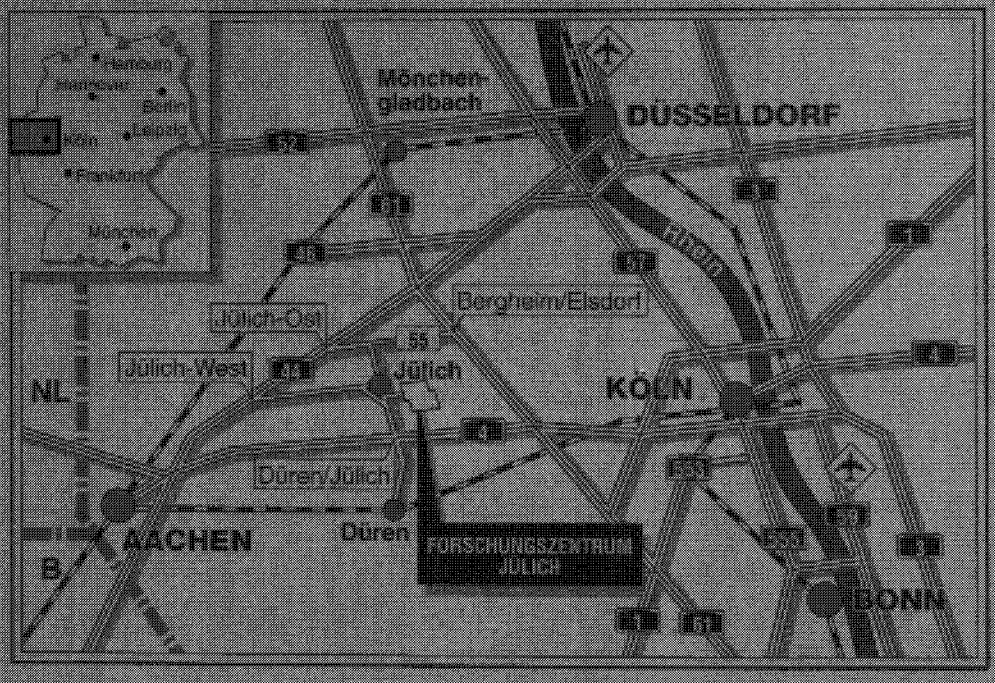
Forschungszentrum Jülich

Zentralinstitut für Angewandte Mathematik

Entwurf paralleler iterativer Verfahren mit kurzen Rekursionen für große dünnbesetzte lineare Gleichungssysteme

Hanns Martin Bücker

JÜL-3478



Berichte des Forschungszentrums Jülich ; 3478
ISSN 0944-2952
Zentralinstitut für Angewandte Mathematik Jül-3478
D 82 (Diss. RWTH Aachen)

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek
D-52425 Jülich · Bundesrepublik Deutschland
☎ 02461/61-6102 · Telefax: 02461/61-6103 · e-mail: zb-publikation@fz-juelich.de

Entwurf paralleler iterativer Verfahren mit kurzen Rekursionen für große dünnbesetzte lineare Gleichungssysteme

Zur Lösung großer dünnbesetzter linearer Gleichungssysteme mit allgemeiner nicht-hermitescher Koeffizientenmatrix werden Krylov–Teilraumverfahren mit kurzen Rekursionen entwickelt. Der Entwurf dieser iterativen Methoden trennt den zugrundeliegenden Prozeß zur Erzeugung einer Basis der Krylov–Teilräume von der Festlegung eines in diesen Vektorräumen enthaltenen Vektors als aktuelle Iterierte. Diese Trennung erlaubt die Rückführung des Entwurfs von parallelen Varianten zur iterativen Lösung linearer Systeme auf die Herleitung paralleler Prozesse zum Aufspannen der zugrundeliegenden Krylov–Teilräume. Es werden parallele Varianten des nicht-hermiteschen Lanczos–Algorithmus ohne Look-Ahead benutzt, um neue Formulierungen der Methoden der quasi-minimalen Residuen (QMR) und der bikonjugierten Gradienten (BCG) herzuleiten, die in jedem Iterationsschritt einen einzigen globalen Synchronisationspunkt besitzen. Daraus resultiert eine im Vergleich zu den ursprünglichen Versionen höhere Skalierbarkeit, die auf den massivparallelen Rechnersystemen Intel Paragon XP/S 10 und CRAY T3E mit bis zu 512 Prozessoren demonstriert wird. Zur Festlegung der aktuellen Iterierten eines Krylov–Teilraumverfahrens wird außerdem der 1-Norm-Ansatz der quasi-minimalen Residuen vorgestellt, der zu Verfahren mit kurzen Rekursionen führt, etwa wenn der nicht-hermitesche Lanczos–Algorithmus, die Methode der quadrierten konjugierten Gradienten (CGS) oder die Methode der bikonjugierten stabilisierten Gradienten (Bi-CGSTAB) zur Erzeugung der zugrundeliegenden Krylov–Teilräume verwendet werden.

Design of Parallel Iterative Methods Based on Short Recurrences for Large Sparse Linear Systems

For the solution of large sparse systems of linear equations with general non-Hermitian coefficient matrices, Krylov subspace methods based on short recurrences are developed. The design of these methods draws a distinction between generating a basis of the Krylov subspaces and defining a vector therein as the actual iterate. By this means, the design of parallel iterative variants for the solution of linear systems is reduced to the derivation of parallel underlying processes to span the Krylov subspaces. Parallel variants of the non-Hermitian Lanczos algorithm without look-ahead are used to derive versions of the quasi-minimal residual method (QMR) and the biconjugate gradient method (BCG) consisting of a single global synchronization point per iteration. Using the Intel Paragon XP/S 10 and CRAY T3E with up to 512 nodes, it is demonstrated that these new versions are significantly more scalable than their original equivalents implemented on massively parallel computers. Moreover, the 1-norm quasi-minimal residual approach is introduced to define the actual iterates of a Krylov subspace method and it is shown that this approach leads to short recurrences, e.g., when the non-Hermitian Lanczos algorithm, the conjugate gradient squared method (CGS) or the biconjugate gradient stabilized method (Bi-CGSTAB) are used to span the underlying Krylov subspaces.

Danksagung

Die vorliegende Arbeit ist im Rahmen des an der RWTH Aachen eingerichteten Graduiertenkollegs “Informatik und Technik” mit Sprecher Prof. Dr. O. Spaniol entstanden.

Herrn Prof. Dr. F. Hoßfeld, dem Inhaber des Lehrstuhls für Technische Informatik und Computerwissenschaften der RWTH Aachen und Direktor am Zentralinstitut für Angewandte Mathematik des Forschungszentrums Jülich, bin ich für seine Betreuung und wohlwollende Unterstützung, die er meiner Arbeit in den letzten Jahren hat zukommen lassen, dankbar. Herrn Prof. Dr. K. Indermark, Inhaber des Lehrstuhls II für Informatik der RWTH Aachen, danke ich für die Übernahme des Korreferates sowie seine zahlreichen, stets konstruktiven Beiträge anläßlich meiner Vorträge in Aachen. Bedanken möchte ich mich auch bei Herrn Dr. P. Weidner für verschiedene Verbesserungsvorschläge und die kritische Durchsicht der Arbeit. Nicht zuletzt gilt mein herzlicher Dank allen Kollegen des Zentralinstituts für Angewandte Mathematik, deren Hilfsbereitschaft und Kompetenz zum Gelingen dieser Arbeit beigetragen haben. Namentlich erwähnen möchte ich an dieser Stelle Herrn M. Sauren und Herrn Dr. R. von Seggern, deren Anregungen während vieler fruchtbarer Diskussionen meine Arbeit wesentlich beeinflußt haben.

Schließlich gebührt ein ganz besonderer Dank meinen Eltern, die mich auf vielfältige Weise nicht nur bei dieser Arbeit unterstützt haben.

Aachen, im August 1997 Martin Bücker

Inhaltsverzeichnis

Kapitel 1

Numerische Simulation in Forschung, Entwicklung und Produktion

Im Verlaufe der letzten 15 Jahre ist die mathematische Modellierung von Abläufen aus Naturwissenschaften und Technik sowie deren Vorausberechnung mit Hilfe moderner Computer, kurz numerische Simulation genannt, zum unentbehrlichen Werkzeug in zahlreichen Schlüsselbereichen der Industrie wie dem Automobil- und Flugzeugbau, der Elektro- und chemischen Industrie geworden. Die numerische Simulation erlaubt das Verstehen und Beherrschen von Vorgängen aus Grundlagenforschung, Entwicklung und Fabrikation zu einem Zeitpunkt, der dem Aufstellen von Theorien und der eigentlichen Fertigung vorausgeht. Einerseits entfällt dadurch der kostenintensive Aufbau komplexer Versuchsanordnungen, und andererseits werden Fortschritte in Wissenschaften und deren technischen Anwendungen überhaupt erst ermöglicht. In der Automobilindustrie ersetzt die Crash-Simulation weitgehend den extensiven Verbrauch teurer Prototypen. In der Flugzeugindustrie löst die Tragflächensimulation das Experiment im Windkanal ab. In der Elektroindustrie werden Anordnung und Verbindungen elektronischer Bauelemente durch Simulation der auftretenden Stromflüsse optimiert. In der chemischen Industrie tritt die verfahrenstechnische Simulation an die Stelle der physikalischen Integration ganzer Anlagenblöcke. Der Einsatz von Techniken der numerischen Simulation reduziert nicht nur den finanziellen und zeitlichen Aufwand in Forschung, Entwicklung und Produktion, sondern spart wertvolle Rohstoffe, senkt die Umweltbelastung und beansprucht Geräte weniger. Nicht zuletzt führt die numerische Simulation zu einem immensen Erkenntnisgewinn, dessen Bedeutung in den Hochtechnologien als unterstützendes Element von Theorie und Experiment kontinuierlich wächst.

Die Problemstellungen, die heutzutage mit Hilfe der numerischen Simulation angegangen werden, entspringen einer Vielzahl von unterschiedlichen Disziplinen: tatsächlich gibt es derzeit kaum einen Bereich von Wissenschaft und Technik, in der der Rechner dafür nicht als Arbeitsinstrument eingesetzt wird. Während in früherer Zeit die Lösung einer den Rechner involvierenden Aufgabenstellung durch informationstechnologisches Zusatzwissen innerhalb der jeweiligen Disziplin gesucht wurde, tritt heute der multidisziplinäre Ansatz mehr und mehr in den Vordergrund. Die "wirklich schweren" Probleme beinhalten heute — und es gibt keinen zwingenden Grund, warum sich dies in Zukunft ändern sollte — eine Fülle von Teilaspekten verschiedenster Disziplinen. Die Lösung solcher Aufgaben bedingt den konzertierten Brückenschlag zwischen Mathematik, Informatik, Natur- und Ingenieurwissenschaften. Probleme werden nicht mehr durch eine einzelne Disziplin gelöst, sondern durch das symbiotische Zusammenspiel von Disziplinen. In diesem Sinne erweist sich die numerische Simulation als inhärent interdisziplinär.

Um Antworten auf harte Fragestellungen zu finden, müssen mathematische Modelle der physikalisch-technischen Abläufe aufgestellt, Algorithmen zur Lösung der in dem Modellierungsprozeß auftretenden Teilaspekte entworfen und eine Abbildung dieser Algorithmen auf adäquate Rech-

nersysteme gefunden werden. Die Formulierung der zugrundeliegenden mathematischen Modelle erfolgt typischerweise auf der Basis von Differentialgleichungen, seien es gewöhnliche oder partielle. In den meisten technischen Anwendungen sind explizite analytische Lösungen dieser Differentialgleichungen unbekannt, beispielsweise aufgrund zu berücksichtigender komplizierter geometrischer Strukturen. Für diese Fälle stellt die numerische Mathematik Näherungsverfahren mit meist iterativer Grundstruktur bereit. Das folgende Beispiel aus der Halbleitertechnik erlaubt einen kurzen Einblick in eine physikalische Problemstellung, seine mathematische Modellierung und numerische Lösung, bei der auch Ergebnisse der vorliegenden Arbeit eingesetzt werden.

In der Halbleitertechnik sind Fortschritte ohne numerische Simulationen heute nicht mehr vorstellbar [12]. Die simulierten physikalischen Vorgänge basieren auf der Energiedifferenz zwischen Leitungs- und Valenzband, die bei Halbleitern so gering ist, daß bei Raumtemperatur thermische Bewegung ausreicht, um einzelne Elektronen vom Valenzband in das Leitungsband anzuheben. Die so entstehenden Paare von Elektronen und Löchern sind die Ursache für die schwache elektrische Leitfähigkeit von *intrinsischen* Halbleitern bei Raumtemperatur. Durch gezieltes — wenn auch geringfügiges — Anreichern eines reinen Halbleiters mit Fremdatomen kann die elektrische Leitfähigkeit um einen Faktor von 10^3 erhöht werden. Dieser Vorgang wird als *Dotierung* bezeichnet, und das auf diese Weise entstandene dotierte Material wird *extrinsischer* Halbleiter genannt. Dieses physikalische Phänomen der Halbleitertechnologie bildet den Ausgangspunkt bei der Herstellung von hochintegrierten Schaltkreisen, deren industrielle Fabrikation einige hundert Arbeitsschritte umfaßt. Zur Unterstützung der zugrundeliegenden technischen Abläufe entwickelt die Elektroindustrie Simulationswerkzeuge wie beispielsweise die SIEMENS AG [49, 57].

Wurden in der Anfangsphase der Halbleitersimulation eindimensionale Modelle verwendet, sind heute aufgrund der schnell fortschreitenden Miniaturisierung der verwendeten Bauelemente zweidimensionale Modellierungen notwendig und in Zukunft wohl dreidimensionale Simulationen unabdingbar. Die linke Seite der Abb. 1.1 stellt einen zweidimensionalen Schnitt durch eine typische Struktur eines Ausgangssubstrats dar. Dabei ist der aus Silizium (Si) bestehende Kern bereits mit einer dünnen Schicht Siliziumdioxid (SiO_2) überzogen. Teile der SiO_2–Schicht werden von Siliziumnitrid (Si_3N_4) bedeckt. Die numerische Simulation schließt mehrere physikalische Phänomene ein, von denen an dieser Stelle jedoch lediglich Oxidation und Dotierung exemplarisch angedeutet werden.

Bei der Oxidation wird dem Halbleitersubstrat Sauerstoff bei Temperaturen zwischen 900 °C und 1200 °C zugeführt. Der Sauerstoff diffundiert an den Stellen durch die SiO_2–Schicht hindurch, die nicht mit dem Nitrid bedeckt sind, und gelangt so bis zum Silizium. Dort treten Sauerstoff und Silizium in eine chemische Reaktion ein. Durch das entstehende Oxid ändert sich während des Vorgangs der Verlauf der Grenzschicht Si–SiO_2. Da das gebildete SiO_2 außerdem ein um den Faktor 2,3 größeres Volumen als Silizium aufweist, tritt zusätzlich eine Deformation auf, die auf der rechten Seite der Abb. 1.1 schematisch gezeigt ist. Darüber hinaus entsteht eine neue Schicht, die aus einem Gemisch von Silizium und Siliziumdioxid besteht. Durch die Bewegung der Grenzschicht Si–SiO_2 sowie die Volumenexpansion treten mechanische Verformungen auf, deren geometrische Veränderungen bei den numerischen Berechnungen durch eine Anpassung des zugrundeliegenden Diskretisierungsgitters zu berücksichtigen sind.

Bei der Dotierung wird das Silizium mit Fremdatomen in ionisierter Form angereichert. Um diese Fremdatome in das Silizium hineinzubringen, wird die Kraftwirkung ausgenutzt, die ein geladenes Teilchen in einem elektrostatischen Feld erfährt. Die ionisierten Fremdatome werden nämlich durch Anlegen eines elektrostatischen Feldes in die Halbleiterstruktur regelrecht "hineingeschossen". Dabei wird die freie Beweglichkeit der Fremdatome sowohl durch die hohen Temperaturen als auch durch das Vorhandensein des Sauerstoffs gefördert. Durch Kollisionen mit Atomen des reinen Siliziums verlieren die ionisierten Fremdatome Energie, werden abgebremst und kommen schließlich zur Ruhe.

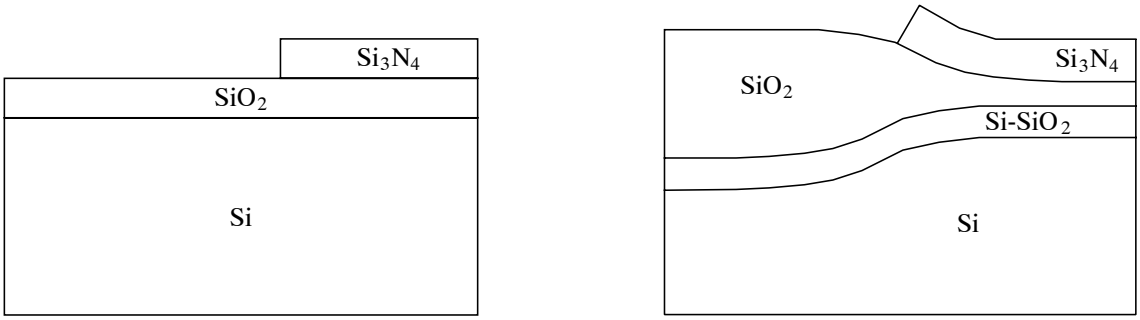


Abbildung 1.1: Zweidimensionaler Schnitt durch eine typische Halbleiterstruktur zum Ausgangszeitpunkt (links) und nach der Verformung durch Oxidation (rechts).

Die Beschreibung der Oxidation, der Dotierung und weiterer physikalischer Phänomene, die hier nicht skizziert sind, wird auf mathematischer Seite durch partielle Differentialgleichungen vorgenommen, beispielsweise werden Kontinuitäts- und Diffusionsgleichungen aufgestellt. Die partiellen Differentialgleichungen, bei denen entsprechende Randbedingungen anzugeben sind, beschreiben das Verhalten derjenigen Kenngrößen, die für den Entwickler von Halbleiterbauelementen relevant sind. Da die elektrischen Eigenschaften der Halbleiterstruktur von der Verteilung der Fremdatome abhängen, ist eine relevante Kenngröße zum Beispiel die Anzahl der Fremdatome in einem Einheitsvolumen. Ein Ziel der numerischen Simulation ist die Berechnung dieses Konzentrationsfeldes.

