= \quad \tilde{\boldsymbol{C}}^I_{N+1}\,(t_0, \frac{t_1+t_{-1}}{2},\dots,\frac{t_{N-1}+t_{-(N-1)}}{2},0)'\,,\\
(\tilde{d}_1,\dots,\tilde{d}_{N-1})' \quad &:= \quad \tilde{\boldsymbol{S}}^I_{N-1}\,(\frac{t_{-1}-t_1}{2},\dots,\frac{t_{-(N-1)}-t_{N-1}}{2})'\,.
\end{aligned}$$

Die Matrix–Vektor–Multiplikation $\hat{\boldsymbol{x}} := \boldsymbol{T}\boldsymbol{x}$ benötigt somit eine DCT–I$(N+1)$ und eine DST–I$(N-1)$, um die Einträge der Diagonalmatrizen $\boldsymbol{D}$ und $\tilde{\boldsymbol{D}}$ vorzuberechnen. Wir bilden nun $\hat{\boldsymbol{y}}_C := \boldsymbol{C}\boldsymbol{x}$ und $\hat{\boldsymbol{y}}_S := \boldsymbol{S}\boldsymbol{x}$ und anschließend

$$\hat{\boldsymbol{x}} = \boldsymbol{C}'(\boldsymbol{D}\,\hat{\boldsymbol{y}}_C + \tilde{\boldsymbol{D}}\,\hat{\boldsymbol{y}}_S) \;+\; \boldsymbol{S}'(\boldsymbol{D}\,\hat{\boldsymbol{y}}_S - \tilde{\boldsymbol{D}}\,\hat{\boldsymbol{y}}_C)\,.$$

Zählen wir die Rechenoperationen, so erhalten wir nach Tabelle 1.1 die Behauptung. ■

Ähnliche Ergebnisse wurden kürzlich ebenfalls in [52] gezeigt. Es ist wichtig, daß man die Zerlegungen bezüglich verschiedener diskreter trigonometrischer Transformationen kennt. Dies wird in Bemerkung 3.29 deutlich.

Lemma 1.7 liefert eine Grundlage, um in Kapitel 3 effiziente iterative Verfahren zur Lösung eines linearen Gleichungssystems mit Toeplitz–Koeffizientenmatrix anzugeben. Oft sind rechteckige Toeplitz–Matrizen $\boldsymbol{A} = (a_{j-k})_{j=0,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N}$ (siehe [24, 25, 74]) von Interesse. In vielen Fällen ist $M \gg N$ und es ist eine schnelle Multiplikation $\boldsymbol{A}'\boldsymbol{A}\boldsymbol{x}_N$ mit $\boldsymbol{x}_N = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$ gesucht. In [24] und [25] wird vorgeschlagen, $\boldsymbol{A}$ bzw. $\boldsymbol{A}'$ in eine zirkulante (M,M)–Matrix einzubetten und diese Matrix anschließend mit der Fourier–Matrix zu diagonalisieren. Eine reelle Variante, um $\boldsymbol{y} := \boldsymbol{A}\boldsymbol{x}_N$ zu berechnen, ist z.B. die Multiplikation

$$\operatorname{toep}((a_0,a_{-1},\dots,a_{1-N},\boldsymbol{o}'_{M-N}),(a_0,a_1,\dots,a_{M-1}))\;(x_0,x_1,\dots,x_{N-1},\boldsymbol{o}'_{M-N})' \tag{1.14}$$

mit Hilfe von Lemma 1.2. In vielen Algorithmen ist jedoch mit der (N,N)–Matrix $\boldsymbol{A}'\boldsymbol{A}$ zu multiplizieren. Natürlich soll die Matrix $\boldsymbol{A}'\boldsymbol{A}$ nicht explizit berechnet werden. Da $\boldsymbol{A}'\boldsymbol{A}$ eine (N,N)–Matrix ist, wollen wir auch eine Erweiterung zu einer (M,M)–Matrix vermeiden. Wir schlagen deshalb eine Multiplikation mit $\boldsymbol{A}'\boldsymbol{A}$ basierend auf einer Zerlegung in (N,N)–Matrizen vor. Mit dieser Zerlegung werden wir erstmals einen

Algorithmus angeben, der die Matrix–Vektor–Multiplikation $\boldsymbol{A}'\boldsymbol{A}\boldsymbol{x}_N$ in $\mathcal{O}(N \log N)$ Schritten berechnet. Nur in der Vorberechnung sind $\mathcal{O}(M \log M)$ Schritte nötig. Dieser Zugang ist besonders dann von Vorteil, wenn wir mit der Matrix $\boldsymbol{A}'\boldsymbol{A}$ oft multiplizieren müssen (siehe Abschnitt 3.4). Ähnliche Zerlegungen wurden in [25] und [71] benutzt, um Vorkonditionierer zu bestimmen.

Lemma 1.8 *Sei $\boldsymbol{A} = (a_{j-k})_{j=0,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N}$ eine gegebene Toeplitz–Matrix mit $M \geq N$. Dann läßt sich $\boldsymbol{A}'\boldsymbol{A} \in \mathbb{R}^{N,N}$ in der Form*

$$\boldsymbol{A}'\boldsymbol{A} \;=\; \boldsymbol{T}_N \;+\; \boldsymbol{L}'_N\boldsymbol{L}_N \;+\; \boldsymbol{U}'_N\boldsymbol{U}_N \tag{1.15}$$

zerlegen, wobei $\boldsymbol{T}_N \in \mathbb{R}^{N,N}$ eine symmetrische Toeplitz–Matrix, $\boldsymbol{U}_N \in \mathbb{R}^{N,N}$ eine obere Toeplitz–Dreiecksmatrix und $\boldsymbol{L}_N \in \mathbb{R}^{N,N}$ eine untere Toeplitz–Dreiecksmatrix mit

$$\boldsymbol{T}_N \;:=\; \mathrm{stoep}(t_0, t_1, \ldots, t_{N-1}) \;, \tag{1.16}$$

$$\boldsymbol{U}_N \;:=\; \mathrm{toep}((0, a_{-1}, a_{-2}, \ldots, a_{1-N}), \boldsymbol{o}'_N), \tag{1.17}$$

$$\boldsymbol{L}_N \;:=\; \mathrm{toep}(\boldsymbol{o}'_N, (0, a_{M-N+1}, a_{M-N+2}, \ldots, a_{M-1})) \tag{1.18}$$

ist.

Beweis: Wir zerlegen $\boldsymbol{A}$ in die Summe der Matrizen $\boldsymbol{L}$, $\boldsymbol{M}$ und $\boldsymbol{U}$ mit

$$\begin{aligned}
\boldsymbol{L} &:= \mathrm{toep}(\boldsymbol{o}'_N, (\boldsymbol{o}'_{M-N}, a_{M-N+1}, a_{M-N+2}, \ldots, a_{M-1})) \;, \\
\boldsymbol{M} &:= \mathrm{toep}(\boldsymbol{o}'_N, (a_0, a_1, \ldots, a_{M-N-1}, \boldsymbol{o}'_N)) \;, \\
\boldsymbol{U} &:= \mathrm{toep}((0, a_{-1}, \ldots, a_{1-N}), \boldsymbol{o}'_M) \;.
\end{aligned}$$

Somit ergibt sich

$$\boldsymbol{A}'\boldsymbol{A} = \boldsymbol{M}'\boldsymbol{M} \;+\; \boldsymbol{L}'\boldsymbol{M} \;+\; \boldsymbol{M}'\boldsymbol{L} \;+\; \boldsymbol{U}'\boldsymbol{M} \;+\; \boldsymbol{M}'\boldsymbol{U} \;+\; \boldsymbol{L}'\boldsymbol{L} + \boldsymbol{U}'\boldsymbol{U} \;. \tag{1.19}$$

Es gilt

$$\begin{aligned}
\boldsymbol{M}'\boldsymbol{M} &= \mathrm{stoep}(\boldsymbol{M}'\boldsymbol{M}\,\boldsymbol{e}_0)' \;, \\
\boldsymbol{M}'\boldsymbol{L} \;+\; \boldsymbol{L}'\boldsymbol{M} &= \mathrm{stoep}(\boldsymbol{M}'\boldsymbol{L}\,\boldsymbol{e}_0)' \;, \\
\boldsymbol{M}'\boldsymbol{U} \;+\; \boldsymbol{U}'\boldsymbol{M} &= \mathrm{stoep}(\boldsymbol{U}'\boldsymbol{M}\,\boldsymbol{e}_0)'
\end{aligned}$$

und somit

$$\boldsymbol{T}_N = \mathrm{stoep}(t_0, t_1, \ldots, t_{N-1}) \tag{1.20}$$

mit

$$\begin{aligned}
(t_0, t_1, \ldots, t_{N-1})' &= (\boldsymbol{M}'\boldsymbol{M} + \boldsymbol{M}'\boldsymbol{L} + \boldsymbol{U}'\boldsymbol{M})\,\boldsymbol{e}_0 \\
&= \boldsymbol{M}'\boldsymbol{A}\,\boldsymbol{e}_0 \;+\; \boldsymbol{U}'_N\,(a_0, a_1, \ldots, a_{N-1})' \;.
\end{aligned} \tag{1.21}$$

Beachten wir nun $\boldsymbol{L}'\boldsymbol{L} = \boldsymbol{L}'_N\boldsymbol{L}_N$ und $\boldsymbol{U}'\boldsymbol{U} = \boldsymbol{U}'_N\boldsymbol{U}_N$, so folgt die Behauptung. ■

Um den Vektor $\boldsymbol{t}_N = (t_0, t_1, \dots, t_{N-1})'$ vorzuberechnen, müssen wir die Matrix $\boldsymbol{M}$ zu einer (M_1, M_1)–Matrix mit $M_1 := 2^{\lfloor 1+\log(M-1)/\log(2)\rfloor}$ erweitern. Im folgenden Algorithmus benutzen wir wieder nur die Zerlegungen bezüglich DCT–II und DST–II. Zur Vereinfachung setzen wir $\boldsymbol{C} := \boldsymbol{Z}_{N,2}\boldsymbol{C}_N^{II}$ und $\boldsymbol{S} := \boldsymbol{Z}_{N,1}\boldsymbol{S}_N^{II}$.

Algorithmus 1.9 (*Schnelle Matrix–Vektor–Multiplikation* $\boldsymbol{y} := \boldsymbol{A}'\boldsymbol{A}\,\boldsymbol{x}_N$)
Eingabe: $N = 2^n$, $M \geq N$,

$$\boldsymbol{A} = (a_{j-k})_{j=0,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N},$$

$$x_k \in \mathbb{R} \; (k = 0, \dots, N-1).$$

Schritt 0. *Vorberechne für* $M_1 := 2^{\lfloor 1+\log(M-1)/\log(2)\rfloor}$ *die Größen*

$$\begin{aligned}(d_0, \dots, d_N)' &:= \frac{1}{2}\,\tilde{\boldsymbol{C}}_{N+1}^{I}\,(0, a_{-1}, \dots, a_{1-N}, 0)',\\ (\tilde{d}_1, \dots, \tilde{d}_{N-1})' &:= \frac{1}{2}\,\tilde{\boldsymbol{S}}_{N-1}^{I}\,(a_{-1}, \dots, a_{1-N})',\end{aligned}$$

$$\boldsymbol{D} := \operatorname{diag}(d_0, \dots, d_N)'\,,\ \tilde{\boldsymbol{D}} := \operatorname{diag}(0, \tilde{d}_1, \dots, \tilde{d}_{N-1}, 0)',$$

$$\begin{aligned}(d_0, \dots, d_N)' &:= \frac{1}{2}\,\tilde{\boldsymbol{C}}_{N+1}^{I}\,(0, a_{M-N-1}, \dots, a_{M-1}, 0)',\\ (\tilde{d}_1, \dots, \tilde{d}_{N-1})' &:= \frac{1}{2}\,\tilde{\boldsymbol{S}}_{N-1}^{I}\,(-a_{M-N-1}, \dots, -a_{M-1})',\end{aligned}$$

$$\boldsymbol{E} := \operatorname{diag}(d_0, \dots, d_N)'\,,\ \tilde{\boldsymbol{E}} := \operatorname{diag}(0, \tilde{d}_1, \dots, \tilde{d}_{N-1}, 0)',$$

$$\tilde{a}_k := \begin{cases} 0 & k = -M_1+1, \dots, -N\,,\\ a_k & k = -N+1, \dots, M-1\,,\\ 0 & k = M-1, \dots, M_1-1\,.\end{cases}$$

Vorberechne ferner

$$\begin{aligned}\boldsymbol{t} := (t_j)_{j=0}^{M_1-1} &= \operatorname{toep}((\tilde{a}_0, \tilde{a}_1, \dots, \tilde{a}_{M_1-1}), (\tilde{a}_0, \boldsymbol{o}'_{M_1-1}))\,(\tilde{a}_0, \tilde{a}_1, \dots, \tilde{a}_{M_1-1})'\\ \tilde{\boldsymbol{t}} := (\tilde{t}_j)_{j=0}^{N-1} &= (t_j)_{j=0}^{N-1} + \operatorname{toep}((0, a_{-1}, a_{-2}, \dots, a_{1-N}), \boldsymbol{o}'_N)\,(a_0, a_1, \dots, a_{N-1})'\end{aligned}$$

mit Hilfe von Lemma 1.7. *Bilde*

$$\begin{aligned}(d_0, \dots, d_N)' &:= \tilde{\boldsymbol{C}}_{N+1}^{I}\,(\tilde{t}_0, \tilde{t}_1, \dots, \tilde{t}_{N-1}, 0)',\\ \boldsymbol{F} &:= \operatorname{diag}(d_0, \dots, d_N)'\,.\end{aligned}$$

Schritt 1. *Bilde* $\boldsymbol{x_C} := \boldsymbol{C}\,\boldsymbol{x}$, $\boldsymbol{x_S} := \boldsymbol{S}\,\boldsymbol{x}$.
Schritt 2. *Berechne* $\boldsymbol{u} = \boldsymbol{L}'_N\,\boldsymbol{x}$ *und* $\boldsymbol{v} = \boldsymbol{U}'_N\,\boldsymbol{x}$ *mittels*

$$\begin{aligned}\boldsymbol{u} &:= \boldsymbol{C}'(\boldsymbol{D}\boldsymbol{x_C} + \tilde{\boldsymbol{D}}\boldsymbol{x_S}) + \boldsymbol{S}'(\boldsymbol{D}\boldsymbol{x_S} - \tilde{\boldsymbol{D}}\boldsymbol{x_C}) , \\ \boldsymbol{v} &:= \boldsymbol{C}'(\boldsymbol{E}\boldsymbol{x_C} + \tilde{\boldsymbol{E}}\boldsymbol{x_S}) + \boldsymbol{S}'(\boldsymbol{E}\boldsymbol{x_S} - \tilde{\boldsymbol{E}}\boldsymbol{x_C})\end{aligned}$$

Schritt 3. *Berechne* $\boldsymbol{u_C} := \boldsymbol{C}\,\boldsymbol{u}$, $\boldsymbol{u_S} := \boldsymbol{S}\,\boldsymbol{u}$, $\boldsymbol{v_C} := \boldsymbol{C}\,\boldsymbol{v}$, $\boldsymbol{v_S} := \boldsymbol{S}\,\boldsymbol{v}$.
Schritt 4. *Berechne* $\boldsymbol{y} = \boldsymbol{T}_N\boldsymbol{x} + \boldsymbol{L}_N\boldsymbol{u} + \boldsymbol{U}_N\boldsymbol{v}$ *mittels*

$$\begin{aligned}\boldsymbol{y} &:= \boldsymbol{C}'(\boldsymbol{F}\boldsymbol{x_C} + \boldsymbol{D}\boldsymbol{u_C} - \tilde{\boldsymbol{D}}\boldsymbol{u_S} + \boldsymbol{E}\boldsymbol{v_C} - \tilde{\boldsymbol{E}}\boldsymbol{v_S}) \\ &+ \boldsymbol{S}'(\boldsymbol{F}\boldsymbol{x_S} + \boldsymbol{D}\boldsymbol{u_S} + \tilde{\boldsymbol{D}}\boldsymbol{u_C} + \boldsymbol{E}\boldsymbol{v_S} + \tilde{\boldsymbol{E}}\boldsymbol{v_C}) .\end{aligned}$$

Ausgabe: y_k $(k = 0, \ldots , N-1)$.

Dieser Algorithmus benötigt $6n \cdot 2^n + \mathcal{O}(2^n)$ reelle Multiplikationen und $18n \cdot 2^n + \mathcal{O}(2^n)$ reelle Additionen. Nur in der Vorberechnung sind $\mathcal{O}(M_1 \log M_1)$ Schritte nötig. Diese schnelle Matrix–Vektor–Multiplikation werden wir in iterativen Verfahren in Abschnitt 3.4 anwenden.

Kapitel 2

Schnelle Polynomtransformationen

Bei vielen mathematischen Problemen ist eine gegebene Funktion durch ein Polynom zu approximieren. Ein Vorteil bei der Verwendung von Polynomen liegt darin, daß diese leicht ausgewertet werden können. Außerdem sind Polynome einfach zu integrieren und differenzieren.

Die Wahl einer geeigneten Basis von Π_N ist dabei von entscheidender Bedeutung. Die Basis der Monome $1, x, x^2, \ldots, x^N$ ist nur in einer Umgebung des Nullpunktes numerisch günstig (siehe [82], S. 133). Ein beliebiges kompaktes Intervall können wir durch eine affine Abbildung auf das Intervall $I := [-1, 1]$ transformieren. Auf dem Intervall I sind die Chebyshev–Polynome erster Art $\{T_n\}_{n=0}^N$ als Basis von Π_n besonders empfehlenswert (siehe z.B. [42], S. 196). Wir werden deshalb in Abschnitt 2.1 zeigen, daß man mit Polynomen in Chebyshev–Darstellung (2.1) einfach rechnen kann. Die Auswertung von Polynomen in Chebyshev–Form führt auf die diskreten trigonometrischen Transformationen aus Abschnitt 1.1. Die Multiplikation von Polynomen in Chebyshev–Form können wir ebenfalls mit den diskreten trigonometrischen Transformationen beschreiben (siehe Algorithmus 2.1). Es sind somit numerisch stabile und schnelle Algorithmen bekannt, die das Rechnen mit Polynomen sehr hohen Grades ermöglichen.

Aus verschiedenen Gründen sind auch andere orthogonale Polynome $\{P_k\}_{k=0}^N$ als Basen von Π_N von Bedeutung, beispielsweise bei der numerischen Lösung von Differentialgleichungen oder singulären Integralgleichungen (siehe z.B. [48]). Eine effiziente Zerlegung einer Funktion in eine Linearkombination von P_k $(k = 0, \ldots, N-1)$ ist dabei eine wichtige Aufgabenstellung. Deshalb wollen wir in Abschnitt 2.2 einen Algorithmus vorschlagen, der eine Basistransformation in die Basis der Chebyshev–Polynome realisiert (siehe Algorithmus 2.4).
Die tragenden Ideen des Algorithmus, nämlich die Teile–und–Herrsche–Strategie, die Kaskadensummation und eine Verallgemeinerung der Drei–Term–Rekursion, werden klar herausgestellt. Im Fall der Legendre–Polynome haben Alpert und Rokhlin [1] einen approximativen Algorithmus der Komplexität $\mathcal{O}(N \log 1/\varepsilon)$ vorgeschlagen, wobei ε eine geforderte Genauigkeit ist. Dabei wird die Basistransformationsmatrix angenähert.

Driscoll und Healy [43] gaben erstmalig einen exakten Algorithmus der Komplexität $\mathcal{O}(N \log^2 N)$ an. Dieser Algorithmus wurde ebenfalls für die Legendre–Polynome formuliert und in [44, 70] verallgemeinert. Dazu wurde die diskrete Fourier–Transformation (DFT) verwendet. Wir sind erstmals in der Lage, diesen Algorithmus mit Hilfe der Polynomarithmetik und einer Teile–und–Herrsche–Strategie zu formulieren. Dazu sind $\mathcal{O}(N \log^2 N)$ arithmetische Operationen nötig. Die Nutzung der diskreten trigonometrischen Transformationen erlaubt eine einfachere Darstellung dieses Algorithmus. Außerdem haben diese Algorithmen eine geringere arithmetische Komplexität. Die Herleitung der Basistransformation von $\{P_k\}_{k=0}^N$ in die Basis $\{T_k\}_{k=0}^N$ der Chebyshev–Polynome benötigt nur die Drei–Term–Rekursion der orthogonalen Polynome.

Es sei bemerkt, daß wir diesen Zugang [77] genutzt haben, um eine schnelle Fourier–Transformation auf der Sphäre S^2 zu beschreiben. Dazu verwenden wir eine Drei–Term–Rekursion und eine Zwei–Term–Rekursion der assoziierten Legendre–Funktionen. Die klare Darstellung des Algorithmus ermöglicht eine einfache heuristische Stabilisierung (siehe [78]). Eine andere heuristische Stabilisierung des Driscoll–Healy–Algorithmus wurde zuvor von Moore [69] vorgeschlagen.

2.1 Polynomarithmetik in Chebyshev–Darstellung

Das Ziel der folgenden Betrachtungen ist die effiziente Auswertung von Polynomen sowie von Polynomoperationen. Als Basis von Π_n verwenden wir die Chebyshev–Polynome T_k $(k = 0, \ldots, n)$ auf dem Intervall $I := [-1, 1]$. Ein beliebiges Polynom $P \in \Pi_n$ läßt sich eindeutig in der Form

$$P(x) = \sum_{k=0}^{n} a_k \, T_k(x) \quad (x \in I) \tag{2.1}$$

mit $a_k \in \mathbb{R}$ $(k = 0, \ldots, n)$ darstellen. Es ist bekannt, daß man mit Polynomen in Chebyshev–Darstellung gut rechnen kann. Dies wollen wir im folgenden genauer erklären.

Zur effizienten Berechnung eines Polynomwertes $P_n(x_0)$ $(x_0 \in I)$ benutzt man oft das Horner–Schema. Dieser Algorithmus reduziert den Polynomgrad von P_n schrittweise mittels der Rekursionsformel (1.2) und ist als Clenshaw–Algorithmus [40] bekannt. Für die Berechnung weniger Polynomwerte ist dieser Algorithmus der Komplexität $\mathcal{O}(n)$ geeignet. Möchte man viele Werte eines Polynoms (2.1) auf dem Gitter $\{c_{2j+1}^{2M} : \; j = 0, \ldots, M-1\}$ mit $M \geq n+1$ berechnen, bietet sich die Nutzung eines reellen schnellen DCT–III–Algorithmus an. Wir setzen $a_j := 0$ $(j = n+1, \ldots, M-1)$ und erhalten somit

$$(P(c_{2j+1}^{2M}))_{j=0}^{M-1} := \tilde{\boldsymbol{C}}_M^{III} \; \mathrm{diag}(2, 1, \ldots, 1)' \, (a_k)_{k=0}^{M-1}.$$

Umgekehrt ergeben sich die Koeffizienten a_k $(k = 0, \ldots, N)$ des Polynoms (2.1) durch Interpolation an den Knoten c_{2j+1}^{2M} $(j = 0, \ldots, M-1)$. Diese Interpolationseigenschaft erlaubt eine schnelle Multiplikation von Polynomen in der Chebyshev–Darstellung. Da

dieser Algorithmus in Abschnitt 2.2 in einer speziellen Form benötigt wird, geben wir ihn hier an.
Ein Polynom $P \in \Pi_n$ $(n \in \mathbb{N})$ sei in der Chebyshev–Darstellung (2.1) durch die reellen Koeffizienten a_k $(k = 0, \ldots, n)$ gegeben. Weiterhin seien von einem Polynom $Q \in \Pi_m$ $(m \in \mathbb{N})$ die Funktionswerte $Q(c_{2j+1}^{2M})$ $(j = 0, \ldots, M-1)$ bekannt. Wir wählen $M = 2^s$ $(s \in \mathbb{N})$ mit $M/2 \le m+n < M$. Der folgende Algorithmus berechnet die Chebyschev–Koeffizienten b_k $(k = 0, \ldots, m+n)$ des Polynoms

$$R(x) := P(x)\,Q(x) = \sum_{k=0}^{n+m} b_k\, T_k(x) \quad (x \in I)$$

in $\mathcal{O}(M \log M)$ Schritten. Er ist Grundlage für die schnelle Polynomtransformation (siehe Algorithmus 2.4) im folgenden Abschnitt.

Algorithmus 2.1 (*Schnelle Polynommultiplikation in Chebyshev–Darstellung*)

Eingabe: $M = 2^s$ $(s \in \mathbb{N})$ *mit* $M/2 \le m+n < M$,

$Q \in \Pi_m$ *mit* $Q(c_{2j+1}^{2M}) \in \mathbb{R}$ $(j = 0, \ldots, M-1)$,

$P \in \Pi_m$ *der Form* (2.1) *mit* $a_k \in \mathbb{R}$ $(k = 0, \ldots, n)$.

Schritt 1. *Berechne*

$$(P(c_{2j+1}^{2M}))_{j=0}^{M-1} := \tilde{\boldsymbol{C}}_M^{III}\, \mathrm{diag}(2, 1, \ldots, 1)'\, (a_k)_{k=0}^{M-1}$$

mittels schneller DCT–III (M) *von* $(a_k)_{k=0}^{M-1}$ *mit* $a_k := 0$ $(k = n+1, \ldots, M-1)$.

Schritt 2. *Bilde die* M *Produkte*

$$R(c_{2j+1}^{2M}) := P(c_{2j+1}^{2M})\, Q(c_{2j+1}^{2M}) \quad (j = 0, \ldots, M-1)\,.$$

Schritt 3. *Berechne*

$$(b_k)_{k=0}^{M-1} := \frac{2}{M}\, \mathrm{diag}(1/2, 1, \ldots, 1)'\, \tilde{\boldsymbol{C}}_M^{II} (R(c_{2j+1}^{2M}))_{j=0}^{M-1}$$

mittels schneller DCT–II (M) *von* $(R(c_{2j+1}^{2M}))_{j=0}^{M-1}$.

Ausgabe: $b_k \in \mathbb{R}$ $(k = 0, \ldots, m+n)$.

Bemerkung 2.2 Ein ähnlicher Algorithmus wurde in [5] angegeben. Dort wurden die Polynome auf dem Gitter c_j^N $(j = 0, \ldots, N)$ ausgewertet. Analoge Ergebnisse kann man bei Verwendung von Chebyshev–Polynomen zweiter Art erhalten. □

Weiterhin ist es möglich, mit einfachen Algorithmen Polynome in der Form (2.1) zu differenzieren und integrieren (siehe [86]).

2.2 Diskrete Polynomtransformationen

In Abschnitt 2.1 haben wir gesehen, daß man mit Polynomen gut rechnen kann, wenn diese in der Chebyshev–Darstellung (2.1) gegeben sind. Oft ist man an Entwicklungen bezüglich anderer orthogonaler Polynome interessiert. So möchte man Funktionen z.B. nach Jacobi–Polynomen oder Gegenbauer–Polynomen entwickeln. Umgekehrt ist eine Auswertung dieser Polynome auf einem Gitter ebenfalls von Interesse. Solche Fragestellungen sollen in diesem Abschnitt untersucht werden. Dazu wollen wir die Polynomarithmetik aus Abschnitt 2.1 nutzen. Diese wichtigen Aufgabenstellungen der angewandten Mathematik finden viele Anwendungen, so z.B. in der digitalen Signalverarbeitung, bei Quadraturverfahren oder beim Lösen von Differentialgleichungen mittels Spektralmethoden.

Um die Aufgabe genauer zu formulieren, führen wir zunächst folgende Bezeichnungen ein. Sei w eine nichtnegative, integrierbare Gewichtsfunktion mit

$$\int_{-1}^{1} w(x)\,\mathrm{d}x \;>\; 0\,.$$

Ferner sei $L^2_w(I)$ mit $I := [-1, 1]$ der reelle Hilbert–Raum mit dem Skalarprodukt

$$\langle f,\, g\rangle \;:=\; \int_{-1}^{1} w(x)\, f(x)\, g(x)\,\mathrm{d}x \quad (f,\, g \in L^2_w(I)).$$

Die zugehörige Norm von $f \in L^2_w(I)$ ist dann $\|f\|_{2,w} := \langle f, f\rangle^{1/2}$. Wir betrachten z.B. die Gewichtsfunktion

$$w(x) \;:=\; (1-x^2)^{\lambda-1/2} \quad (\lambda > -1/2;\; x \in (-1,\, 1))\,. \tag{2.2}$$

Sei $\{P_n\}_{n\in\mathbb{N}_0}$ eine Folge von orthogonalen Polynomen $P_n \in \Pi_n$ bezüglich $\langle\cdot,\cdot\rangle$, dann kann jedes $P \in \Pi_N$ als endliche Summe in der Form

$$P \;=\; \sum_{k=0}^{N} \frac{\langle P, P_k\rangle}{\|P_k\|^2}\, P_k\,, \tag{2.3}$$

geschrieben werden. Die Skalarprodukte $\langle P, P_k\rangle$ können dabei mittels einer Quadraturformel berechnet werden, z.B. durch

$$\langle P, P_k\rangle \;=\; \sum_{j=0}^{2N} w_j^{2N}\, P(c_j^{2N})\, P_k(c_j^{2N}) \tag{2.4}$$

mit den Gewichten

$$w_j^{2N} := \int_{-1}^{1} w(x) \frac{L(x)}{L'(c_j^{2N})(x-c_j^{2N})}\,\mathrm{d}x, \quad L(x) := \prod_{j=0}^{2N}(x-c_j^{2N})$$

und den Chebyshev–Knoten

$$c_j^N := \cos \frac{j\pi}{N} \qquad (j = 0, \dots, N)\,.$$

Für $w := 1$, d.h. für die Legendre–Polynome P_k, ergibt die Quadraturformel (2.4) die *Clenshaw–Curtis–Quadratur* mit den positiven Gewichten

$$w_j^{2N} := \frac{1}{N}\,(\varepsilon_j^{2N})^2 \sum_{l=0}^{N} (\varepsilon_l^N)^2 \,\frac{-2}{4l^2-1}\, c_{lj}^N \quad (j = 0, \dots, 2N).$$

Ähnliche Formeln für w_j^{2N} können für die Gewichtsfunktionen (2.2) gefunden werden. Diese Gewichte können mit der DCT–I(N) berechnet werden. Die Aufgabe besteht nun darin, folgende Probleme effizient zu lösen: Es seien $M,\, N \in \mathbb{N}$ mit $M \geq N$ zwei gegebene Zweierpotenzen.

1. Gegeben seien die Koeffizienten $a_k \in \mathbb{R}$ $(k = 0, \dots, N)$. Man berechne die *diskrete Polynomtransformation* DPT$(N+1, M+1)$: $\mathbb{R}^{N+1} \to \mathbb{R}^{M+1}$, die durch

$$\hat{a}_j := \sum_{k=0}^{N} a_k\, P_k(c_j^M) \quad (j = 0, \dots, M) \tag{2.5}$$

definiert ist. Die Transformationsmatrix $\boldsymbol{P} := (P_k(c_j^M))_{j,k=0}^{M,N}$ wird *Vandermonde–ähnliche Matrix* genannt. Mit $\boldsymbol{a} = (a_k)_{k=0}^N \in R^{N+1}$ und $\hat{\boldsymbol{a}} = (\hat{a}_k)_{k=0}^M \in \mathbb{R}^{M+1}$ gilt

$$\hat{\boldsymbol{a}} = \boldsymbol{P}\,\boldsymbol{a}\,.$$

2. Gegeben seien die Koeffizienten $b_j \in \mathbb{R}$ $(j = 0, \dots, M)$. Man berechne die *transponierte diskrete Polynomtransformation* TDPT$(M+1, N+1)$: $\mathbb{R}^{M+1} \to \mathbb{R}^{N+1}$, die durch

$$\tilde{b}_k := \sum_{j=0}^{M} b_j\, P_k(c_j^M) \quad (k = 0, \dots, N) \tag{2.6}$$

definiert ist. Mit $\boldsymbol{b} := (b_k)_{k=0}^M \in \mathbb{R}^{M+1}$ und $\tilde{\boldsymbol{b}} = (\hat{a}_k)_{k=0}^N \in \mathbb{R}^{N+1}$ gilt

$$\tilde{\boldsymbol{b}} := \boldsymbol{P}'\,\boldsymbol{b}\,.$$

Im ersten Fall wollen wir ein Polynom $P \in \Pi_N$, welches in der Form (2.3) gegeben ist, an den Chebyshev–Knoten c_j^M $(j = 0, \dots, M)$ berechnen. Im zweiten Fall wollen wir die Fourier–Koeffizienten von $P \in \Pi_N$ mit Hilfe einer Quadraturformel berechnen. In Abschnitt 2.1 haben wir gesehen, daß diese Probleme sehr effizient für den Fall $P_k = T_k$ gelöst werden können. Aus diesem Grund wollen wir im folgenden einen Basiswechsel von $\{P_n\}_{n=0}^N$ zu $\{T_n\}_{n=0}^N$ realisieren, damit wir das Problem (2.5) mit einer schnellen Kosinus–Transformation lösen können. Im Falle der Legendre–Polynome haben Alpert und Rokhlin [1] einen approximativen Algorithmus der Komplexität $\mathcal{O}(N \log 1/\varepsilon)$ vorgeschlagen, wobei ε eine geforderte Genauigkeit ist. Dazu wird die Basistransformationsmatrix angenähert. Unser neuer Algorithmus berechnet den Basiswechsel in $\mathcal{O}(N \log^2 N)$ arithmetischen Operationen unter Benutzung der Polynomarithmetik von Abschnitt 2.1 und einer Teile–und–Herrsche–Strategie. Diesen Algorithmus können wir für beliebige Polynome, die einer Drei–Term–Rekursion genügen, herleiten. Dazu ist es notwendig, $\mathcal{O}(N \log N)$ Polynomwerte von sogenannten zugeordneten Polynomen vorzuberechnen und zu speichern.

Die Gleichungen (2.3) – (2.4) zeigen, daß die Probleme (2.5) – (2.6) mit $M = 2N$, $a_k = \|P_k\|^{-2} \langle P, P_k \rangle$ und $b_j = w_j^{2N} P(c_j^{2N})$ „invers“ in dem Sinne sind, daß die zugehörige Transformationsmatrix $\boldsymbol{P}$ die Gleichungen

$$\begin{aligned} \boldsymbol{P} \operatorname{diag}(\|P_k\|^{-2})_{k=0}^{N} \boldsymbol{P}' \operatorname{diag}(w_j^{2N})_{j=0}^{2N} &= \boldsymbol{I}_{2N+1}, \\ \boldsymbol{P}' \operatorname{diag}(w_j^{2N})_{j=0}^{2N} \boldsymbol{P} \operatorname{diag}(\|P_k\|^{-2})_{k=0}^{N} &= \boldsymbol{I}_{N+1} \end{aligned}$$

erfüllt.

Eine Realisierung von (2.5) oder (2.6) erfordert im allgemeinen $\mathcal{O}(NM)$ arithmetische Operationen. Wir wollen erstmals einen Algorithmus angeben, der auf Polynomarithmetik beruht und der diese Probleme in $\mathcal{O}(N \log^2 N) + \mathcal{O}(M \log M)$ arithmetischen Operationen löst. Ein schneller Algorithmus für das Problem (2.5) impliziert eine Faktorisierung der Transformationsmatrix $\boldsymbol{P}$ in ein Produkt von dünnbesetzten Matrizen. Sollte also ein schneller Algorithmus für (2.5) bekannt sein, so können wir in einfacher Weise einen schnellen Algorithmus für das „transponierte“ Problem (2.6) mit der Transformationsmatrix $\boldsymbol{P}'$ ableiten. Deshalb werden wir uns im folgenden auf das Problem (2.5) beschränken.

Sei $\{P_n\}_{n \in \mathbb{N}_0}$ eine Folge von Polynomen, die durch die Drei–Term–Rekursion

$$\begin{aligned} P_n(x) &= (\alpha_n x + \beta_n) P_{n-1}(x) + \gamma_n P_{n-2}(x) \quad (n = 1, 2, \dots), \\ P_{-1}(x) &:= 0, \quad P_0(x) := 1 \end{aligned} \tag{2.7}$$

mit $\alpha_n, \beta_n, \gamma_n \in \mathbb{R}$ und $\alpha_n > 0, \gamma_n \neq 0$ $(n \in \mathbb{N})$ gegeben ist. Nach Favard's Satz ist $\{P_n\}_{n=0}^{\infty}$ eine Folge von orthogonalen Polynomen bezüglich einer Gewichtsfunktion w (siehe [39], Theorem 4.4).
Verschieben wir in (2.7) den Index n bei α_n, β_n, γ_n um $c \in \mathbb{N}_0$, so erhalten wir die *zugeordneten Polynome* $P_n(\,\cdot\,, c)$ von P_n, die durch

$$\begin{aligned} P_n(x,c) &:= (\alpha_{n+c} x + \beta_{n+c}) P_{n-1}(x,c) + \gamma_{n+c} P_{n-2}(x,c) \quad (n = 1, 2, \dots), \\ P_{-1}(x,c) &:= 0, \quad P_0(x,c) := 1. \end{aligned}$$

definiert sind.

Wir erhalten mittels vollständiger Induktion folgende Verallgemeinerung der Drei–Term–Rekursion (2.7):

Lemma 2.3 (siehe [6]) *Für beliebige $c,\, n \in \mathbb{N}_0$ gilt*

$$P_{c+n}(x) \;=\; P_n(x,c)\, P_c(x) \;+\; \gamma_{c+1} P_{n-1}(x, c+1) P_{c-1}(x)\,.$$

Aus Lemma 2.3 folgt

$$\begin{pmatrix} P_{c+n} \\ P_{c+n+1} \end{pmatrix} = \boldsymbol{U}'_n(\cdot, c) \begin{pmatrix} P_{c-1} \\ P_c \end{pmatrix} \tag{2.8}$$

mit

$$\boldsymbol{U}_n(x,c) \;:=\; \begin{pmatrix} \gamma_{c+1} P_{n-1}(x, c+1) & \gamma_{c+1} P_n(x, c+1) \\ P_n(x,c) & P_{n+1}(x,c) \end{pmatrix}.$$

Seien $N = 2^t$ und $M = 2^s$ $(s,\, t \in \mathbb{N};\; s \geq t)$ gegeben. Wir betrachten das Polynom

$$P \;:=\; \sum_{k=0}^{N} a_k\, P_k \in \Pi_N$$

mit bekannten Koeffizienten a_k. Unser Ziel ist die schnelle Berechnung von $P(c_j^M)$ $(j = 0, \ldots, M)$ in $\mathcal{O}(N \log^2 N) + \mathcal{O}(M \log M)$ anstelle von $\mathcal{O}(MN)$ arithmetischen Operationen. Der Hauptteil unseres Algorithmus realisiert einen Basiswechsel in Π_N von $\{P_k\}_{k=0}^N$ zu $\{T_k\}_{k=0}^N$ und berechnet die Chebyshev–Koeffizienten $\tilde{a}_k$ von

$$P \;=\; \sum_{k=0}^{N} \tilde{a}_k\, T_k\,. \tag{2.9}$$

Kennen wir diese Koeffizienten $\tilde{a}_k$, so können die Werte $P(c_j^M)$ $(j = 0, \ldots, M)$ mit einer schnellen DCT–I $(M+1)$ in $\mathcal{O}(M \log M)$ arithmetischen Operationen berechnet werden (siehe [85, 5]). In einem letzten Schritt erhalten wir

$$(P(c_j^M))_{j=0}^M \;=\; \tilde{\boldsymbol{C}}_{M+1}^I\, (\tilde{a}_k)_{k=0}^M\,. \tag{2.10}$$

Dabei haben wir $\tilde{a}_k := 0$ für $k = N+1, \ldots, M$ zu setzen.

Im folgenden beschreiben wir den *Basiswechsel*. Im *ersten Schritt* nutzen wir (2.7) und $T_1(x) = x$, um

$$P \;=\; \sum_{k=0}^{N-1} a_k^{(0)}\, P_k \;=\; \sum_{k=0}^{N/4-1} \left(\sum_{l=0}^{3} a_{4k+l}^{(0)}\, P_{4k+l} \right)$$

mit

$$\begin{aligned} a_k^{(0)} \;&:=\; a_k \quad (k = 0, \ldots, N-3)\,, \\ a_{N-2}^{(0)} \;&:=\; a_{N-2} \;+\; \gamma_{N-1} a_N\,, \\ a_{N-1}^{(0)} \;&:=\; a_{N-1} \;+\; \beta_{N-1} a_N \;+\; \alpha_{N-1} a_N\, T_1 \end{aligned} \tag{2.11}$$

zu erhalten. Wir fahren mit der *Kaskadensummation*, wie in Abbildung 2.1 schematisch dargestellt ist, fort. Nutzen wir (2.8) mit $n = 1$ und $c = 4k+1$ $(k = 0, \dots, N/4-1)$, so folgt

$$(a^{(0)}_{4k+2},\, a^{(0)}_{4k+3}) \begin{pmatrix} P_{4k+2} \\ P_{4k+3} \end{pmatrix} = (a^{(0)}_{4k+2},\, a^{(0)}_{4k+3})\, \boldsymbol{U}'_1(\,\cdot\,, 4k+1) \begin{pmatrix} P_{4k} \\ P_{4k+1} \end{pmatrix} .$$

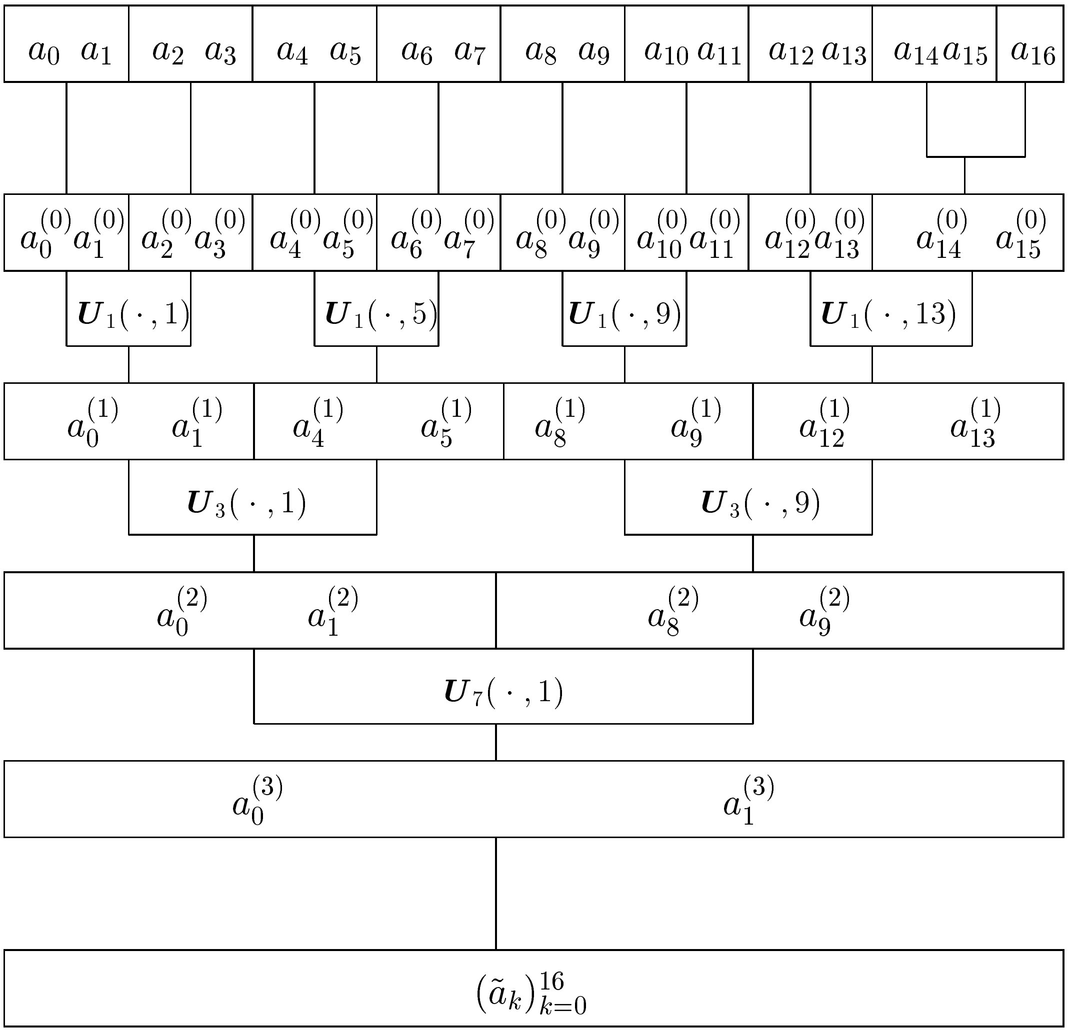

Abbildung 2.1: Kaskadensummation für den Basiswechsel im Fall $N = 16$

Somit erhalten wir

$$P = \sum_{k=0}^{N/4-1} (\, a^{(1)}_{4k}\, P_{4k} + a^{(1)}_{4k+1}\, P_{4k+1})$$

mit

$$\begin{pmatrix} a^{(1)}_{4k} \\ a^{(1)}_{4k+1} \end{pmatrix} := \begin{pmatrix} a^{(0)}_{4k} \\ a^{(0)}_{4k+1} \end{pmatrix} + \boldsymbol{U}_1(\,\cdot\,, 4k+1) \begin{pmatrix} a^{(0)}_{4k+2} \\ a^{(0)}_{4k+3} \end{pmatrix} . \tag{2.12}$$

Der Polynomgrad des Produktes in (2.12) ist höchstens 3, so daß die Berechnung bezüglich der Chebyshev–Polynome mittels Algorithmus 2.1 mit $M = 4$ realisiert werden kann. Somit benötigt die Berechnung der Chebyshev–Koeffizienten der Polynome $a^{(1)}_{4k}$, $a^{(1)}_{4k+1} \in \Pi_3$ $(k = 0, \ldots, N/4 - 1)$ in Schritt 1 nur $11N$ Multiplikationen und $12N$ Additionen.

In *Schritt* τ $(1 < \tau < t)$ berechnen wir mit Hilfe der Gleichung (2.8) mit $n = 2^\tau - 1$ die Chebyshev–Koeffizienten der Polynome $a^{(\tau)}_{2^{\tau+1}k}$, $a^{(\tau)}_{2^{\tau+1}k+1} \in \Pi_{2^{\tau+1}-1}$ $(k = 0, \ldots, N/2^{\tau+1} - 1)$ wie folgt

$$\begin{pmatrix} a^{(\tau)}_{2^{\tau+1}k} \\ a^{(\tau)}_{2^{\tau+1}k+1} \end{pmatrix} := \begin{pmatrix} a^{(\tau-1)}_{2^{\tau+1}k} \\ a^{(\tau-1)}_{2^{\tau+1}k+1} \end{pmatrix} + \boldsymbol{U}_{2^\tau-1}(\,\cdot\,, 2^{\tau+1}k+1) \begin{pmatrix} a^{(\tau-1)}_{2^{\tau+1}k+2^\tau} \\ a^{(\tau-1)}_{2^{\tau+1}k+2^\tau+1} \end{pmatrix}, \tag{2.13}$$

wobei wir Algorithmus 2.1 (mit $M = 2^{\tau+1}$) für das Produkt der Polynome nutzen. Die $4N$ Werte der Polynome $U_{2^\tau-1}(c^{2^{\tau+2}}_{2l+1}, 2^{\tau+1}k+1)$ für $k = 0, \ldots, N/2^{\tau+1}$ und $l = 0, \ldots, 2^{\tau+1} - 1$ seien durch einen Clenshaw–Algorithmus (siehe [40] oder [96], S. 165 – 172) vorberechnet. Somit benötigt der τ–te Schritt $(2\tau + 8 + 2^{1-\tau})N$ Multiplikationen und $(6\tau + 5 + 2^{1-\tau})N$ Additionen und kann in der Form

$$P = \sum_{k=0}^{N/2^{\tau+1}-1} \left(a^{(\tau)}_{2^{\tau+1}k} P_{2^{\tau+1}k} + a^{(\tau)}_{2^{\tau+1}k+1} P_{2^{\tau+1}k+1} \right).$$

geschrieben werden.
Wir beschreiben nun *Schritt* t. Nach $t-1$ Schritten ergibt diese Kaskadensummation

$$P = a^{(t-1)}_0 P_0 + a^{(t-1)}_1 P_1 .$$

Nun ist $P_0(x) = 1$, $P_1(x) = \alpha_1 x + \beta_1$ und

$$x\,T_0(x) = T_1(x)\,, \quad x\,T_n(x) = \frac{1}{2}\,(T_{n+1}(x) + T_{n-1}(x)) \quad (n = 1, 2, \ldots)\,.$$

Wenn also

$$a^{(t-1)}_0 = \sum_{n=0}^{N-1} a^{(t-1)}_{0,n} T_n\,, \qquad a^{(t-1)}_1 = \sum_{n=0}^{N-1} a^{(t-1)}_{1,n} T_n\,,$$

dann gilt

$$a^{(t)}_1 := a^{(t-1)}_1 P_1 = \sum_{n=0}^{N} a^{(t)}_{1,n} T_n$$

mit

$$(a^{(t)}_{1,n})_{n=0}^N = (\alpha_1 \boldsymbol{B}'_{N+1} + \beta_1 \boldsymbol{I}_{N+1})(a^{(t-1)}_{1,n})_{n=0}^N\,, \tag{2.14}$$

wobei wir $a_{0,N}^{(t-1)} = a_{1,N}^{(t-1)} := 0$ setzen und $\boldsymbol{B}_{N+1}$ eine tridiagonale $(N+1, N+1)$–Matrix

$$\boldsymbol{B}_{N+1} := \begin{pmatrix} 0 & 1 & & & \\ 1/2 & 0 & 1/2 & & \\ & \ddots & \ddots & \ddots & \\ & & 1/2 & 0 & 1/2 \\ & & & 1 & 0 \end{pmatrix} \tag{2.15}$$

ist. Dies führt zu

$$P = a_0^{(t-1)} + a_1^{(t)} .$$

Eine Addition der Chebyshev–Koeffizienten von $a_0^{(t-1)}$ und $a_1^{(t)}$ ergibt die gesuchten Chebyshev–Koeffizienten von P, d.h.

$$(\tilde{a}_n)_{n=0}^N = (a_{0,n}^{(t-1)})_{n=0}^N + (a_{1,n}^{(t)})_{n=0}^N . \tag{2.16}$$

Wir fassen zusammen:

Algorithmus 2.4 (*Schnelle Polynomtransformation* (FPT))

Eingabe: $N = 2^t$, $M = 2^s$ $(s, t \in \mathbb{N};\ s \geq t)$,
$a_k \in \mathbb{R}$ $(k = 0, \dots, N)$,
$\boldsymbol{U}_{2^\tau-1}(c_{2l+1}^{2^{\tau+2}}, 2^{\tau+1}k+1)$ $(\tau = 1, \dots, t-1;\ k = 0, \dots, 2^{t-\tau-1};$
$l = 0, \dots, 2^{\tau+1}-1)$.

Schritt 0. *Berechne* $a_k^{(0)}$ $(k = 0, \dots, 2^t - 1)$ *durch* (2.11).

Für $\tau = 1, \dots, t-1$ *berechne:*

Schritt τ. *Für jedes* $k = 0, \dots, 2^{t-\tau-1} - 1$ *bilde* (2.13) *mittels Algorithmus* 2.1 *für die schnelle Polynommultiplikation.*

Schritt t. *Berechne* $\tilde{a}_n$ $(n = 0, \dots, N)$ *durch* (2.14) *und* (2.16).

Schritt $t+1$. *Berechne* (2.10) *mittels schneller* DCT–I $(M+1)$.

Ausgabe: $P(c_j^M)$ $(j = 0, \dots, M)$.

Insgesamt müssen wir $4N(\log N - 1)$ Daten vorberechnen. Zählen wir die arithmetischen Operationen in jedem Schritt, so erkennen wir, daß der Basiswechsel in diesem Algorithmus $N \log^2 N + 7N \log N + \mathcal{O}(N)$ Multiplikationen und $3N \log^2 N + 2N \log N + \mathcal{O}(N)$ Additionen benötigt.
Schließlich erfordert die Berechnung von (2.10) noch eine DCT–I $(M+1)$, also $\frac{1}{2} M \log M + \mathcal{O}(M)$ Multiplikationen und $\frac{3}{2} M \log M + \mathcal{O}(M)$ Additionen. Die schnelle Polynomtransformation ist effizienter als der Clenshaw–Algorithmus, falls $N \geq 128$ ist. Dies erscheint sinnvoll, weil auch die schnelle Fourier–Transformation erst für $N \geq 64$ effizienter als eine direkte Berechnung ist.

Bemerkung 2.5 Eine schnelle Berechnung von (2.6), d.h. eine schnelle Multiplikation mit $\boldsymbol{P}'$, können wir einfach durch „Inversion“ von Algorithmus 2.4 erhalten. Mit anderen Worten, wir müssen die Flußrichtung von Algorithmus 2.4 umdrehen.

Es existiert bereits ein Algorithmus der Komplexität $\mathcal{O}(N \log^2 N)$ für das Problem (2.5) (als auch für (2.6)), welcher ursprünglich von Driscoll und Healy [43] bezüglich der Legendre–Polynome formuliert wurde. Dieser Algorithmus wurde für beliebige Polynome, die einer Drei–Term–Rekursion genügen, verallgemeinert (siehe [44, 50]). Der Zugang für diese schnellen Algorithmen, um das Problem (2.6) zu lösen, beruht hauptsächlich auf einer schnellen Toeplitz–Matrix–Vektor–Multiplikation. Dazu wird die schnelle Fourier–Transformation benutzt. Wir haben nicht nur die DFT durch diskrete trigonometrische Transformationen ersetzt, sondern außerdem die Herleitung des Algorithmus vereinfacht. Somit ist es durch den genannten Zugang möglich, eine schnelle diskrete Fourier–Transformation auf der Sphäre S^2 zu beschreiben. Dabei ist allerdings noch eine einfache heuristische Stabilisierung nötig (siehe [78]). Erstmals wurde eine heuristische Stabilisierung für den Driscoll–Healy–Algorithmus [43] von Moore [69] vorgeschlagen, um eine diskrete Fourier–Transformation auf der Sphäre S^2 zu berechnen. □

2.3 Numerische Ergebnisse

Wir haben den Algorithmus 2.4 in C auf einer Sun SPARCstation 20 implementiert und getestet. Die schnellen DCT–Algorithmen wurden von G. Baszenski in C programmiert (siehe [4]).

Beispiel 2.6 Wir betrachten die ultrasphärischen Polynome P_n^λ $(\lambda > -1/2)$, die durch

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

gegeben sind. Diese Polynome sind orthogonal bezüglich der Gewichtsfunktion (2.2). Für $\lambda = 1/2$ erhalten wir die Legendre–Polynome. Für gegebene $a_k \in \mathbb{R}$ $(k = 0, \ldots, N)$ berechnen wir

$$\hat{a}_j = \sum_{k=0}^{N} a_k\, P_k^\lambda(c_j^N) \quad (j = 0, \ldots, N) \tag{2.17}$$

auf verschiedene Weisen, nämlich mittels des Clenshaw–Algorithmus (CA) in doppelter Genauigkeit, des Clenshaw–Algorithmus mit einer hohen Genauigkeit von 64 Dezimalstellen in Maple (CA_{64}) und mit der schnellen Polynomtransformation (FPT) in doppelter Genauigkeit. In Tabelle 2.1 vergleichen wir die Ergebnisse für verschiedene Transformationslängen N und verschiedene Parameter λ. Die dritte Spalte von Tabelle 2.1 enthält die gegebenen Koeffizienten a_k. In der vierten und in der letzten Spalte geben wir den relativen Fehler $\varepsilon(\mathrm{CA})$ für den Clenshaw–Algorithmus

$$\varepsilon(\mathrm{CA}) := \max_{0 \le j \le N} |\hat{a}_j(\mathrm{CA}) - \hat{a}_j(\mathrm{CA}_{64})| / \max_{0 \le j \le N} |\hat{a}_j(\mathrm{CA}_{64})|$$

und den relativen Fehler für den Algorithmus 2.4

$$\varepsilon(\mathrm{FPT}) := \max_{0\le j\le N} |\hat{a}_j(\mathrm{FPT}) - \hat{a}_j(\mathrm{CA}_{64})| / \max_{0\le j\le N} |\hat{a}_j(\mathrm{CA}_{64})|$$

an. Hierbei bedeuten $\hat{a}_j(\mathrm{CA})$, $\hat{a}_j(\mathrm{CA}_{64})$ und $\hat{a}_j(\mathrm{FPT})$ die entsprechenden Ergebnisse von (2.17) unter Verwendung von CA, CA_{64} und FPT.

N	λ	a_k	$\varepsilon(\mathrm{CA})$	$\varepsilon(\mathrm{FPT})$
256	0.5	$1/(k+1)$	$3.88E-16$	$3.77E-13$
512	0.5	$1/(k+1)$	$1.59E-14$	$5.73E-12$
1024	0.5	$1/(k+1)$	$4.21E-13$	$8.98E-12$
2048	0.5	$1/(k+1)$	$2.11E-12$	$3.19E-11$
256	1.5	$1/(k+1)$	$1.88E-13$	$8.36E-13$
512	1.5	$1/(k+1)$	$6.12E-13$	$1.29E-11$
1024	1.5	$1/(k+1)$	$1.26E-12$	$8.00E-11$
256	5	$1/(k+1)$	$1.15E-13$	$2.72E-13$
512	5	$1/(k+1)$	$5.15E-13$	$4.37E-12$
1024	5	$1/(k+1)$	$1.04E-12$	$5.18E-12$
256	2	1	$2.44E-13$	$7.52E-13$
512	2	1	$8.61E-13$	$6.61E-12$
1024	2	1	$1.71E-12$	$4.82E-12$

Tabelle 2.1: Relative Fehler für verschiedene Transformationslängen

Der Clenshaw–Algorithmus und die schnelle Polynomtransformation realisieren die jeweilige diskrete Polynomtransformation mit etwa der gleichen Genauigkeit. □

Das folgende Beispiel zeigt, das die schnelle Polynomtransformation eine viel schnellere Implementierung von (2.17) als der Clenshaw–Algorithmus erlaubt.

Beispiel 2.7 Wie in Beispiel 2.6 nutzen wir hier die ultrasphärischen Polynome. Es ist bekannt, daß der Clenshaw–Algorithmus N^2 Multiplikationen und $3N^2$ Additionen benötigt. Der Algorithmus 2.4 der Komplexität $\mathcal{O}(N \log^2 N)$ ist schneller für $N \geq 128$. Die dritte und vierte Spalte von Tabelle 2.2 zeigt die CPU–Zeiten $t(\mathrm{CA})$ und $t(\mathrm{FPT})$ (in Sekunden) für den Clenshaw–Algorithmus und für den Algorithmus 2.4. Die letzte Spalte von Tabelle 2.2 enthält den relativen Fehler

$$\tilde{\varepsilon}(\mathrm{FPT}) := \max_{0\le j\le N} |\hat{a}_j(\mathrm{FPT}) - \hat{a}_j(\mathrm{CA})| / \max_{0\le j\le N} |\hat{a}_j(\mathrm{CA})|\,.$$

Hier haben wir die Eingabedaten a_k $(k = 0, \ldots, N)$ zufällig aus den Intervall $[-0.5, 0.5]$ gewählt. □

Wir haben die diskrete Polynomtransformation (2.5) und (2.6) bezüglich der $2N+1$ Chebshev–Knoten c_j^{2N} $(j = 0, \ldots, 2N)$ betrachtet. Dies scheint besonders nützlich,

N	λ	t(CA)	t(FPT)	$\tilde{\varepsilon}$(FPT)
128	0.5	0.05	0.04	$3.59E-14$
256	0.5	0.21	0.07	$4.35E-12$
512	0.5	0.82	0.19	$4.93E-12$
1024	0.5	3.27	0.39	$5.78E-11$
2048	0.5	13.70	0.85	$2.09E-10$
4096	0.5	55.41	1.92	$1.04E-09$
8192	0.5	220.05	4.26	$5.04E-08$
4096	2.5	55.43	1.91	$1.72E-09$
4096	4.0	55.42	1.91	$6.41E-10$
4096	5.0	55.42	1.92	$3.35E-10$

Tabelle 2.2: Rechenzeiten für verschiedene Transformationslängen

wenn wir die Clenshaw–Curtis–Quadratur für die Berechnung der Fourier–Koeffizienten in (2.3) nutzen. Wollen wir Quadraturformeln mit anderen Knoten benutzen, so haben wir

$$\sum_{k=0}^{N} a_k \, P_k(x_j^M) \quad (j = 0, \dots, M; M \geq N)$$

für beliebige Knoten $x_j^M \in [-1, 1]$ und $a_k \in \mathbb{R}$ effizient zu berechnen. Der Algorithmus 2.4 ist *nicht* von der Wahl der Knoten x_j^M abhängig. Somit stellt sich die Frage, wie wir

$$\sum_{k=0}^{N} \tilde{a}_k \, T_k(x_j^M) \quad (j = 0, \dots, M)\,, \tag{2.18}$$

schnell berechnen können. Eine Möglichkeit, (2.18) mit Hilfe der Multipol–Methode zu realisieren, wurde in [45] beschrieben (siehe auch [46]). Damit ergeben sich weitere Anwendungsmöglichkeiten der FPT, auf die hier nicht weiter eingegangen werden soll.

Kapitel 3

Trigonometrische Vorkonditionierer für Toeplitz–Matrizen

Eine (N,N)–Matrix

$$\boldsymbol{T}_N \ := \begin{pmatrix} a_0 & a_{-1} & \dots & a_{1-N} \\ a_1 & a_0 & \dots & a_{2-N} \\ \vdots & \vdots & & \vdots \\ a_{N-2} & a_{N-3} & \dots & a_{-1} \\ a_{N-1} & a_{N-2} & \dots & a_0 \end{pmatrix} = (a_{j-k})_{j,k=0}^{N-1} \in \mathbb{R}^{N,N}$$

wird *Toeplitz–Matrix* genannt. In vielen Bereichen der Mathematik treten Toeplitz–Matrizen auf, z.B. in der digitalen Signal– und Bildverarbeitung, bei der numerischen Lösung von Differential– bzw. Integralgleichungen, in der Approximationstheorie sowie in der Statistik. Diese vielfältigen Anwendungen motivierten Mathematiker und Ingenieure spezielle Löser für ein lineares Gleichungssystem mit einer Toeplitz–Koeffizientenmatrix (kurz: Toeplitz–System) zu entwickeln. Es gibt direkte und iterative Lösungsmethoden für Toeplitz–Systeme. Ein erstes Verfahren besteht darin, die spezielle Struktur der Toeplitz-Matrix so weit wie möglich in einem direkten Verfahren auszunutzen. Hier gibt es eine Vielzahl von Algorithmen der Komplexität $\mathcal{O}(N^2)$ (siehe z.B. Levinson (1946), Baxter (1961), Trench (1996)). Außerdem wurden Toeplitz–Löser mit der Komplexität $\mathcal{O}(N(\log N)^2)$ entwickelt (siehe z.B. Brent, Gustavson und Yun (1980), Ammar und Gragg (1988)). Diese Methoden nutzen die Gohberg–Semencul–Formel für die Inverse einer Toeplitz–Matrix aus. Es gibt aber eine große Klasse von Toeplitz–Matrizen, bei deren Lösung die direkten Verfahren numerisch instabil sind, z.B. indefinite oder nichtsymmetrische Toeplitz–Matrizen (siehe Bunch (1985)).
Eine zweite Möglichkeit, um ein großes Toeplitz–System $\boldsymbol{T}_N\, \boldsymbol{x} = \boldsymbol{b}$ zu lösen, ist ein iteratives Verfahren zur Bestimmung einer approximativen Lösung $\boldsymbol{x}$. Dies erscheint auch deshalb sinnvoll, weil wir in der Regel an der Lösung $\boldsymbol{x}$ nur bis auf eine Genauigkeit ε interessiert sind. Iterative Verfahren sind besonders gut geeignet, wenn die Matrix–Vektor–Multiplikation schnell ausgeführt werden kann (siehe Abschnitt 1.4). Zur iterativen Lösung von linearen Gleichungssystemen mit symmetrischer, positiv definiter

Koeffizientenmatrix hat sich mittlerweile das Verfahren der konjugierten Gradienten (CG–Verfahren) durchgesetzt. In Verbindung mit einer Vorkonditionierung wurde die PCG–Methode (Preconditioned Conjugate Gradient Method) in den letzten Jahren intensiv untersucht. Das Hauptergebnis ist dabei, daß sich gewisse Klassen von Toeplitz–Systemen in $\mathcal{O}(N \log N)$ arithmetischen Operationen numerisch stabil lösen lassen.
Ein weiteres iteratives Verfahren, daß sich besonders beim schnellen Lösen von großen linearen Gleichungssystemen hervorgetan hat, ist das *Mehrgitterverfahren*. Dieses Verfahren wurde auch zum Lösen von Toeplitz–Systemen verwendet [19, 91]. Einen schönen Überblick von diesem aktuellen Forschungsgegenstand geben die Arbeiten [28, 29].
In der vorliegenden Arbeit wollen wir iterative Löser für reelle Toeplitz–Systeme betrachten. Dazu konstruieren wir Vorkonditionierer für das PCG–Verfahren. Oft liefern diese Verfahren bessere Ergebnisse als die direkten Verfahren [60].
Strang [87] schlug vor, zirkulante Vorkonditionierer zur Lösung von Toeplitz–Systemen zu nutzen. Motiviert wird dies durch die bekannte Tatsache, daß sich zirkulante Matrizen durch die Fourier–Matrix diagonalisieren lassen (siehe Abschnitt 1.2). Somit können effiziente Verfahren angegeben werden, wenn wir die schnelle Fourier–Transformation benutzen. Numerische Tests zeigten eine sehr schnelle Konvergenz für eine große Klasse von Toeplitz–Matrizen. Dies wurde dann theoretisch in [32] bewiesen. Die Entwicklung weiterer zirkulanter Vorkonditionierer war die Folge (siehe [64, 92, 34, 58]). Zu den wichtigsten Typen zählt neben dem Strang–Vorkonditionierer der optimale zirkulante Vorkonditionierer [37].
Es gibt bereits einige Arbeiten, die anstelle der DFT die DST–I [14, 12, 59, 31, 10] bzw. die DCT–II [14, 20] zur Konstruktion von Vorkonditionierern für symmetrische Toeplitz–Matrizen benutzen. Wir werden hier einen *neuen einheitlichen Zugang* beschreiben, der es uns erlaubt, optimale trigonometrische Vorkonditionierer für *beliebige Matrizen* zu berechnen (siehe Abschnitt 3.2). In Abschnitt 3.3 werden wir dann Vorkonditionierer für symmetrische Toeplitz–Matrizen konstruieren. Außerdem wollen wir in Abschnitt 3.4 Vorkonditionierer für unsymmetrische und rechteckige Toeplitz–Matrizen angeben, indem wir zu der zugehörigen Normalgleichung übergehen. Erstmals sind wir in der Lage, optimale Vorkonditionierer für die Matrix $\boldsymbol{A}'\boldsymbol{A}$ in $\mathcal{O}(M \log M)$ arithmetischen Operationen zu konstruieren. Dabei bezeichnet $\boldsymbol{A} \in \mathbb{R}^{M,N}$ $(M \geq N)$ eine rechteckige Toeplitz–Matrix.
Falls die erzeugende Funktion φ für die Folge von Toeplitz–Matrizen $\boldsymbol{T}_N$ bekannt ist (siehe Abschnitte 3.3 und 3.4), empfehlen wir die vereinfachten Strang–Typ–Vorkonditionierer $\boldsymbol{M}_N(\varphi)$. Diese sind besonders für schlecht konditionierte Toeplitz–Matrizen geeignet. Während der Fertigstellung dieser Arbeit fanden wir einen Preprint [61] von Huckle. Dort wurden ebenfalls die vereinfachten Strang–Typ–Vorkonditionierer bezüglich der DST–I betrachtet. Wir sind jedoch erstmals in der Lage zu zeigen, daß unter zusätzlichen Voraussetzungen die Eigenwerte von der Matrix $\boldsymbol{M}_N^{-1}(\varphi)\boldsymbol{T}_N$ um 1 geclustert sind (siehe Satz 3.22). Die numerischen Ergebnisse bestätigen, daß diese neuen Vorkonditionierer bessere Ergebnisse als die bisher bekannten Vorkonditionierer [10, 18, 33, 83, 73] liefern.