Die numerische Lösung der Aufgabenstellung trennt Diskretisierungen in Raum und Zeit durch die Linienmethode voneinander, wobei die räumliche Diskretisierung mittels der Methode der finiten Elemente erfolgt. Auftretende nichtlineare Gleichungssysteme werden mit dem Verfahren von Newton linearisiert. Details der numerischen Lösung werden in [57] beschrieben. Diese Vorgehensweise ist in folgendem Sinne typisch für die numerische Lösung von partiellen Differentialgleichungen auf Digitalrechnern [47]. Sie verlangt das Ersetzen der kontinuierlichen Problemstellung durch geeignete diskretisierte Entsprechungen. Diskretisierung und möglicherweise Linearisierung von Differentialgleichungen führen schließlich zu dem Problem der Lösung von linearen Gleichungssystemen als einem der Kernbestandteile der numerischen Simulation. Um die vorliegende Arbeit auch für andere technische Anwendungen verfügbar zu machen, wird nicht länger das spezielle Problem aus der Halbleitertechnik behandelt. Stattdessen wird die Problemstellung auf die abstrakte Ebene der Lösung von linearen Gleichungssystemen angehoben. Dadurch sind die Resultate dieser Arbeit auf allgemeinere Problemstellungen aus Naturwissenschaft und Technik übertragbar. Häufig sind die Koeffizientenmatrizen der auftretenden Gleichungssysteme groß und dünnbesetzt. Dabei werden in der vorliegenden Arbeit unter letzterem Begriff solche Systeme verstanden, bei denen Lösungsverfahren einen Vorteil aus Anzahl und Position der Nichtnull-Elemente ziehen können. In den letzten Jahren ist ein deutlicher Trend bezüglich des Typs der zur Lösung von großen dünnbesetzten linearen Systemen angewendeten Verfahren zu erkennen. Während bisher *direkte Verfahren*, die auf einer Faktorisierung der Koeffizientenmatrix beruhen, bevorzugt wurden, nimmt heute mehr und mehr der Anteil an *iterativen Verfahren* zu, die in aufeinanderfolgenden Schritten geeignete Approximationen an die exakte Lösung berechnen. Daß dieser Richtungswechsel in der grundsätzlichen Vorgehensweise stattfindet, hat vor allem zwei Gründe: die Entwicklung effizienter iterativer Verfahren und die rasant fortschreitende Zunahme der Größe der Gleichungssysteme.

Effiziente iterative Verfahren für die Klasse der reell symmetrischen oder komplex hermiteschen Systeme sind seit langer Zeit bekannt und werden in der Literatur ausführlich behandelt [23]. Für diese Problemklasse existieren Verfahren, die Effizienz und Optimalität miteinander

vereinen. Effizienz bezieht sich dabei auf die Eigenschaft eines Verfahrens, in dessen Verlaufe jeweils den gleichen moderaten Aufwand an Speicherplatz und Rechenzeit in jedem Iterationsschritt zu benötigen. Solche Methoden werden mit dem Begriff *kurze Rekursionen* charakterisiert, weil die Berechnung eines Vektors lediglich auf eine feste, zudem kleine Anzahl von Vektoren aus vorherigen Iterationsschritten zurückgreift. Das Stichwort Optimalität kennzeichnet die Güte einer Approximation an die exakte Lösung. Beispielsweise konstruiert die Methode der minimierten Residuen (MINRES) [58] "optimale" Approximationen an die exakte Lösung in dem Sinne, daß deren Residuen in jedem Iterationsschritt die kleinste euklidische Norm aufweisen. Ist die Koeffizientenmatrix $\mathbf{A}$ außerdem positiv definit, minimiert die Methode der konjugierten Gradienten (CG) [46] stattdessen die $\mathbf{A}$-Norm des Fehlers [62]. Da MINRES und CG sowohl auf kurzen Rekursionen basieren als auch über ausgezeichnete Approximationseigenschaften verfügen, sind sie für hermitesche Systeme die Methoden der ersten Wahl.

Beim Übergang von hermiteschen zu allgemeineren nicht-hermiteschen Systemen ändert sich diese Situation drastisch. Hier gibt es keine einfachen und generell anwendbaren Methoden der ersten Wahl, und man kann sogar zeigen, daß sich Effizienz und Optimalität gegenseitig ausschließen [21]. Methoden mit "optimalen" Approximationen kommen nicht länger mit kurzen Rekursionen aus, sondern verwenden *lange Rekursionen*, deren Aufwand an Speicherplatz und Rechenzeit linear in der Iterationsanzahl zunimmt. Umgekehrt erweisen sich Verfahren mit kurzen Rekursionen als suboptimal hinsichtlich ihrer Approximationseigenschaften. In praktischen Anwendungen treten bei Verfahren mit langen Rekursionen häufig Speicherplatzprobleme auf. Die Entwicklung von Methoden mit kurzen Rekursionen zur Lösung von nicht-hermiteschen Systemen ist deshalb ein aktives Forschungsgebiet und hat eine Vielzahl unterschiedlicher Verfahren hervorgebracht [62]. Da jede Methode bezüglich Optimalität ihre Vor- und Nachteile besitzt, ist es unerläßlich, je nach Problemstellung aus einer Fülle von praktisch anwendbaren iterativen Techniken auswählen zu können. An diesem Punkt setzt die vorliegende Arbeit an, indem sie neue effiziente iterative Verfahren zur Lösung von linearen Gleichungssystemen mit allgemeiner nichthermitescher Koeffizientenmatrix entwirft und damit das Arsenal an potentiellen Kandidaten mit kurzen Rekursionen zu deren numerischen Lösung vergrößert.

Der zweite Grund für das gestiegene Interesse an iterativen Verfahren ist deren im Gegensatz zu direkten Verfahren geringer Speicherplatzbedarf. Bei den heute gebräuchlichen Modellen realer Anwendungen stoßen direkte Verfahren mit ihrem hohen Speicherplatzbedarf an ihre Grenzen. Die Lösung von dreidimensionalen partiellen Differentialgleichungen oder zweidimensionalen mit vielen Freiheitsgraden erzwingt geradezu die Verwendung von iterativen Methoden mit moderaten Anforderungen an Speicherplatz und Rechenzeit. Zudem sind sie auf Hochleistungs- und Parallelrechnern oft mit einem geringeren Aufwand zu implementieren als direkte Verfahren, die dann speziell auf dünnbesetzte Systeme zugeschnitten sind [45, 62].

Da die numerische Simulation und damit die iterativen Verfahren zur Lösung von linearen Gleichungssystemen integraler Bestandteil der Behandlung gerade schwieriger Problemstellungen aus Wissenschaften und deren technischen Anwendungen sind, ist es nicht überraschend, daß man den dort auftretenden großen Datenmengen und hohen Rechenzeiten mit dem Einsatz der jeweils modernsten Rechnertechnologie begegnet. Parallelrechner, bei denen mehrere Prozessoreinheiten gleichzeitig an *einer* Problemstellung arbeiten, sind deshalb bevorzugte Architekturen. Bei der algorithmischen Lösung von Problemen mit Hilfe von parallelen Rechnersystemen werden zwei grundsätzlich verschiedene Ansätze verfolgt. Erstens werden traditionelle sequentielle Algorithmen in ihrer ursprünglichen Form verwendet, wobei soviel Parallelismus wie möglich extrahiert wird. Der Vorteil dieses Konzeptes besteht darin, daß die dabei entstehenden Implementierungen einfach zu handhaben sind und bei deren Analyse auf Eigenschaften des sequentiellen Algorithmus verwiesen werden kann. Unglücklicherweise erreichen traditionelle Algorithmen, die für serielle Rechner entwickelt wurden, häufig bei weitem nicht die erhoffte Effizienz, wenn sie auf Parallel-

rechner portiert werden. Der Grund dafür ist die Übernahme der vorgegebenen seriellen Grundstruktur des Algorithmus, die oft eine effiziente Implementierung auf Parallelrechnern verhindert. Dagegen stellt ein zweiter Parallelisierungsansatz Gesichtspunkte der Parallelverarbeitung bereits in der Entwurfsphase in den Vordergrund. Dem Vorteil einer potentiellen Verbesserung der dabei zu erreichenden Effizienz steht eine Erhöhung des Entwicklungsaufwandes gegenüber. Außerdem ist eine Analyse der neuen Algorithmen notwendig, bei der häufig ein Verlust der numerischen Stabilität beobachtet wird [19]. Die vorliegende Arbeit verfolgt diesen zweiten Ansatz des parallelen Algorithmenentwurfes. Sie entwickelt nicht nur neue (sequentielle) iterative Verfahren zur Lösung linearer Gleichungssysteme, sondern achtet bei deren Entwurf gezielt auf deren Ausführung auf parallelen Rechnersystemen mit verteiltem Speicher, die über eine hohe Anzahl von Prozessoren verfügen. Dazu werden die mathematischen Eigenschaften der zugrundeliegenden iterativen Prozesse analysiert. Speziell werden die von diesen Prozessen erzeugten Vektorräume untersucht. Entscheidungen, die den parallelen Algorithmenentwurf betreffen, werden aufgrund von aufgespannten Vektorräumen und nicht durch eine Restrukturierung traditioneller serieller Algorithmen oder Programme getroffen. Dies entspricht einer Berücksichtigung von parallelen Aspekten zu einem frühen Zeitpunkt der Entwurfsphase und nicht erst zu einem zu späten Zeitpunkt, an dem der Algorithmus oder das Programm bereits der Parallelverarbeitung entgegenstehende Konstruktionselemente enthalten. Das Konzept, bei iterativen Verfahren die Betrachtung der zugrundeliegenden Vektorräume für den parallelen Algorithmenentwurf auszunutzen, erweist sich als tragfähig. Mit Hilfe dieses Konzeptes wird hier eine Reihe von parallelen iterativen Methoden hergeleitet.

Der Aufbau der vorliegenden Arbeit wird durch die Grundstruktur der entwickelten Verfahren zur Lösung linearer Gleichungssysteme vorgegeben. Diese iterativen Verfahren gehören zu der Klasse der sogenannten *Krylov–Teilraumverfahren*, die die Approximation an die exakte Lösung innerhalb von Teilräumen einer ganz speziellen Form konstruieren. Daher befaßt sich der erste Teil dieser Arbeit, der aus den Kapiteln 2–4 besteht, mit der Erzeugung dieser speziellen Teilräume. In einem zweiten Teil, den die Kapitel 5 und 6 bilden, wird dann festgelegt, welcher Vektor aus diesen Teilräumen die nächste Approximation an die exakte Lösung darstellt.

Detaillierter gliedert sich die Arbeit wie folgt. In Kapitel 2 werden bekannte iterative Prozesse zum Aufspannen dieser Teilräume zur Verfügung gestellt und deren Eigenschaften in Sätzen gesammelt, auf die später Bezug genommen wird. Eine Analyse dieser Verfahren hinsichtlich deren Ausführung auf Parallelrechnern mit verteiltem Speicher folgt in Kapitel 3. Dabei wird das Isoeffizienzmodell eingesetzt, das neben dem zu untersuchenden Algorithmus auch die zugrundegelegte Parallelrechnerarchitektur berücksichtigt. Durch asymptotische Betrachtungen erhöht die vorgenommene Isoeffizienzanalyse das Verständnis für das Gesamtverhalten des Systems aus parallelem Algorithmus und Architektur, ohne detaillierte Laufzeitprognosen zu beabsichtigen. Sie erlaubt jedoch, Anforderungen an den Entwurf paralleler iterativer Methoden abzuleiten. Für Parallelrechner mit einer hohen Prozessoranzahl identifiziert die Analyse Operationen mit *globaler* Synchronisation als limitierenden Faktor, der einer hocheffizienten und skalierbaren parallelen Implementierung von Krylov–Teilraumverfahren entgegensteht. In solchen Operationen kommunizieren alle beteiligten Prozessoren zum selben Zeitpunkt miteinander. Die Fortsetzung der Berechnung ist erst dann möglich, wenn die dabei ausgetauschten Daten bei allen Prozessoren verfügbar sind. Die Erkenntisse der Isoeffizienzanalyse werden im darauffolgenden Kapitel 4 benutzt, um eine neue Variante des nicht-hermiteschen Lanczos–Algorithmus herzuleiten, in der alle Operationen mit globaler Synchronisation innerhalb eines Iterationsschrittes voneinander unabhängig sind und somit simultan berechnet werden können. Diese parallele Variante des Lanczos–Algorithmus wird später in Krylov–Teilraumverfahren als zugrundeliegendes Verfahren zur Berechnung einer Basis der speziellen Teilräume verwendet. Die Herleitung dieser parallelen Variante erfolgt durch eine Betrachtung der aufgespannten Vektorräume und damit zu einem frühen Zeitpunkt in der Entwurfsphase des Algorithmus.

Neben der Konstruktion einer geeigneten Basis der speziellen Teilräume besteht der Entwurf von Krylov–Teilraumverfahren konzeptionell aus einem zweiten Bestandteil, nämlich einer Angabe darüber, welcher Vektor aus dem aktuell aufgespannten Teilraum als nächste Approximation an die exakte Lösung ausgewählt wird. Diese Festlegung der aktuellen Iterierten wird in der vorliegenden Arbeit unter Verwendung unterschiedlicher Minimierungsprobleme vorgenommen. Präzise ausgedrückt wird dabei die Lösung einer Folge von Minimierungsproblemen einer speziellen Form benötigt. In Kapitel 5 werden sowohl die spezielle Form dieser Minimierungsprobleme vorgestellt als auch Techniken zu deren effizienten Lösung in unterschiedlichen Normen bereitgestellt. Mit Hilfe dieser Techniken werden in Kapitel 6 neue iterative Verfahren mit kurzen Rekursionen entwickelt. In diesem Kapitel werden nicht nur neue parallele Varianten bekannter Methoden hergeleitet, sondern darüber hinaus auch neue serielle iterative Methoden entworfen. Letztere entstehen durch die Anwendung eines neuen allgemeinen Ansatzes zur Herleitung von iterativen Methoden. Dieser Ansatz abstrahiert von dem speziellen Prozeß zum Aufspannen der zugrundeliegenden Teilräume, so daß aus unterschiedlichen Prozessen jeweils unterschiedliche Methoden zur Lösung von linearen Gleichungssystemen resultieren. Dieser Ansatz erlaubt insbesondere die Verwendung der in Kapitel 4 eingeführten parallelen Varianten zum Aufspannen der Teilräume. Numerische Experimente und Ergebnisse der parallelen Implementierungen werden in Kapitel 7 beschrieben und bewertet.

Kapitel 2

Erzeugung von Krylov–Teilräumen

In den letzten zehn Jahren hat sich eine Klasse von iterativen Verfahren zur Lösung von großen dünnbesetzten linearen Gleichungssystemen etabliert, die man als Krylov–Teilraumverfahren bezeichnet. Diese Verfahren sind dadurch charakterisiert, daß sie die Approximationen an die exakte Lösung des Gleichungssystems in Vektorräumen konstruieren, die eine spezielle Form besitzen. Konzeptionell bestehen Krylov–Teilraumverfahren aus zwei unterschiedlichen Bestandteilen, die getrennt voneinander betrachtet werden können: Die Erzeugung der speziellen Vektorräume, die Krylov–Teilräume genannt werden, und die Definition der aktuellen Approximation innerhalb dieser Krylov–Teilräume. Dieses Kapitel widmet sich ersterem und trägt bekannte Verfahren zusammen, die eine numerisch stabile Basis der Krylov–Teilräume generieren. Eine Möglichkeit, solche Verfahren zu gewinnen, besteht darin, bekannte Krylov–Teilraumverfahren zur Lösung von Gleichungssystemen in ihre zwei Bestandteile aufzuspalten und nur denjenigen Teil zu verwenden, der die Basisvektoren der Krylov–Teilräume generiert. Diese Vorgehensweise wird in diesem Kapitel bei zwei bekannten Krylov–Teilraumverfahren verfolgt. Dabei steht nicht eine präzise Beschreibung dieser Verfahren in ihrer ursprünglichen Form im Vordergrund. Vielmehr ist das Ziel ihre Restrukturierung derart, daß sie in nachfolgenden Kapiteln beim Entwurf neuer Krylov–Teilraumverfahren unmittelbar eingesetzt werden können.

Nach einer kurzen Beschreibung der in dieser Arbeit verwendeten Notationen und einem einführenden Abschnitt, der sich mit den Grundlagen von Krylov–Teilraumverfahren beschäftigt, werden in drei weiteren Abschnitten der biorthogonale Lanczos–Algorithmus, das Verfahren der quadrierten konjugierten Gradienten (CGS) und die Methode der bikonjugierten stabilisierten Gradienten (Bi-CGSTAB) vorgestellt. Diese drei Verfahren werden in späteren Teilen der vorliegenden Arbeit als zugrundeliegende Technik zur Erzeugung von Krylov–Teilräumen in neuen Iterationsverfahren zur Lösung linearer Gleichungssysteme eingesetzt.

2.1 Notationen

In dieser Arbeit werden Matrizen und Vektoren durch in Fettdruck gesetzte lateinische Buchstaben gekennzeichnet, während skalare Größen in griechischen Buchstaben notiert werden. Zu gegebenen Vektoren $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q \in \mathbb{C}^N$ bezeichnet die Schreibweise $[\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_q] \in \mathbb{C}^{N\times q}$ diejenige Matrix, deren Spalten aus diesen Vektoren aufgebaut sind. Die Menge aller Linearkombinationen dieser Vektoren wird durch die Notation $\mathrm{span}\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ beschrieben. Für die Diagonalmatrix mit Elementen $\alpha_1, \alpha_2, \ldots, \alpha_q \in \mathbb{C}$ auf der Hauptdiagonalen wird die Schreibweise $\mathrm{diag}(\alpha_1, \alpha_2, \ldots, \alpha_q) \in \mathbb{C}^{q\times q}$ verwendet. Der Nullvektor des $\mathbb{R}^N$ wird durch das Symbol $\mathbf{0}_N$ dargestellt. Für den j-ten Spaltenvektor der N-dimensionalen Einheitsmatrix $\mathbf{I}_N$ wird die Notation $\mathbf{e}_j^{(N)}$ verwendet, wobei die Kennzeichnung (N) der Dimension entfällt, wenn sie aus

dem Kontext hervorgeht. Komplexe Konjugation wird durch einen Strich notiert, beispielsweise $\alpha \cdot \overline{\alpha} = |\alpha|^2$. Die Menge aller Polynome vom Grad höchstens n wird mit $\mathcal{P}_n$ bezeichnet. Für zwei Funktionen $f(N)$ und $g(N)$, die von der Menge der natürlichen Zahlen nach der Menge der positiven reellen Zahlen abbilden, wird zur Beschreibung eines asymptotischen Funktionenverhaltens die Notation $f(N) = \Theta(g(N))$ verwendet, wenn es positive reelle Konstanten c_1 und c_2 mit $c_1 g(N) \leq f(N) \leq c_2 g(N)$ für alle hinreichend große N gibt.