Wir bemerken noch, daß sich viele unserer Ergebnisse auf weitere strukturierte reelle Matrizen (wie z.B. Toeplitz–ähnliche Matrizen, Hankel–Matrizen, Toeplitz–plus–

Hankel–Matrizen [63, 29] oder Toeplitz–plus–Band–Matrizen [27, 29]) übertragen lassen. Damit kann die numerische Komplexität der entsprechenden Algorithmen reduziert werden. Außerdem sind in vielen Fällen die trigonometrischen Vorkonditionierer für reelle Toeplitz–Blockmatrizen besser geeignet als zirkulante Matrizen. Besonders die vereinfachten Strang–Typ–Vorkonditionierer sind für schlecht konditionierte Toeplitz–Blockmatrizen besser als bisher bekannte Vorkonditionierer [38, 21, 66, 73] geeignet. Da die Übertragung auf den Block–Fall schon in [79] erfolgte, geben wir hier nur numerische Ergebnisse für den vereinfachten Strang–Typ–Vorkonditionierer an. Auch hier sind keine Vorkonditionierer bekannt, die in weniger Iterationsschritten gleich gute Ergebnisse liefern würden. Außerdem sind diese Vorkonditionierer besonders interessant, da sie keine zusätzlichen diskreten trigonometrischen Transformationen benötigen (siehe Bemerkung 3.29).

3.1 Bedingungen an Vorkonditionierer

In diesem Abschnitt wollen wir die Grundlagen für die iterative Lösung eines linearen Gleichungssystems

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{3.1}$$

mittels CG–Verfahren bereitstellen, wobei $\boldsymbol{A} \in \mathbb{R}^{N,N}$ eine symmetrische, positiv definite Matrix und $\boldsymbol{b} \in \mathbb{R}^N$ ist. Wir verzichten an dieser Stelle auf Einzelheiten und verweisen auf die schöne Darstellung in [2], S. 449 ff. Die Lösung $\boldsymbol{x}$ von (3.1) wird durch Vektoren $\boldsymbol{x}_k$ aus dem affinen Teilraum V_k approximiert, der durch

$$V_k := \boldsymbol{x}_0 + \operatorname{span}\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \ldots, \boldsymbol{A}^{k-1}\boldsymbol{r}_0\} \quad (k = 1, \ldots, n)\ ,$$

mit $V_0 := \{\boldsymbol{x}_0\}$ und $\boldsymbol{r}_0 := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$ erklärt ist. Wir definieren ein problemangepaßtes Skalarprodukt

$$(\boldsymbol{x}, \boldsymbol{y})_A \ := \boldsymbol{x}'\ \boldsymbol{A}\ \boldsymbol{y}$$

und die zugeordnete Norm (*Energienorm* von $\boldsymbol{A}$)

$$\|\boldsymbol{y}\|_A \ := \ \sqrt{(\boldsymbol{y}, \boldsymbol{y})_A}\ .$$

Die Lösung $\boldsymbol{x}_k$ des Minimierungsproblems

$$\|\boldsymbol{x}_k - \boldsymbol{x}\|_A \ = \ \min_{\boldsymbol{y} \in V_k} \|\boldsymbol{y} - \boldsymbol{x}\|_A \tag{3.2}$$

heißt *Ritz–Galerkin–Approximation* von $\boldsymbol{x}$ in V_k. Dies ergibt das von Hestenes und Stiefel [53] vorgestellte CG–Verfahren, welches das folgende Fehlerverhalten hat:

Satz 3.1 (siehe [42], S. 257) *Der Approximationsfehler des* CG–*Verfahrens läßt sich in der Energienorm durch*

$$\|\boldsymbol{x} - \boldsymbol{x}_k\|_A \ \le \ 2 \left(\frac{\sqrt{\operatorname{cond}(\boldsymbol{A})} - 1}{\sqrt{\operatorname{cond}(\boldsymbol{A})} + 1} \right)^k \ \|\boldsymbol{x} - \boldsymbol{x}_0\|_A \tag{3.3}$$

abschätzen, wobei $\operatorname{cond}(\boldsymbol{A})$ *die Konditionszahl von* $\boldsymbol{A}$ *bezüglich der Spektralnorm bezeichnet.*

Beweis: Da $\boldsymbol{x}_k$ die Lösung des Minimierungsproblems (3.2) ist, gilt

$$\|\boldsymbol{x}-\boldsymbol{x}_k\|_A \;=\; \min_{P\in\Pi_k, P(0)=1} \|P(\boldsymbol{A})\;(\boldsymbol{x}-\boldsymbol{x}_0)\|_A$$

Seien nun λ_j die Eigenwerte von $\boldsymbol{A}$ und $\boldsymbol{z}_j$ zugehörige orthonormierte Eigenvektoren, d.h.

$$\boldsymbol{A}\boldsymbol{z}_j = \lambda_j \boldsymbol{z}_j\;, \quad \boldsymbol{z}_j'\boldsymbol{z}_k = \delta_{j,k} \quad (j,k=1,\ldots,N)\;.$$

Dann gilt

$$\boldsymbol{x}-\boldsymbol{x}_0 \;=\; \sum_{j=1}^{N} \varrho_j \boldsymbol{z}_j \quad (\varrho_j \in \mathbb{R})$$

und somit

$$\|\boldsymbol{x}-\boldsymbol{x}_k\|_A^2 = (\boldsymbol{x}-\boldsymbol{x}_0)'\;\boldsymbol{A}\;(\boldsymbol{x}-\boldsymbol{x}_0) \;=\; \sum_{j=1}^{N} \lambda_j \varrho_j^2\;.$$

Wir betrachten nun

$$\begin{aligned}
\|P(\boldsymbol{A})\;(\boldsymbol{x}-\boldsymbol{x}_0)\|_A^2 &= (\boldsymbol{x}-\boldsymbol{x}_0)'P(\boldsymbol{A})'\;\boldsymbol{A}\;P(\boldsymbol{A})(\boldsymbol{x}-\boldsymbol{x}_0)\\
&= (\boldsymbol{x}-\boldsymbol{x}_0)'P(\boldsymbol{A})^2\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{x}_0)\\
&= (\boldsymbol{x}-\boldsymbol{x}_0)'\sum_{j=1}^{N}\lambda_j P(\lambda_j)^2\varrho_j\boldsymbol{z}_j\\
&= \sum_{j=1}^{N}\lambda_j P(\lambda_j)^2\varrho_j^2\\
&\le (\max_{j=1,\ldots,N} P(\lambda_j)^2)\;\sum_{k=1}^{N}\lambda_k\varrho_k^2\\
&\le (\max_{j=1,\ldots,N} |P(\lambda_j)|)^2\;\|\boldsymbol{x}-\boldsymbol{x}_0\|_A^2\;.
\end{aligned}$$

Folglich gilt

$$\|P(\boldsymbol{A})\;(\boldsymbol{x}-\boldsymbol{x}_0)\|_A \le (\max_{j=1,\ldots,N}|P(\lambda_j)|)\|\boldsymbol{x}-\boldsymbol{x}_0\|_A$$

und somit

$$\begin{aligned}
\|\boldsymbol{x}-\boldsymbol{x}_k\|_A &= \min_{P\in\Pi_k,P(0)=1} \|P(\boldsymbol{A})\;(\boldsymbol{x}-\boldsymbol{x}_0)\|_A\\
&\le \min_{P\in\Pi_k,P(0)=1}\;\max_{j=1,\ldots,N}|P(\lambda_j)|\;\|\boldsymbol{x}-\boldsymbol{x}_0\|_A\\
&\le \min_{P\in\Pi_k,P(0)=1}\;\max_{\lambda\in[\lambda_1,\lambda_N]}|P(\lambda)|\;\|\boldsymbol{x}-\boldsymbol{x}_0\|_A\;.
\end{aligned} \tag{3.4}$$

Aus der Minimax–Eigenschaft (siehe [42], S. 196) der Chebyshev–Polynome erster Art

$$\min_{P\in\Pi_k, P(0)=1} \big(\max_{\lambda\in[\lambda_1,\lambda_N]} |P(\lambda)|\big) \;=\; \left(T_k\Big(\frac{\lambda_N+\lambda_1}{\lambda_N-\lambda_1}\Big)\right)^{-1}$$

folgt die Behauptung. ■

Die Abschätzung der Konvergenzgeschwindigkeit hängt von der Konditionszahl $\mathrm{cond}(\boldsymbol{A})$ bezüglich der Spektralnorm ab. Daraus ergibt sich die folgende Frage: Wie kann man die Konditionszahl der Matrix $\boldsymbol{A}$ verringern? Dies versucht man durch sogenannte *Vorkonditionierung* zu erreichen.

Die Lösung von $\boldsymbol{Ax} = \boldsymbol{b}$ mit einer symmetrischen, positiv definiten Matrix $\boldsymbol{A}$ bleibt unverändert, wenn wir beide Seiten der Gleichung von links mit einer invertierbaren Matrix $\boldsymbol{M}^{-1}$ multiplizieren. Dabei müssen wir darauf achten, daß $\boldsymbol{M}$ symmetrisch und positiv definit ist. Nur so bleibt das CG–Verfahren anwendbar.
Eine ausführliche Herleitung des CG–Verfahrens und eine Einführung in die Krylov–Raum–Methoden ist z.B. in [2, 47] oder [81] zu finden. Wir geben hier nur das *vorkonditionierte* CG–*Verfahren* oder kurz PCG–Verfahren an.

Algorithmus 3.2 (PCG–*Verfahren*)

Eingabe: $N \in \mathbb{N}$,
$\boldsymbol{A}$, $\boldsymbol{M} \in \mathbb{R}^{N,N}$ *symmetrisch, positiv definit,*
$\boldsymbol{b} \in \mathbb{R}^N$, $\varepsilon > 0$.
Schritt 1. *Setze* $\boldsymbol{x} := \boldsymbol{o}_N$, $\beta := 0$, $\boldsymbol{r} := \boldsymbol{b}_N$, $\boldsymbol{p} := \boldsymbol{o}_N$.
Schritt 2. *Bestimme* $\boldsymbol{z} \in \mathbb{R}^N$ *mit* $\boldsymbol{Mz} = \boldsymbol{r}$.
Schritt 3. *Berechne* $\delta_0 := \boldsymbol{z}'\boldsymbol{r}$.
Schritt 4. *Ist* $\|\boldsymbol{r}\|_2 > \varepsilon\, \|\boldsymbol{b}\|_2$, *dann berechne*
$\boldsymbol{p} := \boldsymbol{z} + \beta\boldsymbol{p}$,
$\boldsymbol{y} := \boldsymbol{A}\,\boldsymbol{p}$,
$\alpha := \delta_0/(\boldsymbol{p}'\,\boldsymbol{y})$,
$\boldsymbol{x} := \boldsymbol{x} + \alpha\boldsymbol{p}$,
$\boldsymbol{r} := \boldsymbol{r} - \alpha\boldsymbol{y}$.
Bestimme $\boldsymbol{z} \in \mathbb{R}^N$ *mit* $\boldsymbol{Mz} = \boldsymbol{r}$.
$\delta_1 := \delta_0$, $\delta_0 := \boldsymbol{z}'\boldsymbol{r}$, $\beta := \delta_0/\delta_1$.
Gehe zu Schritt 4.
sonst stop.
Ausgabe: $\boldsymbol{x} \in \mathbb{R}^N$.

Der Approximationsfehler des PCG–Verfahrens (siehe [42], S. 262) läßt sich in der entsprechenden Energienorm durch

$$|||\boldsymbol{x} - \boldsymbol{x}_k||| \;\leq\; 2\left(\frac{\sqrt{\mathrm{cond}(\boldsymbol{M}^{-1}\boldsymbol{A})}-1}{\sqrt{\mathrm{cond}(\boldsymbol{M}^{-1}\boldsymbol{A})}+1}\right)^k \; |||\boldsymbol{x} - \boldsymbol{x}_0|||$$

abschätzen, wobei $|||\boldsymbol{x}|||^2 := \boldsymbol{x}'\boldsymbol{M}^{-1/2}\boldsymbol{A}\boldsymbol{M}^{-1/2}\boldsymbol{x}$ ist.

Bemerkung 3.3 Die obige Abschätzung wird wie Satz 3.1 bewiesen. Für die Konvergenz des PCG–Verfahrens ist aber nicht nur die Konditionszahl, also der größte und der kleinste Eigenwert der Matrix $\boldsymbol{M}^{-1}\boldsymbol{A}$ von Bedeutung, sondern die Lage aller Eigenwerte (siehe [2], S. 558 ff und Gleichung (3.4)). Man beachte, daß für jede Verteilung der Eigenwerte λ_j und für jedes k ein Startvektor $\boldsymbol{x}_0$ existiert, so daß in (3.4) das Gleichheitszeichen gilt. Ausgehend von dieser Ungleichung kann man die Konvergenzgeschwindigkeit des PCG–Verfahrens in bestimmten Fällen genauer abschätzen (siehe Satz 3.5). □

Das PCG–Verfahren heißt *superlinear* konvergent, wenn für jedes $\varepsilon \in (0,1)$ eine positive Konstante $C(\varepsilon)$ existiert, so daß für alle $k \in \mathbb{N}_0$

$$\frac{|||\boldsymbol{x}_{k+1} - \boldsymbol{x}|||}{|||\boldsymbol{x}_0 - \boldsymbol{x}|||} \le C(\varepsilon)\,\varepsilon^k \quad (\boldsymbol{x}_0 \neq \boldsymbol{x})$$

ist (siehe [28]). Dies bedeutet, daß die Anzahl der Iterationen, um eine geforderte Genauigkeit der Lösung $\boldsymbol{x}$ zu erreichen, unabhängig von der Dimension N beschränkt ist.

Die folgende Aufgabe ist nun zu lösen: Gesucht sind Vorkonditionierer $\boldsymbol{M} \in \mathbb{R}^{N,N}$ $(N \in \mathbb{N})$ mit den folgenden vier Eigenschaften:

(E1) Die Matrizen $\boldsymbol{M}$ sind symmetrisch und positiv definit.

(E2) Die arithmetische Komplexität zur Berechnung von $\boldsymbol{M}$ ist nicht größer als die arithmetische Komplexität bei einer Matrix–Vektor–Multiplikation mit der Koeffizientenmatrix $\boldsymbol{A}$.

(E3) Die arithmetische Komplexität zur Lösung von $\boldsymbol{M}\boldsymbol{z} = \boldsymbol{r}$ ist nicht größer als die arithmetische Komplexität bei einer Matrix–Vektor–Multiplikation mit der Koeffizientenmatrix $\boldsymbol{A}$.

(E4) Die Matrizen $\boldsymbol{M}^{-1}\boldsymbol{A}$ haben „geclusterte“ Eigenwerte um 1.

Die arithmetische Komplexität des CG–Verfahrens wird durch die Matrix–Vektor–Multiplikation mit der Koeffizientenmatrix $\boldsymbol{A}$ bestimmt. Im PCG–Verfahren ist zusätzlich in jedem Schritt noch mit der Matrix $\boldsymbol{M}^{-1}$ zu multiplizieren. Die Eigenschaften (E1) – (E3) ergeben sich somit aus der Forderung, ein effizientes PCG–Verfahren zu konstruieren.
Im folgenden wollen wir die Forderung (E4) begründen. Dazu ist es notwendig, den Begriff „geclusterte“ Eigenwerte zu erläutern. Anschließend soll gezeigt werden, daß diese Forderung zu einer superlinearen Konvergenz des PCG–Verfahrens führt.

Manchmal liegen die meisten Eigenwerte einer Folge von symmetrischen Matrizen $\boldsymbol{A}_N$ in der Nähe einer Menge. Solch eine Menge nennt man Cluster. Wir betrachten nur einelementige Cluster $\{p\}$ mit $p \in \{0,1\}$. Seien $\lambda_k(\boldsymbol{A}_N)$ $(k = 0, \ldots, N-1)$ die Eigenwerte von $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$. Für beliebiges $\varepsilon > 0$ bezeichnen wir mit $\gamma_N(\varepsilon)$ die Anzahl der derjenigen Eigenwerte $\lambda_k(\boldsymbol{A}_N)$ mit der Eigenschaft

$$|p - \lambda_k(\boldsymbol{A}_N)| > \varepsilon\,.$$

Dann heißt die Menge $\{p\}$ ein *echtes Cluster*, wenn für jedes $N \in \mathbb{N}$ und alle $\varepsilon > 0$

$$\gamma_N(\varepsilon) < K(\varepsilon)$$

gilt, wobei $K(\varepsilon) > 0$ eine von N unabhängige Konstante ist.

Oft ist es leichter anstelle von (E4) die folgende Eigenschaft (E4′) zu zeigen:

(E4′) Seien $\boldsymbol{A}_N, \boldsymbol{M}_N \in \mathbb{R}^{N,N}$ ($N \in \mathbb{N}$) zwei Folgen von symmetrischen, positiv definiten Matrizen mit $\|\boldsymbol{M}_N^{-1}\|_2 < \gamma$ für alle $N \in \mathbb{N}$. Für alle $\varepsilon > 0$ existieren Indizes $M, n \in \mathbb{N}$ mit $(n > 2M)$, so daß für alle $N > n$ gilt

$$\boldsymbol{A}_N - \boldsymbol{M}_N = \boldsymbol{W}_N + \boldsymbol{V}_N \ ,$$

wobei $\boldsymbol{W}_N \in \mathbb{R}^{N,N}$ eine symmetrische Matrix mit Rang $\leq M$ und $\boldsymbol{V}_N \in \mathbb{R}^{N,N}$ eine symmetrische Matrix mit $\|\boldsymbol{V}_N\|_2 \leq \varepsilon/\gamma$ ist.

Deshalb wollen wir im folgenden Satz zeigen, daß aus der Eigenschaft (E4′) die Eigenschaft (E4) folgt.

Satz 3.4 *Es gelte die Eigenschaft* (E4′). *Dann sind die Eigenwerte von* $\boldsymbol{M}_N - \boldsymbol{A}_N$ *um* 0 *und die Eigenwerte von* $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$ *um* 1 *geclustert. Falls außerdem* $\|\boldsymbol{A}_N\|_2 \leq \delta < \infty$ *für alle* $N \in \mathbb{N}$ *ist, konvergiert das* PCG*–Verfahren für* $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$ *superlinear.*

Beweis: Es ist klar, daß alle Eigenwerte von $\boldsymbol{V}_N$ betragsmäßig kleiner oder gleich ε/γ sind. Da $\boldsymbol{W}_N$ eine symmetrische Matrix mit Rang $\leq M$ mit $N > 2M$ ist, gilt nach dem Verflechtungssatz von Weyl (siehe [56], S. 184)

$$\lambda_k(\boldsymbol{V}_N) \leq \lambda_{k+M}(\boldsymbol{V}_N + \boldsymbol{W}_N) \leq \lambda_{k+2M}(\boldsymbol{V}_N) \quad (k = 1, 2, \ldots, N-2M) \ .$$

Somit können höchstens $2M$ Eigenwerte von $\boldsymbol{A}_N - \boldsymbol{M}_N$ außerhalb von $[-\varepsilon/\gamma, \varepsilon/\gamma]$ liegen. Es gilt

$$\begin{aligned} \boldsymbol{M}_N^{-1/2}\boldsymbol{A}_N\boldsymbol{M}_N^{-1/2} - \boldsymbol{I}_N &= \boldsymbol{M}_N^{-1/2}\boldsymbol{W}_N\boldsymbol{M}_N^{-1/2} + \boldsymbol{M}_N^{-1/2}\boldsymbol{V}_N\boldsymbol{M}_N^{-1/2} \\ &= \tilde{\boldsymbol{W}}_N + \tilde{\boldsymbol{V}}_N \ . \end{aligned}$$

Dabei ist $\|\tilde{\boldsymbol{V}}_N\|_2 \leq \varepsilon$ und der Rang von $\tilde{\boldsymbol{W}}_N$ ist höchstens M. Damit sind die Eigenwerte von $\boldsymbol{M}_N^{-1/2}\boldsymbol{A}_N\boldsymbol{M}_N^{-1/2} - \boldsymbol{I}_N$ um 0 geclustert. Somit sind die Eigenwerte

$$\lambda_k(\boldsymbol{M}_N^{-1/2}\boldsymbol{A}_N\boldsymbol{M}_N^{-1/2}) = \lambda_k(\boldsymbol{M}_N^{-1}\boldsymbol{A}_N)$$

um 1 geclustert. Die Clusterung führt dann zu einer superlinearen Konvergenz des PCG–Verfahrens (siehe z.B. [32, 17]), falls $\|\boldsymbol{A}_N\|_2$ gleichmäßig beschränkt ist. ∎

Eine bessere obere Schranke, um die benötigte Anzahl von Iterationen des PCG–Verfahrens abzuschätzen, findet man folgendermaßen:

Satz 3.5 (siehe [2], S. 573) *Die Eigenwerte von* $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$ *seien wie folgt mit gewissen* $a, b \in \mathbb{R}$ $(0 < a < b < \infty)$ *angeordnet:*

$$\begin{aligned} 0 < \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{q-1} \; &< \; a \leq \lambda_{q+1} \leq \ldots \leq \lambda_{N-p-1} \leq b \\ &< \; \lambda_{N-p} \leq \lambda_{N-p+1} \leq \ldots \leq \lambda_{N-1} \, . \end{aligned}$$

Dann konvergiert das PCG*–Verfahren in höchstens*

$$\left\lceil (\ln \frac{2}{\tau} + \sum_{k=0}^{q-1} \frac{b}{\lambda_k}) \; / \; \ln \frac{1 + (\frac{a}{b})^{1/2}}{1 - (\frac{a}{b})^{1/2}} \right\rceil + p + q$$

Schritten. Dabei ist $\tau > 0$ *die geforderte Genauigkeit*

$$\frac{|||\boldsymbol{x}_{k+1} - \boldsymbol{x}|||}{|||\boldsymbol{x}_0 - \boldsymbol{x}|||} \leq \tau$$

des PCG*–Verfahrens.*

Dieser Satz zeigt, daß die Anzahl der Iterationen nur von a, b und den Eigenwerten außerhalb von $[a, b]$ abhängt.

Wir fassen zusammen: Gesucht ist ein Vorkonditionierer $\boldsymbol{M} = \boldsymbol{M}_N$ von $\boldsymbol{A} = \boldsymbol{A}_N$ mit (E1) – (E4) bzw. mit (E1) – (E3) und (E4′). In den folgenden Abschnitten werden wir für unsere Vorkonditionierer $\boldsymbol{M}^{-1}$ diese Eigenschaften nachweisen.

3.2 Optimale trigonometrische Vorkonditionierer

In diesem Abschnitt wollen wir zeigen, wie man optimale trigonometrische Vorkonditionierer berechnen kann. Dieses Verfahren werden wir dann in den Abschnitten 3.3 und 3.4 anwenden.
Der Vorkonditionierer $\boldsymbol{M}_N$ soll aus einer Algebra

$$\mathcal{A}_{\boldsymbol{O}_N} := \{\boldsymbol{O}'_N \,(\mathrm{diag}\, \boldsymbol{d})\, \boldsymbol{O}_N : \boldsymbol{d} \in \mathbb{R}^N\} \tag{3.5}$$

gewählt werden. Dazu sei $\boldsymbol{O}_N$ eine gegebene orthogonale Matrix. Mit anderen Worten, Matrizen von der Algebra (3.5) können durch $\boldsymbol{O}_N$ „diagonalisiert“ werden. Motiviert wird dieses Vorgehen durch die Tatsache, daß Vorkonditionierer aus der Algebra

$$\tilde{\mathcal{A}}_{\boldsymbol{F}_N} \; := \; \{\bar{\boldsymbol{F}}_N \,(\mathrm{diag}\, \boldsymbol{d})\, \boldsymbol{F}_N : \boldsymbol{d} \in \mathbb{C}^N\}$$

als *zirkulante Vorkonditionierer* bezeichnet werden. Als orthogonale Matrizen werden wir die trigonometrischen Matrizen aus Abschnitt 1.1 benutzen. Zunächst definieren wir ein Skalarprodukt in $\mathbb{R}^{N,N}$. Für $\boldsymbol{A}_N, \boldsymbol{B}_N \in \mathbb{R}^{N,N}$ erklären wir

$$\langle \boldsymbol{A}_N, \boldsymbol{B}_N \rangle := \mathrm{tr}(\boldsymbol{A}'_N \boldsymbol{B}_N) = \sum_{j,k=0}^{N-1} a_{j,k}\, b_{j,k}. \tag{3.6}$$

Dieses Skalarprodukt induziert die *Frobenius–Norm*

$$\|\boldsymbol{A}_N\|_F := (\mathrm{tr}(\boldsymbol{A}'_N\boldsymbol{A}_N))^{1/2} \ .$$

Für eine gegebene Matrix $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ wird eine Matrix $\boldsymbol{M}_N^{\mathcal{O}} := \boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{A}_N, \boldsymbol{O}_N) \in \mathbb{R}^{N,N}$ *optimaler Vorkonditionierer* von $\boldsymbol{A}_N$ in $\mathcal{A}_{\boldsymbol{O}_N}$ genannt, falls (siehe [37])

$$||\boldsymbol{M}_N^{\mathcal{O}} - \boldsymbol{A}_N||_F = \min\{||\boldsymbol{B}_N - \boldsymbol{A}_N||_F : \boldsymbol{B}_N \in \mathcal{A}_{O_N}\} \tag{3.7}$$

gilt. Optimale Vorkonditionierer wurden bereits bezüglich der DFT [37] (*optimaler zirkulanter Vorkonditionierer*), der DST–I [31], der DCT–II [20] und der diskreten Hartley–Transformation [13] betrachtet. Ist $\boldsymbol{O}_N$ eine orthogonale trigonometrische Matrix aus Abschnitt 1.1, so wird $\boldsymbol{M}_N^{\mathcal{O}}$ als *optimaler trigonometrischer Vorkonditionierer* von $\boldsymbol{A}_N$ bezeichnet. Insbesondere erkennen wir

$$||\boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}'_N||_F^2 = \mathrm{tr}(\boldsymbol{O}_N\boldsymbol{A}'_N\boldsymbol{O}'_N\boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}'_N) = \mathrm{tr}(\boldsymbol{A}'_N\boldsymbol{A}_N) = ||\boldsymbol{A}_N||_F^2 \ . \tag{3.8}$$

Mit dem Skalarprodukt (3.6) wird $\mathbb{R}^{N,N}$ ein Hilbert–Raum. Das Minimierungsproblem (3.7) ist somit eindeutig lösbar. Das folgende Lemma zeigt zwei Wege, um den optimalen Vorkonditionierer für $\boldsymbol{A}_N$ zu beschreiben.

Lemma 3.6 *Sei $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$. Ferner sei $\boldsymbol{O}_N \in \mathbb{R}^{N,N}$ eine orthogonale Matrix und $\mathcal{A}_{\boldsymbol{O}_N}$ die Algebra* (3.5)*. Dann lautet der optimale Vorkonditionierer von $\boldsymbol{A}_N$ bezüglich $\boldsymbol{O}_N$*

$$\boldsymbol{M}_N^{\mathcal{O}} = \boldsymbol{O}'_N\,\delta(\boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}'_N)\,\boldsymbol{O}_N \ . \tag{3.9}$$

Ist $\{\boldsymbol{B}_0, \dots, \boldsymbol{B}_{N-1}\}$ eine Basis von $\mathcal{A}_{\boldsymbol{O}_N}$, so kann $\boldsymbol{M}_N^{\mathcal{O}}$ in der Form

$$\boldsymbol{M}_N^{\mathcal{O}} := \sum_{k=0}^{N-1} \alpha_k \boldsymbol{B}_k \tag{3.10}$$

dargestellt werden. Dabei ist der Koeffizientenvektor $\boldsymbol{\alpha} := (\alpha_0, \dots, \alpha_{N-1})'$ die eindeutige Lösung von $\boldsymbol{G}\boldsymbol{\alpha} = \boldsymbol{\beta}$ mit

$$\boldsymbol{G} := (\langle \boldsymbol{B}_k, \boldsymbol{B}_j\rangle)_{j,k=0}^{N-1} \ , \ \boldsymbol{\beta} := (\langle \boldsymbol{A}_N, \boldsymbol{B}_j\rangle)_{j=0}^{N-1} \quad .$$

Beweis: 1. Unter Beachtung von (3.8) folgt für $\boldsymbol{M}_N^{\mathcal{O}} = \boldsymbol{O}'_N\,(\mathrm{diag}\,\boldsymbol{d})\,\boldsymbol{O}_N \in \mathcal{A}_{\boldsymbol{O}_N}$, daß

$$||\boldsymbol{M}_N^{\mathcal{O}} - \boldsymbol{A}_N||_F = ||\,\mathrm{diag}\,\boldsymbol{d} - \boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}'_N||_F$$

ist. Benutzen wir die Definition (3.7) des optimalen Vorkonditionierers, so erhalten wir die Behauptung (3.9) (siehe [22] und [89]).
2. Da $\mathbb{R}^{N,N}$ mit dem Skalarprodukt (3.6) ein Hilbert–Raum ist, ist die Bestimmung des optimalen Vorkonditionierers mit der Berechnung der besten Approximation von $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ in dem linearen Teilraum $\mathcal{A}_{\boldsymbol{O}_N}$ äquivalent. Diese Aufgabe kann mit dem Galerkin–Verfahren gelöst werden. Dazu setzen wir (3.10) in die Forderung

$$\langle \boldsymbol{M}_N^{\mathcal{O}} - \boldsymbol{A}_N, \boldsymbol{B}_j\rangle \ = 0 \quad (j = 0, \dots, N-1)$$

ein und erhalten das lineare Gleichungssystem $\boldsymbol{G\alpha} = \boldsymbol{\beta}$. ∎

Mit dem folgenden Lemma erkennen wir, daß der optimale Vorkonditionierer die Eigenschaft (E1) erfüllt, falls $\boldsymbol{A}_N$ symmetrisch und positiv definit ist.

Lemma 3.7 *Sei $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ symmetrisch und positiv definit. Außerdem sei $\mathcal{A}_{\boldsymbol{O}_N}$ durch* (3.5) *bezüglich einer orthogonalen Matrix $\boldsymbol{O}_N$ gegeben. Dann ist der optimale Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}} = \boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{A}_N, \boldsymbol{O}_N)$ symmetrisch und positiv definit.*

Beweis: Die Symmetrie von $\boldsymbol{M}_N^{\mathcal{O}}$ folgt aus der Definition von $\mathcal{A}_{\boldsymbol{O}_N}$.
Aus (3.9) ist ersichtlich, daß die Eigenwerte der Matrix $\boldsymbol{M}_N^{\mathcal{O}}$ durch die Diagonalelemente $(\boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}_N')_{k,k}$ $(k = 0, \ldots, N-1)$ der Matrix $\boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}_N'$ gegeben sind. Da die Matrix $\boldsymbol{A}_N$ positiv definit ist, ergibt sich

$$\begin{aligned} 0 &< \min\left\{\frac{\boldsymbol{x}_N'\boldsymbol{A}_N\boldsymbol{x}_N}{\boldsymbol{x}_N'\boldsymbol{x}_N} : \boldsymbol{x}_N \neq \boldsymbol{o}_N\right\} = \min\left\{\frac{\boldsymbol{y}_N'\boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}_N'\boldsymbol{y}_N}{\boldsymbol{y}_N'\boldsymbol{y}_N} : \boldsymbol{y}_N \neq \boldsymbol{o}_N\right\} \\ &\leq \boldsymbol{e}_k'(\boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}_N')\boldsymbol{e}_k = (\boldsymbol{O}_N\boldsymbol{A}_N\boldsymbol{O}_N')_{k,k} \quad (k = 0, \ldots, N-1) \end{aligned}$$

und somit die Behauptung. ∎

Im folgenden wollen wir erklären, wie die optimalen trigonometrischen Vorkonditionierer mit Hilfe des Galerkin–Verfahrens konstruiert werden. Wir beschränken uns wieder auf Vorkonditionierer bezüglich $\boldsymbol{C}_N^{II}$. Die Konstruktion bezüglich der anderen trigonometrischen Matrizen verläuft analog.
Wir benutzen die Darstellung (3.10) von $\boldsymbol{M}_N^{\mathcal{O}}$ mit der Basis $\{\boldsymbol{B}_k^{II} : k = 0, \ldots, N-1\}$ von $\mathcal{A}_{\boldsymbol{C}_N^{II}}$ (siehe [15, 51]):

$$\boldsymbol{B}_k^{II} := (\boldsymbol{C}_N^{II})' \ \operatorname{diag}(U_k(c_l^N))_{l=0}^{N-1} \ \boldsymbol{C}_N^{II} .$$

Dabei ist $c_l^N := \cos \dfrac{l\pi}{N}$. Mit U_k bezeichnen wir das k–te Chebyshev–Polynom zweiter Art (siehe (1.3)). Außerdem sei $\{\boldsymbol{B}_k^I : k = 0, \ldots, N-1\}$ mit

$$\begin{aligned} \boldsymbol{B}_k^I &:= (\boldsymbol{C}_N^{II})' \ \operatorname{diag}(T_k(c_l^N))_{l=0}^{N-1} \ \boldsymbol{C}_N^{II} \\ &= \frac{1}{2}(\varepsilon_k^N)^{-2}\left(\operatorname{stoep} \ \boldsymbol{e}_k' \ + \ \operatorname{shank} \ \boldsymbol{e}_{k-1}'\right) \quad (k = 0, \ldots, N-1) \end{aligned} \tag{3.11}$$

und $\boldsymbol{e}_{-1} := \boldsymbol{o}_N$ eine weitere Basis von $\mathcal{A}_{\boldsymbol{C}_N^{II}}$. Dabei gilt der Zusammenhang

$$\begin{aligned} \boldsymbol{B}_0^I &= \boldsymbol{B}_0^{II} = \boldsymbol{I} \ , \ \boldsymbol{B}_1^{II} = 2\boldsymbol{B}_1^I \, , \\ \boldsymbol{B}_j^{II} &= \boldsymbol{B}_{j-2}^{II} \ + \ 2\boldsymbol{B}_j^I \quad (j = 2, \ldots, N-1) \, , \end{aligned} \tag{3.12}$$

wobei die letzte Gleichung aus der Rekursionsformel $U_j = U_{j-2} + 2T_j$ $(j > 2)$ folgt. Andererseits erhalten wir mit Hilfe der Darstellung (3.10), daß

$$\boldsymbol{M}_N^{\mathcal{O}} = \sum_{k=0}^{N-1} \alpha_k \, \boldsymbol{B}_k^{II}$$

mit

$$\boldsymbol{\alpha} = \boldsymbol{G}^{-1} \boldsymbol{\beta}^{II} , \tag{3.13}$$

$$\boldsymbol{G} := \left(\langle \boldsymbol{B}_j^{II}, \boldsymbol{B}_k^{II} \rangle\right)_{j,k=0}^{N-1} , \ \boldsymbol{\beta}^{II} := \left(\langle \boldsymbol{A}_N, \boldsymbol{B}_j^{II} \rangle\right)_{j=0}^{N-1}$$

ist. Damit wir im Algorithmus 3.2 mit dem optimalen trigonometrischen Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}}$ schnell multiplizieren können, ist derjenige Vektor $\boldsymbol{d} \in \mathbb{R}^N$ mit der Eigenschaft

$$\boldsymbol{M}_N^{\mathcal{O}} = (\boldsymbol{C}_N^{II})' \, (\operatorname{diag} \boldsymbol{d}) \, \boldsymbol{C}_N^{II} \tag{3.14}$$

gesucht. Falls $\boldsymbol{\alpha} \in \mathbb{R}^N$ bekannt ist, erhalten wir $\boldsymbol{d}$ durch

$$\operatorname{diag} \boldsymbol{d} = \sum_{k=0}^{N-1} \alpha_k \, \boldsymbol{C}_N^{II} \, \boldsymbol{B}_k^{II} \, (\boldsymbol{C}_N^{II})' = \sum_{k=0}^{N-1} \alpha_k \, \operatorname{diag} \left(U_k(c_l^N)\right)_{l=0}^{N-1} .$$

Mit Hilfe der Definition von U_k ergibt sich

$$d_0 = \sum_{k=0}^{N-1} (k+1) \, \alpha_k \, , \ d_k = \frac{\hat{\alpha}_k}{\sin \frac{k\pi}{N}} \quad (k = 1, \ldots , N-1) , \tag{3.15}$$

$$(\hat{\alpha}_k)_{k=1}^{N-1} := \tilde{\boldsymbol{S}}_{N-1}^{I} \, (\alpha_{k-1})_{k=1}^{N-1} . \tag{3.16}$$

Daraus ist erkennbar, daß für die Konstruktion von $\boldsymbol{d}$ aus den gegebenen Daten $\boldsymbol{\alpha}$ nur $\mathcal{O}(N \log N)$ arithmetische Operationen nötig sind. Es soll nun gezeigt werden, wie wir den Koeffizientenvektor $\boldsymbol{\alpha}$ effizient berechnen können.

Lemma 3.8 *Für $\boldsymbol{G} := \left(\langle \boldsymbol{B}_j^{II}, \boldsymbol{B}_k^{II} \rangle\right)_{j,k=0}^{N-1}$ gilt*

$$\boldsymbol{G}^{-1} = \frac{1}{2N} \begin{pmatrix} 3 & 0 & -1 & & & & \\ 0 & 2 & 0 & -1 & & & \\ -1 & 0 & 2 & 0 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & -1 & 0 & 2 & 0 & -1 \\ & & & -1 & 0 & 3 & -2 \\ & & & & -1 & -2 & \frac{3N-2}{N} \end{pmatrix} ,$$

so daß eine Matrix–Vektor–Multiplikation mit $\boldsymbol{G}^{-1}$ in $\mathcal{O}(N)$ Rechenoperationen gebildet werden kann.

Beweis: Wir zeigen, daß $\boldsymbol{G}^{-1}\ \boldsymbol{G} = \boldsymbol{I}_N$ gilt. Für den (k,l)–ten Eintrag $g_{k,l}$ von $\boldsymbol{G}$ erkennen wir

$$\begin{aligned} g_{k,l} &:= \langle \boldsymbol{B}_k^{II}\,,\, \boldsymbol{B}_l^{II}\rangle \;=\; \mathrm{tr}\,\left((\boldsymbol{B}_l^{II})'\,\boldsymbol{B}_k^{II}\right) \\ &= \sum_{j=0}^{N-1} U_l(c_j^N)\, U_k(c_j^N)\,. \end{aligned}$$

Dann erhalten wir für $l = 2,\dots,N-3$

$$2g_{k,l} - g_{k,l-2} - g_{k,l+2} \;=\; \sum_{j=0}^{N-1} U_k(c_j^N)\ \left(2U_l(c_j^N) - U_{l-2}(c_j^N) - U_{l+2}(c_j^N)\right)\ .$$

Mit der Identität

$$2U_l(x) - U_{l-2}(x) - U_{l+2}(x) \;=\; 4(1-x^2)\, U_l(x)$$

und der Definition von U_l folgt für alle $k = 0,\dots,N-1$ und $l = 2,\dots,N-3$ dann

$$2g_{k,l} - g_{k,l-2} - g_{k,l+2} \;=\; 4\sum_{j=1}^{N-1} \sin\frac{(k+1)j\pi}{N}\sin\frac{(l+1)j\pi}{N} \;=\; 2N\ \delta_{k,l}\ .$$

Eine einfache Rechnung für die restlichen Fälle $l = 0, 1, N-2, N-1$ vervollständigt den Beweis. ■

Lemma 3.9 *Der Vektor $\boldsymbol{\beta}^{II}$ kann aus den Daten*

$$\boldsymbol{\beta}^{I} := \left(\langle \boldsymbol{A}_N, \boldsymbol{B}_j^I\rangle\right)_{j=0}^{N-1}\ . \tag{3.17}$$

in $\mathcal{O}(N)$ Additionen berechnet werden.

Beweis: Mit der Rekursionsformel (3.12) erhalten wir

$$\beta_0^{II} = \beta_0^I\ ,\ \beta_1^{II} = 2\beta_1^I\ ,\ \beta_k^{II} = \beta_{k-2}^{II} + \beta_k^I \quad (k = 2,\dots,N-1)\,. \tag{3.18}$$

und somit die Behauptung. ■

Wir sind jetzt in der Lage, den optimalen trigonometrischen Vorkonditionierer für beliebige Matrizen $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ in $\mathcal{O}(N^2)$ Rechenoperationen zu berechnen. Dazu müssen wir nur die Skalarprodukte (3.17) berechnen. Mit Hilfe von Lemma 3.9, der Formel (3.13) und einer DST können wir den optimalen trigonometrischen Vorkonditionierer dann in $\mathcal{O}(N\log N)$ Operationen berechnen. In Abschnitt 3.3 werden wir den optimalen trigonometrischen Vorkonditionierer für Toeplitz–Matrizen in $\mathcal{O}(N\log N)$ Operationen berechnen. In Abschnitt 3.4 wird gezeigt, daß wir den optimalen Vorkonditionierer auch für die Lösung der Normalgleichung $\boldsymbol{A}'\boldsymbol{A}\ \boldsymbol{x} = \boldsymbol{A}'\boldsymbol{b}$ in $\mathcal{O}(M\log M)$ Operationen berechnen können. Dabei ist $\boldsymbol{A} \in \mathbb{R}^{M,N}$ mit $M \geq N$ eine beliebige rechteckige Toeplitz–Matrix.

Bemerkung 3.10 In [54, 55] wird ein optimaler trigonometrischer Vorkonditionierer für Matrizen $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$, die durch Diskretisierung einer partiellen Differentialgleichung entstehen, berechnet. Unser Zugang erlaubt ebenfalls eine schnelle und einfache Berechnung der Skalarprodukte (3.17) für derartige dünnbesetzte Matrizen und somit eine einfache Berechnung des optimalen trigonometrischen Vorkonditionierers. □

Bemerkung 3.11 Wir können ähnliche Ideen für die Konstruktion der optimalen trigonometrischen Vorkonditionierer bezüglich $\boldsymbol{C}_N^{IV}$, $\boldsymbol{S}_{N-1}^{I}$, $\boldsymbol{S}_N^{II}$ and $\boldsymbol{S}_N^{IV}$ nutzen. Für die entsprechenden Basen von $\mathcal{A}_{\boldsymbol{O}_N}$ mit

$$\begin{aligned} \boldsymbol{B}_k^I &:= \boldsymbol{O}'_{\dim} \operatorname{diag}(T_k(c_l))_{l=0}^{\dim -1} \boldsymbol{O}_{\dim} \\ &= \frac{1}{2}(\varepsilon_k^N)^{-2} (\operatorname{stoep} \boldsymbol{e}'_k + \operatorname{hank}(\boldsymbol{u}'_k, \boldsymbol{v}'_k)) , \\ \boldsymbol{B}_k^{II} &:= \boldsymbol{O}'_{\dim} \operatorname{diag}(U_k(c_l))_{l=0}^{\dim -1} \boldsymbol{O}_{\dim} \end{aligned}$$

gilt, daß

$$\boldsymbol{G}^{-1} = \frac{1}{K} \begin{pmatrix} 3 & 0 & -1 & & & & \\ 0 & 2 & 0 & -1 & & & \\ -1 & 0 & 2 & 0 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & -1 & 0 & 2 & 0 & -1 \\ & & & -1 & 0 & g_1 & g_2 \\ & & & & -1 & g_2 & g_3 \end{pmatrix}$$

ist. Dabei sind die Größen $K, g_1, g_2, g_3, \dim, c_l, \boldsymbol{u}_k$ und $\boldsymbol{v}_k$ entsprechend der gewählten orthogonalen trigonometrischen Transformation $\boldsymbol{O}_{\dim}$ gemäß Tabelle 3.1 einzusetzen.

$\boldsymbol{O}_{\dim}$	c_l	dim	$\boldsymbol{u}_k$	$\boldsymbol{v}_k$	K	g_1	g_2	g_3
$\boldsymbol{C}_N^{II}$	$\cos \frac{l\pi}{N}$	N	$\boldsymbol{e}_{k-1}$	$\boldsymbol{e}_{k-1}$	$2N$	3	-2	$\frac{3N-2}{N}$
$\boldsymbol{C}_N^{IV}$	$\cos \frac{(2l+1)\pi}{2N}$	N	$\boldsymbol{e}_{k-1}$	$-\boldsymbol{e}_{k-1}$	$2N$	1	0	1
$\boldsymbol{S}_{N-1}^{I}$	$\cos \frac{(l+1)\pi}{N}$	$N-1$	$-\boldsymbol{e}_{k-2}$	$-\boldsymbol{e}_{k-2}$	$2N+2$	2	0	3
$\boldsymbol{S}_N^{II}$	$\cos \frac{(l+1)\pi}{N}$	N	$-\boldsymbol{e}_{k-1}$	$-\boldsymbol{e}_{k-1}$	$2N$	3	2	$\frac{3N-2}{N}$
$\boldsymbol{S}_N^{IV}$	$\cos \frac{(2l+1)\pi}{2N}$	N	$-\boldsymbol{e}_{k-1}$	$\boldsymbol{e}_{k-1}$	$2N$	1	0	1

Tabelle 3.1: Größen für die optimalen trigonometrischen Vorkonditionierer

Benutzen wir die Basis $\boldsymbol{B}_j^I$ als Ansatzmatrizen für das Galerkin–Verfahren, so ist die schnelle Konstruktion des optimalen Vorkonditionierers bezüglich $\boldsymbol{C}_N^{IV}$ und $\boldsymbol{S}_N^{IV}$ ebenfalls möglich. Man erkennt leicht, daß $\tilde{\boldsymbol{G}} := \left(\langle \boldsymbol{B}_j^I, \boldsymbol{B}_k^I\rangle\right)_{j,k=0}^{N-1} = 2N\,\mathrm{diag}(2,1,\dots,1)$ gilt. $\square$

3.3 Vorkonditionierer für symmetrische Toeplitz–Matrizen

Sei $C_{2\pi}$ die Menge aller reellwertigen, stetigen 2π–periodischen Funktionen. Für $\varphi \in C_{2\pi}$ sei

$$a_k \;=\; a_k(\varphi) := \frac{1}{2\pi}\int_{-\pi}^{\pi} \varphi(t)\,\mathrm{e}^{-\mathrm{i}kt}\,\mathrm{d}t \quad (k \in \mathbb{Z}) \tag{3.19}$$

der k–te Fourier–Koeffizient von φ. Sei $\boldsymbol{T}_N(\varphi) \in \mathbb{C}^{N,N}$ $(N \geq 1)$ die Toeplitz-Matrix mit den Einträgen a_{j-k} $(0 \leq j,k \leq N)$. Die Funktion φ wird *erzeugende Funktion* für die Folge von Toeplitz–Matrizen $\boldsymbol{T}_N(\varphi)$ genannt. Da wir hier nur reellwertige Funktionen $\varphi \in C_{2\pi}$ betrachten, ist $a_{-k} = \bar{a}_k$ $(k \in \mathbb{Z})$. Somit sind die Matrizen $\boldsymbol{T}_N(\varphi)$ hermitesche Matrizen.
Das folgende Lemma gibt eine Beziehung zwischen $\varphi \in C_{2\pi}$ und den Eigenwerten von $\boldsymbol{T}_N(\varphi)$ an. Wir bezeichnen mit

$$\varphi_{\min} := \min\left\{\varphi(t) \;:\; t \in [-\pi,\pi]\right\}$$

das Minimum und mit

$$\varphi_{\max} := \max\left\{\varphi(t) \;:\; t \in [-\pi,\pi]\right\}$$

das Maximum der Funktion $\varphi \in C_{2\pi}$.

Lemma 3.12 (siehe [49], S. 64) *Sei $\varphi \in C_{2\pi}$. Für alle Eigenwerte $\lambda_j(\boldsymbol{T}_N(\varphi))$ $(j = 0,\dots,N-1)$ von $\boldsymbol{T}_N(\varphi)$ gilt*

$$\lambda_j(\boldsymbol{T}_N(\varphi)) \in [\varphi_{\min}, \varphi_{\max}]\;.$$

Insbesondere ist

$$\|\boldsymbol{T}_N(\varphi)\|_2 \leq \varphi_{\max}\;.$$

Falls $\varphi_{\min} > 0$ ist, so ist $\boldsymbol{T}_N(\varphi)$ positiv definit für alle $N \in \mathbb{N}$.

Dieses Lemma wurde in [49], S. 63 – 65 bewiesen. Eine wichtige Konsequenz von [18] ist, daß alle Matrizen $\boldsymbol{T}_N(\varphi)$ positiv definit sind, falls $\varphi \in C_{2\pi}$, wobei für alle $t \in [-\pi,\pi]$ gilt $\varphi(t) \geq 0$ und für ein $t_0 \in [-\pi,\pi]$ gilt $\varphi(t_0) > 0$. Wir wollen hier einen allgemeineren Satz beweisen.

Satz 3.13 *Sei $\varrho \in C_{2\pi}$ eine positive Funktion mit*

$$\varrho_{\min} := \min\{\varrho(t) \,:\, t \in [-\pi,\pi]\},$$
$$\varrho_{\max} := \max\{\varrho(t) \,:\, t \in [-\pi,\pi]\}.$$

und $0 < \varrho_{\min} < \varrho_{\max} < \infty$. Ferner seien $\varphi, \psi \in C_{2\pi}$ nichtnegative Funktionen mit $\varphi(t) = \psi(t)\varrho(t)$, wobei $\psi(t_0) > 0$ für ein $t_0 \in [-\pi,\pi]$ gilt. Dann liegen alle Eigenwerte der Matrix $\boldsymbol{T}_N^{-1}(\psi)\,\boldsymbol{T}_N(\varphi)$ im offenen Intervall $(\varrho_{\min}, \varrho_{\max})$.

Beweis: Da $\boldsymbol{T}_N(\psi)$ eine hermitesche, positiv definite Matrix [18] ist, gilt für den Rayleigh–Quotienten (siehe [42], S. 264) mit $\boldsymbol{x}_N \in \mathbb{C}_N$ $(x \neq \boldsymbol{o}_N)$

$$\begin{aligned}
\frac{\bar{\boldsymbol{x}}'_N\,\boldsymbol{T}_N^{-1/2}(\psi)\,\boldsymbol{T}_N(\varphi)\;\boldsymbol{T}_N^{-1/2}(\psi)\,\boldsymbol{x}_N}{\bar{\boldsymbol{x}}'_N\boldsymbol{x}_N} &= \frac{\bar{\boldsymbol{y}}'_N\,\boldsymbol{T}_N(\varphi)\,\boldsymbol{y}_N}{\bar{\boldsymbol{y}}'_N\,\boldsymbol{T}_N(\psi)\,\boldsymbol{y}_N} \\
&= \frac{\displaystyle\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} a_{k-l}(\varphi) y_l \bar{y}_k}{\displaystyle\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} a_{k-l}(\psi) y_l \bar{y}_k}\,.
\end{aligned}$$

Nun gilt mit Formel (3.19) (siehe auch [49], S. 64)

$$\begin{aligned}
\bar{\boldsymbol{y}}'_N\,\boldsymbol{T}_N(\varphi)\,\boldsymbol{y}_N &= \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} a_{k-l}(\varphi) y_l \bar{y}_k \\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi} |\sum_{k=0}^{N-1} y_k \mathrm{e}^{-\mathrm{i}kt}|^2 \varphi(t)\,\mathrm{d}t\,.
\end{aligned} \tag{3.20}$$

Mit dem erweiterten Mittelwertsatz der Integralrechnung erhalten wir

$$\varrho_{\min}\,\bar{\boldsymbol{y}}'_N\,\boldsymbol{T}_N(\psi)\,\boldsymbol{y}_N < \bar{\boldsymbol{y}}'_N\,\boldsymbol{T}_N(\varphi)\,\boldsymbol{y}_N < \varrho_{\max}\,\bar{\boldsymbol{y}}'_N\,\boldsymbol{T}_N(\psi)\,\boldsymbol{y}_N\,.$$

Da $\boldsymbol{T}_N(\psi)$ positiv definit ist, folgt wegen

$$\varrho_{\min} < \frac{\bar{\boldsymbol{x}}'_N\,\boldsymbol{T}_N^{-1/2}(\psi)\,\boldsymbol{T}_N(\varphi)\,\boldsymbol{T}_N^{-1/2}(\psi)\,\boldsymbol{x}_N}{\bar{\boldsymbol{x}}'_N\boldsymbol{x}_N} < \varrho_{\max}$$

die Behauptung. ■

Das Lemma 3.12 kann in ähnlicher Weise mit Hilfe des ersten Mittelwertsatzes der Integralrechnung gezeigt werden. Einen anderen Beweis von Satz 3.13 findet man in [10].

Einer beliebigen reellwertigen, stetigen Funktion $f := I \to \mathbb{R}$ $(I := [-1,1])$ ordnen wir die Funktion $\varphi := f(\cos) : [0,\pi] \to \mathbb{R}$ zu. Wir setzen φ durch $\varphi(t) := f(\cos t)$ $(t \in \mathbb{R})$

auf $\mathbb{R}$ fort. Somit ist φ eine gerade, 2π–periodische Funktion. Mit $C(I)$ bezeichnen wir den reellen Banach–Raum aller stetigen Funktionen $f:\ I \to \mathbb{R}$ mit der Norm

$$\|f\|_\infty := \max\,\{|f(x)|:\ x \in I\}\ .$$

In diesem Abschnitt wollen wir reellwertige, gerade Funktionen φ betrachten, so daß die Matrix $\boldsymbol{T}_N(\varphi)$ reell und symmetrisch ist. Deshalb übertragen wir die Theorie für die stetigen 2π–periodischen Funktionen auf stetige Funktionen auf dem Intervall I. Für $f \in C(I)$ sei

$$a_k[f] := \frac{2}{\pi}\int_{-1}^{1} w(x)\,f(x)\,T_k(x)\,\mathrm{d}x \quad (k \in \mathbb{N}_0)$$

mit dem *Chebyshev–Gewicht* $w(x) := (1-x^2)^{-1/2}$ $(x \in (-1,1))$ der *k–te Chebyshev–Koeffizient* von f. Die Funktion f wird *erzeugende Funktion* für die Folge von symmetrischen Toeplitz–Matrizen

$$\boldsymbol{T}_N(f) := \frac{1}{2}\ \mathrm{stoep}(a_0[f], a_1[f], \ldots, a_{N-1}[f])$$

genannt. Falls $\varphi := f(\cos)$ ist, besteht zwischen den Fourier–Koeffizienten von $a_k(\varphi)$ und den Chebyshev–Koeffizienten $a_k[f]$ der Zusammenhang

$$2\,a_k(\varphi)\ =\ a_k[f] \quad (k \in \mathbb{N}_0)\ .$$

Somit erzeugen die Funktionen $\varphi = f(\cos)$ und f die gleichen Toeplitz–Matrizen.

Die *Chebyshev–Reihe* von f lautet

$$\sum_{k=0}^{\infty} (\varepsilon_k)^2\ a_k[f]\ T_k\ .$$

Die n–te Teilsumme $\mathcal{S}_n f$ dieser Reihe heißt n–te *Chebyshev–Summe*. Es sei $L^2_w(I)$ der reelle Hilbert–Raum aller meßbaren Funktionen $f: I \to \mathbb{R}$ mit

$$\int_{-1}^{1} w(x)\ f(x)^2\,\mathrm{d}x\ < \infty\ .$$

Das Skalarprodukt in $L^2_w(I)$ lautet

$$\langle f, g\rangle\ := \frac{1}{\pi}\int_{-1}^{1} w(x) f(x) g(x)\,\mathrm{d}x \quad (f, g \in L^2_w(I))\ .$$

Die zugehörige Norm von $f \in L^2_w(I)$ ist dann $\|f\|_{2,w} := \langle f, f\rangle^{1/2}$. Wie üblich werden in $L^2_w(I)$ fast überall gleiche Funktionen identifiziert. In [93, 94] werden Toeplitz–Matrizen betrachtet, die von quadratisch integrierbaren Funktionen erzeugt werden. Dies führt dann aber zu keiner superlinearen Konvergenz des PCG–Verfahrens. Deshalb führen wir für jede positive Zahl $s \in \mathbb{R}$ Teilräume $L^2_{w,s}(I)$ von $L^2_w(I)$ durch

$$L^2_{w,s}(I)\ :=\ \{f \in L^2_w(I):\ \|f\|_{2,w,s} < \infty\}$$

mit der Norm vom Sobolev–Typ

$$\|f\|_{2,w,s} := \left(\sum_{k=0}^{\infty} (1+k)^{2s} \, a_k[f]^2 \right)^{1/2}$$

ein. Wir fassen einige Eigenschaften der Räume $L^2_{w,s}(I)$ in dem folgenden Satz zusammen.

Satz 3.14 (siehe [11]) *Es gilt*
(i) $L^2_w(I) = L^2_{w,0}(I)$,
(ii) $L^2_{w,s}(I) \subseteq L^2_{w,t}(I)$ *mit* $\|f\|_{2,w,t} \leq \|f\|_{2,w,s}$ *für* $f \in L^2_{w,s}(I)$ *im Fall* $0 \leq t \leq s$,
(iii) $L^2_{w,s}(I) \subseteq C(I)$ *für* $s > 1/2$.

Diese Aussagen werden wir im folgenden Abschnitt benutzen.

Das Lemma 3.12 und der Satz 3.13 lassen sich in einfacher Weise übertragen. Im folgenden wollen wir wieder eine Beziehung zwischen $f \in C(I)$ und den Eigenwerten von $\boldsymbol{T}_N(f)$ angeben. Wir bezeichnen mit

$$f_{\min} := \min \{f(x) \,:\, x \in I\}$$

den kleinsten und mit

$$f_{\max} := \max \{f(x) \,:\, x \in I\}$$

den größten Funktionswert der Funktion $f \in C(I)$.

Lemma 3.15 *Sei* $f \in C(I)$. *Für alle Eigenwerte* $\lambda_j(\boldsymbol{T}_N(f))$ $(j = 0, \ldots, N-1)$ *von* $\boldsymbol{T}_N(f)$ *gilt*

$$\lambda_j(\boldsymbol{T}_N(f)) \in [f_{\min}, f_{\max}] \,.$$

Insbesondere ist

$$\|\boldsymbol{T}_N(f)\|_2 \leq f_{\max}.$$

Im Fall $f_{\max} > f_{\min} \geq 0$ *ist* $\boldsymbol{T}_N(f)$ *positiv definit für alle* $N \in \mathbb{N}$.

Beweis: Für $\varphi := f(\cos) \in C_{2\pi}$ sind diese Aussagen in Lemma 3.12 angegeben. Da $2a_k(\varphi) = a_k[f]$ ist, werden die gleichen Toeplitz–Matrizen erzeugt. Für $f_{\min} = 0$ folgt die Aussage von [18]. ■

3.3.1 Strang–Typ–Vorkonditionierer

Strang [87] führte zuerst zirkulante Vorkonditionierer für Toeplitz–Matrizen $\boldsymbol{T}_N = (t_{j-k})_{j,k=0}^{N-1} \in \mathbb{C}^{N,N}$ ein. Für gerades N ist der *zirkulante Strang–Vorkonditionierer* durch

$$\mathrm{circ}(t_0, t_{-1}, \ldots, t_{-N/2}, t_{N/2-1}, \ldots, t_1)$$

gegeben. Dieser einfache Vorkonditionierer kann von einer Fourier–Reihe abgeleitet werden [34]. Im trigonometrischen Fall wollen wir ähnlich vorgehen. Wegweiser sind die Diagonalisierungsaussagen (siehe Satz 1.4). Unser Ziel ist die Übereinstimmung der symmetrischen Toeplitz–Matrix $\boldsymbol{T}_N$ mit dem Toeplitz–Anteil des Vorkonditionierers. Dies werden wir im Lemma 3.16 zeigen. Wir werden die Strang–Typ–Vorkonditionierer mit Hilfe der Chebyshev–Reihe definieren. Diese können wir in einfacher Weise ändern und Vorkonditionierer in analoger Weise wie in [34] ableiten. Auch diese Vorkonditionierer sind Matrizen aus der Algebra (3.5).

Die *Strang–Typ–Vorkonditionierer* von $\boldsymbol{T}_N(f)$ bezüglich einer diskreten trigonometrischen Transformation sind wie folgt erklärt:

$$\begin{array}{lrcl}
\text{DCT–I:} & \boldsymbol{M}^{\mathcal{S}}_{N-1}(\boldsymbol{C}^{I}_{N+1}) & := & \boldsymbol{R}'_N \boldsymbol{C}^{I}_{N+1}\ \operatorname{diag}\left(\mathcal{S}_N\, f(c^N_j)\right)_{j=0}^{N}\ \boldsymbol{C}^{I}_{N+1}\boldsymbol{R}_N, \\
\text{DST–I:} & \boldsymbol{M}^{\mathcal{S}}_{N-1}(\boldsymbol{S}^{I}_{N-1}) & := & \boldsymbol{S}^{I}_{N-1}\ \boldsymbol{R}'_N\ \operatorname{diag}\left(\mathcal{S}_N\, f(c^N_j)\right)_{j=1}^{N-1}\ \boldsymbol{R}_N\boldsymbol{S}^{I}_{N-1}, \\
\text{DCT–II:} & \boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{C}^{II}_{N}) & := & (\boldsymbol{C}^{II}_{N})'\operatorname{diag}\left(\mathcal{S}_N\, f(c^N_j)\right)_{j=0}^{N-1}\ \boldsymbol{C}^{II}_{N}, \\
\text{DST–II:} & \boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{S}^{II}_{N}) & := & (\boldsymbol{S}^{II}_{N})'\operatorname{diag}\left(\mathcal{S}_N\, f(c^N_j)\right)_{j=1}^{N}\ \boldsymbol{S}^{II}_{N}, \\
\text{DCT–IV:} & \boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{C}^{IV}_{N}) & := & \boldsymbol{C}^{IV}_{N}\ \operatorname{diag}\left(\mathcal{S}_N\, f(c^{2N}_{2j+1})\right)_{j=0}^{N-1}\ \boldsymbol{C}^{IV}_{N}, \\
\text{DST–IV:} & \boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{S}^{IV}_{N}) & := & \boldsymbol{S}^{IV}_{N}\ \operatorname{diag}\left(\mathcal{S}_N\, f(c^{2N}_{2j+1})\right)_{j=0}^{N-1}\ \boldsymbol{S}^{IV}_{N},
\end{array}$$

Ein sogenannter Displacement–Zugang wurde kürzlich in [62] gegeben. Für die DST–I sind die Vorkonditionierer als τ–Vorkonditionierer bekannt (siehe z.B. [7, 12, 14]). Im folgenden untersuchen wir nur den Vorkonditionierer bezüglich der DCT–II. Für die anderen Vorkonditionierer ergeben sich analoge Aussagen.

Lemma 3.16 *Sei $f \in L^2_w(I)$ die erzeugende Funktion der symmetrischen Toeplitz–Matrix $\boldsymbol{T}_N := \frac{1}{2}\operatorname{stoep}(a_0[f], a_1[f], \ldots, a_{N-1}[f]) \in \mathbb{R}^{N,N}$. Für den Strang–Typ–Vorkonditionierer von $\boldsymbol{T}_N$ gilt*

$$\boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{C}^{II}_{N}) \;=\; \boldsymbol{T}_N \;+\; \boldsymbol{H}_N$$

mit $\boldsymbol{H}_N := \frac{1}{2}\operatorname{shank}(a_1[f], a_2[f], \ldots, a_{N-1}[f], 0)$. Die Eigenwerte des Vorkonditionierers $\boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{C}^{II}_{N})$ sind durch

$$\lambda_j\left(\boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{C}^{II}_{N})\right) \;= \mathcal{S}_N f\ (c^N_j) \quad (j = 0, \ldots, N-1)$$

gegeben. Insbesondere gilt für $f \in C(I)$ mit $f > 0$

$$\|\boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{C}^{II}_{N})\|_2 \le f_{\max} \quad \textit{und} \quad \|(\boldsymbol{M}^{\mathcal{S}}_{N}(\boldsymbol{C}^{II}_{N}))^{-1}\|_2 \le \frac{1}{f_{\min}}\,. \tag{3.21}$$

Beweis: Für die N–te Chebyshev–Summe gilt

$$\mathcal{S}_N f \;= \sum_{k=0}^{N} (\varepsilon_k)^2\ a_k[f]\ T_k$$

und somit

$$\mathcal{S}_N f(c_j^N) = \sum_{k=0}^{N} (\varepsilon_k)^2 \, a_k[f] \, T_k(c_j^N) \quad (j = 0, \dots, N) \, .$$

Da f die erzeugende Funktion für die Toeplitz–Matrix $\boldsymbol{T}_N$ ist, gilt mit $a_N[f] := 0$

$$(\mathcal{S}_N f(c_j^N))_{j=0}^{N} \;=\; \tilde{\boldsymbol{C}}_{N+1}^{I} \, (a_n[f])_{n=0}^{N} \, . \tag{3.22}$$

Nach Diagonalisierungssatz 1.4 (ii) folgt

$$(\boldsymbol{C}_N^{II})' \; \mathrm{diag}(\mathcal{S}_N f(c_j^N))_{j=0}^{N-1} \, \boldsymbol{C}_N^{II} \;=\; \boldsymbol{T}_N \; + \boldsymbol{H}_N \, .$$

Da die Matrix $\boldsymbol{C}_N^{II}$ orthogonal ist, können wir die Eigenwerte von $\boldsymbol{M}^{\mathcal{S}}(\boldsymbol{C}_N^{II})$ aus der Diagonalmatrix ablesen. Unter Beachtung der Symmetrie von $\boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{C}_N^{II})$ folgen die Ungleichungen (3.21). ■

Satz 3.17 *Für eine Folge von symmetrischen Toeplitz–Matrizen* $\boldsymbol{T}_N = \frac{1}{2}\,\mathrm{stoep}\,\boldsymbol{a}_N'$ $(N \in \mathbb{N})$ *und* $\boldsymbol{a}_N = (a_k)_{k=0}^{N-1} \in \mathbb{R}^N$ *gelte*

$$\sum_{k=1}^{\infty} k \, a_k^2 \; < \; \infty \, .$$

Dann existieren für alle $\varepsilon > 0$ *Indizes* $M \in \mathbb{N}$ *und* $n \in \mathbb{N}$ $(n > M)$, *so daß für alle* $N > n$

$$\boldsymbol{T}_N - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{C}_N^{II}) = \boldsymbol{W}_N + \boldsymbol{V}_N$$

gilt, wobei $\boldsymbol{W}_N$ *eine Matrix mit Rang* $\leq 2M$ *ist und* $\|\boldsymbol{V}_N\|_2 \leq \varepsilon$ *gilt.*

Beweis: Da

$$\begin{aligned} \|\boldsymbol{T}_N - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{C}_N^{II})\|_2 \;&\leq\; \|\boldsymbol{T}_N - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{C}_N^{II})\|_F \\ &=\; \|\,\mathrm{shank}(a_1, a_2, \dots, a_{N-1}, 0)\|_F \\ &=\; (2 \sum_{k=0}^{N-1} k \, a_k^2)^{1/2} \end{aligned}$$

gilt, existiert für ein hinreichend großes N ein Index $M \in \mathbb{N}$, so daß

$$\boldsymbol{W}_N = \mathrm{shank}(a_1, a_2, \dots, a_M, 0, \dots, 0)'$$

eine Matrix mit Rang $\leq 2M$ und

$$\boldsymbol{V}_N = \mathrm{shank}(0, \dots, 0, a_{M+1}, \dots, a_{N-1}, 0)'$$

eine Matrix mit kleiner Spektralnorm $\|\boldsymbol{V}_N\|_2 \leq \varepsilon$ ist. ■

Folgerung 3.18 *Wird die symmetrische Toeplitz–Matrix $\boldsymbol{T}_N(f) \in \mathbb{R}^{N,N}$ von einer positiven Funktion $f \in L^2_{w,1/2}(I)$ erzeugt, so erfüllt der Strang–Typ–Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{C}_N^{II})$ für hinreichend große N die Eigenschaften* (E1) – (E4). *Ferner gilt dies auch für $f \in C(I)$ mit $f > 0$.*

Beweis: Der Strang–Typ–Vorkonditionierer ist nach Definition symmetrisch. Die Eigenwerte dieses Vorkonditionierers sind nach Lemma 3.16 durch die Funktionswerte der N–ten Chebyshev–Summe $\mathcal{S}_N f$ gegeben. Da die Chebyshev–Reihe sogar in $L^2_w(I) \supset L^2_{w,1/2}(I)$ konvergiert (siehe Satz 3.14), sind für positive f alle Eigenwerte positiv (siehe [76], Lemma 2.1). Aus den Einträgen der Toeplitz–Matrix $\boldsymbol{T}_N$ können wir mit einer $\tilde{\boldsymbol{C}}^I_{N+1}$ den Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{C}_N^{II})$ in $\mathcal{O}(N \log N)$ Operationen nach der Formel (3.22) berechnen. Damit sind die Eigenschaften (E1) – (E3) erfüllt. Für die Chebyshev–Koeffizienten $a_k := a_k[f]$ von $f \in L^2_{w,1/2}(I)$ gilt

$$\sum_{k=0}^{\infty} (1+k)\; a_k^2 \;< \infty \,.$$

Somit folgt die Eigenschaft (E4) aus den Sätzen 3.17 und 3.4.
Diese Aussage ist auch für $f \in C(I)$ gültig (siehe auch [35]). Dazu benutzen wir den Approximationssatz von Weierstraß. Zu $f \in C(I)$ existiert ein Polynom $g_M \in \Pi_M$, so daß $\|f - g_M\|_\infty \le \varepsilon/2$ ist. Wir zeigen, daß für alle $N > n > 2M$ gilt

$$\boldsymbol{T}_N(f) - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{T}_N(f), \boldsymbol{C}_N^{II}) = \boldsymbol{W}_N + \boldsymbol{V}_N \,,$$

wobei $\boldsymbol{W}_N$ eine Matrix mit Rang $\le 2M$ ist und $\|\boldsymbol{V}_N\|_2 \le \varepsilon$ gilt. Dazu nutzen wir (siehe auch [35])

$$\begin{aligned} \boldsymbol{T}_N(f) - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{T}_N(f), \boldsymbol{C}_N^{II}) &= \boldsymbol{T}_N(f - g_M) - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{T}_N(f - g_M), \boldsymbol{C}_N^{II}) \\ &+ \boldsymbol{T}_N(g_M) - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{T}_N(g_M), \boldsymbol{C}_N^{II}) \,. \end{aligned}$$

Da g_M ein Polynom höchstens vom Grade M ist, hat die Matrix

$$\boldsymbol{W}_N = \boldsymbol{T}_N(g_M) - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{T}_N(g_M), \boldsymbol{C}_N^{II})$$

einen Rang $\le 2M$. Nun gilt nach Lemma 3.15, daß $\|\boldsymbol{T}_N(f - g_M)\|_2 \le \|f - g_M\|_\infty$, und nach Formel (3.21), daß $\|\boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{T}_N(f - g_M), \boldsymbol{C}_N^{II})\|_2 \le \|f - g_M\|_\infty$. Somit gilt für $\boldsymbol{V}_N = \boldsymbol{T}_N(f - g_M) - \boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{T}_N(f - g_M), \boldsymbol{C}_N^{II})$, daß $\|\boldsymbol{V}_N\|_2 \le \varepsilon$ ist. ■

Analoge Aussagen gelten für die anderen trigonometrischen Vorkonditionierer.