2.2 Grundlagen von Krylov–Teilraumverfahren

Die in dieser Arbeit betrachteten iterativen Verfahren, die als Krylov–Teilraumverfahren bezeichnet werden, sind in der Mitte der 80er Jahre populär geworden. Umfangreichere Darstellungen von Krylov–Teilraumverfahren findet man in den Büchern [1, 23, 40, 62, 70]. Eine kurze Einführung gibt der Übersichtsartikel [31]. Neuere Entwicklungen sind in [39, 44] zusammengefaßt. Krylov–Teilraumverfahren stellen derzeit ein mächtiges Instrument zur Lösung von linearen Gleichungssystemen der Form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.1}$$

dar, wobei $\mathbf{A}$ eine allgemeine unsymmetrische, komplexe $N \times N$ Matrix und $\mathbf{x}, \mathbf{b} \in \mathbb{C}^N$ ist. Außerdem wird im Verlaufe der gesamten Arbeit angenommen, daß $\mathbf{A}$ nicht singulär ist und somit für (2.1) eine eindeutige Lösung $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$ existiert. Es wird weiterhin implizit vorausgesetzt, daß die Koeffizientenmatrix $\mathbf{A}$ dünnbesetzt und deren Ordnung N groß ist. Andernfalls könnten direkte Verfahren wie die Gauß–Elimination angewendet werden, deren Anforderungen an Speicher und Rechenzeit für große Systeme jedoch inakzeptabel sind. Ebenso werden keine Verfahren betrachtet, die spezielle Eigenschaften der Koeffizientenmatrix wie etwa Symmetrie oder positive Definitheit voraussetzen oder die deren vorhandene Struktur, etwa Bandstruktur, gezielt ausnutzen.

Die grundsätzliche Idee, die hinter den hier betrachteten Verfahren steht, ist die der Projektion. Dabei wird eine Näherung $\mathbf{x}_n$ an die exakte Lösung $\mathbf{x}^*$ innerhalb eines niedrig-dimensionalen Teilraumes des $\mathbb{C}^N$ gesucht. Das Symbol $\mathcal{K}$ bezeichne diesen Teilraum, der die Dimension n besitze. Um in $\mathcal{K}$ einen Vektor als Approximation $\mathbf{x}_n$ zu definieren, werden dann n Bedingungen in Form von Orthogonalitätsbeziehungen gestellt. Dabei wird von dem Residuum $\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$ die Orthogonalität zu n linear unabhängigen Vektoren gefordert. Diese Vektoren spannen einen weiteren Vektorraum $\mathcal{L}$ der Dimension n auf. Die Beziehungen zwischen den Räumen $\mathcal{K}$ und $\mathcal{L}$ werden in der folgenden Definition zu einer groben Klassifikation von Projektionsmethoden herangezogen.

Definition 2.1. Bezeichnet $\mathbf{x}_0 \in \mathbb{C}^N$ einen beliebigen Startvektor zur iterativen Lösung von (2.1) und sind $\mathcal{K}$ und $\mathcal{L}$ zwei Teilräume von $\mathbb{C}^N$ der Dimension n, dann heißt

$$\text{Finde} \quad \mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K} \quad \text{mit} \quad \mathbf{b} - \mathbf{A}\mathbf{x}_n \perp \mathcal{L}$$

eine *Projektionsmethode auf den Teilraum $\mathcal{K}$ orthogonal zu $\mathcal{L}$*. Ist $\mathcal{K} = \mathcal{L}$, nennt man die Projektionsmethode *orthogonal*, andernfalls *schief*.

Diese Klassifikation unterscheidet eine Projektionsmethode danach, ob der Teilraum $\mathcal{K}$ der gleiche wie $\mathcal{L}$ ist. Die Vorgehensweise im Fall $\mathcal{K} = \mathcal{L}$, also im Falle einer orthogonalen Projektionsmethode, besteht darin, das Residuum zum Raum der aktuellen Approximation zu orthogonalisieren, und wird *Galerkin–Ansatz* genannt. Im Gegensatz dazu spricht man in einer schiefen Projektionsmethode, in der $\mathcal{K} \neq \mathcal{L}$ gilt, von einem *Petrov–Galerkin–Ansatz*. Hier wird eine Approximation $\mathbf{x}_n$ in einem Teilraum so bestimmt, daß das Residuum orthogonal zu einem anderen, geeignet gewählten Teilraum der gleichen Dimension ist.

Die in obiger Definition beschriebene Vorgehensweise charakterisiert einen einzelnen Schritt einer Projektionsmethode in allgemeiner Form. Üblicherweise bestehen iterative Verfahren aus einer Folge von solchen Schritten. Dabei werden in jedem Schritt zwei neue Räume $\mathcal{K}$ und $\mathcal{L}$ sowie ein neuer Startvektor $\mathbf{x}_0$ verwendet, der gleich der Approximation aus dem letzten Projektionsschritt ist. Für wichtige Spezialfälle von Projektionsmethoden kann man Optimalitätseigenschaften nachweisen [62]. An dieser Stelle werden jedoch nur solche Methoden betrachtet, bei denen der Teilraum $\mathcal{K}$ durch eine spezielle Form festgelegt ist.

Definition 2.2. Bezeichnet $\mathbf{x}_0 \in \mathbb{C}^N$ einen beliebigen Startvektor zur iterativen Lösung von (2.1) und $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ das Startresiduum, dann heißt eine (orthogonale oder schiefe) Projektionsmethode auf den Teilraum

$$\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0) := \operatorname{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{n-1}\mathbf{r}_0\} \tag{2.2}$$

Krylov–Teilraumverfahren. Der Raum $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ wird als n-ter von $\mathbf{A}$ und $\mathbf{r}_0$ erzeugter *Krylov–Teilraum* bezeichnet.

Bei einem Krylov–Teilraumverfahren ist der Teilraum, aus dem die Approximation $\mathbf{x}_n$ konstruiert wird, festgelegt, nicht jedoch der Teilraum $\mathcal{L}$, zu dem das Residuum orthogonalisiert wird. Die bekanntesten Krylov–Teilraumverfahren lassen sich durch unterschiedliche Wahl des Teilraumes $\mathcal{L}$ in die zwei folgenden Klassen einteilen. Die erste Klasse ist durch die Beziehungen $\mathcal{L} = \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ oder $\mathcal{L} = \mathbf{A}\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ charakterisiert. Im Fall $\mathcal{L} = \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ erhält man beispielsweise das Verfahren der konjugierten Gradienten (**C**onjugate **G**radients, CG) [46] und ein bekannter Vertreter des Falles $\mathcal{L} = \mathbf{A}\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ ist die Methode der verallgemeinerten Minimierung der Residuen (**G**eneralized **M**inimal **RES**idual, GMRES) [63]. Die zweite Klasse basiert auf der Wahl $\mathcal{L} = \mathcal{K}_n(\mathbf{A}^T, \widetilde{\mathbf{r}}_0)$, also auf dem Krylov–Teilraum bezüglich $\mathbf{A}^T$ mit einem geeigneten Vektor $\widetilde{\mathbf{r}}_0 \in \mathbb{C}^N$. Bekannte Vertreter dieser Klasse sind das Verfahren der bikonjugierten Gradienten (**Bi**Conjugate **G**radients, BCG) [24, 52] und die Methode der quasi-minimalen Residuen (**Q**uasi-**M**inimal **R**esidual, QMR) [35].

Unter praktischen Gesichtspunkten besteht das Konzept einer Krylov–Teilraummethode aus den folgenden zwei Bestandteilen:

- Die numerisch stabile Konstruktion einer Basis für $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ und
- die Festlegung der aktuellen Iterierten $\mathbf{x}_n$.

Die in (2.2) angegebene Basis von $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ ist aus numerischer Sicht unbrauchbar; denn nach der Potenzmethode [71] zeigen die Vektoren $\mathbf{A}^i\mathbf{r}_0$ für wachsendes i mehr und mehr in Richtung des dominanten Eigenvektors und werden in endlicher Arithmetik linear abhängig. Ziel dieses Kapitels ist es, geeignete Verfahren zur Erzeugung von Basisvektoren $\mathbf{v}_i$ mit

$$\operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} = \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0) \tag{2.3}$$

anzugeben. Die nächsten drei Abschnitte stellen jeweils unterschiedliche Algorithmen vor, die in folgenden Kapiteln der vorliegenden Arbeit zu diesem Zwecke benutzt werden.

Zur Festlegung der aktuellen Iterierten $\mathbf{x}_n$, die von der Form

$$\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$$

ist, kann es günstig sein, Krylov–Teilraumverfahren in polynomialer Darstellung zu betrachten. Da die Basisvektoren von $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ nach der Repräsentation (2.2) auf $\mathbf{r}_0$ angewandte Polynome in der Matrix $\mathbf{A}$ sind, gilt

$$\mathbf{x}_n = \mathbf{x}_0 + \phi_{n-1}(\mathbf{A})\mathbf{r}_0,$$

wobei ϕ_{n-1} ein Polynom vom Grad $n-1$ ist. Daraus folgt unmittelbar

$$\mathbf{r}_n = \psi_n(\mathbf{A})\mathbf{r}_0 \qquad \text{mit} \qquad \psi_n(0) = 1$$

für ein Polynom ψ_n vom Grad n. Daher ist es möglich, die Iterierte $\mathbf{x}_n$ durch eine geeignete Wahl des sogenannten *Residualpolynoms* ψ_n zu definieren.

2.3 Biorthogonaler Lanczos–Algorithmus

Um 1950 entwickelte Cornelius Lanczos ein iteratives Verfahren [51] zur Berechnung von Eigenwerten einer nicht notwendig symmetrischen Matrix. Den Kern dieses Verfahrens bildet eine Ähnlichkeitstransformation dieser Matrix auf tridiagonale Form, so daß im Anschluß an diese Transformation eine beliebige effiziente und numerisch stabile Methode zur Berechnung der Eigenwerte einer Tridiagonalmatrix angewendet werden kann. Das iterative Verfahren, das unter dem Begriff *Lanczos–Algorithmus* bekannt ist, wird seit langer Zeit in unterschiedlichen Formen in vielen Anwendungen verwendet, in denen die Matrix hermitesch ist. Für diesen Spezialfall ist der Lanczos–Algorithmus gut analysiert und theoretisch fundiert [59]. Aufgrund numerischer Instabilitäten und der Möglichkeit eines vorzeitigen Abbruchs ist dagegen der allgemeine, nicht-hermitesche Fall oft mit Skepsis betrachtet worden, obwohl stabile Implementierungen für praktische Anwendungen bekannt waren [16]. Erst seit der Entwicklung von sogenannten *Look-Ahead-Techniken*, die hohe Sicherheit gegenüber Instabilität und Abbruch bieten, wird der nicht-hermitesche Lanczos–Algorithmus in der numerischen Analysis mehr beachtet.

Im Zusammenhang mit der vorliegenden Arbeit, in der es nicht um Eigenwertprobleme geht, ist eine weitere Eigenschaft des Lanczos–Algorithmus von zentraler Bedeutung. Der Lanczos–Algorithmus kann benutzt werden, um eine Basis von $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ zu erzeugen. Damit kann er als zugrundeliegende Technik von Krylov–Teilraumverfahren zur Konstruktion der in (2.3) gesuchten Basisvektoren $\mathbf{v}_i$ dienen. Dieser Abschnitt befaßt sich mit dem Lanczos–Algorithmus für den nicht-hermiteschen Fall. Dabei werden zunächst allgemeine Rahmenbedingungen beschrieben, die grundsätzlich auf alle Implementierungen des Lanczos–Algorithmus zutreffen. Die zwei folgenden Unterabschnitte skizzieren dann Implementierungen, die sich im Hinblick auf die Struktur der zur Erzeugung der Basisvektoren verwendeten Rekursionen unterscheiden. Abschließend werden Look-Ahead-Techniken betrachtet.

2.3.1 Allgemeine Rahmenbedingungen

Der Lanczos–Algorithmus wird in späteren Kapiteln der vorliegenden Arbeit bei dem Entwurf unterschiedlicher schiefer Projektionsmethoden auf den Teilraum $\mathcal{K} = \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ orthogonal zu $\mathcal{L} = \mathcal{K}_n(\mathbf{A}^T, \widetilde{\mathbf{r}}_0)$ verwendet. Der Lanczos–Algorithmus konstruiert dabei eine Basis sowohl für $\mathcal{K}$ als auch für $\mathcal{L}$, also Krylov–Teilräume bezüglich einer Matrix sowie deren Transponierten. Die Bezeichnungen $\mathbf{r}_0$ und $\widetilde{\mathbf{r}}_0$ kennzeichnen den Bezug zu der Verwendung des Lanczos–Algorithmus als zugrundeliegende Technik zur Lösung von linearen Gleichungssystemen. Da aber der Lanczos–Algorithmus vorerst losgelöst von dieser Aufgabenstellung betrachtet wird, wird er hier mit zwei allgemeinen Startvektoren $\mathbf{v}_1, \mathbf{w}_1 \in \mathbb{C}^N$ formuliert.

Zu gegebener Matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$ und zwei Startvektoren $\mathbf{v}_1 \neq \mathbf{0}$ und $\mathbf{w}_1 \neq \mathbf{0}$ erzeugt der Lanczos–Algorithmus ein Paar von Folgen von Vektoren $\{\mathbf{v}_i\}_{i=1,2,3,\ldots}$ und $\{\mathbf{w}_i\}_{i=1,2,3,\ldots}$, die den

folgenden drei Eigenschaften genügen:

$$\mathbf{v}_n \in \mathcal{K}_n\left(\mathbf{A}, \mathbf{v}_1\right), \tag{2.4}$$

$$\mathbf{w}_n \in \mathcal{K}_n\left(\mathbf{A}^T, \mathbf{w}_1\right), \tag{2.5}$$

$$\mathbf{w}_i^T \mathbf{v}_j = \begin{cases} 0, & \text{falls } i \neq j, \\ \delta_i \neq 0, & \text{falls } i = j. \end{cases} \tag{2.6}$$

Die durch den Lanczos–Algorithmus generierten Vektoren $\mathbf{v}_i$ und $\mathbf{w}_i$ werden *Lanczos–Vektoren* genannt. Die in (2.6) beschriebene Eigenschaft wird als *Biorthogonalität* bezeichnet. Auf dieser Eigenschaft beruht die Bezeichnung des Verfahrens als *biorthogonaler Lanczos–Algorithmus*. Soweit nicht anders daraufhingewiesen wird, wird im Verlaufe der gesamten Arbeit abkürzend der Begriff Lanczos–Algorithmus verwendet, so daß der hermitesche Spezialfall jeweils explizit erwähnt wird.

Implementierungen des Lanczos–Algorithmus unterscheiden sich hinsichtlich mehrerer Aspekte. Viele Implementierungen, besonders in Textbüchern [38, 62, 71], bieten lediglich die Möglichkeit, eine der beiden Folgen der Lanczos–Vektoren gegen Über- oder Unterlauf abzusichern. Solche Implementierungen werden hier nicht betrachtet. Stattdessen besteht in allen hier vorgestellten Implementierungen eine Normalisierungsmöglichkeit sowohl für die Vektoren $\{\mathbf{v}_i\}_{i=1,2,3,\ldots}$ als auch $\{\mathbf{w}_i\}_{i=1,2,3,\ldots}$. Ein weiteres Unterscheidungsmerkmal für Implementierungen des Lanczos–Algorithmus ist die Länge der zur Berechnung eines Lanczos–Vektors verwendeten Rekursion. In den folgenden beiden Unterabschnitten werden zuerst Drei-Term-Rekursionen und anschließend gekoppelte Zwei-Term-Rekursionen beschrieben.

2.3.2 Implementierung mit Drei-Term-Rekursionen

Eine kompakte Formulierung des Lanczos–Algorithmus erfolgt in Matrixnotation. Dazu faßt man die Lanczos–Vektoren als Spalten von $N \times n$ Matrizen

$$\mathbf{V}_n := [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_n] \qquad \text{und} \qquad \mathbf{W}_n := [\mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_n]$$

auf. In dieser Notation wird die Biorthogonalität (2.6) der Lanczos–Vektoren durch die Matrixgleichung

$$\mathbf{W}_n^T \mathbf{V}_n = \mathbf{D}_n \tag{2.7}$$

beschrieben, wobei die nicht-singuläre $n \times n$ Matrix

$$\mathbf{D}_n = \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_n)$$

eingeführt wird. Damit ist der Lanczos–Algorithmus bis zum n-ten Schritt, in dem die Lanczos–Vektoren $\mathbf{v}_{n+1}$ und $\mathbf{w}_{n+1}$ berechnet werden, durch die zwei Matrixgleichungen

$$\mathbf{A}\mathbf{V}_n = \mathbf{V}_n \mathbf{T}_n + \rho_{n+1} \mathbf{v}_{n+1} \mathbf{e}_n^T, \tag{2.8a}$$

$$\mathbf{A}^T \mathbf{W}_n = \mathbf{W}_n \mathbf{D}_n^{-1} \mathbf{T}_n^T \mathbf{D}_n + \frac{\gamma_{n+1}\delta_n}{\delta_{n+1}} \mathbf{w}_{n+1} \mathbf{e}_n^T, \tag{2.8b}$$

charakterisiert, wobei neben $\mathbf{e}_n = (0, \ldots, 0, 1)^T \in \mathbb{R}^n$ die tridiagonale $n \times n$ Matrix

$$\mathbf{T}_n := \begin{bmatrix} \alpha_1 & \gamma_2 & & & \\ \rho_2 & \alpha_2 & \gamma_3 & & \mathbf{0} \\ & \rho_3 & \ddots & \ddots & \\ & & \ddots & \ddots & \gamma_n \\ \mathbf{0} & & & \rho_n & \alpha_n \end{bmatrix} \tag{2.9}$$

benutzt wird. Im nächsten Iterationsschritt werden dann die berechneten Lanczos–Vektoren $\mathbf{v}_{n+1}$ und $\mathbf{w}_{n+1}$ den Matrizen $\mathbf{V}_n$ und $\mathbf{W}_n$ als letzte Spalten hinzugefügt. Außerdem werden im n-ten Schritt die skalaren Größen α_n, γ_{n+1} und ρ_{n+1} berechnet.