Bemerkung 3.19 Die Definition der Strang–Typ–Vorkonditionierer kann man analog zu [34] ändern. Wir erhalten dann Vorkonditionierer, die auf verschiedenen Kernen der Approximationstheorie basieren. In Abschnitt 3.2 behandelten wir die optimalen trigonometrischen Vorkonditionierer. Für symmetrische Toeplitz–Matrizen werden wir die optimalen trigonometrischen Vorkonditionierer in Abschnitt 3.3.2 betrachten. Für die optimalen trigonometrischen Vorkonditionierer bezüglich $\boldsymbol{C}_N^{IV}$ und $\boldsymbol{S}_N^{IV}$ erhalten wir – wie im zirkulanten Fall – die Aussage, daß sich diese Vorkonditionierer mit Hilfe der Fejér–Kerne (siehe [79]) beschreiben lassen. Es gilt

$$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{IV}) \;=\; \boldsymbol{C}_N^{IV} \operatorname{diag}\left(\frac{1}{N}\sum_{k=0}^{N-1} \mathcal{S}_k\, f(c_{2j+1}^{2N})\right)_{j=0}^{N-1} \boldsymbol{C}_N^{IV}\,,$$

$$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{IV}) \;=\; \boldsymbol{S}_N^{IV} \operatorname{diag}\left(\frac{1}{N}\sum_{k=0}^{N-1} \mathcal{S}_k\, f(c_{2j+1}^{2N})\right)_{j=0}^{N-1} \boldsymbol{S}_N^{IV}\,.$$

Um diese Formeln zu beweisen, ist es lediglich nötig, die Koeffizienten von (3.10) mit den Basis–Matrizen $\boldsymbol{B}_k^I(\boldsymbol{C}_N^{IV})$ zu berechnen. In diesem Fall ist die Galerkin–Matrix $\tilde{\boldsymbol{G}}$ eine Diagonalmatrix (siehe Bemerkung 3.11). □

Im folgenden wollen wir den wichtigen Fall betrachten, daß $f \geq 0$ Nullstellen hat. In diesem Fall gilt $\operatorname{cond}(\boldsymbol{T}_N(f)) \to \infty$ für $N \to \infty$. Da die bekannten zirkulanten Vorkonditionierer in diesen Fällen keine guten Ergebisse liefern (siehe z.B. [83]), wurden Band–Toeplitz–Vorkonditionierer (siehe [18, 33, 7, 83]) bzw. Toeplitz–Vorkonditionierer (siehe [26]) vorgeschlagen. Wir wollen uns hier auf Vorkonditionierer aus der Algebra (3.5) beschränken, weil in diesem Fall das PCG–Verfahren in jedem Schritt nur N Multiplikationen mehr als das CG–Verfahren benötigt (siehe Abschnitt 3.3.3). Für erzeugende Funktionen mit Nullstellen ist der Strang–Typ–Vorkonditionierer oft nicht positiv definit, d.h., in der Diagonalmatrix des Vorkonditionierers haben wir negative Einträge. In vielen Anwendungen ist die erzeugende Funktion explizit bekannt. Dies ist eine Voraussetzung für die Konstruktion der Vorkonditionierer in [18, 33, 7, 83]. Wir definieren deshalb *vereinfachte Strang–Typ–Vorkonditionierer* durch

$$\begin{aligned}\boldsymbol{M}_N(f, \boldsymbol{S}_N^{II}) \;&:=\; (\boldsymbol{S}_N^{II})' \operatorname{diag}\big(f(c_j^N)\big)_{j=1}^{N} \boldsymbol{S}_N^{II}\,,\\ \boldsymbol{M}_N(f, \boldsymbol{C}_N^{II}) \;&:=\; (\boldsymbol{C}_N^{II})' \operatorname{diag}\big(f(c_j^N)\big)_{j=0}^{N-1} \boldsymbol{C}_N^{II}\,.\end{aligned} \tag{3.23}$$

Dabei ist wieder $f(x) = \varphi(\arccos x)$ $(x \in [-1, 1])$, d.h., in der Diagonalmatrix stehen nicht Funktionswerte der N–ten Chebyshev–Summe, sondern die Funktionswerte der erzeugenden Funktion. Im Fall $f \geq 0$ ist der Vorkonditionierer symmetrisch und positiv semidefinit. Im Fall $f(c_j^N) > 0$ für $j = 1, \ldots, N$ ist $\boldsymbol{M}_N(f, \boldsymbol{S}_N^{II})$ positiv definit. Im Fall $f(c_j^N) > 0$ für $j = 0, \ldots, N-1$ ist $\boldsymbol{M}_N(f, \boldsymbol{C}_N^{II})$ positiv definit. In [76] haben wir für hermitische Matrizen Vorkonditionierer für schlecht kondtionierte Matrizen aus der Formel (3.20) unter Nutzung der Trapez–Regel hergeleitet. Für die symmetrischen Matrizen erhalten wir Formel (3.23). Solche Vorkonditionierer wurden kürzlich auch unabhängig in [61] für die DST–I vorgeschlagen. Die vereinfachten Strang–Typ–Vorkonditionierer bezüglich der anderen Transformationen können wir in analoger Weise definieren. Wir sind erstmals in der Lage, für diese Vorkonditionierer die Eigenschaften (E1) – (E4) zu beweisen. Falls f ein trigonometrisches Polynom vom Grade kleiner N ist, gilt $\boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{O}_N) = \boldsymbol{M}_N(f, \boldsymbol{O}_N)$, wobei $\boldsymbol{O}_N$ eine trigonometrische Matrix ist.

Lemma 3.20 *Falls $f \in C(I)$ mit $f > 0$ ist, so erfüllt der vereinfachte Strang–Typ–Vorkonditionierer die Eigenschaften* (E1) – (E4).

Beweis: Mit dem Approximationssatz von Weierstraß folgt in analoger Weise wie im Beweis von Folgerung 3.18 die Behauptung. ∎

Im folgenden betrachten wir nur Funktionen $f \in C(I)$, die in $x = 1$ eine Nullstelle ganzzahliger Vielfachheit haben und sonst $f(x) > 0$ $(x \in [-1,1))$ gilt. Wir setzen $\boldsymbol{M}_N(f) = \boldsymbol{M}_N(f, \boldsymbol{S}_N^{II})$. Falls die erzeugende Funktion $\varphi \in C_{2\pi}$ endlich viele Nullstellen gerader Ordnung hat, siehe [76].

Satz 3.21 *Sei $s \in \mathbb{N}$. Ferner sei $f \in C(I)$ mit $f(x) > 0$ für $x \in [-1,1)$ und $f(1) = 0$ gegeben, wobei $x = 1$ eine s–fache Nullstelle von f ist. Ist f die erzeugende Funktion der symmetrischen Toeplitz–Matrix $\boldsymbol{T}_N(f)$, dann ist die Anzahl der Iterationschritte, die das* PCG*–Verfahren mit dem Vorkonditionierer $\boldsymbol{M}_N(f)$ bei dem Toeplitz–System*

$$\boldsymbol{T}_N(f)\,\boldsymbol{x} \;=\; \boldsymbol{b}$$

für eine vorgegebene Genauigkeit benötigt, unabhängig von N.

Beweis: Wir bezeichnen mit $\boldsymbol{R}_N(m)$ eine beliebige (N,N)–Matrix vom Rang m. Nach Voraussetzung können wir die Funktion f in der Form

$$f(x) = p_s(x)\, r(x) \quad (x \in I)$$

mit dem Polynom $p_s(x) = (1-x)^s$ darstellen. Für $r = \text{const}$ ist die Aussage trivial. Deshalb seien $r_{\min} := \min\{r(x) : \; x \in I\}$ und $r_{\max} := \max\{r(x) : \; x \in I\}$ Konstanten mit der Ungleichung $0 < r_{\min} < r_{\max} < \infty$. Wir werden nun zeigen, daß $N - 2s$ Eigenwerte von $(\boldsymbol{M}_N(f))^{-1}\boldsymbol{T}_N(f)$ im Intervall $(\frac{r_{\min}}{r_{\max}}, \frac{r_{\max}}{r_{\min}})$ liegen. Dazu betrachten wir den Rayleigh–Quotienten (siehe auch [9]) für $\boldsymbol{x}_N \neq \boldsymbol{o}_N$

$$\frac{\boldsymbol{x}_N'\,\boldsymbol{T}_N(f)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N} = \frac{\boldsymbol{x}_N'\,\boldsymbol{T}_N(f)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{T}_N(p_s)\,\boldsymbol{x}_N}\;\frac{\boldsymbol{x}_N'\,\boldsymbol{T}_N(p_s)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N}\,. \tag{3.24}$$

Nach Voraussetzung ist $\boldsymbol{T}_N(p_s)$ eine Bandmatrix der Bandbreite $2s+1$. Somit läßt sich die Toeplitz–Matrix $\boldsymbol{T}_N(p_s)$ in der Form

$$\boldsymbol{T}_N(p_s) \;=\; \boldsymbol{M}_N(p_s) + \boldsymbol{R}_N(2s)$$

schreiben. Die symmetrische Matrix $\boldsymbol{R}_N(2s)$ kann in der Form

$$\boldsymbol{R}_N(2s) \;=\; \boldsymbol{Q}_N'\,\boldsymbol{D}_N^+\,\boldsymbol{Q}_N \;+\; \boldsymbol{Q}_N'\,\boldsymbol{D}_N^-\,\boldsymbol{Q}_N \;=\; \boldsymbol{R}_N(s_1) \;+\; \boldsymbol{R}_N(2s - s_1) \tag{3.25}$$

dargestellt werden. Dabei ist $\boldsymbol{Q}_N$ eine orthogonale Matrix und $\boldsymbol{D}_N^+$ und $\boldsymbol{D}_N^-$ sind Diagonalmatrizen, die die s_1 positiven bzw. $2s - s_1$ negativen Eigenwerte der Matrix $\boldsymbol{R}_N(2s)$ als Einträge enthalten. Setzen wir nun

$$a(\boldsymbol{x}_N) \;:=\; \frac{\boldsymbol{x}_N'\,\boldsymbol{T}_N(f)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{T}_N(p_s)\,\boldsymbol{x}_N}\,, \quad m(\boldsymbol{x}_N) \;:=\; \frac{\boldsymbol{x}_N'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N}\,,$$

so erhalten wir mit (3.24) und (3.25)

$$\frac{\boldsymbol{x}_N' \boldsymbol{T}_N(f)\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N} = a(\boldsymbol{x}_N)\, m(\boldsymbol{x}_N) + a(\boldsymbol{x}_N)\, \frac{\boldsymbol{x}_N' \boldsymbol{R}_N(s_1)\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N} + a(\boldsymbol{x}_N)\, \frac{\boldsymbol{x}_N' \boldsymbol{R}_N(2s-s_1)\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N} .$$

Mit Satz 3.13 und nach Konstruktion von $\boldsymbol{M}_N$ folgt für alle $\boldsymbol{x}_N \in \mathbb{R}^N$ $(\boldsymbol{x}_N \neq \boldsymbol{o}_N)$, daß

$$a(\boldsymbol{x}_N) \in [r_{\min}, r_{\max}] , \quad m(\boldsymbol{x}_N) \in [\frac{1}{r_{\max}}, \frac{1}{r_{\min}}]$$

ist. Da für alle $\boldsymbol{x}_N \in \mathbb{R}^N$ $(\boldsymbol{x}_N \neq \boldsymbol{o}_N)$

$$\frac{\boldsymbol{x}_N' \boldsymbol{R}_N(s_1)\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N} \geq 0 , \quad \frac{\boldsymbol{x}_N' \boldsymbol{R}_N(2s-s_1)\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N} \leq 0$$

gilt, erhalten wir

$$\frac{\boldsymbol{x}_N' \boldsymbol{T}_N(f)\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N} \leq \frac{r_{\max}}{r_{\min}} + \frac{\boldsymbol{x}_N' [r_{\max} \boldsymbol{R}_N(s_1) + r_{\min} \boldsymbol{R}_N(2s-s_1)]\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N}$$

und damit

$$\frac{\boldsymbol{x}_N' [\boldsymbol{T}_N(f) - (r_{\max} \boldsymbol{R}_N(s_1) + r_{\min} \boldsymbol{R}_N(2s-s_1))]\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N} \leq \frac{r_{\max}}{r_{\min}} .$$

Die Matrix $r_{\max}\boldsymbol{R}_N(s_1) + r_{\min}\boldsymbol{R}_N(2s-s_1)$ hat wieder s_1 positive und $2s - s_1$ negative Eigenwerte. Somit folgt aus den Eigenschaften des Rayleigh–Quotienten und nach dem Verflechtungssatz von Weyl (siehe [56], S. 184), daß höchstens s_1 Eigenwerte von $\boldsymbol{M}_N(f)^{-1}\boldsymbol{T}_N(f)$ größer als $\frac{r_{\max}}{r_{\min}}$ sind. Ähnlich erhalten wir

$$\frac{\boldsymbol{x}_N' [\boldsymbol{T}_N(f) - (r_{\min} \boldsymbol{R}_N(s_1) + r_{\max} \boldsymbol{R}_N(2s-s_1))]\, \boldsymbol{x}_N}{\boldsymbol{x}_N' \boldsymbol{M}_N(f)\, \boldsymbol{x}_N} \geq \frac{r_{\min}}{r_{\max}} .$$

Somit sind höchstens $2s - s_1$ Eigenwerte von $\boldsymbol{M}_N(f)^{-1}\boldsymbol{T}_N(f)$ kleiner als $\frac{r_{\min}}{r_{\max}}$. Damit liegen nur $2s$ Eigenwerte von $(\boldsymbol{M}_N(f))^{-1}\boldsymbol{T}_N(f)$ außerhalb des Intervalls $(\frac{r_{\min}}{r_{\max}}, \frac{r_{\max}}{r_{\min}})$. Nach Satz 3.5 ist – unabhängig von N – nur eine beschränkte Anzahl von Iterationen für das PCG–Verfahren bei vorgegebener Genauigkeit notwendig. ■

Erstmals nutzte R. Chan (siehe [18]) Band–Toeplitz–Matrizen $\boldsymbol{B}_N$ als Vorkonditionierer für Toeplitz–Matrizen $\boldsymbol{T}_N(\varphi)$, falls $\varphi \in C_{2\pi}$, wobei $\varphi(t_0) > 0$ für ein $t_0 \in [-\pi, \pi]$ gilt und $\varphi(t) \geq 0$ für alle $t \in [-\pi, \pi]$ gilt. Dabei wurde gezeigt, daß alle Eigenwerte von $\boldsymbol{B}_N^{-1}\boldsymbol{T}_N$ in einem beschränkten Intervall – unabhängig von N – liegen. Diese Methode wurde dann in [10, 33, 83, 73] verallgemeinert und weiterentwickelt.
Im folgenden werden wir zeigen, daß die Eigenwerte von $\boldsymbol{M}_N(f)^{-1}\boldsymbol{T}_N(f)$ sogar um 1 geclustert sind. Diese führt zu besseren numerischen Ergebnissen (siehe Abschnitt 3.3.3). Außerdem benötigt der vereinfachte Strang–Typ–Vorkonditionierer keine zusätzlichen diskreten trigonometrischen Transformationen (siehe Bemerkung 3.29).

Satz 3.22 *Sei* $s \in \mathbb{N}$. *Ferner sei* $f \in C(I)$ *mit* $f(x) > 0$ *für* $x \in [-1, 1)$ *und* $f(1) = 0$ *gegeben, wobei* $x = 1$ *eine* s*–fache Nullstelle von* f *ist. Ist* f *die erzeugende Funktion der symmetrischen Toeplitz–Matrix* $\boldsymbol{T}_N(f)$*, dann besitzt der vereinfachte Strang–Typ–Vorkonditionierer* $\boldsymbol{M}_N(f)$ *die Eigenschaften* (E1) – (E4).

Beweis: Die Eigenschaften (E1) – (E3) sind erfüllt, da $f(c_j^N) > 0$ $(j = 1, \ldots, N)$ und $M_N(f) \in \mathcal{A}_{\boldsymbol{S}_N^{II}}$. Damit bleibt zu zeigen, daß die Eigenwerte von $\boldsymbol{M}_N(f)^{-1}\boldsymbol{T}_N$ um 1 geclustert sind. Es sei N hinreichend groß gewählt. Nach Voraussetzung ist $r = f/p_s$ eine stetige positive Funktion. Wir benutzen den Approximationssatz von Weierstraß, so daß für beliebiges $\varepsilon > 0$ ein Polynom $q_n \in \Pi_n$ mit

$$q_n(x) - \frac{1}{2}\,\varepsilon r_{\min} \;\le\; r(x) \;\le\; q_n(x) + \frac{1}{2}\,\varepsilon r_{\min} \qquad (x \in I) \tag{3.26}$$

existiert. Da $p_s(x) \ge 0$ ist, gilt

$$q_n(x)\,p_s(x) - \frac{1}{2}\,\varepsilon r_{\min}\,p_s(x) \;\le\; f(x) \;\le\; q_n(x)\,p_s(x) + \frac{1}{2}\,\varepsilon r_{\min}\,p_s(x) \qquad (x \in I)\,. \tag{3.27}$$

Aus der rechten Ungleichung erhalten wir mit Formel (3.20)

$$\boldsymbol{x}_N' \,\boldsymbol{T}_N(f)\,\boldsymbol{x}_N \;\le\; \boldsymbol{x}_N'\,\boldsymbol{T}_N(q_n\,p_s)\,\boldsymbol{x}_N \;+\; \frac{1}{2}\,\varepsilon r_{\min}\,\boldsymbol{x}_N'\,\boldsymbol{T}_N(p_s)\,\boldsymbol{x}_N\,.$$

Da die Matrix $\boldsymbol{M}_N(f)$ positiv definit ist, folgt für alle $\boldsymbol{x}_N \in \mathbb{R}^N$ $(\boldsymbol{x}_N \neq \boldsymbol{o}_N)$

$$\frac{\boldsymbol{x}_N'\,\boldsymbol{T}_N(f)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N} \;\le\; \frac{\boldsymbol{x}_N'\,\boldsymbol{T}_N(q_n\,p_s)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N} \;+\; \frac{1}{2}\,\varepsilon r_{\min}\,\frac{\boldsymbol{x}_N'\,\boldsymbol{T}_N(p_s)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N}\,. \tag{3.28}$$

Wiederum bezeichne $\boldsymbol{R}_N(m)$ eine (N, N)–Matrix vom Rang m. Da sich die Band–Toeplitz–Matrix $\boldsymbol{T}_N(p_s)$ in der Form

$$\boldsymbol{T}_N(p_s) \;=\; \boldsymbol{M}_N(p_s) \;+\; \boldsymbol{R}_N(2s) \tag{3.29}$$

darstellen läßt, gilt mit einem Ergebnis von Benedetto [8]

$$\boldsymbol{T}_N(q_n\,p_s) = \boldsymbol{T}_N(q_n)\,\boldsymbol{T}_N(p_s) \;+\; \boldsymbol{R}_N(2n + 2s)$$

die folgende Formel

$$\begin{aligned} \boldsymbol{T}_N(q_n p_s) \;&=\; (\boldsymbol{M}_N(q_n) \;+\; \boldsymbol{R}_N(2n))\,(\boldsymbol{M}_N(p_s) \;+\; \boldsymbol{R}_N(2s)) \;+\; \boldsymbol{R}_N(2n + 2s) \\ &=\; \boldsymbol{M}_N(q_n)\,\boldsymbol{M}_N(p_s) \;+\; \boldsymbol{R}_N(m)\,. \end{aligned} \tag{3.30}$$

Dabei ist $\boldsymbol{R}_N(m)$ eine symmetrische Matrix mit Rang $m \le 4n + 4s + \min\{2n, 2s\}$. Substituieren wir (3.29) und (3.30) in (3.28), so erhalten wir

$$\begin{aligned} \frac{\boldsymbol{x}_N'\,\boldsymbol{T}_N(f)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N} \;&\le\; \frac{\boldsymbol{x}_N'\,\boldsymbol{M}_N(q_n)\,\boldsymbol{M}_N(p_s)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N} \;+\; \frac{\boldsymbol{x}_N'\,\boldsymbol{R}_N(m)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N} \\ &\quad+\; \frac{1}{2}\,\varepsilon r_{\min}\,\frac{\boldsymbol{x}_N'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N} \;+\; \frac{1}{2}\,\varepsilon r_{\min}\,\frac{\boldsymbol{x}_N'\,\boldsymbol{R}_N(2s)\,\boldsymbol{x}_N}{\boldsymbol{x}_N'\,\boldsymbol{M}_N(f)\,\boldsymbol{x}_N}\,. \end{aligned}$$

Da
$$\frac{\boldsymbol{x}_N' \, \boldsymbol{M}_N(p_s) \, \boldsymbol{x}_N}{\boldsymbol{x}_N' \, \boldsymbol{M}_N(f) \, \boldsymbol{x}_N} \leq \frac{1}{r_{\min}}$$
folgt
$$\frac{\boldsymbol{x}_N' \, [\boldsymbol{T}_N(f) - \boldsymbol{R}_N(\tilde{m})] \, \boldsymbol{x}_N}{\boldsymbol{x}_N' \, \boldsymbol{M}_N(f) \, \boldsymbol{x}_N} \leq \frac{\boldsymbol{x}_N' \, \boldsymbol{M}_N(q_n) \, \boldsymbol{M}_N(p_s) \, \boldsymbol{x}_N}{\boldsymbol{x}_N' \, \boldsymbol{M}_N(f) \, \boldsymbol{x}_N} + \frac{1}{2}\varepsilon$$
mit $\tilde{m} \leq m + 2s$. Setzen wir nun $\boldsymbol{y}_N := \boldsymbol{M}_N(p_s)^{1/2} \boldsymbol{x}_N$ und nutzen die Identität $\boldsymbol{M}_N(f) = \boldsymbol{M}_N(r) \, \boldsymbol{M}_N(p_s)$, so folgt
$$\frac{\boldsymbol{x}_N' \, [\boldsymbol{T}_N(f) - \boldsymbol{R}_N(\tilde{m})] \, \boldsymbol{x}_N}{\boldsymbol{x}_N' \, \boldsymbol{M}_N(f) \, \boldsymbol{x}_N} \leq \frac{\boldsymbol{y}_N' \, \boldsymbol{M}_N(q_n) \, \boldsymbol{y}_N}{\boldsymbol{y}_N' \, \boldsymbol{M}_N(r) \, \boldsymbol{y}_N} + \frac{1}{2}\varepsilon \, . \tag{3.31}$$
Schließlich bekommen wir mit (3.26) und der Definition von $\boldsymbol{M}_N$ für alle $\boldsymbol{y}_N \in \mathbb{R}^N$ $(\boldsymbol{y}_N \neq \boldsymbol{o}_N)$ die Ungleichung
$$\boldsymbol{y}_N' \, \boldsymbol{M}_N(q_n) \, \boldsymbol{y}_N \leq \boldsymbol{y}_N' \, \boldsymbol{M}_N(r) \, \boldsymbol{y}_N + \frac{1}{2}\varepsilon \, r_{\min} \, \boldsymbol{y}_N' \, \boldsymbol{y}_N \, .$$
Da nun $0 < \dfrac{\boldsymbol{y}_N' \, \boldsymbol{y}_N}{\boldsymbol{y}_N' \, \boldsymbol{M}_N(r) \, \boldsymbol{y}_N} \leq \dfrac{1}{r_{\min}}$ ist, ergibt sich
$$\frac{\boldsymbol{y_N}' \, \boldsymbol{M}_N(q_n) \, \boldsymbol{y}_N}{\boldsymbol{y}_N' \, \boldsymbol{M}_N(r) \, \boldsymbol{y}_N} \leq 1 + \frac{1}{2}\varepsilon \, .$$
Nutzen wir diese Ungleichung in (3.31), so erhalten wir
$$\frac{\boldsymbol{x}_N' \, [\boldsymbol{T}_N(f) - \boldsymbol{R}_N(\tilde{m})] \, \boldsymbol{x}_N}{\boldsymbol{x}_N' \, \boldsymbol{M}_N(f) \, \boldsymbol{x}_N} \leq 1 + \varepsilon \, .$$
In analoger Weise folgt aus der linken Ungleichung von (3.27)
$$\frac{\boldsymbol{x}_N' \, [\boldsymbol{T}_N(f) - \boldsymbol{R}_N(\tilde{m})] \, \boldsymbol{x}_N}{\boldsymbol{x}_N' \, \boldsymbol{M}_N(f) \, \boldsymbol{x}_N} \geq 1 - \varepsilon \, .$$
Somit sind höchstens $\tilde{m}$ Eigenwerte von $\boldsymbol{M}_N(f)^{-1} \boldsymbol{T}_N(f)$ nicht in $[1-\varepsilon, 1+\varepsilon]$ enthalten und die Eigenschaft (E4) ist gezeigt. ■

In Abschnitt 3.3.3 werden wir in numerischen Tests diese Vorkonditionierer vergleichen. Außerdem werden wir die Effizienz der vereinfachten Strang–Typ–Vorkonditionierer auch im Block–Fall verdeutlichen (siehe [76]).

3.3.2 Optimale trigonometrische Vorkonditionierer

Die Konstruktion des optimalen trigonometrischen Vorkonditionierers für beliebige Matrizen $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ haben wir ausführlich in Abschnitt 3.2 beschrieben. Dieser Zugang soll nun auf symmetrische Toeplitz–Matrizen angewendet werden. Für symmetrische

Toeplitz–Matrizen wurden die optimalen Vorkonditionierer bezüglich der Sinus–Matrix $\boldsymbol{S}_{N-1}^{I}$ in [14, 31] untersucht. In [14] wurden die Matrizen angegeben, welche durch $\boldsymbol{C}_{N+1}^{I}, \boldsymbol{C}_{N}^{II}$ bzw. $\boldsymbol{S}_{N}^{II}$ diagonalisiert werden. Diese Idee wurde in [20] genutzt, um den optimalen Vorkonditionierer für die DCT–II zu berechnen. Unser neuer, allgemeiner Zugang für die optimale Vorkonditionierung erlaubt eine sehr einfache Berechnung bezüglich aller trigonometrischen Matrizen. Dies soll im folgenden erklärt werden.
Um den optimalen trigonometrischen Vorkonditionierer zu berechnen, müssen wir den Vektor $\boldsymbol{d} \in \mathbb{R}^{N}$ in (3.14) berechnen. Dies ist mit einer DST–I$(N-1)$ aus $\boldsymbol{\alpha} \in \mathbb{R}^{N}$ möglich (siehe (3.15) und (3.16)). Deshalb brauchen wir im folgenden Satz nur die Berechnung des Vektors $\boldsymbol{\alpha} = (\alpha_j)_{j=0}^{N-1} \in \mathbb{R}^{N}$ angeben.

Satz 3.23 *Sei* $\boldsymbol{T}_N = \operatorname{stoep}(t_0, t_1, \ldots, t_{N-1}) \in \mathbb{R}^{N,N}$. *Der optimale trigonometrische Vorkonditionierer* $\boldsymbol{M}_N^{\mathcal{O}} = \boldsymbol{M}_N(\boldsymbol{T}_N, \boldsymbol{C}_N^{II})$ *bezüglich der* DCT–II *kann mit einer* DST–I *aus den Daten*

$$\begin{aligned} \alpha_j &= \frac{1}{N}\left((N-j)t_j \;-\; 2t_{j+1} \;-\; (N-j-2)t_{j+2}\right) \quad (j = 0, \ldots, N-2) \\ \alpha_{N-1} &= \frac{2}{N^2}\sum_{k=0}^{N-2} k\, t_k \;+\; \frac{3N-2}{N^2}\, t_{N-1}\,. \end{aligned}$$

mit Hilfe der Formeln (3.15) *und* (3.16) *berechnet werden.*

Beweis: Wir berechnen zunächst die Daten

$$\boldsymbol{\beta}^{I} := \left(\langle \boldsymbol{T}_N, \boldsymbol{B}_j^{I}\rangle\right)_{j=0}^{N-1}$$

(siehe (3.17)). Die Basis $\boldsymbol{B}_k^{I}$ ist durch (3.11) gegeben. Also gilt

$$\beta_j^{I} = \langle \boldsymbol{T}_N, \boldsymbol{B}_j^{I}\rangle = \langle \boldsymbol{T}_N, \operatorname{stoep} \boldsymbol{e}_j'\rangle \;+\; \langle \boldsymbol{T}_N, \operatorname{shank} \boldsymbol{e}_{j-1}'\rangle \quad (j = 0, \ldots, N-1).$$

Unter Beachtung der Definition des Skalarproduktes (3.6) erkennen wir

$$\begin{aligned} \langle \boldsymbol{T}_N, \operatorname{stoep} \boldsymbol{e}_j\rangle &= 2(\varepsilon_j^N)^2 (N-j)\, t_j \quad (j = 0, \ldots, N-1)\,, \\ \langle \boldsymbol{T}_N, \operatorname{shank} \boldsymbol{e}_0\rangle &= 2t_0\,, \\ \langle \boldsymbol{T}_N, \operatorname{shank} \boldsymbol{e}_1\rangle &= 4t_1\,, \\ \langle \boldsymbol{T}_N, \operatorname{shank} \boldsymbol{e}_{j-2}\rangle &= \langle \boldsymbol{T}_N, \operatorname{shank} \boldsymbol{e}_j\rangle \;+\; 4\,t_j \quad (j = 2, \ldots, N-1)\,. \end{aligned}$$

Benutzen wir nun Lemma 3.9, so erhalten wir die Daten $(\beta_j^{II})_{j=0}^{N-1}$. Anschließend folgt mit Lemma 3.8 und (3.13) die Behauptung. ■

Bemerkung 3.24 Für die optimalen trigonometrischen Vorkonditionierer bezüglich $\boldsymbol{C}_N^{IV}$, $\boldsymbol{S}_{N-1}^{I}$, $\boldsymbol{S}_N^{II}$ bzw. $\boldsymbol{S}_N^{IV}$ erhalten wir folgenden Vektor $\boldsymbol{\alpha} = (\alpha_j)_{j=0}^{N-1} \in \mathbb{R}^N$:
Für den optimalen trigonometrischen Vorkonditionierer bezüglich DCT–IV(N) gilt

$$\begin{aligned}
\alpha_j &= \frac{1}{N}\left((N-j)t_j \;-\; (N-j-2)t_{j+2}\right) \quad (j = 0, \ldots, N-2)\,, \\
\alpha_{N-1} &= \frac{t_{N-1}}{N}\,.
\end{aligned}$$

Für den optimalen trigonometrischen Vorkonditionierer bezüglich DST–I($N-1$) gilt

$$\begin{aligned}
\alpha_0 &= t_0 \;-\; \frac{(N-3)}{N}\,t_2\,, \\
\alpha_j &= \frac{1}{N}\,\left((N+1-j)t_j \;-\; (N-j-3)t_{j+2}\right) \quad (j = 1, \ldots, N-3)\,, \\
\alpha_{N-2} &= \frac{3t_{N-2}}{N}\,.
\end{aligned}$$

Für den optimalen trigonometrischen Vorkonditionierer bezüglich DST–II(N) gilt

$$\begin{aligned}
\alpha_j &= \frac{1}{N}\left((N-j)t_j \;+\; 2t_{j+1} \;-\; (N-j-2)t_{j+2}\right) \quad (j = 0, \ldots, N-2), \\
\alpha_{N-1} &= \frac{2}{N^2}\sum_{k=0}^{N-2} (-1)^{k+1+N}\,k\,t_k \;+\; \frac{3N-2}{N^2}\,t_{N-1}\,.
\end{aligned}$$

Für den optimalen trigonometrischen Vorkonditionierer bezüglich DST–IV($N-1$) gilt

$$\begin{aligned}
\alpha_j &= \frac{1}{N}\,\left((N-j)t_j \;-\; (N-j-2)t_{j+2}\right) \quad (j = 0, \ldots, N-2)\,, \\
\alpha_{N-1} &= \frac{t_{N-1}}{N}\,.
\end{aligned}$$

Mit Hilfe von Bemerkung 3.11 können wir die Diagonalmatrix (3.9) des entsprechenden Vorkonditionierers bestimmen. □

Satz 3.25 *Für die Folge von symmetrischen Toeplitz–Matrizen* $\boldsymbol{T}_N = \operatorname{stoep} \boldsymbol{a}'_N$ $(N \in \mathbb{N})$ *mit* $\boldsymbol{a}_N := (a_k)_{k=0}^{N-1}$ *gelte*

$$\sum_{k=1}^{\infty} k\,a_k^2 \;<\; \infty\,.$$

Dann existieren für alle $\varepsilon > 0$ *Indizes* $M \in \mathbb{N}$ *und* $n \in \mathbb{N}$ $(n > M)$, *so daß für alle* $N > n$

$$\boldsymbol{T}_N - \boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{T}_N, \boldsymbol{C}_N^{II}) = \boldsymbol{W}_N + \boldsymbol{V}_N$$

gilt, wobei $\boldsymbol{W}_N$ *eine Matrix mit Rang* $\leq 2M$ *ist und* $\|\boldsymbol{V}_N\|_2 \leq \varepsilon$ *gilt.*

Beweis: Es gilt
$$\boldsymbol{T}_N - \boldsymbol{M}_N^{\mathcal{O}} = \boldsymbol{T}_N - \boldsymbol{M}_N^{\mathcal{S}} + \boldsymbol{M}_N^{\mathcal{S}} - \boldsymbol{M}_N^{\mathcal{O}} .$$
Da sich der Strang–Typ–Vorkonditionierer durch $\boldsymbol{C}_N^{II}$ diagonalisieren läßt, erhalten wir mit der Eigenschaft (3.9)
$$\boldsymbol{M}_N^{\mathcal{S}} - \boldsymbol{M}_N^{\mathcal{O}} = -(\boldsymbol{C}_N^{II})' \left(\delta(\boldsymbol{C}_N^{II}(\boldsymbol{T}_N - \boldsymbol{M}_N^{\mathcal{S}})(\boldsymbol{C}_N^{II})') \right) \boldsymbol{C}_N^{II} .$$
Nach Satz 3.17 ergibt sich
$$\begin{aligned}\boldsymbol{T}_N - \boldsymbol{M}_N^{\mathcal{O}} = \boldsymbol{W}_N + \boldsymbol{V}_N &- (\boldsymbol{C}_N^{II})' \left(\delta(\boldsymbol{C}_N^{II}\boldsymbol{W}_N(\boldsymbol{C}_N^{II})') \right) \boldsymbol{C}_N^{II} \\ &- (\boldsymbol{C}_N^{II})' \left(\delta(\boldsymbol{C}_N^{II}\boldsymbol{V}_N(\boldsymbol{C}_N^{II})') \right) \boldsymbol{C}_N^{II}\end{aligned}$$
mit der Matrix $\boldsymbol{W}_N := (w_{j,k})_{j,k=0}^{N-1}$ und $w_{j,k} = 0$ für $M \le j+k \le 2N-2-M$ aus Satz 3.17. Nun gilt
$$\begin{aligned}\|\boldsymbol{C}_N^{II}\delta((\boldsymbol{C}_N^{II})'\boldsymbol{V}_N\boldsymbol{C}_N^{II})(\boldsymbol{C}_N^{II})'\|_2 &= \|\delta((\boldsymbol{C}_N^{II})'\boldsymbol{V}_N\boldsymbol{C}_N^{II})\|_2 \\ &\le \|(\boldsymbol{C}_N^{II})'\boldsymbol{V}_N\boldsymbol{C}_N^{II}\|_2 = \|\boldsymbol{V}_N\|_2 < \varepsilon/2 .\end{aligned}$$
Mit $\omega := \sum\limits_{j,k=0}^{N-1} |w_{j,k}|$ folgt dann die Abschätzung
$$\begin{aligned}||\delta(\boldsymbol{C}_N^{II}\, \boldsymbol{W}\, (\boldsymbol{C}_N^{II})')||_2 &\le \max_{n=0,\dots,N-1} \left| \frac{2}{N} (\varepsilon_n^N)^2 \sum_{j,k=0}^{N-1} w_{j,k} \cos\frac{n(2j+1)\pi}{2N} \cos\frac{n(2k+1)\pi}{2N} \right| \\ &\le 2\omega/N < \varepsilon/2\end{aligned}$$
für $N > 4\omega/\varepsilon$. Beachten wir, daß für festes M der Wert ω nicht von N abhängt, so erhalten wir die Behauptung. ■

Folgerung 3.26 *Falls die Folge der symmetrischen Toeplitz–Matrizen*
$$\boldsymbol{T}_N := \frac{1}{2}(a_{|j-k|}[f])_{j,k=0}^{N-1} \quad (N \in \mathbb{N})$$
von einer positiven Funktion $f \in L^2_{w,1/2}(I)$ *erzeugt wird, so erfüllt der optimale trigonometrische Vorkonditionierer* $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{T}_N, \boldsymbol{C}_N^{II})$ *die Eigenschaften* (E1) – (E4). *Ferner gilt dies auch für* $f \in C(I)$ *mit* $f > 0$.

Beweis: Die Eigenschaft (E1) ist nach Lemma 3.7 erfüllt. Aus den Einträgen der Toeplitz–Matrix $\boldsymbol{T}_N$ können wir nach Satz 3.23 den Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{T}_N, \boldsymbol{C}_N^{II})$ in $\mathcal{O}(N \log N)$ Operationen berechnen. Damit sind die Eigenschaften (E2) – (E3) erfüllt. Für die Chebyshev–Koeffizienten $a_k := a_k[f]$ von $f \in L^2_{w,1/2}(I)$ gilt nach Definition
$$\sum_{k=0}^{\infty} (1+k)\, a_k^2 < \infty .$$

Somit folgt die Eigenschaft (E4) aus dem Satz 3.25 und Satz 3.4. Analoges Vorgehen wie in Folgerung 3.18 zeigt, daß diese Aussagen auch für $f \in C(I)$ gültig sind. Dies wurde in [31] für den Vorkonditionierer $\boldsymbol{M}^{\mathcal{O}}_{N-1}(\boldsymbol{T}_{N-1}, \boldsymbol{S}^{I}_{N-1})$ bewiesen. ■

Bemerkung 3.27 Falls die symmetrische Toeplitz–Matrix $\boldsymbol{T}_N$ indefinit ist, d.h., die erzeugende Funktion f hat positive und negative Funktionswerte, dann können wir die hier beschriebenen Algorithmen und Vorkonditionierer nicht anwenden. In diesem Fall gehen wir zu der Normalgleichung über (siehe Abschnitt 3.4 und Beispiel auf S. 81). Folgende andere Lösungsmöglichkeit bietet sich für diese Klasse von Matrizen an. Man nutzt eine Verallgemeinerung des CG–Verfahrens, um indefinite Probleme zu lösen (z.B. SYMMLQ oder MINRES, siehe [47], S. 166 ff). Ein offenes Problem ist die Entwicklung spezieller Vorkonditionierer für diese Verfahren. □

Bemerkung 3.28 Für dünnbesetzte Matrizen $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ sind approximative Vorkonditionierer bekannt [2]. Dazu ist ein Vorkonditionierer $\boldsymbol{M}_N$ mit zusätzlichen Forderungen gesucht, so daß $||\boldsymbol{I}_N - \boldsymbol{M}_N\boldsymbol{A}_N||_F$ minimal wird. E. Tyrtyshnikov berechnete für Toeplitz–Matrizen $\boldsymbol{T}_N$ in [92] die nichtsinguläre, zirkulante Matrix $\tilde{\boldsymbol{M}}_N$, für die

$$||\boldsymbol{I}_N - \tilde{\boldsymbol{M}}_N\boldsymbol{T}_N||_F = \min\{||\boldsymbol{I}_N - \tilde{\boldsymbol{B}}_N\boldsymbol{T}_N||_F : \tilde{\boldsymbol{B}}_N \in \tilde{\mathcal{A}}_{\boldsymbol{F}_N}, \tilde{\boldsymbol{B}}_N \text{ regulär}\}$$

gilt. Diese Matrix $\tilde{\boldsymbol{M}}_N$ wird *superoptimaler zirkulanter Vorkonditionierer* von $\boldsymbol{T}_N$ genannt. Wir definieren analog die superoptimalen trigonometrischen Vorkonditionierer. Für eine gegebene Matrix $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ wird eine Matrix $\tilde{\boldsymbol{M}}_N(\boldsymbol{A}_N, \boldsymbol{O}_N) \in \mathbb{R}^{N,N}$ *superoptimaler trigonometrischer Vorkonditionierer* von $\boldsymbol{A}_N$ in $\mathcal{A}_{\boldsymbol{O}_N}$ genannt, wenn (siehe [92])

$$||\boldsymbol{I}_N - \tilde{\boldsymbol{M}}_N(\boldsymbol{A}_N, \boldsymbol{O}_N)\boldsymbol{A}_N||_F = \min\{||\boldsymbol{I}_N - \boldsymbol{B}_N\boldsymbol{A}_N||_F : \boldsymbol{B}_N \in \mathcal{A}_{\boldsymbol{O}_N}, \boldsymbol{B}_N \text{ regulär}\}$$

gilt. In [23] wurde für zirkulante Vorkonditionierer gezeigt, daß die Eigenwerte der Matrix $\tilde{\boldsymbol{M}}_N\boldsymbol{T}_N$ um 1 geclustert sind. Der superoptimale Vorkonditionierer läßt sich leicht berechnen, wenn man den optimalen Vorkonditionierer für die Matrizen $\boldsymbol{T}'_N\boldsymbol{T}_N$ und $\boldsymbol{T}_N$ berechnen kann. Die Ergebnisse aus [22, 89] lassen sich in einfacher Weise auf Matrizen übertragen, die von einer trigonometrischen Matrix diagonalisiert werden können. Auch im trigonometrischen Fall kann man aus $\boldsymbol{M}^{\mathcal{O}}_N(\boldsymbol{T}'_N\boldsymbol{T}_N, \boldsymbol{O}_N)$ und aus $\boldsymbol{M}^{\mathcal{O}}_N(\boldsymbol{T}_N, \boldsymbol{O}_N)$ den superoptimalen trigonometrischen Vorkonditionierer konstruieren. Die schnelle Berechnung von $\boldsymbol{M}^{\mathcal{O}}_N(\boldsymbol{T}'_N\boldsymbol{T}_N, \boldsymbol{O}_N)$ werden wir in Abschnitt 3.4.2 betrachten. Da man oft nicht weiß, wie „gut" die Clusterung bei den Vorkonditionierern ausfällt, ist man auf numerische Tests angewiesen (siehe [90]). Die numerischen Ergebnisse des superoptimalen zirkulanten Vorkonditionierers zeigen, daß dieser oft schlechtere Ergebnisse als die zirkulanten Strang–Vorkonditionierer bzw. die optimalen zirkulanten Vorkonditionierer [88] liefert. Für die superoptimalen trigonometrischen Vorkonditionierer können wir diese Beobachtung bestätigen. Deshalb wollen wir die Ergebnisse auch nicht übertragen, sondern nur numerische Testergebnisse angeben (siehe Abschnitt 3.3.3). □

3.3.3 Numerische Beispiele

In diesem Abschnitt wollen wir die Strang–Typ–Vorkonditionierer, die optimalen trigonometrischen Vorkonditionierer und die superoptimalen trigonometrischen Vorkonditionierer an verschiedenen Beispielen testen. Wir benutzen die Vorkonditionierer bezüglich $\boldsymbol{C}_N^{II}, \boldsymbol{C}_N^{IV}, \boldsymbol{S}_N^{II}$ bzw. $\boldsymbol{S}_N^{IV}$. Die schnelle Berechnung der Vorkonditionierer und des PCG–Verfahrens wurden in MATLAB auf einer Sun SPARCstation 20 implementiert und getestet. Die schnellen trigonometrischen Transformationen wurden von G. Baszenski in C implementiert [4]. Mit Hilfe des cmex–Programms können wir diese Programme von MATLAB aus starten. Somit ist eine schnelle Berechnung der trigonometrischen Transformationen in MATLAB möglich.
Um die Anzahl der trigonometrischen Transformationen in jedem PCG–Schritt zu reduzieren, gehen wir ähnlich wie in [60] vor. Wir zerlegen die Toeplitz–Matrix mit Hilfe der diskreten trigonometrischen Transformation, welche wir für den Vorkonditionierer benutzen. Dies wird am Beispiel der Vorkonditionierung mit der DCT–II verdeutlicht. Mit dem PCG–Verfahren haben wir das lineare Gleichungssystem

$$\boldsymbol{M}_N^{-1}\, \boldsymbol{T}_N\, \boldsymbol{x}_N = \boldsymbol{M}_N^{-1}\, \boldsymbol{b}_N \tag{3.32}$$

zu lösen. Dabei ist $\boldsymbol{T}_N$ eine symmetrische Toeplitz–Matrix und $\boldsymbol{M}_N$ ein trigonometrischer Vorkonditionierer, der durch eine DCT–II diagonalisiert werden kann, d.h.

$$\begin{aligned} \boldsymbol{M}_N &= (\boldsymbol{C}_N^{II})'\boldsymbol{D}\, \boldsymbol{C}_N^{II}\, , \\ \boldsymbol{M}_N^{-1} &= (\boldsymbol{C}_N^{II})'\boldsymbol{D}^{-1}\, \boldsymbol{C}_N^{II} \end{aligned}$$

mit einer Diagonalmatrix $\boldsymbol{D}$. Für die symmetrische Toeplitz–Matrix $\boldsymbol{T}_N$ benutzen wir die Zerlegung (siehe Satz 1.4 (ii))

$$\boldsymbol{T}_N = (\boldsymbol{C}_N^{II})'\boldsymbol{E}\, \boldsymbol{C}_N^{II} \; + \; (\boldsymbol{S}_N^{II})'\tilde{\boldsymbol{E}}\, \boldsymbol{S}_N^{II}$$

mit Diagonalmatrizen $\boldsymbol{E}$ und $\tilde{\boldsymbol{E}}$. Somit würden wir in jedem PCG–Schritt 6 diskrete trigonometrische Transformationen benötigen. Wir ersetzen (3.32) durch

$$(\boldsymbol{C}_N^{II}\boldsymbol{M}_N^{-1}\, \boldsymbol{T}_N(\boldsymbol{C}_N^{II})')\; (\boldsymbol{C}_N^{II}\, \boldsymbol{x}_N) = \boldsymbol{C}_N^{II}\, \boldsymbol{M}_N^{-1}\, \boldsymbol{b}_N\; .$$

Mit der Darstellung des Vorkonditionierers und der Toeplitz–Matrix erhalten wir das lineare Gleichungssystem

$$\boldsymbol{D}^{-1}\, (\boldsymbol{E} \; + \; \boldsymbol{C}_N^{II}(\boldsymbol{S}_N^{II})'\; \tilde{\boldsymbol{E}}\, \boldsymbol{S}_N^{II}\; (\boldsymbol{C}_N^{II})')\; \boldsymbol{y}_N \; = \; \tilde{\boldsymbol{b}}_N \tag{3.33}$$

mit $\boldsymbol{y}_N := \boldsymbol{C}_N^{II}\, \boldsymbol{x}_N$ und $\tilde{\boldsymbol{b}}_N := \boldsymbol{C}_N^{II}\, \boldsymbol{M}_N^{-1}\, \boldsymbol{b}_N$. Die Matrizen in (3.32) und (3.33) haben die gleichen Eigenwerte. Wir wenden also das PCG–Verfahren mit dem Vorkonditionierer $\boldsymbol{D}$ auf die Matrix $\boldsymbol{C}_N^{II}\boldsymbol{T}_N(\boldsymbol{C}_N^{II})'$ an.

Bemerkung 3.29 Verwenden wir für die symmetrischen Toeplitz–Matrizen $\boldsymbol{T}_N \in \mathbb{R}^{N,N}$ ($N \in \mathbb{N}$) Zerlegungen mit den gleichen trigonometrischen Matrizen $\boldsymbol{O}_N$, so braucht das PCG–Verfahren mit dem Vorkonditionierer $\boldsymbol{O}'_N \boldsymbol{D}\boldsymbol{O}_N$ nur N Multiplikationen mehr als das CG–Verfahren. Es sind dadurch in jedem PCG–Schritt 4 diskrete trigonometrische Transformationen ($8N \log N + \mathcal{O}(N)$ arithmetische Operationen) nötig. Arbeitet man dagegen mit der DFT für reelle Eingabedaten, so sind $10N \log N + \mathcal{O}(N)$ arithmetische Operationen notwendig (siehe [60]). Aus diesem Grunde bevorzugen wir die diskreten trigonometrischen Transformationen. □

Als Transformationslänge wählen wir $N = 2^n$. Die rechte Seite $\boldsymbol{b}_N$ von (3.32) ist der Vektor, der N Einsen enthält. Das PCG–Verfahren starten wir mit dem Nullvektor. Die Iteration wird abgebrochen, wenn $\|\boldsymbol{r}^{(j)}\|_2 / \|\boldsymbol{r}^{(0)}\|_2 < 10^{-7}$ gilt. Dabei bezeichnet $\boldsymbol{r}^{(j)}$ den Residuumvektor nach j Iterationen.
In den Tabellen im Anhang tragen wir in der zweiten Spalte die Anzahl der Iterationen ein, die das CG–Verfahren benötigt. Die 3. – 6. Spalten bzw. 7. – 10. Spalten geben die Anzahl der Iterationen an, die das PCG–Verfahren mit einem Strang–Typ–Vorkonditionierer bzw. einen optimalen trigonometrischen Vorkonditionierer bezüglich der jeweiligen diskreten trigonometrischen Transformation benötigt. Die letzten vier Spalten geben die Anzahl der Iterationen an, die das PCG–Verfahren bei Benutzung eines superoptimalen trigonometrischen Vorkonditionierers braucht.

Beispiel (i): Wir benutzen zunächst die erzeugende Funktion $f(x) := x^4 + 1$ ($x \in [-1, 1]$), um symmetrische Toeplitz–Matrizen $\boldsymbol{T}_N$ zu definieren. In diesem Fall sind die Chebyshev–Koeffizienten durch die Darstellung

$$x^4 + 1 = \frac{1}{8} T_4(x) \; + \; \frac{1}{2} T_2(x) \; + \; \frac{11}{8} T_0(x)$$

gegeben. Nach Lemma 3.15 gilt für die Eigenwerte λ_j der Matrix $\boldsymbol{T}_N$ die Ungleichung $1 \leq \lambda_j(\boldsymbol{T}_N(f)) \leq 2$. Wir können nach Satz 3.5 eine obere Schranke für die Anzahl der nötigen Iterationen berechnen, um diese Toeplitz–Systeme ohne Vorkonditionierer zu lösen. Für $a = 1$, $b = 2$, $p = 0$, $q = 0$ und $\tau = 10^{-7}$ erhalten wir 10 Iterationen. Die Anzahl der Iterationen für verschiedene Vorkonditionierer haben wir in Tabelle 3.2 angegeben. Wir betrachten zwei weitere Beispiele aus der Literatur.
Beispiel (ii): Die Einträge der Toeplitz–Matrix sind durch $t_k = 1/(k+1)$ ($k = 0, \ldots, 2^n - 1$) (siehe [14] und Tabelle 3.3) gegeben. In diesem Fall sind die Voraussetzungen von Satz 3.17 und Satz 3.25 nicht erfüllt, denn $\sum\limits_{k=1}^{\infty} k t_k^2$ divergiert.
Beispiel (iii): Die erzeugende Funktion φ ist durch $\varphi(t) = t^4 + 1 \quad (t \in [-\pi, \pi])$ gegeben (siehe [28, 31] und Tabelle 3.4). Nach Satz 3.5 sind maximal 84 Iterationen nötig, um die geforderte Genauigkeit zu erhalten.

Aus vielen weiteren numerischen Beispielen mit Matrizen aus [14, 31, 88, 90, 92] ergeben sich folgende Beobachtungen:

1. Unter den Voraussetzungen von Folgerung 3.18 und Folgerung 3.26 konvergiert das PCG–Verfahren mit dem Strang–Typ–Vorkonditionierer, mit dem optimalen trigonometrischen Vorkonditionierer und mit dem superoptimalen trigonometrischen Vorkonditionierer superlinear.
2. In allen Fällen liefert ein trigonometrischer Vorkonditionierer in weniger Schritten ein gleich gutes Ergebnis wie ein zirkulanter Vorkonditionierer. Da die arithmetische Komplexität mit einer diskreten trigonometrischen Transformation geringer als bei einer DFT mit reellen Eingabedaten (siehe Bemerkung 1.6 und Gleichung (3.33)) ist, sind diese Vorkonditionierer für reelle Toeplitz–Matrizen vorzuziehen.
3. In allen getesteten Beispielen liefern die superoptimalen trigonometrischen Vorkonditionierer keine besseren Ergebnisse als die Strang–Typ–Vorkonditionierer oder die optimalen trigonometrischen Vorkonditionierer. Wir betrachten sie deshalb in den weiteren Beispielen nicht mehr. Dies wurde für den superoptimalen zirkulanten Vorkonditionierer bereits in [88] beobachtet.

Es ist bekannt, daß trigonometrische Vorkonditionierer auch dann gute Ergebnisse liefern, wenn die erzeugende Funktion $f \geq 0$ Nullstellen hat [8]. In den folgenden 3 Beispielen untersuchen wir Toeplitz–Bandmatrizen. In diesen Fällen sind die vereinfachten Strang–Typ– und die Strang–Typ–Vorkonditionierer identisch.
Beispiel (iv): Zunächst sei $f(x) := (x^2+1)(1-x)$ $(x \in [-1,1])$ die erzeugende Funktion für die Folge von symmetrischen Toeplitz–Matrizen. Die Chebyshev–Koeffizienten können wir wieder explizit berechnen. Tabelle 3.5 zeigt die Anzahl der Iterationen, um $\boldsymbol{T}_N^{-1}\boldsymbol{b}_N$ mit der geforderten Genauigkeit zu berechnen. Diese Toeplitz–Matrizen sind Bandmatrizen mit der Bandbreite 6. Es ist klar, daß der Strang–Typ–Vorkonditionierer bezüglich DCT–II für diese Funktion nicht invertierbar ist, weil $f(1) = \mathcal{S}_N f(1) = 0$ $(N \geq 4)$ gilt. In diesem Fall liefert der optimale trigonometrische Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$ schlechtere Ergebnisse als $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$. Die besten Ergebnisse erhalten wir mit dem Strang–Typ–Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{S}}(\boldsymbol{S}_N^{II})$.
Beispiel (v): Sei nun $f(x) := (x^2+1)(1+x)$ $(x \in [-1,1])$ die erzeugende Funktion für die Folge von symmetrischen Toeplitz–Matrizen. In diesem Fall können wir den Strang–Typ–Vorkonditionierer bezüglich DST–II nicht invertieren, weil $f(-1) = \mathcal{S}_N f(-1) = 0$ $(N \geq 4)$ gilt (siehe Tabelle 3.6).
Beispiel (vi): Auch für mehrfache Nullstellen sind diese Vorkonditionierer noch gut geeignet. Wir wählen dazu die erzeugende Funktion $f(x) = (x^2+1)(1+x)^2$ $(x \in [-1,1])$ (siehe Tabelle 3.7).
Um unsere trigonometrischen Vorkonditionierer mit den Vorkonditionierern aus [83] zu vergleichen, verwenden wir die gleichen Testbeispiele.
Beispiel (vii): Sei $\varphi(t) := (t^2-1)^2$ $(t \in [-\pi,\pi])$. Die Fourier–Koeffizienten können wir explizit berechnen. In Tabelle 3.8 haben wir die Iterationszahlen eingetragen. In diesem Fall ist der Strang–Typ–Vorkonditionierer nicht positiv definit, d.h., in der Diagonalmatrix des Vorkonditionierers haben wir negative Einträge. Dies kennzeichnen wir in den folgenden Tabellen durch einen Stern hinter der Anzahl der benötigten Iterationen. Vergleichen wir diese Ergebnisse mit denen aus [83], so erkennen wir, daß die vereinfachten Strang–Typ–Vorkonditionierer die besten Ergebnisse liefern. Ist die erzeugende

Funktion φ bekannt, so sollte man den Vorkonditionierer $\boldsymbol{M}_N(f)$ benutzen. Die letzte Spalte von Tabelle 3.8 gibt die Anzahl der Iterationen bezüglich $\boldsymbol{M}_N(f, \boldsymbol{S}_N^{II})$ an. Der Vorkonditionierer $\boldsymbol{M}_N(f, \boldsymbol{C}_N^{II})$ ermöglicht in diesem Fall die gleichen Ergebnisse.
Beispiel (viii): In der Tabelle 3.9 ist die Anzahl der Iterationen angegeben, falls die Toeplitz–Matrizen $\boldsymbol{T}_N$ von der Funktion $\varphi(t) := t^4$ $(t \in [-\pi, \pi])$ erzeugt werden (vgl. [83, 18, 26]). Zu beachten ist, daß für dieses Beispiel der vereinfachte Strang–Typ–Vorkonditionierer $\boldsymbol{M}_N(f, \boldsymbol{C}_N^{II})$ bezüglich der DCT–II singulär ist, da $\varphi(0) = 0$ ist. Auch hier liefert der Vorkonditionierer $\boldsymbol{M}_N(f, \boldsymbol{S}_N^{II})$ die besten Ergebnisse. Die numerischen Ergebnisse bestätigen, daß die vereinfachten Strang–Typ–Vorkonditionierer auch für schlecht konditionierte positiv definite Matrizen geeignet sind. Die Wahl des Vorkonditionierers hängt von der Lage der Nullstellen ab. Wichtig ist dabei, daß die Vorkonditionierer nicht singulär sind.
Beispiel (ix): Um die Effizienz der vereinfachten Strang–Typ–Vorkonditionierer auch im Block–Fall zu verdeutlichen, wollen wir ein Beispiel angeben. Da die Übertragung auf den Block–Fall schon für viele andere Vorkonditionierer [79, 73, 66] erfolgte, verzichten wir auf eine ausführliche Herleitung. Sei $\varphi(s,t) = s^2 t^4$ $(s, t \in [-\pi, \pi])$ die erzeugende Funktion des symmetrischen Block–Toeplitz – Toeplitz–Block–Systems. Mit $\boldsymbol{M}_N(\varphi, \boldsymbol{S}_N^{II} \otimes \boldsymbol{S}_N^{II})$ bezeichnen wir den vereinfachten Strang–Typ–Vorkonditionierer. Außerdem betrachten wir die erzeugende Funktion $\psi(s,t) = (s^2 + t^2)^2$ $(s, t \in [-\pi, \pi])$ (siehe [73]). Die Ergebnisse stellen wir in Tabelle 3.10 dar. Auch im Block–Fall sind keine Vorkonditionierer bekannt, die für schlecht konditionierte Matrizen bessere Ergebnisse liefern. Außerdem benötigen diese Vorkonditionierer keine zusätzlichen diskreten trigonometrischen Transformationen (siehe Bemerkung 3.29).

3.4 Vorkonditionierer für nichtsymmetrische und rechteckige Toeplitz–Matrizen

Da im symmetrischen Fall die trigonometrischen Vorkonditionierer oft sehr gute Ergebnisse liefern, wollen wir trigonometrische Vorkonditionierer auch für nichtsymmetrische Toeplitz–Matrizen konstruieren. Auf Grund der bestechenden Eigenschaften des CG–Verfahrens für symmetrische, positiv definite Matrizen ist natürlich die Frage von Interesse, welche Eigenschaften sich auf nichtsymmetrische Matrizen übertragen lassen. Zunächst ist darauf hinzuweisen, daß eine beliebige invertierbare Matrix $\boldsymbol{A}$ i.allg. kein Skalarprodukt induziert. Zwei Lösungsmöglichkeiten bieten sich an. Zum einen kann man das CG–Verfahren verallgemeinern (siehe z.B. [81]) und entsprechende Vorkonditionierer konstruieren. Dies wurde kürzlich von E. Tyrtyshnikov [94] untersucht. Er betrachtete zirkulante Vorkonditionierer für GMRES. Falls man auf jeden Fall die Drei–Term–Rekursion des CG–Verfahrens erhalten will, so geht man i.allg. zu der Normalgleichung

$$\boldsymbol{A}'\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{A}'\,\boldsymbol{b}$$

über und realisiert ein CG–Verfahren für die symmetrische, positiv definite Matrix $\boldsymbol{A}'\boldsymbol{A}$. Dieses Vorgehen hat jedoch wegen cond $(\boldsymbol{A}'\boldsymbol{A}) = (\text{cond}\ \boldsymbol{A})^2$ zur Folge, daß in der Abschätzung (3.3) zur Konvergenzgeschwindigkeit der Faktor cond $(\boldsymbol{A})$ durch $(\text{cond}\ \boldsymbol{A})^2$ zu ersetzen ist. Da dies oft eine signifikante Verschlechterung für die Konvergenzgeschwindigkeit bedeutet, wollen wir in diesem Abschnitt Vorkonditionierer mit den Eigenschaften (E1) – (E4) für die Normalgleichung konstruieren.

Es soll nun der allgemeinere Fall einer rechteckigen Toeplitz–Matrix

$$\boldsymbol{A} = (a_{j-k})_{j=0,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N}$$

mit $M \geq N$ betrachtet werden, wobei Rang$\boldsymbol{A} = N$ ist. Wir erhalten somit ein lineares Ausgleichsproblem. Gesucht ist zu gegebenem $\boldsymbol{b} \in \mathbb{R}^M$ und $\boldsymbol{A} \in \mathbb{R}^{M,N}$ mit $M \geq N$ ein Vektor $\boldsymbol{x} \in \mathbb{R}^N$, so daß

$$\|\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}\|_2 \;=\; \min\,. \tag{3.34}$$

Satz 3.30 (siehe [42], S. 66) *Ein Vektor* $\boldsymbol{x} \in \mathbb{R}^N$ *ist genau dann eine Lösung des linearen Ausgleichsproblems* (3.34)*, wenn er die Normalgleichung*

$$\boldsymbol{A}'\boldsymbol{A}\,\boldsymbol{x} \;=\; \boldsymbol{A}'\boldsymbol{b} \tag{3.35}$$

erfüllt. Insbesondere ist das lineare Ausgleichsproblem genau dann eindeutig lösbar, wenn $\boldsymbol{A}$ *maximalen Rang hat, d.h.* Rang $\boldsymbol{A} = N$*.*

Für die Normalgleichung wurden zirkulante Vorkonditionierer [24] und sogenannte „displacement“ –Vorkonditionierer [25] konstruiert. Ein trigonometrischer Vorkonditionierer wurde in [8] angegeben (siehe auch Satz 3.32).