Unter Verwendung der Biorthogonalität ist es nun möglich, aus den Gleichungen (2.8) einen Algorithmus herzuleiten, bei dem nicht die kompletten Matrizen $\mathbf{V}_n$, $\mathbf{W}_n$ und $\mathbf{T}_n$ abgespeichert werden müssen. Dazu betrachtet man in den Gleichungen (2.8) die jeweils letzte Spalte:

$$\mathbf{A}\mathbf{v}_n = \rho_{n+1}\mathbf{v}_{n+1} + \alpha_n\mathbf{v}_n + \gamma_n\mathbf{v}_{n-1}, \tag{2.10a}$$

$$\mathbf{A}^T\mathbf{w}_n = \frac{\gamma_{n+1}\delta_n}{\delta_{n+1}}\mathbf{w}_{n+1} + \alpha_n\mathbf{w}_n + \frac{\rho_n\delta_n}{\delta_{n-1}}\mathbf{w}_{n-1}, \tag{2.10b}$$

wobei $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$ und $\delta_0 \neq 0$. Die Koeffizienten α_n, γ_{n+1} und ρ_{n+1} werden durch die Forderung der Biorthogonalität bestimmt. Multipliziert man (2.10a) mit $\mathbf{w}_n^T$ und verwendet (2.7), so folgt

$$\alpha_n = \frac{\mathbf{w}_n^T\mathbf{A}\mathbf{v}_n}{\delta_n}. \tag{2.11}$$

Führt man nun die folgenden Größen ein, nämlich

$$\widetilde{\mathbf{v}}_{n+1} := \mathbf{A}\mathbf{v}_n - \alpha_n\mathbf{v}_n - \gamma_n\mathbf{v}_{n-1}, \tag{2.12a}$$

$$\widetilde{\mathbf{w}}_{n+1} := \mathbf{A}^T\mathbf{w}_n - \alpha_n\mathbf{w}_n - \frac{\rho_n\delta_n}{\delta_{n-1}}\mathbf{w}_{n-1}, \tag{2.12b}$$

und

$$\xi_{n+1} := \gamma_{n+1}\delta_n/\delta_{n+1}, \tag{2.13}$$

dann gelten nach (2.10) die Beziehungen

$$\widetilde{\mathbf{v}}_{n+1} = \rho_{n+1}\mathbf{v}_{n+1}, \tag{2.14a}$$

$$\widetilde{\mathbf{w}}_{n+1} = \xi_{n+1}\mathbf{w}_{n+1}. \tag{2.14b}$$

Die Biorthogonalität der Lanczos–Vektoren $\mathbf{v}_{n+1}$ und $\mathbf{w}_{n+1}$ impliziert

$$\rho_{n+1}\xi_{n+1}\delta_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T\widetilde{\mathbf{v}}_{n+1}.$$

Dadurch hat man die Möglichkeit, die Lanczos–Vektoren durch

$$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2, \tag{2.15a}$$

$$\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2, \tag{2.15b}$$

in der euklidischen Norm auf Einheitslänge zu normieren. Formt man (2.12b) mit Hilfe von (2.13) in

$$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T\mathbf{w}_n - \alpha_n\mathbf{w}_n - \frac{\gamma_n\rho_n}{\xi_n}\mathbf{w}_{n-1} \tag{2.16}$$

mit $\xi_1 \neq 0$ um, so folgt durch korrektes Zusammenfügen der hergeleiteten Gleichungen der resultierende Prozeß. Die Tridiagonalität der Matrix $\mathbf{T}_n$ führt zu den Rekursionen (2.12) für die unskalierten Lanczos–Vektoren $\widetilde{\mathbf{v}}_{n+1}$ und $\widetilde{\mathbf{w}}_{n+1}$. Die Struktur dieser Rekursionen zeigt, daß diese Implementierung auf *Drei-Term-Rekursionen* basiert.

ALGORITHMUS 2.1 (LANCZOS–ALGORITHMUS MIT DREI-TERM-REKURSIONEN)

Wähle $\mathbf{v}_1, \mathbf{w}_1 \in \mathbb{C}^N$ so, daß $\|\mathbf{v}_1\|_2 = \|\mathbf{w}_1\|_2 = 1$ und $\delta_1 = \mathbf{w}_1^T \mathbf{v}_1 \neq 0$
Setze $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$ und $\gamma_1 = \rho_1 = 0$, $\xi_1 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**

$\alpha_n = \mathbf{w}_n^T \mathbf{A} \mathbf{v}_n / \delta_n$	Gl. (2.11)
$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{v}_n - \alpha_n \mathbf{v}_n - \gamma_n \mathbf{v}_{n-1}$	Gl. (2.12a)
$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T \mathbf{w}_n - \alpha_n \mathbf{w}_n - \frac{\gamma_n \rho_n}{\xi_n} \mathbf{w}_{n-1}$	Gl. (2.16)
$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$	Gl. (2.15a)
$\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$	Gl. (2.15b)
$\mathbf{v}_{n+1} = \frac{1}{\rho_{n+1}} \widetilde{\mathbf{v}}_{n+1}$	Gl. (2.14a)
$\mathbf{w}_{n+1} = \frac{1}{\xi_{n+1}} \widetilde{\mathbf{w}}_{n+1}$	Gl. (2.14b)
$\delta_{n+1} = \mathbf{w}_{n+1}^T \mathbf{v}_{n+1}$	Gl. (2.6)
$\gamma_{n+1} = \xi_{n+1} \delta_{n+1} / \delta_n$	Gl. (2.13)

enddo

Algorithmus 2.1 bricht vorzeitig ab, falls im Verlaufe des iterativen Prozesses $\delta_n = 0$ für ein n. Wie man einen solchen Fall behandelt, beschreibt ein späterer Teilabschnitt über Look-Ahead-Techniken. Unter der Annahme, daß kein solcher Abbruch auftritt, faßt der folgende Satz die Ergebnisse der Implementierung des Lanczos–Algorithmus mit Drei-Term-Rekursionen zusammen.

Satz 2.1 (Lanczos–Algorithmus mit Drei-Term-Rekursionen). *Falls Alg. 2.1 bis zum n-ten Schritt nicht abbricht, bilden die Lanczos–Vektoren $\{\mathbf{v}_i\}_{1 \leq i \leq n}$ und $\{\mathbf{w}_i\}_{1 \leq i \leq n}$ in exakter Arithmetik ein biorthogonales System, also*

$$\mathbf{W}_n^T \mathbf{V}_n = \mathbf{D}_n. \tag{2.17}$$

Außerdem bilden diese Lanczos–Vektoren eine Basis der Vektorräume

$$\mathcal{K}_n(\mathbf{A}, \mathbf{v}_1) = \operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}, \tag{2.18}$$
$$\mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1) = \operatorname{span}\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}, \tag{2.19}$$

und es gelten die Beziehungen

$$\mathbf{A}\mathbf{V}_n = \mathbf{V}_n \mathbf{T}_n + \rho_{n+1} \mathbf{v}_{n+1} \mathbf{e}_n^T, \tag{2.20}$$
$$\mathbf{A}^T \mathbf{W}_n = \mathbf{W}_n \mathbf{D}_n^{-1} \mathbf{T}_n^T \mathbf{D}_n + \xi_{n+1} \mathbf{w}_{n+1} \mathbf{e}_n^T, \tag{2.21}$$
$$\mathbf{W}_n^T \mathbf{A} \mathbf{V}_n = \mathbf{D}_n \mathbf{T}_n. \tag{2.22}$$

Beweis. Alle Aussagen zeigt man analog zum Beweis in [62, Proposition 7.1], der eine Implementierung mit orthonormalen Lanczos–Vektoren beschreibt. Abweichend von der Darstellung hier gilt dort $\mathbf{W}_n^T \mathbf{V}_n = \mathbf{I}_n$, wobei $\mathbf{I}_n$ die $n \times n$ Einheitsmatrix bezeichnet. Eine solche Implementierung bietet jedoch lediglich die Möglichkeit, eine der beiden Folgen von Lanczos–Vektoren zu normalisieren. □

2.3.3 Implementierung mit gekoppelten Zwei-Term-Rekursionen

Der klassische Lanczos–Algorithmus, wie er im vorherigen Teilabschnitt beschrieben ist, basiert auf Drei-Term-Rekursionen zur Berechnung der Lanczos–Vektoren. In diesem Abschnitt wird eine unterschiedliche Implementierung skizziert, die auf gekoppelten Zwei-Term-Rekursionen beruht. In exakter Arithmetik sind diese beiden Implementierungen mathematisch äquivalent. Obwohl Lanczos [52] die gekoppelte Zwei-Term-Technik bereits um 1950 benutzte, befaßt sich die Mehrheit der Veröffentlichungen mit der klassischen Drei-Term-Implementierung. Freund und Nachtigal [36] verwenden die Implementierung mit gekoppelten Zwei-Term-Rekursionen als Technik zum Aufspannen von Krylov–Teilräumen in der Methode der quasi-minimalen Residuen zur iterativen Lösung von linearen Gleichungssystemen. Die dort beschriebenen numerischen Experimente zeigen, daß diese Version bezüglich der numerischen Stabilität günstiger als eine dazu korrespondierende Implementierung mit Drei-Term-Rekursionen ist.

Die Implementierung mit gekoppelten Zwei-Term-Rekursionen geht von der Annahme aus, daß die tridiagonale Matrix $\mathbf{T}_n$ aus (2.9) eine LU-Zerlegung der Form

$$\mathbf{T}_n = \mathbf{L}_n \mathbf{U}_n$$

mit bidiagonalen Faktoren

$$\mathbf{L}_n = \begin{bmatrix} \beta_1 & & & \mathbf{0} \\ \rho_2 & \beta_2 & & \\ & \rho_3 & \beta_3 & \\ & & \ddots & \ddots \\ \mathbf{0} & & & \rho_n \quad \beta_n \end{bmatrix} \in \mathbb{C}^{n \times n} \tag{2.23}$$

und

$$\mathbf{U}_n = \begin{bmatrix} 1 & \frac{\xi_2 \delta_2}{\varepsilon_1} & & \mathbf{0} \\ & 1 & \frac{\xi_3 \delta_3}{\varepsilon_2} & \\ & & 1 & \ddots \\ & & & \ddots \quad \frac{\xi_n \delta_n}{\varepsilon_{n-1}} \\ \mathbf{0} & & & 1 \end{bmatrix} \in \mathbb{C}^{n \times n} \tag{2.24}$$

besitzt. Zusätzlich zu dem Paar von Basisvektoren $\{\mathbf{v}_i\}_{i=1,2,3,\ldots}$ und $\{\mathbf{w}_i\}_{i=1,2,3,\ldots}$, die genau wie in der Drei-Term-Implementierung enthalten sind, wird hier ein zweites Paar von Basisvektoren $\{\mathbf{p}_i\}_{i=1,2,3,\ldots}$ und $\{\mathbf{q}_i\}_{i=1,2,3,\ldots}$ generiert, für die die Beziehungen

$$\mathcal{K}_n(\mathbf{A}, \mathbf{v}_1) = \operatorname{span}\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}, \tag{2.25}$$

$$\mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1) = \operatorname{span}\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n\}, \tag{2.26}$$

gelten. Wegen (2.18) und (2.19) stehen damit sowohl für das Aufspannen von $\mathcal{K}_n(\mathbf{A}, \mathbf{v}_1)$ als auch für $\mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1)$ je zwei Sätze linear unabhängiger Vektoren zur Verfügung.

Faßt man die Vektoren $\mathbf{p}_i$ und $\mathbf{q}_i$ ebenfalls als Spalten von $N \times n$ Matrizen

$$\mathbf{P}_n := [\mathbf{p}_1 \ \mathbf{p}_2 \ \cdots \ \mathbf{p}_n] \qquad \text{und} \qquad \mathbf{Q}_n := [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_n]$$

auf, dann lautet die Darstellung mit gekoppelten Zwei-Term-Rekursionen:

$$\mathbf{V}_n = \mathbf{P}_n \mathbf{U}_n, \tag{2.27a}$$

$$\mathbf{A}\mathbf{P}_n = \mathbf{V}_n \mathbf{L}_n + \rho_{n+1} \mathbf{v}_{n+1} \mathbf{e}_n^T, \tag{2.27b}$$

$$\mathbf{W}_n = \mathbf{Q}_n \mathbf{E}_n^{-1} \mathbf{L}_n^T \mathbf{D}_n, \tag{2.27c}$$

$$\mathbf{A}^T \mathbf{Q}_n = \mathbf{W}_n \mathbf{D}_n^{-1} \mathbf{U}_n^T \mathbf{E}_n + \xi_{n+1} \mathbf{w}_{n+1} \mathbf{e}_n^T, \tag{2.27d}$$

wobei die nicht-singuläre $n \times n$ Matrix

$$\mathbf{E}_n = \operatorname{diag}(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$$

benutzt wird.

Zur Herleitung der gekoppelten Zwei-Term-Implementierung vergleicht man in den Gleichungen (2.27) jeweils die n-te Spalte und erhält

$$\mathbf{v}_n = \mathbf{p}_n + \frac{\xi_n \delta_n}{\varepsilon_{n-1}} \mathbf{p}_{n-1}, \tag{2.28a}$$

$$\mathbf{A}\mathbf{p}_n = \rho_{n+1} \mathbf{v}_{n+1} + \beta_n \mathbf{v}_n, \tag{2.28b}$$

$$\mathbf{w}_n = \frac{\beta_n \delta_n}{\varepsilon_n} \mathbf{q}_n + \frac{\rho_n \delta_n}{\varepsilon_{n-1}} \mathbf{q}_{n-1}, \tag{2.28c}$$

$$\mathbf{A}^T \mathbf{q}_n = \xi_{n+1} \mathbf{w}_{n+1} + \frac{\varepsilon_n}{\delta_n} \mathbf{w}_n, \tag{2.28d}$$

wobei $\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{0}$, $\xi_1 = \rho_1 = 0$ und $\varepsilon_0 \neq 0$. Wegen (2.19) und (2.26) sind die Vektoren $\mathbf{q}_j$ als Linearkombination der $\{\mathbf{w}_i\}_{1 \leq i \leq j}$ darstellbar. Unter Verwendung der Biorthogonalität der Lanczos–Vektoren folgt daher aus (2.28c) durch Multiplikation mit $\mathbf{v}_n$ die Beziehung

$$\varepsilon_n = \beta_n \mathbf{q}_n^T \mathbf{v}_n.$$

Damit führt die Multiplikation von (2.28b) mit $\mathbf{q}_n^T$ analog zu

$$\varepsilon_n = \mathbf{q}_n^T \mathbf{A} \mathbf{p}_n. \tag{2.29}$$

Mit den Setzungen

$$\widetilde{\mathbf{v}}_{n+1} := \mathbf{A}\mathbf{p}_n - \beta_n \mathbf{v}_n,$$
$$\widetilde{\mathbf{w}}_{n+1} := \mathbf{A}^T \mathbf{q}_n - \frac{\varepsilon_n}{\delta_n} \mathbf{w}_n,$$

und

$$\beta_n \delta_n / \varepsilon_n = 1 \tag{2.30}$$

folgen aus (2.28) die Darstellungen der Lanczos–Vektoren

$$\mathbf{v}_{n+1} = \widetilde{\mathbf{v}}_{n+1} / \rho_{n+1}, \tag{2.31a}$$

$$\mathbf{w}_{n+1} = \widetilde{\mathbf{w}}_{n+1} / \xi_{n+1}, \tag{2.31b}$$

und die zu (2.28) äquivalenten Darstellungen

$$\mathbf{p}_n = \mathbf{v}_n - \frac{\xi_n \delta_n}{\varepsilon_{n-1}} \mathbf{p}_{n-1}, \tag{2.32a}$$

$$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \beta_n \mathbf{v}_n, \tag{2.32b}$$

$$\mathbf{q}_n = \mathbf{w}_n - \frac{\rho_n \delta_n}{\varepsilon_{n-1}} \mathbf{q}_{n-1}, \tag{2.32c}$$

$$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T \mathbf{q}_n - \beta_n \mathbf{w}_n. \tag{2.32d}$$

Wie im Fall der Drei-Term-Rekursionen erreicht man durch die Wahl der Skalierungsfaktoren

$$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2, \tag{2.33a}$$

$$\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2, \tag{2.33b}$$

eine Normierung der Lanczos–Vektoren auf Einheitslänge. Fügt man relevante, oben aufgeführte Gleichungen in der richtigen Reihenfolge zusammen, so erhält man den folgenden Algorithmus, der in [36, Alg. 3.2] auf eine andere Weise hergeleitet wird.

ALGORITHMUS 2.2 (LANCZOS–ALGORITHMUS MIT ZWEI-TERM-REKURSIONEN)

Wähle $\mathbf{v}_1, \mathbf{w}_1 \in \mathbb{C}^N$ so, daß $\|\mathbf{v}_1\|_2 = \|\mathbf{w}_1\|_2 = 1$ und $\mathbf{w}_1^T \mathbf{v}_1 \neq 0$
Setze $\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{0}$ und $\xi_1 = \rho_1 = 0, \varepsilon_0 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**

$\delta_n = \mathbf{w}_n^T \mathbf{v}_n$	Gl. (2.6)
$\mathbf{p}_n = \mathbf{v}_n - \frac{\xi_n \delta_n}{\varepsilon_{n-1}} \mathbf{p}_{n-1}$	Gl. (2.32a)
$\mathbf{q}_n = \mathbf{w}_n - \frac{\rho_n \delta_n}{\varepsilon_{n-1}} \mathbf{q}_{n-1}$	Gl. (2.32c)
$\varepsilon_n = \mathbf{q}_n^T \mathbf{A} \mathbf{p}_n$	Gl. (2.29)
$\beta_n = \varepsilon_n / \delta_n$	Gl. (2.30)
$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \beta_n \mathbf{v}_n$	Gl. (2.32b)
$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T \mathbf{q}_n - \beta_n \mathbf{w}_n$	Gl. (2.32d)
$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$	Gl. (2.33a)
$\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$	Gl. (2.33b)
$\mathbf{v}_{n+1} = \frac{1}{\rho_{n+1}} \widetilde{\mathbf{v}}_{n+1}$	Gl. (2.31a)
$\mathbf{w}_{n+1} = \frac{1}{\xi_{n+1}} \widetilde{\mathbf{w}}_{n+1}$	Gl. (2.31b)

enddo

Der folgende Satz sammelt die Ergebnisse der Implementierung mit *gekoppelten Zwei-Term-Rekursionen*.