Unser nächstes Ziel ist eine geeignete additive Zerlegung von $\boldsymbol{A}'\boldsymbol{A}$. In Lemma 3.33 wird gezeigt, daß unter zusätzlichen Voraussetzungen $\boldsymbol{A}'\boldsymbol{A}$ in die Summe einer symmetrischen Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$, einer Matrix mit Rang $\leq M$ und einer Matrix mit kleiner Spektralnorm zerlegt werden kann. Diese Vorgehen führt dann zur Konstruktion von einfachen Vorkonditionierern. Wir beweisen zunächst:

Lemma 3.31 *Seien* $\boldsymbol{L} \in \mathbb{R}^{N,N}$ *und* $\tilde{\boldsymbol{L}} \in \mathbb{R}^{N,N}$ *untere Toeplitz–Dreiecksmatrizen*

$$\begin{aligned} \boldsymbol{L} &:= \operatorname{toep}(\boldsymbol{o}'_N, (0, l_1, l_2, \ldots, l_{N-1}))\,, \\ \tilde{\boldsymbol{L}} &:= \operatorname{toep}(\boldsymbol{o}'_N, (0, l_{N-1}, l_{N-2}, \ldots, l_1))\,. \end{aligned}$$

Dann gilt

$$\boldsymbol{L}'\boldsymbol{L} \;+\; \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}' \;=\; \operatorname{stoep}((\boldsymbol{L}'\boldsymbol{L}\,\boldsymbol{e}_0)')\,. \tag{3.36}$$

Für obere Toeplitz–Dreiecksmatrizen

$$\begin{aligned} \boldsymbol{U} &:= \operatorname{toep}((0, u_1, u_2, \ldots, u_{N-1}), \boldsymbol{o}'_N)\,, \\ \tilde{\boldsymbol{U}} &:= \operatorname{toep}((0, u_{N-1}, u_{N-2}, \ldots, u_1), \boldsymbol{o}'_N) \end{aligned}$$

gilt

$$\boldsymbol{U}'\boldsymbol{U} \;+\; \tilde{\boldsymbol{U}}\tilde{\boldsymbol{U}}' \;=\; \mathrm{stoep}((\boldsymbol{U}\boldsymbol{U}'\,\boldsymbol{e}_0)')\;. \tag{3.37}$$

Beweis: Die Behauptung verifizieren wir durch einfaches Nachrechnen. ■

Satz 3.32 *Sei $\varphi \in C_{2\pi}$ die erzeugende Funktion einer gegebenen Toeplitz–Matrix*

$$\boldsymbol{A} = (a_{j-k}(\varphi))_{j=0,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N}$$

mit $M \geq N$. Dann läßt sich $\boldsymbol{A}'\boldsymbol{A}$ in der Form

$$\boldsymbol{A}'\boldsymbol{A} \;=\; \tilde{\boldsymbol{T}}_N \;-\; \tilde{\boldsymbol{L}}_N\tilde{\boldsymbol{L}}_N' \;-\; \tilde{\boldsymbol{U}}_N\tilde{\boldsymbol{U}}_N' \tag{3.38}$$

zerlegen. Dabei ist $\tilde{\boldsymbol{T}}_N$ eine symmetrische Toeplitz–Matrix, die von der Funktion $|\varphi|^2$ erzeugt wird. Die Matrix $\tilde{\boldsymbol{U}}_N$ ist eine obere Toeplitz–Dreiecksmatrix und $\tilde{\boldsymbol{L}}_N$ ist eine untere Toeplitz–Dreiecksmatrix mit

$$\tilde{\boldsymbol{T}}_N \;:=\; \mathrm{stoep}(t_0, t_1, \ldots, t_{N-1})\;, \tag{3.39}$$

$$\tilde{\boldsymbol{U}}_N \;:=\; \mathrm{toep}((0, a_{1-N}, a_{2-N}, \ldots, a_{-1}), \boldsymbol{o}_N')\;, \tag{3.40}$$

$$\tilde{\boldsymbol{L}}_N \;:=\; \mathrm{toep}(\boldsymbol{o}_N', (0, a_{M-1}, a_{M-2}, \ldots, a_{M-N+1}))\;. \tag{3.41}$$

Beweis: Wir benutzen die Darstellung (1.15) von Lemma 1.8 und beachten (3.36) sowie (3.37). Dann ist

$$\tilde{\boldsymbol{T}}_N = \boldsymbol{T}_N \;+\; \mathrm{stoep}((\boldsymbol{L}'\boldsymbol{L}\,\boldsymbol{e}_0)') \;+\; \mathrm{stoep}((\boldsymbol{U}\boldsymbol{U}'\,\boldsymbol{e}_0)')\;.$$

Die Einträge der symmetrischen Toeplitz–Matrix $\boldsymbol{T}_N$ der Form (1.20) sind in Lemma 1.8 angegeben. Mit den Bezeichnungen von Lemma 1.8 erhalten wir wegen Gleichung (1.21)

$$\tilde{\boldsymbol{T}}_N = \mathrm{stoep}(\tilde{t}_0, \tilde{t}_1, \ldots, \tilde{t}_{N-1})$$

mit

$$(\tilde{t}_0, \tilde{t}_1, \ldots, \tilde{t}_{N-1})' = (\boldsymbol{M}'\boldsymbol{M} + \boldsymbol{M}'\boldsymbol{L} + \boldsymbol{U}'\boldsymbol{M} + \boldsymbol{L}'\boldsymbol{L} + \boldsymbol{U}\boldsymbol{U}')\,\boldsymbol{e}_0\;.$$

Die Einträge dieser Toeplitz–Matrix können wir in $\mathcal{O}(M \log M)$ Operationen berechnen (siehe Formel (1.14)). Nun gilt aber auch die Gleichung

$$(\tilde{t}_0, \tilde{t}_1, \ldots, \tilde{t}_{N-1})' = \tilde{\boldsymbol{T}}_N\,\boldsymbol{e}_0 \;=\; (\boldsymbol{A}'\boldsymbol{A} \;+\; \tilde{\boldsymbol{L}}_N\tilde{\boldsymbol{L}}_N' \;+\; \tilde{\boldsymbol{U}}_N\tilde{\boldsymbol{U}}_N')\,\boldsymbol{e}_0\;.$$

Schreiben wir dies um, so erhalten wir

$$(\tilde{t}_0, \tilde{t}_1, \ldots, \tilde{t}_{N-1})' = \mathrm{toep}((\boldsymbol{a}_{M+N-1}', \boldsymbol{o}_{N-1}'), (a_{1-N}, \boldsymbol{o}_{N-1}'))((\boldsymbol{J}_{M+N-1}\boldsymbol{a}_{M+N-1})', \boldsymbol{o}_{N-1}')' \tag{3.42}$$

mit der Abkürzung $\boldsymbol{a}_{N+M-1} = (a_j)_{j=1-N}^{M-1}$. Nutzen wir die Polynomdarstellung wie in [8], so erkennen wir, daß die Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$ von der Funktion $|\varphi|^2$ erzeugt wird. Dazu betrachten wir die Laurent–Entwicklung von der erzeugenden Funktion φ in der Form

$$\varphi(z) = \sum_{j=1-N}^{M-1} a_j z^j \ .$$

Die Funktion

$$\bar{\varphi}(z) = \sum_{j=1-N}^{M-1} a_j z^{-j}$$

erzeugt die Toeplitz–Matrix $\boldsymbol{A}'$. Die Berechnung der Laurent–Koeffizienten der Funktion $\bar{\varphi}\varphi$ führt genau auf die Gleichung (3.42). Somit wird die Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$ von der Funktion $|\varphi|^2$ erzeugt. ■

3.4.1 Strang–Typ–Vorkonditionierer

In diesem Abschnitt wollen wir einfache Vorkonditionierer für die Normalgleichung

$$\boldsymbol{A}'\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}'\boldsymbol{b}$$

einführen, wobei $\boldsymbol{A} = (a_{j-k})_{j=0,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N}$ eine rechteckige Toeplitz–Matrix mit $M \geq N$ ist. Diese Vorkonditionierer beruhen auf einer Zerlegung der Matrix $\boldsymbol{A}'\boldsymbol{A}$. Wir werden diese Vorkonditionierer deshalb wieder Strang–Typ–Vorkonditionierer nennen. Wir betrachten zwei Vorkonditionierer $\boldsymbol{M}^I$ und $\boldsymbol{M}^{II}$. Dabei bezeichnen wir mit $\boldsymbol{M}^I$ den optimalen Vorkonditionierer der symmetrischen Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$ von (3.38) und mit $\boldsymbol{M}^{II}$ den Strang–Typ–Vorkonditionierer der symmetrischen Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$ von (3.38).

Lemma 3.33 *Für die rechteckigen Toeplitz–Matrizen $\boldsymbol{A} = (a_{j-k})_{j=0,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N}$ mit $M \geq N$ gelte*

$$\sum_{k=-\infty}^{\infty} |k| \, a_k^2 < \infty \ . \tag{3.43}$$

Dann existieren für alle $\varepsilon > 0$ Indizes $m, n \in \mathbb{N}$ $(n > m)$, so daß für alle $N > n$ die Darstellung

$$\boldsymbol{A}'\boldsymbol{A} - \tilde{\boldsymbol{T}}_N = \boldsymbol{W}_N + \boldsymbol{V}_N$$

gilt, wobei $\tilde{\boldsymbol{T}}_N$ durch (3.39) *erklärt ist, $\boldsymbol{W}_N$ eine Matrix mit Rang $\leq m$ ist und $\|\boldsymbol{V}_N\|_2 \leq \varepsilon$ gilt.*

Beweis: Wir verwenden die Darstellung (3.38). Für die Frobenius–Norm erhalten wir die Abschätzung

$$\|\boldsymbol{L}_N\boldsymbol{L}'_N \ + \ \boldsymbol{U}_N\boldsymbol{U}'_N\|_F \leq \|\boldsymbol{L}_N\|_F \ \|\boldsymbol{L}'_N\|_F \ + \ \|\boldsymbol{U}_N\|_F \ \|\boldsymbol{U}'_N\|_F \,.$$

Wegen

$$\|\boldsymbol{U}_N\|_F = \|\operatorname{toep}((0, a_{1-N}, a_{2-N}, \ldots, a_{-1}), \boldsymbol{o}'_N)\|_F = (\sum_{k=1}^{N-1} k \ a_{-k}^2)^{1/2}$$

existiert für ein hinreichend großes N ein Index $m \in \mathbb{N}$, so daß

$$\boldsymbol{W}_N^1 = \operatorname{toep}((0, \ldots, 0, a_{-m}, a_{1-m}, \ldots, a_{-1}), \boldsymbol{o}'_N)$$

eine Matrix mit Rang $\leq m$ und

$$\boldsymbol{V}_N^1 = \operatorname{toep}((0, a_{1-N}, a_{2-N}, \ldots, a_{-1-m}, 0, \ldots, 0), \boldsymbol{o}'_N)$$

eine Matrix mit einer Spektralnorm $\|\boldsymbol{V}_N^1\|_2 \leq \|\boldsymbol{V}_N^1\|_F \leq (\frac{1}{2}\varepsilon)^{1/2}$ ist. Wir bilden analoge Zerlegungen für die Matrizen $\boldsymbol{U}'_N$, $\boldsymbol{L}_N$ und $\boldsymbol{L}'_N$. Da die Multiplikation einer Matrix mit Rang $\leq m$ mit einer beliebigen Matrix wieder eine Matrix mit Rang $\leq m$ ist, erhalten wir die Behauptung. ■

Bemerkung 3.34 Falls $\varphi = \sum_{k=-\infty}^{\infty} a_k \mathrm{e}^{\mathrm{i}kt}$ die erzeugende Funktion für die Folge der Matrizen $\boldsymbol{A} \in \mathbb{R}^{M,N}$ ist, so bedeutet die Gleichung (3.43), daß φ eine Funktion aus einem periodischen Sobolev–Raum ist.
In Lemma 3.33 haben wir eine symmetrische Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$ gefunden, die die Matrix $\boldsymbol{A}'\boldsymbol{A}$ „approximiert“. Wir können auch analog wie in Lemma 3.6 vorgehen. Finde die symmetrische Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$ mit

$$\tilde{\boldsymbol{T}}_N = \sum_{k=0}^{N-1} a_k \ \operatorname{stoep} \boldsymbol{e}'_k \ ,$$

so daß

$$||\boldsymbol{A}'\boldsymbol{A} - \tilde{\boldsymbol{T}}_N||_F = \min\{||\boldsymbol{A}'\boldsymbol{A} - \boldsymbol{T}_N||_F : \boldsymbol{T}_N = \operatorname{stoep} \boldsymbol{t}'_N \,, \quad \boldsymbol{t}_N \in \mathbb{R}^N\} \,.$$

Dieses Problem können wir wieder in einfacher Weise mit einem Galerkin–Ansatz berechen. Zur schnellen Berechnung der Skalarprodukte $\langle \boldsymbol{A}'\boldsymbol{A}, \operatorname{stoep} \boldsymbol{e}'_k \rangle$ verweisen wir auf die Lemmata 3.36 – 3.38. □

Das Lemma 3.33 zeigt, wie man in einfacher Weise trigonometrische Vorkonditionierer für die Matrix $\boldsymbol{A}'\boldsymbol{A}$ definieren kann.

Mit $\boldsymbol{M}_N^I := \boldsymbol{M}_N^{\mathcal{O}}(\tilde{\boldsymbol{T}}_N, \boldsymbol{O}_N)$ bezeichnen wir den optimalen trigonometrischen Vorkonditionierer der symmetrischen Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$ aus (3.38). Mit $\boldsymbol{M}_N^{II} := \boldsymbol{M}_N^{\mathcal{S}}(\tilde{\boldsymbol{T}}_N, \boldsymbol{O}_N)$ bezeichnen wir den Strang–Typ–Vorkonditionierer der symmetrischen Toeplitz–Matrix $\tilde{\boldsymbol{T}}_N$ aus (3.38).

Folgerung 3.35 *Falls eine Folge von Toeplitz–Matrizen* $\boldsymbol{A} \in \mathbb{R}^{M,N}$ *von einer Funktion* φ *mit* $|\varphi| > 0$ *und* (3.43) *erzeugt wird, so erfüllen die Strang–Typ–Vorkonditionierer* $\boldsymbol{M}_N^I$ *und* $\boldsymbol{M}_N^{II}$ *die Eigenschaften* (E1) – (E4). *Ferner gilt dies auch für* $\varphi \in C_{2\pi}$ *mit* $|\varphi| > 0$.

Beweis: Nach Satz 3.32 sind die Voraussetzungen von den Folgerungen 3.18 und 3.26 erfüllt. Zusammen mit Lemma 3.33 folgt die Behauptung. Analoges Vorgehen wie in Folgerung 3.18 zeigt, daß diese Aussagen auch für $\varphi \in C_{2\pi}$ gültig sind. ■

Wie in Abschnitt 3.3.3 wollen wir unsere Vorkonditionierer auch an schlecht gestellten Problemen testen. In Satz 3.32 haben wir gezeigt, daß die Matrix $\tilde{\boldsymbol{T}}_N$ durch die Funktion $|\varphi|^2$ erzeugt wird, falls die Matrix $\boldsymbol{A}$ durch φ erzeugt wird. Deshalb definieren wir analog *vereinfachte Strang–Typ–Vorkonditionierer* für die Matrix $\boldsymbol{A}'\boldsymbol{A}$ durch

$$\boldsymbol{M}_N(|\varphi|^2, \boldsymbol{S}_N^{II}) := (\boldsymbol{S}_N^{II})' \operatorname{diag}\left(|\varphi(\frac{j\pi}{N})|^2\right)_{j=1}^{N} \boldsymbol{S}_N^{II} ,$$

$$\boldsymbol{M}_N(|\varphi|^2, \boldsymbol{C}_N^{II}) := (\boldsymbol{C}_N^{II})' \operatorname{diag}\left(|\varphi(\frac{j\pi}{N})|^2\right)_{j=0}^{N-1} \boldsymbol{C}_N^{II} .$$

In Abschnitt 3.4.3 werden wir sehen, daß diese Vorkonditionierer die besten numerischen Ergebnisse liefern.

3.4.2 Optimale trigonometrische Vorkonditionierer

Die Konstruktion des optimalen Vorkonditionierers für beliebige Matrizen $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ haben wir ausführlich in Abschnitt 3.2 beschrieben. Der Zugang soll nun auf eine Normalgleichung für nichtsymmetrische Toeplitz–Matrizen (3.35) angewendet werden.
Um den optimalen Vorkonditionierer zu erhalten, müssen wir die Diagonalmatrix diag $\boldsymbol{d}$ in (3.14) berechnen. Dies ist mit einer DST–I$(N-1)$ aus den Daten $\boldsymbol{\alpha}$ möglich (siehe (3.15) und (3.16)). Lemma 3.9 zeigt, wie aus dem Vektor $\boldsymbol{\beta}^I$ der Vektor $\boldsymbol{\beta}^{II}$ bestimmt werden kann und mit der Formel (3.13) der optimale Vorkonditionierer. Aus diesem Grund brauchen wir im folgenden Satz nur die Berechnung der Daten

$$\boldsymbol{\beta}^I = (\langle \boldsymbol{A}_N' \boldsymbol{A}_N, \boldsymbol{B}_j^I \rangle)_{j=0}^{N-1} \tag{3.44}$$

anzugeben.

Wir beschränken uns wieder auf Vorkonditionierer bezüglich $\boldsymbol{C}_N^{II}$. Die Konstruktion wird aber so dargestellt, daß mit Hilfe der Bemerkung 3.11 die anderen trigonometrischen Vorkonditionierer leicht berechnet werden können.
Wir betrachten die Summanden auf der rechten Seite von (1.15). Die Matrix $\boldsymbol{T}_N$ ist eine symmetrische Toeplitz–Matrix. Die Matrizen $\boldsymbol{L}_N'\boldsymbol{L}_N$ und $\boldsymbol{U}_N'\boldsymbol{U}_N$ sind keine

Toeplitz–Matrizen. Für $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ führen wir die Vektoren $\boldsymbol{s}(\boldsymbol{A}_N) := (s_k(\boldsymbol{A}_N))_{k=0}^{N-1}$, $\boldsymbol{h}(\boldsymbol{A}_N) := (h_k(\boldsymbol{A}_N))_{k=0}^{N-1}$ und $\tilde{\boldsymbol{h}}(\boldsymbol{A}_N) := (\tilde{h}_k(\boldsymbol{A}_N))_{k=0}^{N-1}$ durch

$$\begin{aligned}
s_k(\boldsymbol{A}_N) &:= \langle \boldsymbol{A}_N, \operatorname{stoep} \boldsymbol{e}_k' \rangle \quad (k = 0, \dots, N-1)\,, \\
h_k(\boldsymbol{A}_N) &:= \langle \boldsymbol{A}_N, \operatorname{hank}(\boldsymbol{e}_k', \boldsymbol{o}_N') \rangle \quad (k = 0, \dots, N-2)\,, \\
\tilde{h}_k(\boldsymbol{A}_N) &:= \langle \boldsymbol{A}_N, \operatorname{hank}(\boldsymbol{o}_N', \boldsymbol{e}_k') \rangle \quad (k = 0, \dots, N-2)
\end{aligned}$$

ein und definieren $h_{-1} = \tilde{h}_{-1} := 0$. Es folgt mit (3.11), daß

$$\langle \boldsymbol{A}_N, \boldsymbol{B}_k^I \rangle = s_k(\boldsymbol{A}_N) + h_{k-1}(\boldsymbol{A}_N) + \tilde{h}_{k-1}(\boldsymbol{A}_N) \quad (k = 0, \dots, N-1)\,. \tag{3.45}$$

Durch Addition bzw. Subtraktion der Daten $s_k(\boldsymbol{A}_N)$, $h_k(\boldsymbol{A}_N)$, $\tilde{h}_k(\boldsymbol{A}_N)$ können wir leicht unter Beachtung von Tabelle 3.1 die Skalarprodukte bezüglich der Basen der anderen trigonometrischen Vorkonditionierer berechnen.

Lemma 3.36 *Sei* $\boldsymbol{T}_N := \operatorname{stoep}(t_0, t_1, \dots, t_{N-1})$. *Dann gilt*

$$\begin{aligned}
s_k(\boldsymbol{T}_N) &= 2(N-k)t_k \quad (k = 0, \dots, N-1)\,, \\
h_0(\boldsymbol{T}_N) &= t_0\,,\ h_1(\boldsymbol{T}_N) = 2\,t_1\,, \\
h_k(\boldsymbol{T}_N) &= 2\,t_k + h_{k-2}(\boldsymbol{T}_N) \quad (k = 2, \dots, N-2)\,, \\
\tilde{h}_k(\boldsymbol{T}_N) &= h_k(\boldsymbol{T}_N) \quad (k = 0, \dots, N-2)\,.
\end{aligned}$$

Beweis: Da $\boldsymbol{T}_N$ eine symmetrische Toeplitz–Matrix ist, folgt die Behauptung aus der Definition. ∎

Lemma 3.37 *Sei* $\boldsymbol{A}_N := \boldsymbol{U}_N' \boldsymbol{U}_N$, *wobei* $\boldsymbol{U}_N = \operatorname{toep}(\boldsymbol{r}', \boldsymbol{o}_N')$ *eine obere Toeplitz–Dreiecksmatrix mit* $\boldsymbol{r} = (r_k)_{k=0}^{N-1}$ *und* $r_0 = 0$ *ist. Dann gilt*

$$\boldsymbol{s}(\boldsymbol{A}_N) = 2\ \operatorname{hank}((0, (N-1)r_1, (N-2)r_2, \dots, r_{N-1})', (0, \dots, 0, r_{N-1})')\,\boldsymbol{r}\,, \tag{3.46}$$

$$\begin{aligned}
h_0(\boldsymbol{A}_N) &= x_0\,,\ h_1(\boldsymbol{A}_N) = x_1\,, \\
h_k(\boldsymbol{A}_N) &= h_{k-2}(\boldsymbol{A}_N) + x_k \quad (k = 2, \dots, N-2)\,, \qquad (3.47) \\
\tilde{h}_0(\boldsymbol{A}_N) &= y_0/2\,,\ \tilde{h}_1(\boldsymbol{A}_N) = y_1\,, \\
\tilde{h}_k(\boldsymbol{A}_N) &= \tilde{h}_{k-2}(\boldsymbol{A}_N) + y_k \quad (k = 2, \dots, N-2)\,, \qquad (3.48)
\end{aligned}$$

wobei

$$\begin{aligned}
\boldsymbol{x} &:= \boldsymbol{U}_N' \boldsymbol{r}\,, \\
\boldsymbol{y} &:= \operatorname{hank}(2(r_0, r_1, \dots, r_{N-1}), (-r_2, -r_3, \dots, -r_{N-1}, 0, 2r_{N-1}))\,\boldsymbol{r}
\end{aligned}$$

ist.

Beweis: Weil $\boldsymbol{U}_N$ eine obere Toeplitz–Dreiecksmatrix ist, ergibt sich

$$a_{j,k} = a_{j-1,k-1} + r_j r_k \quad (j,k = 1\ldots, N-1)\,. \tag{3.49}$$

Somit erhalten wir (3.46) wegen

$$s_k = 2\sum_{j=0}^{N-k-1} a_{j+k,j} = \sum_{j=0}^{N-1-k} (N-j-k) r_{j+k} r_j\,.$$

Die Rekursionen (3.47) und (3.48) folgen durch einfache Rechnung aus (3.49). ■

Lemma 3.38 *Sei* $\boldsymbol{B}_N := \boldsymbol{L}'_N \boldsymbol{L}_N$ *und sei* $\boldsymbol{J}_N = \operatorname{shank}(\boldsymbol{e}'_{N-1})$ *die Gegenidentität. Dann gilt*

$$\begin{aligned}
s_k(\boldsymbol{B}_N) &= s_k(\boldsymbol{J}_N\,\boldsymbol{B}_N\,\boldsymbol{J}_N) \quad (k=0,\ldots,N-1)\,,\\
h_k(\boldsymbol{B}_N) &= \tilde{h}_k(\boldsymbol{J}_N\,\boldsymbol{B}_N\,\boldsymbol{J}_N) \quad (k=0,\ldots,N-2)\,,\\
\tilde{h}_k(\boldsymbol{B}_N) &= h_k(\boldsymbol{J}_N\,\boldsymbol{B}_N\,\boldsymbol{J}_N) \quad (k=0,\ldots,N-2)\,,
\end{aligned}$$

so daß $\boldsymbol{s}(\boldsymbol{B}_N), \boldsymbol{h}(\boldsymbol{B}_N)$ *und* $\tilde{\boldsymbol{h}}(\boldsymbol{B}_N)$ *mit Hilfe von* (3.46) – (3.48) *berechnet werden können.*

Beweis: Die Relationen für $\boldsymbol{s}(\boldsymbol{B}_N), \boldsymbol{h}(\boldsymbol{B}_N)$ und $\tilde{\boldsymbol{h}}(\boldsymbol{B}_N)$ folgen aus der Definition von $\boldsymbol{J}_N$. Da

$$\boldsymbol{J}_N\,\boldsymbol{L}'_N\,\boldsymbol{L}_N\,\boldsymbol{J}_N = (\boldsymbol{J}_N\,\boldsymbol{L}_N\,\boldsymbol{J}_N)'\,(\boldsymbol{J}_N\,\boldsymbol{L}_N\,\boldsymbol{J}_N)$$

und die Matrix

$$\boldsymbol{J}_N\,\boldsymbol{L}_N\,\boldsymbol{J}_N = \operatorname{toep}((t_0,\ldots,t_{N-1}),(t_0,0,\ldots,0))$$

eine obere Toeplitz–Dreiecksmatrix ist, können wir $\boldsymbol{s}(\boldsymbol{B}_N), \boldsymbol{h}(\boldsymbol{B}_N)$ und $\tilde{\boldsymbol{h}}(\boldsymbol{B}_N)$ durch (3.46) – (3.48) mit $\boldsymbol{A}_N = \boldsymbol{J}_N\,\boldsymbol{B}_N\,\boldsymbol{J}_N$ and $\boldsymbol{r}' = (0, a_{M-N+1}, a_{M-N+2}, \ldots, a_{M-1})$ berechnen. ■

Satz 3.39 *Sei* $\boldsymbol{A} := (a_{j-k})_{j,k=0}^{M-1,N-1} \in \mathbb{R}^{M,N}$ *gegeben. Die Einträge der Matrix* (1.20) *seien vorberechnet. Dann kann der optimale Vorkonditionierer* $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{A}'\boldsymbol{A}, \boldsymbol{C}_N^{II}) \in \mathcal{A}_{\boldsymbol{C}_N^{II}}$ *von* $\boldsymbol{A}'\boldsymbol{A}$ *in* $\mathcal{O}(N\log N)$ *arithmetischen Operationen konstruiert werden.*

Beweis: Wir berechnen den Vektor $\boldsymbol{\beta}^I$ durch (3.44) – (3.45) und mit Lemmata 3.36 – 3.38. Wir beachten, daß die Multiplikation einer Toeplitz–Matrix (siehe Lemma 1.7) oder analog einer Hankel–Matrix mit einem Vektor nur $\mathcal{O}(N\log N)$ Operationen benötigt. Somit erhalten wir den Vektor $\boldsymbol{\beta}^I$ in $\mathcal{O}(N \log N)$ Operationen. Aus dem Vektor $\boldsymbol{\beta}^I$ berechnen wir den Vektor $\boldsymbol{\beta}^{II}$ mit Hilfe von (3.18) in $\mathcal{O}(N)$ Additionen. Nach Lemma 3.8 ergibt sich $\boldsymbol{\alpha} := \boldsymbol{G}^{-1}\boldsymbol{\beta}^{II}$ durch $\mathcal{O}(N)$ Operationen. Schließlich benötigt eine DST–I (siehe (3.16)) der Daten $\boldsymbol{\alpha}$ noch $\mathcal{O}(N \log N)$ Operationen, um die Einträge der Diagonalmatrix (3.14) zu erhalten. Es sind somit nur $\mathcal{O}(N \log N)$ Operationen nötig, um den optimalen Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$ zu berechnen. ■

Folgerung 3.40 *Falls eine Folge von Toeplitz–Matrizen* $\boldsymbol{A} \in \mathbb{R}^{M,N}$ *von einer Funktion* φ *mit* $|\varphi| > 0$ *und* (3.43) *erzeugt wird, so erfüllt der optimale trigonometrische Vorkonditionierer* $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{A}'\boldsymbol{A}, \boldsymbol{C}_N^{II})$ *die Eigenschaften* (E1) – (E4). *Ferner gilt dies auch für* $\varphi \in C_{2\pi}$ *mit* $|\varphi| > 0$.

Beweis: Die Eigenschaft (E1) ist nach Lemma 3.7 erfüllt. Aus den Einträgen der Toeplitz–Matrix $\boldsymbol{A}$ können wir nach Satz 3.39 den Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{A}'\boldsymbol{A}, \boldsymbol{C}_N^{II})$ in $\mathcal{O}(M \log M)$ Operationen berechnen. Damit sind die Eigenschaften (E2) – (E3) erfüllt. Um die Clusterung zu zeigen, gehen wir völlig analog wie in Satz 3.25 vor. Ähnliches Vorgehen wie in Folgerung 3.18 zeigt, daß diese Aussagen auch für $f \in C(I)$ gültig sind. ■

Bemerkung 3.41 Es gibt einige Arbeiten (z.B. [36]), die einen zirkulanten Vorkonditionierer $\boldsymbol{M}_N$ für komplexe nichthermitesche Toeplitz–Matrizen $\boldsymbol{A}_N \in \mathbb{C}^{N,N}$ berechnen. Unter bestimmten Voraussetzungen an die erzeugende Funktion von $\boldsymbol{A}_N$ wurde superlineare Konvergenz des CG–Verfahrens für

$$(\boldsymbol{M}_N^{-1}\boldsymbol{A}_N)^*(\boldsymbol{M}_N^{-1}\boldsymbol{A}_N)\boldsymbol{x}_N = (\boldsymbol{M}_N^{-1}\boldsymbol{A}_N)^*\boldsymbol{M}_N^{-1}\boldsymbol{b}_N \tag{3.50}$$

bewiesen. Nutzen wir trigonometrischen Vorkonditionierer für nichtsymmetrische Matrizen $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$, so führt dieses Vorgehen zu keiner Clusterung der Eigenwerte. Dies kann man leicht am folgenden Beispiel einsehen. Falls $\boldsymbol{A}_N = (a_{j-k})_{j,k=0}^{N-1}$ mit $a_0 = 1$ und $a_k = -a_{-k}$ $(k = 1, \ldots, N-1)$, dann gilt $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{A}_N, \boldsymbol{O}_N) = \boldsymbol{I}_N$, d.h., wir haben keine Vorkonditionierung. Ist z. B. $a_{-1} = -a_1 = 2$ und $a_k = 0$ $(|k| > 1)$, dann erfüllen die Matrizen $\boldsymbol{A}_{2N+1}$ $(N \in \mathbb{N})$ die Voraussetzungen von Folgerung 3.35 oder Folgerung 3.40. Die Eigenwerte von $\boldsymbol{A}'_{2N+1}\boldsymbol{A}_{2N+1}$ sind aber durch $9 - 8\cos(j\pi/(N+1))$ $(j = 0, \ldots, N)$ gegeben. □

3.4.3 Numerische Beispiele

In diesem Abschnitt wollen wir die Strang–Typ–Vorkonditionierer $\boldsymbol{M}_N^I$ sowie $\boldsymbol{M}_N^{II}$ und die optimalen trigonometrischen Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}}$ für die Normalgleichung (3.35) an verschiedenen Beispielen testen. Wir benutzen nur die Vorkonditionierer bezüglich DCT–II und DST–II, weil diese im symmetrischen Fall die besten Ergebnisse liefern. Um die Anzahl der diskreten trigonometrischen Transformationen in jedem PCG–Schritt zu reduzieren, gehen wir ähnlich wie in Abschnitt 3.3.3 vor. Wir zerlegen die Toeplitz–Matrix mit Hilfe der diskreten trigonometrischen Transformation, welche den Vorkonditionierer diagonalisiert (siehe (1.13)). Analoges Vorgehen wie in Abschnitt 3.3.3 zeigt, daß ein Schritt im PCG–Verfahren nur N Multiplikationen mehr als ein Schritt im CG–Verfahren benötigt. Für einen Schritt benötigen wir 12 diskrete trigonometrische Transformationen für allgemeine rechteckige Toeplitz–Matrizen (siehe Algorithmus 1.9) und 8 diskrete trigonometrische Transformationen im Fall $M = N$ (siehe Lemma 1.7). Als Transformationslänge wählen wir $N = 2^n$. Die rechte Seite $\boldsymbol{b}$ von (3.35) ist der Vektor, der M Einsen enthält. Das PCG–Verfahren starten wir mit dem Nullvektor. Wir brechen die Iteration ab, wenn $\|\boldsymbol{r}^{(j)}\|_2/\|\boldsymbol{r}^{(0)}\|_2 < 10^{-7}$, wobei $\boldsymbol{r}^{(j)}$ den Residuumvektor der Normalgleichung nach j Iterationen bezeichnet. Wir vergleichen

1. den optimalen Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}} = \boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{A}'\boldsymbol{A}, \boldsymbol{O}_N)$,
2. den Strang–Typ–Vorkonditionierer $\boldsymbol{M}_N^{I} = \boldsymbol{M}_N^{\mathcal{O}}(\tilde{\boldsymbol{T}}_N, \boldsymbol{O}_N)$, wobei $\tilde{\boldsymbol{T}}_N$ in Lemma 3.32 angegeben ist, und
3. den Strang–Typ–Vorkonditionierer $\boldsymbol{M}_N^{II} = \boldsymbol{M}_N^{\mathcal{S}}(\tilde{\boldsymbol{T}}_N, \boldsymbol{O}_N)$.

Um diese neuen Vorkonditionierer mit den bekannten zirkulanten Vorkonditionierern zu vergleichen, wählen wir die gleichen Beispiele wie in den entsprechenden Arbeiten.

Beispiel (x): (siehe [65] und Tabelle 3.11)

$$a_k := \begin{cases} 1/\log(2-k) & k \le -1 \ , \\ 1/\log(2)+1 & k = 0 \ , \\ 1/(1+k) & k \ge 1 \ . \end{cases}$$

Beispiel (xi): (siehe [65] und Tabelle 3.12)

$$a_k := \begin{cases} 2 & k = 0 \ , \\ -0.7\ a_{k+1} & k \le -1 \ , \\ 0.9\ a_{k-1} & k \ge 1 \ . \end{cases}$$

Wir wollen jetzt die Kleinste–Quadrate–Lösung von rechteckigen Toeplitz–Matrizen mit Vollrang ermitteln. Dafür wurden zirkulante Vorkonditionierer in [71, 24, 25, 30] entwickelt.

Beispiel (xii): (siehe [25, 30] und Tabelle 3.13)

$$a_k := \begin{cases} \mathrm{e}^{-0.1(k-1)^2} & k = -N+1, \dots, 0 \ , \\ \mathrm{e}^{-0.1k^2} & k = 1, \dots, M-1 \ . \end{cases}$$

Beispiel (xiii): (siehe [25] und Tabelle 3.14)

$$a_k := \begin{cases} 0 & k = -N+1, \dots, 0 \ , \\ 1/(2(w+2)) & k = 0, \dots, w-1 \ , \\ 0 & k = w, \dots, M = k+w-2 \ . \end{cases}$$

In allen Fällen lieferte ein trigonometrischer Vorkonditionierer in nicht mehr Schritten ein gleich gutes Ergebnis wie die bekannten Vorkonditionierer [71, 24, 25, 30]. Da die arithmetische Komplexität im trigonometrischen Fall geringer ist (siehe Bemerkung 1.6 und Gleichung (3.33)), sind diese neuen Vorkonditionierer für reelle Toeplitz–Matrizen vorzuziehen.