Satz 2.2 (Lanczos–Algorithmus mit gekoppelten Zwei-Term-Rekursionen). *Falls Alg. 2.2 bis zum n-ten Schritt nicht abbricht, bilden die Lanczos–Vektoren $\{\mathbf{v}_i\}_{1 \leq i \leq n}$ und $\{\mathbf{w}_i\}_{1 \leq i \leq n}$ in exakter Arithmetik ein biorthogonales System, also*

$$\mathbf{W}_n^T \mathbf{V}_n = \mathbf{D}_n. \tag{2.34}$$

Außerdem bilden neben diesen Lanczos–Vektoren auch die Vektoren $\{\mathbf{p}_i\}_{1 \leq i \leq n}$ und $\{\mathbf{q}_i\}_{1 \leq i \leq n}$ eine Basis der Vektorräume

$$\mathcal{K}_n(\mathbf{A}, \mathbf{v}_1) = \operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} = \operatorname{span}\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}, \tag{2.35}$$

$$\mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1) = \operatorname{span}\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\} = \operatorname{span}\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n\}, \tag{2.36}$$

und es gelten die Beziehungen

$$\mathbf{V}_n = \mathbf{P}_n \mathbf{U}_n, \tag{2.37}$$

$$\mathbf{A}\mathbf{P}_n = \mathbf{V}_n \mathbf{L}_n + \rho_{n+1} \mathbf{v}_{n+1} \mathbf{e}_n^T, \tag{2.38}$$

$$\mathbf{W}_n = \mathbf{Q}_n \mathbf{E}_n^{-1} \mathbf{L}_n^T \mathbf{D}_n, \tag{2.39}$$

$$\mathbf{A}^T \mathbf{Q}_n = \mathbf{W}_n \mathbf{D}_n^{-1} \mathbf{U}_n^T \mathbf{E}_n + \xi_{n+1} \mathbf{w}_{n+1} \mathbf{e}_n^T, \tag{2.40}$$

$$\mathbf{W}_n^T \mathbf{A} \mathbf{V}_n = \mathbf{D}_n \mathbf{L}_n \mathbf{U}_n, \tag{2.41}$$

$$\mathbf{Q}_n^T \mathbf{A} \mathbf{P}_n = \mathbf{E}_n. \tag{2.42}$$

Beweis. Die Biorthogonalität (2.34) wird durch Rückführung auf die Drei-Term-Implementierung analog zum Beweis in [10, Theorem 3.1] gezeigt. Gleichungen (2.35)–(2.40) gelten aufgrund der oben gezeigten Konstruktion des Algorithmus. Gleichung (2.41) folgt aus (2.37), (2.38) und der Biorthogonalität der Lanczos–Vektoren, nämlich

$$\begin{aligned}\mathbf{W}_n^T\mathbf{A}\mathbf{V}_n &= \mathbf{W}_n^T\mathbf{A}\mathbf{P}_n\mathbf{U}_n \\ &= \mathbf{W}_n^T\mathbf{V}_n\mathbf{L}_n\mathbf{U}_n + \rho_{n+1}\mathbf{W}_n^T\mathbf{v}_{n+1}\mathbf{e}_n^T\mathbf{U}_n \\ &= \mathbf{D}_n\mathbf{L}_n\mathbf{U}_n.\end{aligned}$$

Gleichung (2.42) folgt aus (2.39), (2.37) und der soeben gezeigten Beziehung (2.41):

$$\begin{aligned}\mathbf{Q}_n^T\mathbf{A}\mathbf{P}_n &= (\mathbf{W}_n\mathbf{D}_n^{-1}\mathbf{L}_n^{-T}\mathbf{E}_n)^T\mathbf{A}\mathbf{V}_n\mathbf{U}_n^{-1} \\ &= \mathbf{E}_n\mathbf{L}_n^{-1}\mathbf{D}_n^{-1}\mathbf{W}_n^T\mathbf{A}\mathbf{V}_n\mathbf{U}_n^{-1} \\ &= \mathbf{E}_n\mathbf{L}_n^{-1}\mathbf{D}_n^{-1}\mathbf{D}_n\mathbf{L}_n\mathbf{U}_n\mathbf{U}_n^{-1} \\ &= \mathbf{E}_n.\end{aligned}$$

□

2.3.4 Look-Ahead-Techniken

Die in den Algorithmen 2.1 und 2.2 vorgestellten Implementierungen des Lanczos–Algorithmus können durch *Look-Ahead-Techniken* gegen vorzeitigen Abbruch geschützt werden. In exakter Arithmetik tritt ein vorzeitiger Abbruch beispielsweise in der in Alg. 2.1 beschriebenen Drei-Term-Implementierung dann auf, wenn $\mathbf{w}_{n+1}^T\mathbf{v}_{n+1} = 0$ für ein n. Diese Situation kann zwei unterschiedliche Ursachen besitzen. Erstens kann $\mathbf{v}_{n+1} = \mathbf{0}$ oder $\mathbf{w}_{n+1} = \mathbf{0}$ sein. In diesem Fall stoppt der Lanczos–Algorithmus regulär mit den dann invarianten Teilräumen $\mathcal{K}_n(\mathbf{A}, \mathbf{v}_1)$ oder $\mathcal{K}_n\left(\mathbf{A}^T, \mathbf{w}_1\right)$. Zweitens scheitert der Algorithmus, falls

$$\mathbf{w}_{n+1}^T\mathbf{v}_{n+1} = 0, \qquad \text{aber} \qquad \mathbf{v}_{n+1} \neq \mathbf{0}, \mathbf{w}_{n+1} \neq \mathbf{0}.$$

Neben diesem Fall, der als *serious breakdown* bezeichnet wird, können in endlicher Arithmetik sogenannte *near breakdowns* auftreten, in denen das Produkt $\mathbf{w}_{n+1}^T\mathbf{v}_{n+1}$ zwar von Null verschieden aber sehr klein ist.

Die Grundidee der Look-Ahead-Techniken basiert darauf, daß man oft das Paar von Lanczos–Vektoren $\mathbf{v}_{n+2}$ und $\mathbf{w}_{n+2}$ angeben kann, obwohl das Paar $\mathbf{v}_{n+1}$ und $\mathbf{w}_{n+1}$ nicht definiert ist. Falls das Paar $\mathbf{v}_{n+2}$ und $\mathbf{w}_{n+2}$ ebenfalls nicht definiert ist, geht man unmittelbar zum Paar $\mathbf{v}_{n+3}$ und $\mathbf{w}_{n+3}$ über und setzt diese Vorgehensweise solange sukzessive fort, bis ein Paar von Lanczos–Vektoren existiert. In formaler Notation faßt man die Lanczos–Vektoren $\{\mathbf{v}_i\}_{1\leq i\leq n}$ und $\{\mathbf{w}_i\}_{1\leq i\leq n}$ jeweils in k Blöcke wie folgt zusammen:

$$\widehat{\mathbf{V}}_l = \left[\mathbf{v}_{n_l}\ \mathbf{v}_{n_l+1}\ \cdots\ \mathbf{v}_{n_{l+1}-1}\right], \quad \widehat{\mathbf{W}}_l = \left[\mathbf{w}_{n_l}\ \mathbf{w}_{n_l+1}\ \cdots\ \mathbf{w}_{n_{l+1}-1}\right], \qquad l = 1, 2, \ldots, k-1,$$

und

$$\widehat{\mathbf{V}}_k = \left[\mathbf{v}_{n_k}\ \mathbf{v}_{n_k+1}\ \cdots\ \mathbf{v}_n\right], \qquad \widehat{\mathbf{W}}_k = \left[\mathbf{w}_{n_k}\ \mathbf{w}_{n_k+1}\ \cdots\ \mathbf{w}_n\right],$$

wobei

$$1 = n_1 < n_2 < \cdots < n_l < \cdots < n_k \leq n < n_{k+1}.$$

Dabei beginnt man genau dann zwei neue Blöcke $\widehat{\mathbf{V}}_l$ und $\widehat{\mathbf{W}}_l$, wenn ein Paar von Lanczos–Vektoren existiert. Erst wenn ein Paar von Lanczos–Vektoren nicht definiert ist, beginnt man mit dem Auffüllen der Blöcke $\widehat{\mathbf{V}}_l$ und $\widehat{\mathbf{W}}_l$ zu Matrizen mit mehr als einer Spalte. Die mit dieser Look-Ahead-Technik erzeugten Vektoren $\mathbf{v}_i$ und $\mathbf{w}_i$ sind dann innerhalb ihrer Blöcke nicht mehr biorthogonal, sondern lediglich biorthogonal zu allen Vektoren aus ihren Vorgängerblöcken

$$\widehat{\mathbf{W}}_i^T \widehat{\mathbf{V}}_j = \begin{cases} \mathbf{0}, & \text{falls } i \neq j, \\ \widehat{\mathbf{D}}_i, & \text{falls } i = j, \end{cases}$$

wobei das Symbol $\widehat{\mathbf{D}}_i$ eine nicht-singuläre Matrix bezeichnet. Nähere Informationen zu dieser vorgestellten Look-Ahead-Technik findet man in [29, 30, 32, 33, 34, 35]. Weitere Look-Ahead-Techniken sind in [6, 48, 60, 61, 72] beschrieben. Gutknecht gibt in [42, 43, 44] eine umfangreiche Beschreibung des Lanczos–Algorithmus.

Abschließend sei bemerkt, daß die Matrizen $\mathbf{T}_n$ und $\mathbf{L}_n$ aus der Drei-Term- und der gekoppelten Zwei-Term-Implementierung bei Verwendung der hier vorgestellten Look-Ahead-Technik nicht mehr tridiagonal und bidiagonal sind. Stattdessen sind $\mathbf{T}_n$ und $\mathbf{L}_n$ obere Hessenberg–Matrizen, für die

$$\mathbf{T}_n \quad \text{block-tridiagonal} \qquad \text{und} \qquad \mathbf{L}_n \quad \text{block-bidiagonal} \tag{2.43}$$

gilt. Damit steigt die Länge der zur Berechnung der Lanczos–Vektoren benötigten Rekursionen in Abhängigkeit der Länge der Blöcke $\widehat{\mathbf{V}}_l$ und $\widehat{\mathbf{W}}_l$ an.

2.4 Verfahren der quadrierten konjugierten Gradienten

In diesem Abschnitt wird das Verfahren der quadrierten konjugierten Gradienten (**C**onjugate **G**radients **S**quared, CGS) [66] zur iterativen Lösung von linearen Gleichungssystemen betrachtet. Dieses Verfahren gehört zu der Klasse der Krylov–Teilraumverfahren und spannt in einem speziellen Teil die Krylov–Teilräume auf. Wird dieser Teil von dem restlichen Prozeß, der die aktuelle Iterierte von CGS bestimmt, getrennt, kann der verbleibende Teil von CGS in einem anderen Krylov–Teilraumverfahren wiederverwendet werden, dessen Iterierte nach einem anderen Kriterium definiert sind. Nach einer knappen Darstellung von CGS in seiner ursprünglichen Form, werden daran im darauffolgenden Teilabschnitt Änderungen so vorgenommen, daß der entstehende iterative Prozeß zur Erzeugung von Krylov–Teilräumen in späteren Kapiteln unmittelbar zur Verfügung steht.

2.4.1 CGS als iterativer Löser

Die bekannte Methode der konjugierten Gradienten (**C**onjugate **G**radients, CG) [46] ist ein mächtiges Hilfsmittel zur Lösung von linearen Gleichungssystemen mit symmetrisch positiv definiter Koeffizientenmatrix. Eine Erweiterung auf allgemeine unsymmetrische Systeme ist die Methode der bikonjugierten Gradienten (**BiC**onjugate **G**radients, BCG) [24, 52], dessen Residuum von der Form

$$\mathbf{r}_n^{\text{BCG}} = \varphi_n(\mathbf{A})\mathbf{r}_0 \qquad \text{mit} \qquad \varphi_n \in \mathcal{P}_n \qquad \text{und} \qquad \varphi_n(0) = 1$$

ist. Quadriert man das Residualpolynom φ_n von BCG und faßt das so entstandene Polynom als Residualpolynom einer dadurch neudefinierten Methode auf, so gelangt man zu der Methode der quadrierten konjugierten Gradienten (**C**onjugate **G**radients **S**quared, CGS) [66]. Das Residuum von CGS ist daher durch

$$\mathbf{r}_n^{\text{CGS}} = \big(\varphi_n(\mathbf{A})\big)^2 \mathbf{r}_0 \tag{2.44}$$

gegeben. Der resultierende Prozeß ist in dem folgenden Algorithmus notiert.

ALGORITHMUS 2.3 (CGS)

Wähle $\mathbf{x}_0 \in \mathbb{C}^N$ und setze $\mathbf{u}_0 = \mathbf{p}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v}_0 = \mathbf{A}\mathbf{p}_0$
Wähle $\tilde{\mathbf{r}}_0$ so, daß $\rho_0 = \tilde{\mathbf{r}}_0^T \mathbf{r}_0 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**

$$\sigma_{n-1} = \tilde{\mathbf{r}}_0^T \mathbf{v}_{n-1}$$
$$\alpha_{n-1} = \rho_{n-1} / \sigma_{n-1}$$
$$\mathbf{q}_n = \mathbf{u}_{n-1} - \alpha_{n-1}\, \mathbf{v}_{n-1}$$
$$\mathbf{x}_n = \mathbf{x}_{n-1} + \alpha_{n-1} \left(\mathbf{u}_{n-1} + \mathbf{q}_n\right)$$
$$\mathbf{r}_n = \mathbf{r}_{n-1} - \alpha_{n-1} \mathbf{A} \left(\mathbf{u}_{n-1} + \mathbf{q}_n\right)$$
$$\rho_n = \tilde{\mathbf{r}}_0^T \mathbf{r}_n$$
$$\beta_n = \rho_n / \rho_{n-1}$$
$$\mathbf{u}_n = \mathbf{r}_n + \beta_n \mathbf{q}_n$$
$$\mathbf{p}_n = \mathbf{u}_n + \beta_n \left(\mathbf{q}_n + \beta_n \mathbf{p}_{n-1}\right)$$
$$\mathbf{v}_n = \mathbf{A}\mathbf{p}_n$$

enddo

In exakter Arithmetik kann CGS für eine allgemeine Matrix $\mathbf{A}$ abbrechen, wenn $\sigma_{n-1} = 0$ oder $\rho_{n-1} = 0$ für ein n. In der Praxis sind solche Abbrüche, die durch Look-Ahead-Techniken [4, 5] vermieden werden können, aber selten. Hier wird angenommen, daß solche Abbrüche nicht auftreten. Damit wird die gleiche Annahme wie in [28] getroffen, in der Alg. 2.3 als zugrundeliegende Technik in der transpositionsfreien Methode der quasi-minimalen Residuen (TFQMR) verwendet wird. Ein weiteres Argument gegen den Einsatz von Look-Ahead-Techniken ist die Tatsache, daß derzeit keine Implementierung von TFQMR mit Look-Ahead bekannt ist, ohne dabei die Anzahl der Matrix-Vektor-Produkte eines Iterationsschrittes signifikant zu erhöhen.* Unter der Annahme, daß kein Abbruch auftritt, gilt

$$\alpha_{n-1} \neq 0 \qquad \text{für alle } n. \tag{2.45}$$

2.4.2 CGS als Erzeuger von Krylov–Teilräumen

Will man CGS als Technik zur Erzeugung von Krylov–Teilräumen benutzen, sind nicht alle Anweisungen von Alg. 2.3 notwendig. Darüber hinaus ist eine reformulierte Version von CGS für spätere Zwecke günstiger. Dazu wird mit den in Alg. 2.3 enthaltenen Suchrichtungen $\mathbf{u}_{n-1}$ und $\mathbf{q}_n$ der Vektor

$$\mathbf{y}_m = \begin{cases} \mathbf{u}_{n-1}, & \text{falls } m = 2n-1, \\ \mathbf{q}_n, & \text{falls } m = 2n. \end{cases}$$

eingeführt. Unter Verwendung des Residualpolynoms φ_n von BCG wird der Vektor

$$\mathbf{w}_m = \begin{cases} \mathbf{r}_0, & \text{falls } m = 1, \\ \varphi_n(\mathbf{A})\varphi_{n-1}(\mathbf{A})\mathbf{r}_0, & \text{falls } m = 2n, \\ \left(\varphi_n(\mathbf{A})\right)^2 \mathbf{r}_0, & \text{falls } m = 2n+1, \end{cases} \tag{2.46}$$

*R.W. Freund, Harburger Sommerschule "Numerische Lineare Algebra", 26.–30.8.1996.

definiert. In [28] wird zwischen den Vektoren $\mathbf{y}_m$ und $\mathbf{w}_m$ der Zusammenhang

$$\alpha_{\lfloor (m-1)/2 \rfloor} \mathbf{A}\mathbf{y}_m = \mathbf{w}_m - \mathbf{w}_{m+1}$$

hergeleitet. Wegen (2.45) gilt mit den $N \times m$ Matrizen

$$\mathbf{Y}_m := [\mathbf{y}_1\ \mathbf{y}_2\ \cdots\ \mathbf{y}_m] \qquad \text{und} \qquad \mathbf{W}_m := [\mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_m]$$

die Matrixgleichung

$$\mathbf{A}\mathbf{Y}_m = \mathbf{W}_{m+1}\widetilde{\mathbf{B}}_m,$$

wobei

$$\widetilde{\mathbf{B}}_m = \begin{bmatrix} 1 & & \mathbf{0} & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & \mathbf{0} & -1 & 1 \\ & & & -1 \end{bmatrix} \left(\operatorname{diag}\left(\alpha_0, \alpha_0, \alpha_1, \alpha_1, \ldots, \alpha_{\lfloor (m-1)/2 \rfloor}\right)\right)^{-1} \tag{2.47}$$

eine $(m+1) \times m$ untere Bidiagonalmatrix bezeichnet. Mit diesen neuen Bezeichnungen erhält man den folgenden Algorithmus.

ALGORITHMUS 2.4 (ERZEUGUNG VON KRYLOV–TEILRÄUMEN MIT CGS)

Wähle $\mathbf{x}_0 \in \mathbb{C}^N$ und setze $\mathbf{w}_1 = \mathbf{y}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v}_0 = \mathbf{A}\mathbf{y}_1$
Wähle $\tilde{\mathbf{r}}_0 \in \mathbb{C}^N$ so, daß $\rho_0 = \tilde{\mathbf{r}}_0^T \mathbf{r}_0 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**
  $\sigma_{n-1} = \tilde{\mathbf{r}}_0^T \mathbf{v}_{n-1}$
  $\alpha_{n-1} = \rho_{n-1} / \sigma_{n-1}$
  $\mathbf{y}_{2n} = \mathbf{y}_{2n-1} - \alpha_{n-1}\mathbf{v}_{n-1}$
  for $m = 2n-1, 2n$ **do**
    $\mathbf{w}_{m+1} = \mathbf{w}_m - \alpha_{n-1}\mathbf{A}\mathbf{y}_m$
  enddo
  $\rho_n = \tilde{\mathbf{r}}_0^T \mathbf{w}_{2n+1}$
  $\beta_n = \rho_n / \rho_{n-1}$
  $\mathbf{y}_{2n+1} = \mathbf{w}_{2n+1} + \beta_n \mathbf{y}_{2n}$
  $\mathbf{v}_n = \mathbf{A}\mathbf{y}_{2n+1} + \beta_n \left(\mathbf{A}\mathbf{y}_{2n} + \beta_n \mathbf{v}_{n-1}\right)$
enddo

In diesem Algorithmus ist die Berechnung der CGS–Iterierten eliminiert; das Residuum von CGS ist wegen (2.46) und (2.44) in der Form

$$\mathbf{w}_{2n+1} = \mathbf{r}_n \tag{2.48}$$

im Algorithmus weiterhin vorhanden.