Beispiel (xiv): Im folgenden wenden wir unsere Methode zur Lösung eines *Anfangswertproblems* an. In [16] wird die sinc–Funktion

$$\operatorname{sinc} t := \begin{cases} \dfrac{\sin(\pi t)}{\pi t} & t \neq 0 \,, \\ 1 & t = 0 \end{cases}$$

und deren Translate benutzt, um das Anfangswertproblem

$$u'(t) = f(t, u(t)) \,, \quad \lim_{t \to -\infty} u(t) = 0$$

zu lösen. Dazu suchen wir eine Näherungslösung u_M mittels Kollokationsverfahren in der Form

$$u_M(t) = \sum_{k=0}^{2M-1} x_k \operatorname{sinc} \left((2M)^{1/2} t - (k - M)\right) \,.$$

Eine kurze Rechnung in [16] zeigt, daß dieses Problem auf das lineare Gleichungssystem

$$\left(\frac{1}{2M}\right)^{1/2} \boldsymbol{T}_{2M}\, \boldsymbol{x}_{2M} = \boldsymbol{b}_{2M} \tag{3.51}$$

mit

$$\boldsymbol{T}_{2M} = \operatorname{atoep}\left(0, -1, \frac{1}{2}, -\frac{1}{3}, \frac{1}{4} \ldots, -\frac{1}{2M-1}\right)$$

führt. Wir wollen hier wie in [16], Beispiel 2.9 vorgehen. Die Funktion

$$u(t) = (\cosh(\pi t))^{-1}$$

ist die eindeutige Lösung des Problems

$$u'(t) \;=\; -\pi \sinh(\pi t) u(t)^2 \qquad \lim_{t \to -\infty} u(t) = 0 \,.$$

Die Bestimmung der Koeffizienten x_k $(k = 0, \ldots, 2M-1)$ erfolgt durch Lösung des linearen Gleichungssystems (3.51). Die rechte Seite $\boldsymbol{b}_{2M}$ ist durch Abtasten der Funktion $g(x) := -\frac{\pi \sinh(\pi x)}{\cosh^2(\pi x)}$ an den Kollokationspunkten $(k-M)(2M)^{-1/2}$ $(k = 0, \ldots, 2M-1)$ gegeben. Wir starten das PCG–Verfahren mit dem Nullvektor und brechen die Iteration ab, wenn $\|\boldsymbol{r}^{(j)}\|_2 / \|\boldsymbol{r}^{(0)}\|_2 < 10^{-7}$ ist, wobei $\boldsymbol{r}^{(j)}$ den Residuumvektor nach j Iterationen bezeichnet. Die Anzahl der Iterationen bezüglich der verschiedenen Vorkonditionierer geben wir in Tabelle 3.15 an. Da $\varphi(x) = \mathrm{i}x$ die erzeugende Funktion für die Folge der antisymmetrischen Toeplitz–Matrizen $\boldsymbol{T}_{2M}$ ist, können wir auch den vereinfachten Strang–Typ–Vorkonditionierer $\boldsymbol{M}_N(|\varphi|^2, \boldsymbol{S}_N^{II})$ berechnen. In der letzten Spalte geben wir den maximalen Fehler an den Kollokationspunkten zwischen der exakten Lösung $u(x)$ und der numerischen Lösung an (siehe [16]).

Diese Ergebnisse zeigen, daß wir das Problem sehr effizient mit dem PCG–Verfahren lösen können, wenn wir die Vorkonditionierer $\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$, $\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$ oder $\boldsymbol{M}_N(|\varphi|^2, \boldsymbol{S}_N^{II})$ wählen.

Beispiel (xv): In diesem Beispiel wollen wir ein Kleinstes–Quadrate–Problem lösen, wobei die rechteckige Toeplitz–Matrix sehr schlecht konditioniert ist. Solche Beispiele treten in vielen Anwendungen auf, so z.B. in der Signal– und Bildrekonstruktion. Da die Matrix schlecht konditioniert ist, ist die Lösung extrem instabil bezüglich kleiner Störungen der rechten Seite. Die Kleinste–Quadrate–Lösung ist deshalb oft nutzlos. Die *Methode der Regularisierung* kann benutzt werden, um Stabilität zu erhalten. Die regularisierte Lösung $\boldsymbol{x} = \boldsymbol{x}(\mu)$ wird berechnet als

$$\min \| \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{o}_N \end{pmatrix} - \begin{pmatrix} \boldsymbol{T} \\ \mu\, \boldsymbol{L} \end{pmatrix} \boldsymbol{x} \|_2 \,,$$

wobei $\mu > 0$ der Regularisationsparameter und $\boldsymbol{L}$ eine Regularisationsmatrix ist. Um eine „glatte“ Lösung zu erhalten, wird $\boldsymbol{L}$ als eine Differenzenmatrix gewählt. Dies bedeutet, daß man von der Lösung fordert, daß sie eine „kleine k–te Ableitung“ besitzt. Ein ähnliches Beispiel wurde in [25] betrachtet. Dort wurden wieder zirkulante Vorkonditionierer verwendet.
Im folgenden wählen wir $\boldsymbol{L} := \mathrm{stoep}(2, -1, \boldsymbol{o}'_{N-2})$ und $\boldsymbol{T} := (\exp(-(k-l)^2/10))_{k,l=0}^{N-1}$. Der Regularisationsparameter sei $\mu := 0.01$. Wir erhalten die Normalgleichung

$$(\boldsymbol{T}'\boldsymbol{T} \;+\; \mu^2 \boldsymbol{L}'\boldsymbol{L})\boldsymbol{x} \;= \boldsymbol{T}'\boldsymbol{b}$$

und konstruieren für die Matrix $\boldsymbol{T}'\boldsymbol{T} \;+\; \mu^2 \boldsymbol{L}'\boldsymbol{L}$ einen Vorkonditionierer.

Die numerischen Ergebnisse zeigen, daß wir mit den Vorkonditionieren $\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$ und $\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$ überraschend gute Ergebnisse erhalten. Dieses wichtige Beispiel wurde mit einem Multigrid–Verfahren in [19] untersucht. Vergleichende Rechnungen der verschiedenen Verfahren müssen noch durchgeführt werden.
Beispiel (xvi): Wir rechnen das gleiche Beispiel wie oben, nur wählen wir jetzt $\boldsymbol{T} := (\exp(-(k-l)^2/100))_{k,l=0}^{N-1}$. Die Anzahl der Iterationen bezüglich der verschiedenen Vorkonditionierer geben wir in Tabelle 3.17 an.

Beispiel (xvii): Wir betrachten die Folge von Toeplitz–Matrizen, die von der Funktion $\varphi(t) := t^2\, \mathrm{e}^{\mathrm{i}t} \quad (t \in [-\pi, \pi])$ erzeugt wird. Die numerischen Experimente in Tabelle 3.18 zeigen, daß der Vorkonditionierer $\boldsymbol{M}_N^{II}$ nicht positiv definit ist. Die Vorkonditionierer bezüglich der DST–II liefern bessere Ergebnisse als die Vorkonditionierer bezüglich der DCT–II. In der letzten Spalte haben wir die Anzahl der Iterationen bezüglich des vereinfachten Strang–Typ–Vorkonditionierers eingetragen. Dieser liefert die besten Ergebnisse. Der Vorkonditionierer $\boldsymbol{M}_N(|\varphi|^2, \boldsymbol{C}_N^{II})$ ist singulär, da $\varphi(0) = 0$ ist.

Beispiel (xviii): Wir betrachten abschließend noch ein Folge von symmetrischen Matrizen $\boldsymbol{T}_N$, die von der Funktion $\varphi(t) = t^2 - 2 \ (t \in [-\pi, \pi])$ erzeugt wird. In diesem Fall sind die Toeplitz–Matrizen $\boldsymbol{T}_N$ symmetrisch und nicht positiv definit. Somit sind die Voraussetzungen von Folgerung 3.18 und Folgerung 3.26 nicht erfüllt. Wir geben die Ergebnisse dennoch in Tabelle 3.19 an. Natürlich sind auch die Vorkonditionierer nicht positiv definit. In Tabelle 3.20 geben wir die Anzahl der Iterationen an, wenn wir die Normalgleichung lösen (siehe Bemerkung 3.27).

Anhang

		Strang–Typ–V.				Optimaler trig. V.				Superoptimaler trig. V.			
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$
7	8	3	3	5	5	5	5	5	5	5	5	5	5
8	8	3	3	5	5	5	4	5	5	5	5	5	5
9	8	3	3	5	5	4	4	5	5	4	4	5	5
10	8	3	3	5	5	4	4	5	5	4	4	5	5
11	8	3	3	5	5	4	4	5	5	4	4	5	5
12	8	3	3	5	5	4	4	5	5	4	4	5	5
13	8	3	3	5	5	4	4	5	5	4	4	5	5

Tabelle 3.2: *Beispiel* (i): $f(x) = x^4 + 1 \quad (x \in [-1, 1])$

		Strang–Typ–V.				Optimaler trig. V.				Superoptimaler trig. V.			
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$
7	19	5	5	7	7	6	6	7	7	5	6	7	7
8	21	5	5	7	7	6	6	7	7	6	6	7	7
9	24	5	5	7	7	6	6	7	7	6	6	7	7
10	26	5	5	8	8	6	6	8	8	6	6	8	8
15	33	5	5	8	8	6	6	8	8	6	6	8	8

Tabelle 3.3: *Beispiel* (ii): $t_k = 1/(k+1), \quad (k = 0, \ldots, 2^n - 1)$

		Strang–Typ–V.				Optimaler trig. V.				Superoptimaler trig. V.			
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$
6	37	5	5	7	7	7	6	8	8	22	6	19	19
7	55	5	5	7	7	7	5	8	8	18	6	14	14
8	67	5	5	7	7	6	5	7	7	13	5	11	11
9	70	5	5	7	7	6	5	7	7	10	5	9	9
10	71	5	5	7	7	5	5	7	7	8	5	8	8
15	71	5	5	7	7	5	5	7	7	5	5	7	7

Tabelle 3.4: *Beispiel* (iii): $\varphi(t) = t^4 + 1 \quad (t \in [-\pi, \pi])$

		Strang–Typ–V.				Optimaler trig. V.			
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$
5	16	$*$	4	8	8	9	6	11	11
6	36	$*$	4	8	8	12	6	14	14
7	75	$*$	4	8	8	15	6	16	16
8	144	$*$	4	8	8	18	5	21	21
9	279	$*$	4	8	8	24	5	25	25
10	545	$*$	4	8	8	30	5	34	34

Tabelle 3.5: *Beispiel* (iv): $f(x) = (x^2 + 1)(1 - x) \quad (x \in [-1, 1])$

		Strang–Typ–V.				Optimaler trig. V.			
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$
5	16	4	$*$	8	8	5	8	11	11
6	34	4	$*$	8	8	5	9	14	14
7	69	4	$*$	8	8	5	12	16	16
8	136	4	$*$	8	8	5	15	19	19
9	265	4	$*$	8	8	5	17	24	24
10	524	4	$*$	9	9	4	23	31	31

Tabelle 3.6: *Beispiel* (v): $f(x) = (x^2 + 1)(1 + x) \quad (x \in [-1, 1])$

		Strang–Typ–V.				Optimaler trig. V.			
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$
5	21	5	$*$	9	9	7	15	22	22
6	59	5	$*$	9	9	8	22	27	27
7	187	5	$*$	10	10	9	34	46	46
8	658	5	$*$	11	10	10	54	73	74
9	> 1000	6	$*$	9	10	14	122	134	141
10	> 1000	5	$*$	10	12	16	305	297	305

Tabelle 3.7: *Beispiel* (vi): $f(x) = (x^2+1)(1+x)^2 \quad (x \in [-1,1])$

		Strang–Typ–V.				Optimaler trig. V.				
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{M}_N(f, \boldsymbol{S}_N^{II})$
5	25	9^*	8^*	13^*	13^*	17	10	20	20	5
6	69	9^*	8^*	13^*	13^*	21	11	26	26	5
7	190	10^*	10^*	15^*	15^*	26	14	27	27	7
8	457	10^*	10^*	14^*	14	33	16	41	41	8
9	> 1000	11	9	15^*	14^*	43	19	46	46	9
10	> 1000	10^*	10^*	15^*	15^*	59	24	52	51	7
11	> 1000	10^*	10^*	16^*	16^*	79	30	67	66	7

Tabelle 3.8: *Beispiel* (vii): $\varphi(t) = (t^2-1)^2 \quad (t \in [-\pi,\pi])$

		Strang–Typ–V.				Optimaler trig. V.				
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{M}_N(f, \boldsymbol{S}_N^{II})$
5	33	12^*	10^*	18^*	15^*	18	10	27	27	6
6	116	18^*	15^*	20^*	20^*	30	13	44	44	7
7	487	27^*	21^*	26^*	27^*	54	16	89	87	8
8	>1000	40^*	33^*	37^*	37^*	155	19	199	200	9
9	>1000	115^*	63^*	46^*	45^*	376	25	403	411	9
10	>1000	218^*	165^*	86^*	67^*	>1000	32	>1000	>1000	10
11	>1000	761^*	533^*	132^*	134^*	>1000	48	>1000	>1000	11

Tabelle 3.9: *Beispiel* (viii): $\varphi(t) = t^4 \quad (t \in [-\pi,\pi])$

N	$\boldsymbol{M}_N(\varphi, \boldsymbol{S}_N^{II} \otimes \boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N(\psi, \boldsymbol{S}_N^{II} \otimes \boldsymbol{S}_N^{II})$
8	13	9
16	16	12
32	22	14
64	29	19
128	36	25
256	43	35
512	52	49

Tabelle 3.10: *Beispiel* (ix): $\varphi(s,t) = s^2\, t^4$ und $\psi(s,t) = (s^2+t^2)^2$ $(s,t \in [-\pi,\pi])$

n	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$
7	24	8	15	7	12	13	13
8	32	8	17	7	13	15	14
9	43	8	19	8	13	15	15
10	57	9	20	9	14	17	16
11	86	9	20	9	16	21	19
12	121	9	22	9	16	24	21
13	176	9	22	10	17	24	21

Tabelle 3.11: *Beispiel* (x)

n	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$
5	20	8	15	9	9	8	8
6	27	9	13	9	8	6	6
7	34	9	12	8	8	5	5
8	43	8	11	8	7	5	5
9	53	7	10	7	7	5	5
10	59	7	9	7	7	5	5
11	50	6	9	7	7	5	5

Tabelle 3.12: *Beispiel* (xi)

n	M	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$
4	32	22	9	20	17*	22*	19*	20*
5	64	46	10	29	31*	32*	29*	34*
6	128	75	9	28	55*	60*	58*	61*
7	256	125	8	23	96*	102*	101*	106*

Tabelle 3.13: *Beispiel* (xii)

n	M	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$
4	23	10	2	10	2	7	8*	10*
5	47	21	2	12	2	7	17*	20*
6	95	41	2	16	2	8	30	34*
7	191	66	2	12	2	9	103*	139*
8	383	123	2	13	2	9	275*	>300

Tabelle 3.14: *Beispiel* (xiii)

n	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N(\lvert\varphi\rvert^2, \boldsymbol{S}_N^{II})$	Fehler
6	36	21	12	23	9	37^*	37^*	5	1.63e-5
7	79	24	11	29	7	60^*	55^*	6	2.30e-7
8	169	31	6	38	6	97^*	89^*	5	2.95e-8
9	358	40	6	47	6	169^*	181^*	5	8.27e-9

Tabelle 3.15: *Beispiel* (xiv): Ein Anfangswertproblem

n	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$
6	58	18	26	22	31	8	9
7	122	15	39	19	42	8	9
8	272	13	39	15	46	8	9
9	477	11	31	13	40	8	9
10	651	10	23	10	30	8	9
11	>800	9	18	9	22	8	9
12	>800	8	18	9	22	8	9

Tabelle 3.16: *Beispiel* (xv): Anzahl der Iterationen für $\boldsymbol{T}'\boldsymbol{T} + 10^{-4}\boldsymbol{L}'\boldsymbol{L}$

n	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$
6	84	40	77	44	87	13	17
7	145	73	148	90	177	14	18
8	343	95	257	98	255	14	18
9	847	109	256	136	292	12	14
10	1836	85	226	103	227	12	14
11	>2000	63	181	79	237	12	14
12	>2000	49	163	61	193	13	15

Tabelle 3.17: *Beispiel* (xvi): Anzahl der Iterationen für $\boldsymbol{T}'\boldsymbol{T} + 10^{-4}\boldsymbol{L}'\boldsymbol{L}$

n	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N(\lvert\varphi\rvert^2, \boldsymbol{S}_N^{II})$
5	84	29	21	34	18	22^*	19^*	11
6	311	52	26	64	22	32^*	26^*	11
7	1226	116	33	139	27	56^*	44^*	14
8	5220	256	40	324	39	96^*	76^*	16
9	>10000	664	74	865	55	200^*	157^*	19
10	>10000	1758	101	2546	78	466^*	357^*	21

Tabelle 3.18: *Beispiel* (xvii): $\varphi(t) := t^2\, \mathrm{e}^{\mathrm{i}t} \quad (t \in [-\pi, \pi])$

		Strang–Typ–V.				Optimaler trig. V.				Superoptimaler trig. V.			
n	$\boldsymbol{I}_N$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$	$\boldsymbol{C}_N^{II}$	$\boldsymbol{S}_N^{II}$	$\boldsymbol{C}_N^{IV}$	$\boldsymbol{S}_N^{IV}$
5	16	5*	5*	7*	7*	10*	7*	10*	10*	11*	9*	16*	16*
6	37	5*	5*	6*	6*	10*	8*	10*	10*	14*	11*	19*	19*
7	80	5*	5*	7*	7*	10*	8*	10*	10*	16*	11*	23*	23*
8	172	4*	4*	6*	6*	9*	7*	9*	9*	19*	13*	27*	27*
9	359	5*	5*	6*	6*	10*	7*	9*	9*	25*	16*	34*	34*
10	720	4*	4*	6*	6*	9*	7*	9*	9*	32*	19*	45*	45*

Tabelle 3.19: *Beispiel* (xviii): $\varphi(t) = t^2 - 2 \quad (t \in [-\pi, \pi])$

n	$\boldsymbol{I}_N$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{\mathcal{O}}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{I}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{C}_N^{II})$	$\boldsymbol{M}_N^{II}(\boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N(\lvert\varphi\rvert^2, \boldsymbol{S}_N^{II})$	$\boldsymbol{M}_N(\lvert\varphi\rvert^2, \boldsymbol{C}_N^{II})$
5	22	14	15	14	11	10*	9*	6	7
6	56	17	17	19	13	10	10	7	8
7	136	18	18	21	15	9*	9*	7	8
8	311	22	20	27	19	10	10	8	8
9	670	30	25	36	23	11	11	8	8
10	1383	36	29	46	29	11	10	9	9

Tabelle 3.20: *Beispiel* (xviii): $\varphi(t) = t^2 - 2 \quad (t \in [-\pi, \pi])$

Literaturverzeichnis

[1] B. Alpert and V. Rokhlin. A fast algorithm for the evaluation of Legendre expansions. SIAM *J. Sci. Statist. Comput.*, 12:158 – 179, 1991.

[2] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, 1996.

[3] O. Axelsson and V. Barker. *Finite Element Solution of Boundary Value Problems, Theory and Computation.* Academic Press, Orlando, 1984.

[4] G. Baszenski. Programmpaket zur Berechnung diskreter trigonometrischer Transformationen. 1995. http://www.iuk.fh-dortmund.de/~baszenski/.

[5] G. Baszenski and M. Tasche. Fast polynomial multiplication and convolution related to the discrete cosine transform. *Linear Algebra Appl.*, 252:1 – 25, 1997.

[6] S. Belmehdi. On the associated orthogonal polynomials. *J. Comput. Appl. Math.*, 32:311 – 319, 1991.

[7] F. D. Benedetto. Analysis of preconditioning techniques for ill–conditioned Toeplitz matrices. SIAM *J. Sci. Comput.*, 16:682 – 697, 1995.

[8] F. D. Benedetto. Iterative solution of Toeplitz systems by preconditioning with the discrete sine transform. In SPIE 2563, San Diego, 1995.

[9] F. D. Benedetto and S. S. Capizzano. A unifying approach to abstract matrix algebra preconditioning. *Preprint*, 1997.

[10] F. D. Benedetto, G. Fiorentino, and S. Serra. C.G. preconditioning for Toeplitz matrices. *Comp. Math. Appl.*, 25:35 – 45, 1993.

[11] D. Berthold, W. Hoppe, and B. Silbermann. A fast algorithm for solving the generalized airfoil equation. *J. Comp. Appl. Math.*, 43:185 – 219, 1992.

[12] D. Bini and F. D. Benedetto. A new preconditioner for the parallel solution of positive definite Toeplitz systems. In *Proc. Second* ACM *Symp. on Parallel Algorithms and Architectures*, pages 220 – 223, Crete, 1990.

[13] D. Bini and P. Favati. On a matrix algebra related to discrete Hartley transform. SIAM *J. Matrix Anal. Appl.*, 14:500 – 507, 1993.

[14] E. Boman and I. Koltracht. Fast transform based preconditioners for Toeplitz equations. SIAM *J. Matrix Anal. Appl.*, 16:628 – 645, 1995.

[15] E. Bozzo and C. D. Fiore. On the use of certain matrix algebras associated with discrete trigonometric transforms in matrix displacement decomposition. SIAM *J. Matrix Anal. Appl.*, 16:312 – 326, 1995.

[16] T. S. Carlson, J. Dockery, and J. Lund. A sinc–collocation method for initial value problems. *Math. Comp.*, 66:215 – 235, 1997.

[17] R. H. Chan. Circulant preconditioners for hermitian Toeplitz systems. SIAM *J. Matrix Anal. Appl.*, 10:542 – 550, 1989.

[18] R. H. Chan. Toeplitz preconditioners for Toeplitz systems with nonnegative generating functions. *IMA J. Numer. Anal.*, 11:333 – 345, 1991.

[19] R. H. Chan, T. F. Chan, and W. Wan. Multigrid for differential–convolution problem arising from image processing. In G. Golub, S. Lui, F. Luk, and R. Plemmons, editors, *Proceedings of the Workshop on Scientific Computing*, Springer–Verlag, Hong Kong, 1997.

[20] R. H. Chan, T. F. Chan, and C. K. Wong. Cosine transform based preconditioners for total variation minimization problems in image processing. Technical report 95–8, Chinese University of Hong Kong, 1995.

[21] R. H. Chan and X.-Q. Jin. A family of block preconditioners for block systems. SIAM *J. Sci. Statist. Comput.*, 13:1218 – 1235, 1992.

[22] R. H. Chan, X.-Q. Jin, and M.-C. Yeung. The circulant operator in the Banach algebra of matrices. *Linear Algebra Appl.*, 149:41 – 53, 1991.

[23] R. H. Chan, X.-Q. Jin, and M.-C. Yeung. The spectra of super–optimal circulant preconditioned Toeplitz systems. SIAM *J. Numer. Anal.*, 28:871 – 879, 1991.

[24] R. H. Chan, J. Nagy, and R. Plemmons. Circulant preconditioned Toeplitz least square problems. SIAM *J. Matrix Anal. Appl.*, 15:80 – 97, 1994.

[25] R. H. Chan, J. Nagy, and R. Plemmons. Displacement preconditioners for Toeplitz least squares iterations. *Elec. Trans. Numer. Anal.*, 2:44 – 56, 1994.

[26] R. H. Chan and K.-P. Ng. Toeplitz preconditioners for hermitian Toeplitz systems. *Linear Algebra Appl.*, 190:181 – 208, 1993.

[27] R. H. Chan and M. K. Ng. Fast iterative solvers for Toeplitz–plus–band systems. SIAM *J. Sci. Comput.*, 14:1013 – 1019, 1993.

[28] R. H. Chan and M. K. Ng. Conjugate gradient methods of Toeplitz systems. SIAM *Review*, 38:427 – 482, 1996.

[29] R. H. Chan and M. K. Ng. Iterative methods for linear systems with matrix structure. In T. Kailath and A. Sayed, editors, *Fast Reliable Algorithms for Matrices with Structure*, SIAM, Philadelphia, 1998.

[30] R. H. Chan, M. K. Ng, and R. J. Plemmons. Generalization of Strang's preconditioner with applications to Toeplitz least square problems. *Numer. Linear Algebra Appl.*, 3:45 – 64, 1996.

[31] R. H. Chan, M. K. Ng, and C. K. Wong. Sine transform based preconditioners for symmetric Toeplitz systems. *Linear Algebra Appl.*, 232:237 – 259, 1996.

[32] R. H. Chan and G. Strang. Toeplitz systems by conjugate gradients with circulant preconditioner. SIAM *J. Sci. Statist. Comput.*, 10:104 – 119, 1989.

[33] R. H. Chan and T. P. Tang. Fast band–Toeplitz preconditioners for hermitian Toeplitz systems. SIAM *J. Sci. Statist. Comput.*, 15:164 – 171, 1994.

[34] R. H. Chan and M.-C. Yeung. Circulant preconditioners constructed from kernels. SIAM *J. Numer. Anal.*, 29:1093 – 1103, 1992.

[35] R. H. Chan and M.-C. Yeung. Circulant preconditioners for Toeplitz matrices with positive continuous generating functions. *Math. Comp.*, 58:233 – 240, 1992.

[36] R. H. Chan and M.-C. Yeung. Circulant preconditioners for complex Toeplitz systems. SIAM *J. Numer. Anal.*, 30:1193 – 1207, 1993.

[37] T. F. Chan. An optimal circulant preconditioner for Toeplitz systems. SIAM *J. Sci. Statist. Comput.*, 9:766 – 771, 1988.

[38] T. F. Chan and J. A. Olkin. Circulant preconditioner for Toeplitz–block systems. *Numer. Algorithms*, 6:89 – 101, 1994.

[39] T. Chihara. *An Introduction to Orthogonal Polynomials*. Gordon and Breach, New York, 1978.

[40] C. W. Clenshaw. A note on the summation of Chebyshev series. *Math. Comp.*, 9:118 – 120, 1955.

[41] P. Davis. *Circulant Matrices*. John Wiley and Sons, New York, 1979.

[42] P. Deuflhard and A. Hohmann. *Numerische Mathematik. Eine algorithmisch orientierte Einführung*. de Gruyter, Berlin, 1991.

[43] J. Driscoll and D. M. Healy. Computing Fourier transforms and convolutions on the 2–sphere. *Adv. Appl. Math.*, 15:202 – 240, 1994.

[44] J. Driscoll, D. M. Healy, and D. N. Rockmore. Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs. SIAM *J. Comput*, 26:1066 – 1099, 1996.

[45] J. O. Droese. Verfahren zur schnellen Fourier–Transformation mit nichtäquidistanten Knoten. Diplomarbeit, TH Darmstadt, 1996.

[46] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration and differentiation. Research Report, Yale University, New Haven, 1993.

[47] B. Fischer. *Polynomial Based Iteration Methods for Symmetric Linear Systems.* Wiley–Teubner, 1996.

[48] D. Funaro. *Polynomial Approximation of Differential Equations.* Springer–Verlag, Berlin, 1992.

[49] U. Grenander and G. Szegö. *Toeplitz Forms and Their Applications.* University of California Press, Los Angeles, 1958.

[50] D. M. Healy, S. S. B. Moore, and D. N. Rockmore. Efficiency and stability issues in the numerical computation of Fourier transforms and convolutions on the 2-Sphere. Technical report, Dartmouth College, Hanover, 1994.

[51] G. Heinig and A. Bojanczyk. Transformation techniques for Toeplitz and Toeplitz–plus–Hankel matrices. *Linear Algebra Appl.* in print.

[52] G. Heinig and K. Rost. Representations of Toeplitz–plus–Hankel matrices using trigonometric transforms with application to fast matrix–vector multiplication. *Preprint*, 1996.

[53] M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409 – 436, 1952.

[54] S. Holmgren and K. Otto. A framework for polynomial preconditioners based on fast transforms I: Theory. *Preprint*, 1997.

[55] S. Holmgren and K. Otto. A framework for polynomial preconditioners based on fast transforms II: PDE applications. *Preprint*, 1997.

[56] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1985.

[57] T. Huckle. Circulant and skew–circulant matrices for solving Toeplitz matrix problems. SIAM *J. Matrix Anal. Appl.*, 13:767 – 777, 1992.

[58] T. Huckle. Some aspects of circulant preconditoners. SIAM *J. Sci. Stat. Comput.*, 14:531 – 541, 1993.

[59] T. Huckle. Fast transforms for tridiagonal linear equations. *Bit*, 34:99 – 112, 1994.

[60] T. Huckle. Iterative methods for Toeplitz–like matrices. Report SCCM–94–05, Stanford University, 1994.

[61] T. Huckle. Iterative methods for ill–conditioned Toeplitz matrices. In *Workshop on Toeplitz Matrices*, Cortona, 1996.

[62] T. Kailath and V. Olshevsky. Displacement structure approach to discrete–trigonometric–transform based preconditioners of G. Strang type and of T. Chan type. *Preprint*, 1996.

[63] T. Ku and C. Kuo. Preconditioned iterative methode for solving Toeplitz–plus–Hankel systems. SIAM *J. Numer. Anal.*, 30:824 – 844, 1993.

[64] T. Ku and C. Kuo. Spectral properties of preconditioned rational Toeplitz matrices. SIAM *J. Matrix Anal. Appl.*, 14:146 – 165, 1993.

[65] T. Ku and C. Kuo. Spectral properties of preconditioned rational Toeplitz matrices: The nonsymmetric case. SIAM *J. Matrix Anal. Appl.*, 14:521 – 544, 1993.

[66] T. Ku and C. J. Kuo. On the spectrum of a family of preconditioned block Toeplitz matrices. SIAM *J. Sci. Statist. Comput.*, 16:951 – 955, 1992.

[67] C. V. Loan. *Computational Frameworks for the Fast Fourier Transform.* SIAM, Philadelphia, 1992.

[68] J. Mason and E. Venturino. Integration methods of Clenshaw–Curtis type, based on four kinds of Chebyshev polynomials. In G. Nürnberger, J. W. Schmidt, and G. Walz, editors, *Multivariate Approximation and Splines*, pages 153 – 165. Birkhäuser, 1997.

[69] S. S. B. Moore. *Efficient stabilization methods for fast polynomial transforms.* PhD thesis, Dartmouth College, 1994.

[70] S. S. B. Moore, D. M. Healy, and D. N. Rockmore. Symmetry stabilization for fast discrete monomial transforms and polynomial evaluation. *Linear Algebra Appl.*, 192:249 – 299, 1993.

[71] J. G. Nagy and R. J. Plemmons. Some fast Toeplitz least squares algorithms. In *Advanced Signal Processing Algorithms, Architectures and Implementations* II. 35 – 46.

[72] M. K. Ng. Fast iterative methods for solving Toeplitz–plus–Hankel least squares problems. *Elec. Trans. Numer. Anal.*, 2:154 – 170, 1994.

[73] M. K. Ng. Band preconditioners for block–Toeplitz –Toeplitz–block–systems. *Linear Algebra Appl.*, 259:307 – 327, 1997.

[74] R. J. Plemmons. Some applications of iterative deconvolution. *Preprint*, 1994.

[75] D. Potts and G. Steidl. Optimal trigonometric preconditioners for nonsymmetric Toeplitz systems. *Preprint*, 1996.

[76] D. Potts and G. Steidl. Preconditioners for ill–conditioned Toeplitz matrices. *Preprint*, 1997.

[77] D. Potts, G. Steidl, and M. Tasche. Fast algorithms for discrete polynomial transforms. *Math. Comp.* in print.

[78] D. Potts, G. Steidl, and M. Tasche. Fast and stable algorithms for discrete spherical Fourier transforms. *Linear Algebra Appl.* in print.

[79] D. Potts, G. Steidl, and M. Tasche. Trigonometric preconditioners for block Toeplitz systems. In G. Nürnberger, J. W. Schmidt, and G. Walz, editors, *Multivariate Approximation and Splines*, pages 219 – 234, Basel, 1997. Birkhäuser.

[80] K. Rao and P. Yip. *Discrete Cosine Transforms.* Academic Press, Boston, 1990.

[81] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publ., Boston, 1996.

[82] R. Schaback and H. Werner. *Numerische Mathematik.* Springer, Berlin, 1992.

[83] S. Serra. Optimal, quasi–optimal and superlinear band–Toeplitz preconditioners for asymptotically ill–conditioned positive definite Toeplitz systems. *Math. Comp.*, 66:651–665, 1997.

[84] G. Steidl. Fast radix–p discrete cosine transform. *Appl. Algebra in Engrg. Comm. Comput.*, 3:39 – 46, 1992.

[85] G. Steidl and M. Tasche. A polynomial approach to fast algorithms for discrete Fourier–cosine and Fourier–sine transforms. *Math. Comp.*, 56:281 – 296, 1991.

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

[87] G. Strang. A proposal for Toeplitz matrix calculations. *Studies in Appl. Math.*, 74:171 – 176, 1986.

[88] V. Strela. Exploration of circulant preconditioning properties. *Matrix Methods and Algorithms,* IVM RAN *Moscow*, 1993. 9 – 46.

[89] V. Strela and E. E. Tyrtyshnikov. Some generalisations of circulant preconditioner. *Matrix Methods and Algorithms,* IVM RAN *Moscow*, 1990. 66 – 73.

[90] V. V. Strela and E. E. Tyrtyshnikov. Which circulant preconditioner is better? *Math. Comp.*, 65:137 – 150, 1996.

[91] H.-W. Sun, R. H. Chan, and Q.-S. Chang. A note on the convergence of the two–grid methode for Toeplitz systems. *Preprint*, 1997.

[92] E. E. Tyrtyshnikov. Optimal and superoptimal circulant preconditioners. SIAM *J. Matrix Anal. Appl.*, 13:459 – 473, 1992.

[93] E. E. Tyrtyshnikov. A unifying approach to some old and new theorems on distribution and clustering. *Linear Algebra Appl*, 232:1 – 43, 1996.

[94] E. E. Tyrtyshnikov, A. Yeremin, and N. Zamarashkin. Clusters – preconditioners – convergence. *Linear Algebra Appl*, 263:25 – 48, 1997.

[95] Z. Wang. Fast algorithms for the discrete W transform and for the discrete Fourier transform. IEEE *Trans. Acoust. Speech Signal Process*, 32:803 – 816, 1984.

[96] J. Wimp. *Computation with Recurrence Relations.* Pitman, Boston, 1984.

Index

Lebenslauf

Geburtsdatum und -ort:	11. Juli 1969 in Malchin
Familienstand:	verheiratet, eine Tochter
Staatsangehörigkeit:	Bundesrepublik Deutschland

Werdegang:

Sept. 1976 – Juli 1986	10–klassige Oberschule in Stavenhagen
Sept. 1986 – Juli 1988	Erweiterte Oberschule in Malchin, Abschluß mit Prädikat „sehr gut“
Nov. 1988 – Jan. 1990	Grundwehrdienst
Febr. 1990 – Aug. 1990	Elektromonteur in Stavenhagen
Sept. 1990 – März 1995	Mathematikstudium (Diplom), *Universität Rostock* Spezialisierungsrichtung: Numerische Mathematik Nebenfach: Theoretische Physik
1.09.93 – 31.03.94	Teilstudium an der *University of Sussex at Brighton* (England) im Rahmen des Erasmus–Programms
24.03.1995	Diplom in Mathematik mit Prädikat „sehr gut“ Thema der Diplomarbeit: „Wavelets auf der Sphäre“ Betreuer: Prof. Dr. M. Tasche
April 1995 – Sept. 1996	Promotionsstipendiat der *Universität Rostock* auf der Grundlage des Landesgraduiertenförderungsgesetzes unter Leitung von Prof. Dr. G. Maeß
seit Okt. 1996	Wissenschaftlicher Mitarbeiter am Institut für Mathematik der *Medizinischen Universität zu Lübeck*