In dem folgenden Satz sind die Ergebnisse für CGS gesammelt, auf die später Bezug genommen wird.

Satz 2.3 (CGS). *Falls Alg. 2.4 bis zum m-ten Schritt nicht abbricht, bilden die Vektoren $\{\mathbf{y}_i\}_{1\leq i\leq m}$ in exakter Arithmetik eine Basis des Vektorraums*

$$\mathcal{K}_m(\mathbf{A},\mathbf{r}_0) = \operatorname{span}\{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_m\} \tag{2.49}$$

und es gilt die Beziehung

$$\mathbf{A}\mathbf{Y}_m = \mathbf{W}_{m+1}\widetilde{\mathbf{B}}_m \tag{2.50}$$

mit $\widetilde{\mathbf{B}}_m$ aus (2.47)

Beweis. Die in diesem Abschnitt vorgestellte Darstellung von CGS folgt der Arbeit von Freund [28], in der alle Beziehungen gezeigt werden. □

2.5 Verfahren der bikonjugierten stabilisierten Residuen

Analog zur Vorgehensweise im vorherigen Abschnitt, in dem CGS beschrieben ist, wird hier ein anderes iteratives Verfahren zur Lösung von linearen Gleichungssystemen auf den Einsatz für die Erzeugung von Krylov–Teilräumen vorbereitet. Dabei werden ausschließlich reelle Systeme betrachtet.

2.5.1 Bi-CGSTAB als iterativer Löser

CGS ist für eine gewisse Klasse von unsymmetrischen linearen Gleichungssystemen eine attraktive Methode, die einen engen Zusammenhang zu BCG aufweist. Als eine weitere Variante von BCG kann die Methode der bikonjugierten stabilisierten Residuen (**BiC**onjugate **G**radient **STAB**ilized, Bi-CGSTAB) [67] aufgefaßt werden. Die Motivation zum Entwurf von Bi-CGSTAB liegt darin, die günstige Konvergenzgeschwindigkeit von CGS beizubehalten und die starken Oszillationen von CGS in der Residuumsnorm durch ein glatteres Konvergenzverhalten zu ersetzen. Die enge Verwandschaft der drei erwähnten Methoden spiegelt sich in deren Residualpolynomen wider. Bezeichnet $\varphi_n \in \mathcal{P}_n$ mit $\varphi_n(0) = 1$ das Residualpolynom von BCG, also

$$\mathbf{r}_n^{\mathrm{BCG}} = \varphi_n(\mathbf{A})\mathbf{r}_0,$$

dann ist CGS durch

$$\mathbf{r}_n^{\mathrm{CGS}} = \varphi_n(\mathbf{A})\varphi_n(\mathbf{A})\mathbf{r}_0$$

charakterisiert. In Bi-CGSTAB setzt man

$$\mathbf{r}_n = \phi_n(\mathbf{A})\varphi_n(\mathbf{A})\mathbf{r}_0,$$

wobei ein weiteres Polynom

$$\phi_n(\lambda) = \prod_{i=1}^{n}(1-\eta_i\lambda) \in \mathcal{P}_n, \qquad \phi_n(0) = 1,$$

mit geeigneten Parametern η_i für $i = 1,2,\ldots,n$ eingeführt wird. Die Parameter η_i wählt man im Verlauf des iterativen Prozesses nach einem Prinzip des lokalen steilsten Abstieges. Da nämlich das Polynom $\phi_n(\lambda)$ im n-ten Iterationsschritt um den linearen Faktor $1-\eta_n\lambda$ ergänzt wird, bietet sich die Wahl von η_n so an, daß das aktuelle Residuum $\mathbf{r}_n$ als Funktion von η_n in der euklidischen Norm minimiert wird. Der resultierende Prozeß ist nachfolgend abgebildet.

ALGORITHMUS 2.5 (BI-CGSTAB)

Wähle $\mathbf{x}_0 \in \mathbb{R}^N$ und setze $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$
Wähle $\tilde{\mathbf{r}}_0 \in \mathbb{R}^N$ so, daß $\tilde{\mathbf{r}}_0^T \mathbf{r}_0 \neq 0$ und setze $\rho_0 = \alpha_0 = \eta_0 = 1$
for $n = 1, 2, 3, \ldots$ **do**

$$\rho_n = \tilde{\mathbf{r}}_0^T \mathbf{r}_{n-1}$$
$$\beta_n = (\rho_n / \rho_{n-1})(\alpha_{n-1} / \eta_{n-1})$$
$$\mathbf{p}_n = \mathbf{r}_{n-1} + \beta_n (\mathbf{p}_{n-1} - \eta_{n-1} \mathbf{v}_{n-1})$$
$$\mathbf{v}_n = \mathbf{A}\mathbf{p}_n$$
$$\alpha_n = \rho_n / \tilde{\mathbf{r}}_0^T \mathbf{v}_n$$
$$\mathbf{s}_n = \mathbf{r}_{n-1} - \alpha_n \mathbf{v}_n$$
$$\mathbf{t}_n = \mathbf{A}\mathbf{s}_n$$
$$\eta_n = \mathbf{t}_n^T \mathbf{s}_n / \mathbf{t}_n^T \mathbf{t}_n$$
$$\mathbf{x}_n = \mathbf{x}_{n-1} + \alpha_n \mathbf{p}_n + \eta_n \mathbf{s}_n$$
$$\mathbf{r}_n = \mathbf{s}_n - \eta_n \mathbf{t}_n$$

enddo

In exakter Arithmetik bricht Alg. 2.5 nach $n_* \leq N$ Iterationsschritten mit der Lösung des Gleichungssystems ab. In diesem Falle ist $\mathbf{s}_{n_*} = \mathbf{0}$ und damit ist η_{n_*} nicht definiert [67]. In endlicher Arithmetik können weitere Abbrüche auftreten, die jedoch nur äußerst selten auftreten und deshalb nicht näher betrachtet werden. Stattdessen wird

$$\alpha_n \neq 0 \qquad \text{und} \qquad \eta_n \neq 0 \qquad \text{für alle } n \tag{2.51}$$

angenommen. Diese Annahme liegt beispielsweise ebenfalls der Arbeit [13] zugrunde.

Eine weitere Eigenschaft, auf die im nächsten Teilabschnitt Bezug genommen wird, bezieht sich auf die Vektoren $\mathbf{s}_n$ und $\mathbf{p}_n$. Man kann leicht zeigen, daß die zu diesen Vektoren korrespondierenden Polynome, die mit $\widetilde{\psi}_{2n-1} \in \mathcal{P}_{2n-1}$ und $\widehat{\psi}_{2n-2} \in \mathcal{P}_{2n-2}$ bezeichnet werden, die Eigenschaft

$$\mathbf{s}_n = \widetilde{\psi}_{2n-1}(\mathbf{A})\mathbf{r}_0 \qquad \text{mit} \qquad \mathrm{Grad}(\widetilde{\psi}_{2n-1}) = 2n-1, \tag{2.52a}$$
$$\mathbf{p}_n = \widehat{\psi}_{2n-2}(\mathbf{A})\mathbf{r}_0 \qquad \text{mit} \qquad \mathrm{Grad}(\widehat{\psi}_{2n-2}) = 2n-2, \tag{2.52b}$$

besitzen, also vollen Grad haben.

2.5.2 Bi-CGSTAB als Erzeuger von Krylov–Teilräumen

Will man Bi-CGSTAB nicht zur Lösung von Gleichungssystemen sondern lediglich als iterativen Prozeß zur Erzeugung von Krylov–Teilräumen benutzen, ist die Berechnung der Bi-CGSTAB–Iterierten $\mathbf{x}_n$ nicht notwendig. Der oben dargestellte Algorithmus wird daher nun wie folgt umgeformt.

Mit den Definitionen

$$\mathbf{y}_m = \begin{cases} \mathbf{p}_n, & \text{falls } m = 2n-1, \\ \mathbf{s}_n, & \text{falls } m = 2n, \end{cases} \tag{2.53}$$

$$\mathbf{w}_m = \begin{cases} \mathbf{r}_{n-1}, & \text{falls } m = 2n-1, \\ \mathbf{s}_n, & \text{falls } m = 2n, \end{cases} \tag{2.54}$$

und

$$\sigma_m = \begin{cases} \alpha_n, & \text{falls } m = 2n-1, \\ \eta_n, & \text{falls } m = 2n, \end{cases} \tag{2.55}$$

werden die Vektoren $\mathbf{p}_n, \mathbf{s}_n$ and $\mathbf{r}_{n-1}$ sowie die Skalare α_n und η_n in Alg. 2.5 ersetzt. Es sei daraufhingewiesen, daß damit

$$\mathbf{y}_{2n} = \mathbf{w}_{2n} \tag{2.56}$$

gilt. Die in Alg. 2.5 enthaltenen Beziehungen $\mathbf{s}_n = \mathbf{r}_{n-1} - \alpha_n \mathbf{v}_n$ und $\mathbf{r}_n = \mathbf{s}_n - \eta_n \mathbf{t}_n$ sind mit den eingeführten Bezeichnungen durch die Formel

$$\sigma_m \mathbf{A}\mathbf{y}_m = \mathbf{w}_m - \mathbf{w}_{m+1} \tag{2.57}$$

beschrieben. Um den Verlauf des gesamten Iterationsprozesses in kompakter Matrixnotation darzustellen, werden die zwei $N \times m$ Matrizen

$$\mathbf{Y}_m := [\mathbf{y}_1\ \mathbf{y}_2\ \cdots\ \mathbf{y}_m] \qquad \text{und} \qquad \mathbf{W}_m := [\mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_m]$$

sowie die $m \times m$ Diagonalmatrix

$$\mathbf{D}_m := \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_m)$$

eingeführt. Wegen (2.55) und (2.51) ist $\mathbf{D}_m$ nicht singulär. Deshalb lautet (2.57) in Matrixnotation

$$\mathbf{A}\mathbf{Y}_m = \mathbf{W}_{m+1}\widetilde{\mathbf{B}}_m,$$

wobei

$$\widetilde{\mathbf{B}}_m = \begin{bmatrix} 1 & & \mathbf{0} \\ -1 & 1 & \\ & \ddots & \ddots \\ \mathbf{0} & & -1 \quad 1 \\ & & \quad -1 \end{bmatrix} \mathbf{D}_m^{-1} \tag{2.58}$$

eine $(m+1) \times m$ untere Bidiagonalmatrix ist. Mit den neuen Bezeichnungen ist die reformulierte Version ohne die Berechnung der Iterierten durch den folgenden Prozeß gegeben.

ALGORITHMUS 2.6 (ERZEUGUNG VON KRYLOV–TEILRÄUMEN MIT BI-CGSTAB)

Wähle $\mathbf{x}_0 \in \mathbb{R}^N$ und setze $\mathbf{w}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{y}_{-1} = \mathbf{0}$
Wähle $\tilde{\mathbf{r}}_0 \in \mathbb{R}^N$ so, daß $\tilde{\mathbf{r}}_0^T \mathbf{r}_0 \neq 0$ und setze $\rho_0 = \sigma_{-1} = \sigma_0 = 1$
for $n = 1, 2, 3, \ldots$ **do**
  $\rho_n = \tilde{\mathbf{r}}_0^T \mathbf{w}_{2n-1}$
  $\beta_n = (\rho_n / \rho_{n-1})(\sigma_{2n-3} / \sigma_{2n-2})$
  $\mathbf{y}_{2n-1} = \mathbf{w}_{2n-1} + \beta_n (\mathbf{y}_{2n-3} - \sigma_{2n-2} \mathbf{A}\mathbf{y}_{2n-3})$
  for $m = 2n-1, 2n$ **do**
    if $m = 2n$ **then** $\mathbf{y}_m = \mathbf{w}_m$ **endif**

$$\sigma_m = \begin{cases} \rho_n / \tilde{\mathbf{r}}_0^T \mathbf{A}\mathbf{y}_m & \textbf{if } m = 2n-1 \\ (\mathbf{A}\mathbf{y}_m)^T \mathbf{w}_m / (\mathbf{A}\mathbf{y}_m)^T \mathbf{A}\mathbf{y}_m & \textbf{if } m = 2n \end{cases}$$

    $\mathbf{w}_{m+1} = \mathbf{w}_m - \sigma_m \mathbf{A}\mathbf{y}_m$
  enddo
enddo

Um die Formulierung des Prozesses erheblich zu verkürzen, sind in der Darstellung von Alg. 2.6 die Vektoren $\mathbf{y}_{2n}$ enthalten. Wegen (2.56) sind diese Vektoren identisch mit den Vektoren $\mathbf{w}_{2n}$, so daß für die Vektoren $\mathbf{y}_{2n}$ kein zusätzlicher Speicherplatz benötigt wird.

Zusammenfassend gilt der folgende Satz.

Satz 2.4 (Bi-CGSTAB). *Falls Alg. 2.6 bis zum m-ten Schritt nicht abbricht, bilden die Vektoren $\{\mathbf{y}_i\}_{1 \leq i \leq m}$ in exakter Arithmetik eine Basis des Vektorraums*

$$\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0) = \operatorname{span}\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\} \tag{2.59}$$

und es gilt die Beziehung

$$\mathbf{A}\mathbf{Y}_m = \mathbf{W}_{m+1} \widetilde{\mathbf{B}}_m \tag{2.60}$$

mit $\widetilde{\mathbf{B}}_m$ aus (2.58).

Beweis. Da die Polynome in (2.52) vollen Grad besitzen, gilt (2.59) wegen der Definition (2.53). □

Kapitel 3

Entwurf paralleler iterativer Verfahren im Isoeffizienzmodell

Zur Erzeugung von Krylov–Teilräumen stellt das vorangehende Kapitel unterschiedliche iterative Verfahren vor. Da iterativen Verfahren eine serielle Abarbeitung der einzelnen Iterationen innewohnt, erfolgt ihre Implementierung auf Parallelrechnern durch parallele Ausführung der in einem Iterationsschritt auftretenden einzelnen Operationen. Diese grundlegende Technik wird in zwei prinzipiell voneinander unterschiedlichen Konzepten der Parallelisierung solcher Methoden eingesetzt. Erstens werden traditionelle sequentielle Algorithmen in ihrer ursprünglichen oder leicht modifizierten Form verwendet, wobei die vorgegebene serielle Struktur des Algorithmus übernommen wird. Der Vorteil dieses Konzeptes besteht darin, daß die dabei entstehenden Implementierungen einfach zu handhaben sind und bei deren Analyse auf Eigenschaften des sequentiellen Algorithmus verwiesen werden kann. Unglücklicherweise erreichen traditionelle Algorithmen, die für serielle Rechner entwickelt wurden, oft bei weitem nicht die erhoffte Effizienz, wenn sie auf Parallelrechner portiert werden. Dies ist der Grund für das Aufkommen eines zweiten Konzeptes, bei dem bereits in der Entwurfsphase des Algorithmus parallele Gesichtspunkte im Vordergrund stehen. Dem Vorteil einer potentiellen Verbesserung der dabei zu erreichenden Effizienz steht eine Erhöhung des Entwicklungsaufwandes gegenüber. Außerdem ist eine Analyse der neuen Algorithmen notwendig, bei der häufig ein Verlust der numerischen Stabiliät beobachtet wird [19].

Dieses Kapitel stellt ein bekanntes Modell zur Analyse von parallelen Algorithmen vor, mit dem sogenannte *Isoeffizienzanalysen* durchgeführt werden. Dabei wird die Architektur des Pararallelrechners, auf dem der Algorithmus implementiert wird, mit berücksichtigt. Dieses Modell wird auf die iterativen Methoden zur Erzeugung von Krylov–Teilräumen angewendet, die im vorherigen Kapitel vorgestellt wurden. Dazu werden in einem vorgezogenem Abschnitt zunächst die in einem Iterationsschritt auftretenden Operationen hinsichtlich sowohl der Datenverteilung auf einem Parallelrechner mit verteiltem Speicher betrachtet als auch die bei der parallelen Berechnung anfallenden Kommunikationszeiten modelliert. Aus einer Isoeffizienzanalyse eines einzelnen Iterationsschrittes werden dann Anforderungen an den Entwurf von parallelen iterativen Methoden hergeleitet. Die so gewonnenen Erkenntnisse werden dann im nächsten Kapitel verwendet, um iterative Verfahren unter dem Gesichtspunkt der Parallelverarbeitung neu zu entwerfen.

3.1 Parallele Berechnung der Grundoperationen

Nach einigen allgemeinen Vorbemerkungen werden in diesem Abschnitt Kommunikationszeiten für die parallele Berechnung der verschiedenen Grundoperationen in zwei unterschiedlichen Datenverteilungen hergeleitet.

3.1.1 Allgemeine Vorbemerkungen

Die im vorherigen Kapitel eingeführten iterativen Methoden zur Erzeugung von Krylov–Teilräumen verwenden ausschließlich die folgenden vier Typen von Operationen

- Skalare Operationen
- Linearkombinationen von Vektoren
- Skalarprodukte von Vektoren
- Matrix-Vektor-Produkte.

Dabei sei angemerkt, daß die Berechnung einer euklidischen Norm der Berechnung eines Skalarproduktes mit anschließendem Radizieren entspricht. Da euklidische Normen somit durch Komposition der bereits aufgeführten Operationen entstehen, werden sie im folgenden nicht weiter berücksichtigt sondern konzeptionell den Skalarprodukten gleichgestellt. In diesem Kapitel werden nur solche iterativen Methoden analysiert, die aus diesen vier unterschiedlichen Operationstypen aufgebaut sind und deren Anzahl von Operationen während des Verlaufes des Verfahrens innerhalb eines Iterationsschrittes konstant ist. Damit wird die Betrachtung auf Methoden mit sogenannten *kurzen Rekursionen* eingeschränkt, und es werden Methoden explizit ausgeschlossen, bei denen der Speicherbedarf und die Anzahl arithmetischer Operationen eines Iterationsschrittes linear in den Iterationen ansteigen. Ein bekannter Vertreter dieser Klasse der Methoden mit *langen Rekursionen* ist GMRES [63], das in der n-ten Iteration $n + 1$ Skalarprodukte berechnet.

Da die im vorherigen Kapitel vorgestellten iterativen Verfahren als zugrundeliegende Technik in Krylov–Teilraummethoden eingesetzt werden, werden die hier betrachteten Matrix-Vektor-Produkte als Produkte mit einer Koeffizientenmatrix eines linearen Gleichungssystems interpretiert. Deshalb wird für dieses Kapitel eine zwar spezielle, jedoch für eine große Klasse von Gleichungssystemen typische Matrixstruktur zugrundegelegt. Die hier betrachtete Matrix $\mathbf{A}$ sei eine $N \times N$ Matrix, die bei einer Diskretisierung einer zweidimensionalen linearen partiellen Differentialgleichung zweiter Ordnung nach der Differenzenmethode unter Verwendung einer 5-Punkt-Formel mit zentralen Differenzenquotienten entsteht. Die natürliche Numerierung der inneren Gitterpunkte eines quadratischen Gebietes führt auf die in Abb. 3.1 dargestellte Besetzungsstruktur der Koeffizientenmatrix des resultierenden linearen Systems. Eine detaillierte Beschreibung dieses Vorgangs ist in [8] angegeben.

Unabhängig von N besitzt eine Zeile der Matrix aus Abb. 3.1 höchstens fünf Nichtnull-Elemente. Für größer werdende N nimmt daher der Anteil der Nichtnull-Elemente im Vergleich zu den N^2 möglichen Einträgen mehr und mehr ab. Man spricht deshalb von *dünnbesetzten* Matrizen, die durch eine von Wilkinson eingeführte Formulierung nur vage definiert sind. Danach sind Matrizen dünnbesetzt, "wenn man aus Anzahl oder Position der Nichtnull-Elemente einen Vorteil ziehen kann." Im Kontext von iterativen Verfahren, die die Koeffizientenmatrix ausschließlich in der Form von Matrix-Vektor-Produkten enthalten, ist der folgende Unterschied zwischen dicht- und dünnbesetzten Matrizen entscheidend. Während eine Operation der Form $\mathbf{Az}$ für einen beliebigen Vektor $\mathbf{z} \in \mathbb{C}^N$ bei dichtbesetzten Matrizen eine Komplexität von $\Theta(N^2)$ besitzt, sei eine dünnbesetzte Matrix durch die folgende Eigenschaft charakterisiert.

Definition 3.1 (Dünnbesetztheit). Sei $\mathbf{A}$ eine komplexe $N \times N$ Matrix und $\mathbf{z} \in \mathbb{C}^N$ ein beliebiger Vektor; dann heißt $\mathbf{A}$ *dünnbesetzt*, falls es eine Datenstruktur gibt, in der ein Matrix-Vektor-Produkt $\mathbf{Az}$ eine Komplexität von $\Theta(N)$ oder $\Theta(N \log N)$ besitzt.

Man beachte, daß durch diese Definition auch Matrizen, bei denen alle Elemente von Null verschieden sind, als dünnbesetzt definiert werden, solange sie eine Struktur besitzen, die bei der

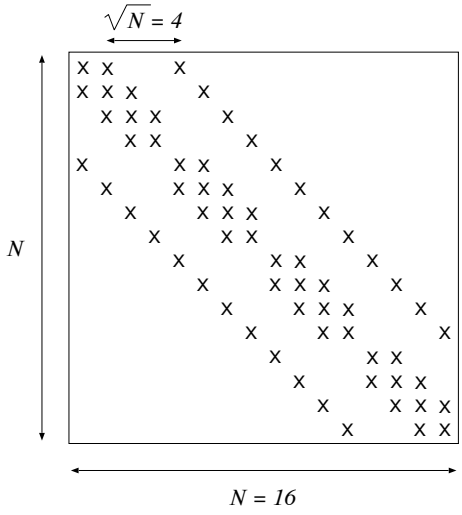


Abbildung 3.1: Besetzungsstruktur der Beispielmatrix

Berechnung eines Matrix-Vektor-Produktes zur Effizienzsteigerung ausgenutzt werden kann. Ein bekanntes Beispiel einer solchen Klasse von dichtbesetzten Matrizen, die im Sinne der Definition als dünnbesetzt gelten, sind die Toeplitz–Matrizen. Matrix-Vektor-Produkte von Toeplitz–Matrizen können durch Rückführung auf *zirkuläre* Matrizen und Fourier–Transformationen mit einem Aufwand von $\Theta(N \log N)$ berechnet werden [38].

Die gesamte Diskussion in diesem Kapitel nimmt an, daß die Matrix $\Theta(N)$ Nichtnull-Elemente besitzt und somit ein Matrix-Vektor-Produkt mit einem Aufwand von $\Theta(N)$ berechnet werden kann. Diese Annahme wird beispielsweise ebenfalls in [55] getroffen. Da alle übrigen Operationen, die in einem Iterationsschritt eines Verfahrens mit kurzen Rekursionen ausgeführt werden, höchstens mit linearem Aufwand in N zu berechnen sind, gilt für die Ausführungszeit des schnellsten sequentiellen Algorithmus zur Berechnung einer einzelner Iteration

$$T_{\text{seq}} = cNt_{\text{a}}, \tag{3.1}$$

wobei c eine Konstante und t_{a} die Zeit zur Berechnung einer arithmetischen Operation auf einem gegebenen seriellen Rechner bezeichnet.

Für das später in diesem Kapitel eingeführte Modell zur Analyse eines Iterationsschrittes werden die Kommunikationszeiten benötigt, die bei der parallelen Berechnung der vier erwähnten Operationstypen anfallen. Im folgenden werden deshalb diese Kommunikationszeiten für die einzelnen Operationen hergeleitet. Gleichzeitig werden dabei zwei unterschiedliche Verteilungen der Daten auf die Prozessoren beschrieben.

3.1.2 Skalare Operationen und Linearkombinationen

In diesem Kapitel wird angenommen, daß es eine natürliche Zahl p gibt, so daß p die Zahl N teilt. Der Vektorraum $\mathbb{C}^N$ wird nun in das kartesische Produkt von p niedriger dimensionalen Vektorräumen $\mathbb{C}^{N/p}$ zerlegt. Bezeichnet p die Anzahl von Prozessoren eines Parallelrechners mit verteiltem Speicher, so werden jedem Prozessor N/p Komponenten eines Vektors zugewiesen. Dabei werden alle Vektoren in der gleichen Weise auf die Prozessoren verteilt. Dies bedeutet, daß die Indizes der Vektorkomponenten, die sich auf einem Prozessor befinden, die gleichen sind. So entsprechen Linearkombinationen von Vektoren aus $\mathbb{C}^N$ genau p unabhängigen Linearkombinationen aus $\mathbb{C}^{N/p}$, die lokal auf den Prozessoren ausgeführt werden. Unter der Annahme, daß alle

skalaren Größen auf jedem Prozessor gespeichert werden, gelten dann für die Kommunikationszeiten von skalaren Operationen $T_{\text{comm}}^{\text{SO}}$und von Linearkombinationen $T_{\text{comm}}^{\text{LK}}$die Beziehungen

$$T_{\text{comm}}^{\text{SO}} = T_{\text{comm}}^{\text{LK}} = 0. \tag{3.2}$$

Die Annahme, daß N durch p teilbar ist, stellt keine gravierenden Einschränkungen für die mathematische Modellierung eines physikalischen Problems dar. In realen Anwendungen gilt fast immer $N \gg p$. Falls daher N nicht durch p teilbar sein sollte, wird der Gesamtaufwand durch den Aufwand der p Prozessoren dominiert, die jeweils einen Teilraum der Dimension $N \operatorname{div} p$ bearbeiten. Der restliche Aufwand bezüglich eines Teilraums $N \bmod p$ ist vernachlässigbar und kann unter Aspekten der Lastbalancierung gehandhabt werden.

3.1.3 Skalarprodukte

Mit der beschriebenen Datenverteilung der Vektoren auf die Prozessoren kann ein Skalarprodukt $\alpha = \mathbf{y}^T\mathbf{z}$ in zwei Phasen berechnet werden. Dabei ist das Ziel der Operation, das Ergebnis α wieder allen Prozessoren verfügbar zu machen. In der ersten Phase der parallelen Berechnung eines Skalarproduktes benötigen die Prozessoren untereinander keine Kommunikation. Hier werden auf allen Prozessoren gleichzeitig die Skalarprodukte von denjenigen Teilvektoren der Dimension N/p berechnet, die auf einem Prozessor vorhanden sind. In der zweiten Phase werden die so entstandenen p Teilergebnisse zu einer globalen Summe aufaddiert, die nun jedoch Kommunikation der Prozessoren untereinander erfordert. Das Kommunikationsmuster, das in dieser zweiten Phase verwendet wird, tritt in anderem Zusammenhang bei vielen weiteren parallelen Berechnungen wie beispielsweise der Minimum- oder Maximumbestimmung von verteilt vorliegenden skalaren Größen auf und wird als *Reduktion* bezeichnet. Eine Reduktion beginnt mit unterschiedlichen Werten auf jedem Prozessor, hier die Ergebnisse der Skalarprodukte der Teilvektoren, und endet mit einem einzigen Ergebnis auf allen Prozessoren, das durch Anwendung einer assoziativen Operation, hier der Addition, auf alle Startwerte entsteht. Daher entspricht die Bestimmung der Kommunikationszeit eines Skalarproduktes im Kern der Bestimmung der Kommunikationszeit einer Reduktion.

Aufgrund der globalen Kommunikationsstruktur hängt die Kommunikationszeit einer Reduktion stark davon ab, in welcher Weise die Prozessoren miteinander verbunden sind. Falls nicht anders erwähnt, wird ein zweidimensionales Gitter von $\sqrt{p}$ Prozessoren in jede Richtung zugrundegelegt, das über die Kanten zu einem Torus verbunden ist. Außerdem werden für einen einzelnen Prozessor folgende Grundannahmen getroffen. Er sendet zu einem Zeitpunkt nur auf einer seiner Ausgangsverbindungen. Ebenso empfängt er zu einem Zeitpunkt nur auf einer seiner Eingangsverbindungen. Ein gleichzeitiges Senden und Empfangen ist sowohl auf einer gleichen Verbindung als auch auf unterschiedlichen Verbindungen möglich. Als Übertragungsprotokoll wird das sogenannte *Cut-Through Routing* angenommen. In diesem Protokoll benötigt die vollständige Übertragung einer Nachricht der Länge l zwischen Prozessoren, die über d Verbindungen benachbart sind,

$$t_{\text{s}} + l t_{\text{w}} + d t_{\text{h}},$$

wobei $t_{\text{s}}, t_{\text{w}}$ und t_{h} die folgenden Größen bezeichnen. Die (Startup-)Zeit t_{s} wird beim sendenden Prozessor zur Initiierung einer Nachrichtenübertragung verbraucht und tritt für jede Nachricht genau einmal auf. Die (Per-Word-)Zeit t_{w} fällt bei der Übertragung eines Wortes zwischen zwei unmittelbar benachbarten Prozessoren an und ist gleich $1/b$, wobei b die Bandbreite des Kommunikationskanals ist. Hier wird vereinfachend angenommen, daß ein Skalar aus $\mathbb{C}$ genau ein Wort an Speicherplatz benötigt. So entspricht die Nachrichtenlänge l der Anzahl zu übertragender Elemente aus $\mathbb{C}$. Und schließlich benötigt der Nachrichtenkopf zwischen zwei unmittelbar benachbarten Prozessoren die (Per-Hop-)Zeit t_{h}.

Nach [50] beträgt unter diesen Voraussetzungen die Kommunikationszeit einer Reduktion von p unterschiedlichen Werten, die jeweils aus l Elementen aus $\mathbb{C}$ bestehen,

$$T_{\text{comm}}^{\text{RED}} = 2\Big[\,(t_{\text{s}} + l t_{\text{w}}) \log p + 2t_{\text{h}}(\sqrt{p} - 1)\,\Big]. \tag{3.3}$$

Dabei wird eine Reduktion in den folgenden zwei Phasen durchgeführt. In der ersten Phase werden die Teilergebnisse durch Anwendung einer assoziativen Operation zu einem Gesamtergebnis kombiniert, das bei einem ausgezeichneten Prozessor gespeichert wird. In der zweiten Phase wird dieses Ergebnis durch Umkehrung der Richtungen und Reihenfolge der Nachrichten aus der ersten Phase zurück an alle Prozessoren gesendet. Die Berechnung eines Skalarproduktes, bei der die zu reduzierenden Nachrichten ein Element aus $\mathbb{C}$ und damit $l = 1$ ist, wird deshalb durch den Ausdruck

$$T_{\text{comm}}^{\text{SP}} \approx 2\Big[\,(t_{\text{s}} + t_{\text{w}}) \log p + 2t_{\text{h}}\sqrt{p}\,\Big] \tag{3.4}$$

approximiert.

In (3.4) geht die spezielle Architektur des zugrundeliegenden Parallelrechners ein. Sie gilt daher nur für den Fall eines zweidimensionalen Torus. Da die Diskussion des nächsten Abschnittes auf (3.4) aufsetzt, sind die dort getroffenen Aussagen ebenfalls nur für einen zweidimensionalen Torus gültig. Das dort benutzte Modell ist jedoch einfach auf andere Rechnerarchitekturen anzupassen, indem (3.4) durch eine entsprechende Gleichung ersetzt wird. Zum Beispiel analysieren Gupta et al. [41] den Hypercube unter Verwendung von $T_{\text{comm}}^{\text{SP}} = 2t_{\text{s}} \log p$ und den Fat-Tree, bei dem sie $T_{\text{comm}}^{\text{SP}} = 0$ annehmen. Für den Hypercube nehmen sie in Übereinstimmung mit der hier gewählten Vorgehensweise an, daß die Reduktion in den oben skizzierten zwei Phasen abläuft. Aber gerade Kommunikationsalgorithmen für den Hypercube sind in der Literatur ausgiebig untersucht [18, 56] und es ist bekannt [50], daß es für den Hypercube eine günstigere Methode zur Berechnung einer Reduktion gibt. Die Kommunikationszeiten für eine Reduktion können nämlich um den Faktor zwei verbessert werden, wenn man das Kommunikationsmuster eines sogenannten *All-To-All Broadcast* geeignet anpaßt. Man beachte, daß es für zu reduzierende Nachrichten mit hoher Länge eine noch bessere Möglichkeit gibt, wenn man nämlich die Nachricht in mehrere kleine Stücke aufteilt [2]. Abschließend sei noch bemerkt, daß die Kommunikationszeiten höchstens um einen Faktor vier ansteigen, wenn man keinen Torus sondern ein zweidimensionales Gitter ohne Verbindungen über die Kanten annimmt.

3.1.4 Matrix-Vektor-Produkte

Für die Berechnung von Matrix-Vektor-Produkten werden zwei unterschiedliche Datenverteilungen der Matrix untersucht, für die in diesem Teilabschnitt jeweils die Kommunikationszeiten hergeleitet werden. Diese unterschiedlichen Ergebnisse werden im nächsten Abschnitt verwendet, um diese beiden Datenverteilungen hinsichtlich ihrer Güte in bezug auf die Parallelisierung von iterativen Methoden zu vergleichen.

Die erste Verteilung der Daten auf die Prozessoren ist äußerst naheliegend, leicht implementierbar, aber für die angegebene Numerierung der Gitterpunkte aus Sicht der Parallelverarbeitung eher ungünstig, wie in diesem Kapitel gezeigt wird. Diese Datenverteilung ist in Abb. 3.2 schematisch dargestellt. Dabei werden die N Zeilen der Matrix in p Blöcke aufgeteilt, von denen jeder N/p aufeinanderfolgende Zeilen der Matrix enthält. Die so entstehenden Blöcke werden nacheinander auf die Prozessoren verteilt, so daß Prozessor i die Nichtnull-Elemente der Matrixzeilen $(i-1)N/p + 1$ bis iN/p speichert. Die Vektoren werden entsprechend verteilt. Diese Verteilung von Matrix und Vektoren wird als *Blockverteilung* bezeichnet. Werden Techniken, die die Dünnbesetztheit der Matrix ausnutzen, angewendet, dann hängen die Kommunikationszeiten zur Berechnung eines Matrix-Vektor-Produktes von der Besetzungsstruktur der Matrix ab.

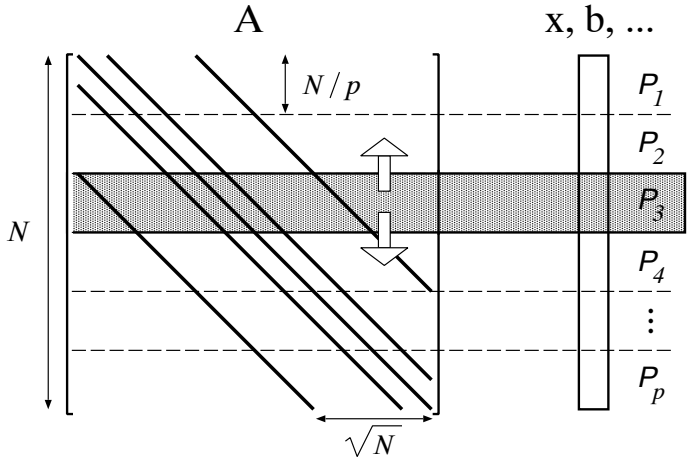


Abbildung 3.2: Blockdatenverteilung

Die spezielle Struktur der hier betrachteten Matrix aus Abb. 3.1 führt dazu, daß während eines Matrix-Vektor-Produktes die i-te Komponente eines Vektors mit Matrixelementen aus den Zeilen $i, i+1, i-1, i-\sqrt{N}$ und $i+\sqrt{N}$ multipliziert wird. Daher wird jede Komponente eines Vektors an Positionen mit einer Entfernung von 1 oder $\sqrt{N}$ zu beiden Seiten gebraucht. Der dazu notwendige Austausch von Vektorkomponenten hängt von der Anzahl der Daten ab, die jeder Prozessor speichert. Zur Vereinfachung wird hier angenommen, daß diese Anzahl, nämlich N/p, größer oder gleich der zurückzulegenden Entfernung $\sqrt{N}$ ist. In diesem Fall ist $\sqrt{N} \geq p$. Nach den oben getroffenen Bemerkungen gilt normalerweise $N \gg p$, so daß diese Annahme durchaus gerechtfertigt ist. Ein Datenaustausch tritt dann nur zwischen unmittelbar benachbarten Prozessoren auf. In Abb. 3.2 ist die zur Berechnung eines Matrix-Vektor-Produktes notwendige Kommunikation von Prozessor P_i mit seinen unmittelbaren Nachbarn P_{i-1} und P_{i+1} für $i = 3$ jeweils duch einen weißen Pfeil angedeutet. Bei einer Kommunikation zwischen zwei Prozessoren werden jeweils $\sqrt{N}$ Vektorkomponenten ausgetauscht. Daher gilt für die Kommunikationszeit eines Matrix-Vektor-Produktes

$$T_{\text{comm}}^{\text{MVP}} = 2(t_{\text{s}} + \sqrt{N} t_{\text{w}} + t_{\text{h}}), \qquad \text{für Blockdatenverteilung.} \tag{3.5}$$

Im Vergleich zur Blockdatenverteilung ist die im folgenden beschriebene *blockzyklische Datenverteilung* weniger offensichtlich, schwieriger zu implementieren, aber vorteilhafter in bezug auf Parallelverarbeitung. Während die Blockdatenverteilung durch Verteilung aufeinanderfolgender Zeilen anhand der Matrixdarstellung in Abb. 3.2 einfach zu beschreiben ist, ist die blockzyklische Datenverteilung leichter zu verstehen, wenn man eine Verteilung der Gitterpunkte bzw. des diskretisierten Gebietes auf die Prozessoren betrachtet. In Abb. 3.3 und Abb. 3.4 ist dieselbe Datenverteilung einmal in der Form der Verteilung von Matrixzeilen und einmal in der Form der Verteilung der Gitterpunkte dargestellt. Beide Darstellungen werden nun nacheinander diskutiert.

Genau wie bei der Blockdatenverteilung erhält jeder Prozessor in der blockzyklischen Datenverteilung N/p Matrixzeilen, die nun aber innerhalb eines Prozessors nicht aufeinanderfolgend numeriert sind. Die N Zeilen der Matrix sind in $\sqrt{Np}$ Blöcke aufgeteilt, die aus jeweils $\sqrt{N/p}$ aufeinanderfolgenden Matrixzeilen bestehen. Diese Blöcke werden zyklisch über je $\sqrt{p}$ Prozessoren verteilt. Um die bei Matrix-Vektor-Produkten auftretenden Kommunikationszeiten anzugeben, ist eine Betrachtung der Verteilung des Diskretisierungsgitters hilfreich. Dazu teilt man das Gebiet in quadratische Felder auf, so daß jeder Prozessor N/p Gitterpunkte erhält. Da jeder Gitterpunkt genau eine Zeile zu der Matrix beiträgt, entspricht diese Verteilung der Gitterpunkte der

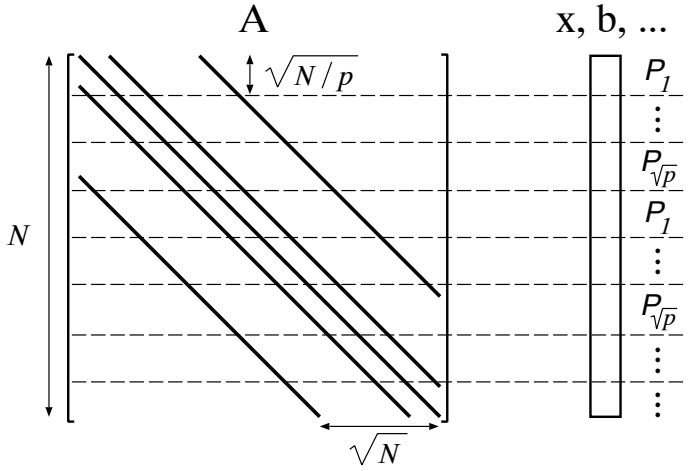


Abbildung 3.3: Blockzyklische Datenverteilung in Matrixrepräsentation

beschriebenen Verteilung der Matrixzeilen. Die Vektoren werden in beiden beschriebenen Datenverteilungen so verteilt, daß ein Prozessor die Vektorkomponente i genau dann erhält, wenn auch die Matrixzeile i dort gespeichert ist.

Man beachte, daß sowohl in der Blockdatenverteilung als auch in der blockzyklischen Datenverteilung jedem Prozessor N/p Matrixzeilen mit entsprechenden Vektorkomponenten zugewiesen werden. Die beiden Verteilungen unterscheiden sich jedoch darin, welche Zeilen bzw. Vektorkomponenten auf einem Prozessor vorhanden sind. Die in (3.2) und (3.4) hergeleiteten Kommunikationszeiten für Linearkombinationen und Skalarprodukten gelten für beide Datenverteilungen.

Die Darstellung der blockzyklischen Datenverteilung über die Verteilung der Gitterpunkte in Abb. 3.4 führt zu einem Ausdruck für die Kommunikationszeiten eines parallel ausgeführten Matrix-Vektor-Produktes. Ein Prozessor P_i, der nicht am Rande des Gebietes positioniert ist, kommunizert mit seinen vier unmittelbar benachbarten Prozessoren $P_{i+\sqrt{p}}$, $P_{i-\sqrt{p}}$, P_{i-1}, und P_{i+1}. Prozessoren am Rande kommunizieren mit entsprechend weniger Nachbarn. In jeder Nachricht sind Daten bezüglich $\sqrt{N/p}$ Gitterpunkte an den Prozessorgrenzen auszutauschen. Daher gilt für die Kommunikationszeit

$$T_{\text{comm}}^{\text{MVP}} = 4\left(t_{\text{s}} + \sqrt{N/p}\, t_{\text{w}} + t_{\text{h}}\right), \qquad \text{für blockzyklische Datenverteilung.} \tag{3.6}$$

Die Kommunikationszeiten zur Berechnung eines Matrix-Vektor-Produktes unterscheiden sich nach (3.5) und (3.6) in den beiden betrachteten Datenverteilungen. Die weißen Pfeile in den Abbildungen 3.2 und 3.4 deuten an, daß ein Prozessor in der Blockdatenverteilung mit zwei anderen Prozessoren kommuniziert, während ein Prozessor in der blockzyklischen Datenverteilung mit vier seiner Nachbarn Daten austauscht. In der blockzyklischen Datenverteilung ist daher die Anzahl der insgesamt versendeten Nachrichten größer; die Länge der Nachrichten ist jedoch geringer und hängt im Gegensatz zur Blockdatenverteilung insbesondere von der Anzahl der verwendeten Prozessoren ab.

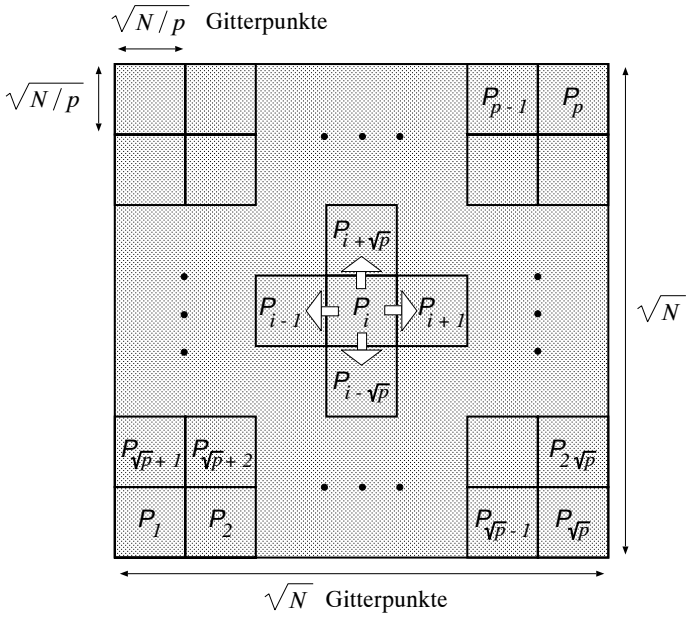


Abbildung 3.4: Blockzyklische Datenverteilung in Gitterpunktrepräsentation

3.2 Isoeffizienzanalyse

In diesem Abschnitt wird ein Konzept zur Modellierung der Skalierbarkeit eines parallelen Algorithmus, der auf einer speziellen Parallelrechnerarchitektur ausgeführt wird, betrachtet. Dieses Konzept wird dann zur Analyse eines einzelnene Iterationsschrittes eines Verfahrens zur Erzeugung von Krylov–Teilräumen herangezogen.

3.2.1 Das Konzept der Isoeffizienz

Sequentielle Algorithmen werden üblicherweise nach ihrer Ausführungszeit bewertet. Die sequentielle Ausführungszeit wird normalerweise als Funktion einer freien Variable ausgedrückt, die man als *Problemgröße* bezeichnet. Da die Erzeugung der Krylov–Teilräume als zugrundeliegende Technik zur iterativen Lösung eines linearen Gleichungssystems eingesetzt wird und man keine Aussagen darüber treffen kann, wie lange das Verfahren für eine hinreichend genaue Lösung benötigt, wird in diesem Kapitel ein einzelner Iterationsschritt als zu untersuchender Algorithmus betrachtet. Die Zeit des schnellsten sequentiellen Algorithmus für diese Aufgabenstellung bei einer Matrix der Dimension N beträgt nach (3.1)

$$T_{\text{seq}}(N) = cNt_{\text{a}} = \Theta(N), \tag{3.7}$$

wobei c eine Konstante und t_{a} die Zeit zur Berechnung einer arithmetischen Operation auf einem gegebenen seriellen Rechner bezeichnet. Beim Übergang von sequentiellen zu parallelen Betrachtungen müssen zusätzliche Aspekte berücksichtigt werden. Die Ausführungszeit eines parallelen Algorithmus T_{par} hängt nicht nur von der Problemgröße N sondern auch von der Prozessorzahl p sowie der speziellen Parallelrechnerarchitektur ab, auf der der Algorithmus implementiert wird. Kumar et al. [50] haben das Konzept der Isoeffizienz eingeführt, das die Ausführungszeit des schnellsten bekannten sequentiellen Algorithmus mit der Anzahl von Prozessoren in Verbindung setzt, die man zur Aufrechterhaltung einer vorgegebenen Effizienz benötigt. Dabei werden sowohl der Algorithmus als auch die Rechnerarchitektur berücksichtigt. Das Ziel ist die Bewertung der

Skalierbarkeit eines Algorithmus auf einer Architektur. Damit ist die Fähigkeit einer zur Prozessoranzahl proportionalen Leistung gemeint.

An dieser Stelle werden die folgenden gebräuchlichen Definitionen erwähnt.

Definition 3.2 (Speedup, Effizienz). Bezeichnet $T_{\text{seq}}(N)$ die Ausführungszeit, ein Problem auf einem einzigen Prozessor mit dem schnellsten bekannten sequentiellen Algorithmus zu lösen, und T_{par} die Ausführungszeit zur Lösung desselben Problems auf einem Parallelrechner mit p Prozessoren, dann heißen die Quotienten

$$S = \frac{T_{\text{seq}}}{T_{\text{par}}} \qquad \text{und} \qquad E = \frac{S}{p}$$

Speedup und *Effizienz*.

Für den optimalen Speedup gilt $S = p$ mit dazugehöriger Effizienz $E = 1$. Das Konzept der Isoeffizienz wird durch folgende zwei Beobachtungen motiviert.

- Bei einer festen Problemgröße — und damit bei einer festen Ausführungszeit des schnellsten bekannten sequentiellen Algorithmus — steigt der Speedup mit zunehmender Prozessoranzahl nicht an sondern tendiert zur Sättigung. Daher sinkt die Effizienz.

- Bei einer festen Prozessoranzahl steigt der Speedup, wenn man den Algorithmus auf größere Probleme anwendet, also bei einer Erhöhung der Problemgröße und damit der sequentiellen Ausführungszeit. Deshalb steigt die Effizienz.

Aufgrund dieser beiden Beobachtungen darf man erwarten, die Effizienz konstant zu halten, wenn man mit zunehmendem p ebenfalls T_{seq} in geeigneter Weise erhöht. Die Stärke, mit der man T_{seq} dabei erhöhen muß, kann als ein Maß für die Skalierbarkeit betrachtet werden.

Implementierungen von Algorithmen auf realen Parallelrechnern erreichen normalerweise nicht den optimalen Speedup. Gründe dafür sind beispielsweise Synchronisation und Kommunikation beim Datenaustausch. Im Konzept der Isoeffizienz werden alle Gründe, die für das Nichterreichen des optimalen Speedup verantwortlich sind, unter dem Begriff "Overhead" zusammengefaßt. Formal wird eine *Overhead–Funktion* durch

$$T_{\text{over}}(N,p) = p\,T_{\text{par}}(N,p) - T_{\text{seq}}(N)$$

definiert. Sie beinhaltet den Teil der gesamten, über die Prozessoren aufsummierten Zeit $p\,T_{\text{par}}$, der nicht durch den schnellsten bekannten sequentiellen Algorithmus bedingt ist. Daher kann die Effizienz als Funktion von T_{over} und T_{seq} ausgedrückt werden

$$E = \frac{T_{\text{seq}}(N)}{p\,T_{\text{par}}(N,p)} = \frac{T_{\text{seq}}(N)}{T_{\text{seq}}(N) + T_{\text{over}}(N,p)} = \frac{1}{1 + T_{\text{over}}(N,p)/T_{\text{seq}}(N)}. \tag{3.8}$$

Gibt man T_{over} und T_{seq} vor, so gibt diese Gleichung die Effizienz eines parallelen Algorithmus auf einer Architektur für eine gewählte Problemgröße N und Prozessoranzahl p an. Man kann diese Gleichung aber auch in einer anderen Art und Weise interpretieren. Für ein festes N, und damit einem festem T_{seq}, sinkt die Effizienz mit zunehmendem p, weil T_{over} normalerweise mit zunehmendem p wächst. Hält man p konstant und setzt voraus, daß T_{over} schwächer als $\Theta(T_{\text{seq}})$ wächst, steigt die Effizienz mit zunehmendem T_{seq} und N. Daher kann die Effizienz auf einem vorgegebenen Wert eingestellt werden, wenn p und T_{seq} gleichzeitig in geeigneter Weise erhöht werden. Ein solches Verhalten wird in der folgenden Definition beschrieben.

Definition 3.3 (Skalierbares paralleles System). Im Konzept der Isoeffizienz nennt man ein Paar aus einem Algorithmus und einer Architektur, auf der dieser implementiert ist, ein *skalierbares paralleles System*, falls der Quotient $T_{\text{over}}(N,p)/T_{\text{seq}}(N)$ bei adäquater gleichzeitiger Erhöhung von T_{seq} und p konstant bleibt.

Die Größenordnung, mit der man T_{seq} in bezug auf p erhöhen muß, um die Effizienz konstant zu halten, dient als Maß für die Güte eines skalierbaren parallelen Systems. Beispielsweise ist ein solches System schlecht skalierbar, wenn man zur Erhaltung einer konstanten Effizienz T_{seq} als eine exponentielle Funktion in p erhöhen muß. Andererseits ist ein skalierbares paralleles System hochskalierbar, wenn man T_{seq} nur linear in p erhöhen muß. Solche Größenordnungen berechnet man von (3.8) oder äquivalent dazu von

$$T_{\text{seq}}(N) = \frac{E}{1-E}\, T_{\text{over}}(N,p), \tag{3.9}$$

wobei E eine gewünschte, zu erhaltende Effizienz bezeichnet. Anstelle der Herleitung von solchen Größenordnungen in bezug auf T_{seq}, die zu sogenannten *Isoeffizienzfunktionen* [50] führen, wird hier anders vorgegangen. Es wird analysiert, wie stark man die Problemgröße N in bezug auf p erhöhen muß, um die Effizienz konstant zu halten. Die Aufgabe besteht daher darin, (3.9) in geschlossener Form nach N als Funktion von p aufzulösen.

3.2.2 Isoeffizienzanalyse eines einzelnen Iterationsschrittes

Nun wird eine Isoeffizienzanalyse eines einzelnen Iterationsschrittes eines Verfahrens zur Erzeugung von Krylov–Teilräumen durchgeführt. Um (3.9) nach N aufzulösen, werden T_{seq} und T_{over} eines Iterationsschrittes benötigt. Die Ausführungszeit des schnellsten bekannten sequentiellen Algorithmus ist durch (3.7) gegeben. Die Overhead–Funktion ist nur auf anfallende Kommunikationszeiten zurückzuführen und ist daher durch $T_{\text{over}} = p\,T_{\text{comm}}$ bestimmt. Weil nach (3.2) bei der Berechnung von skalaren Operationen und Linearkombinationen von Vektoren keine Kommunikationszeiten anfallen, ist die Overhead–Funktion von der Form

$$T_{\text{over}} = sp\,T_{\text{comm}}^{\text{SP}} + mp\,T_{\text{comm}}^{\text{MVP}}, \tag{3.10}$$

wobei s die Anzahl der Skalarprodukte und m die Anzahl der Matrix-Vektor-Produkte eines Iterationsschrittes bezeichnet. Beispielsweise gelten für Alg. 2.2 die Beziehungen $s = 4$ und $m = 2$.

Im vorherigen Abschnitt sind zwei unterschiedliche Datenverteilungen, nämlich Blockdatenverteilung und blockzyklische Datenverteilung, eingeführt worden. Diese zwei Datenverteilungen werden nun mit der Isoeffizienzanalyse verglichen. Beide Datenverteilungen besitzen die gleiche Kommunikationszeit zur Berechnung eines Skalarproduktes, die in (3.4) angegeben ist. Dagegen unterscheiden sich die Kommunikationszeiten zur Berechnung eines Matrix-Vektor-Produktes nach (3.5) und (3.6). Setzt man (3.4)–(3.6) in (3.10) ein und sortiert nach unterschiedlichen Größenordnungen von p, so erhält man

$$T_{\text{over}} = \begin{cases} 2mt_{\text{w}}\sqrt{N}\,p + 2m(t_{\text{s}}+t_{\text{h}})p + 2s(t_{\text{s}}+t_{\text{w}})p\log p + 4st_{\text{h}}p^{3/2} \\ \qquad\qquad \text{für Blockdatenverteilung,} \\ 4mt_{\text{w}}\sqrt{Np} + 4m(t_{\text{s}}+t_{\text{h}})p + 2s(t_{\text{s}}+t_{\text{w}})p\log p + 4st_{\text{h}}p^{3/2} \\ \qquad\qquad \text{für blockzyklische Datenverteilung.} \end{cases} \tag{3.11}$$

Für ein festes p zeigt (3.11), daß T_{over} langsamer als $\Theta(T_{\text{seq}}) = \Theta(N)$ wächst. Daher ist ein Iterationsschritt zur Erzeugung von Krylov–Teilräumen, der auf einem Parallelrechner mit zweidimensionaler Torusstruktur implementiert ist, ein skalierbares paralleles System. Es ist demnach

möglich, aus (3.9) durch Einsetzen von (3.7) und (3.11) Größenordnungen für eine konstante Effizienz herzuleiten. Hier besteht die Overhead–Funktion aus mehreren additiven Termen. In solchen Fällen ist es günstig, (3.9) für jeden einzelnen Term der Overhead–Funktion alleine nach N aufzulösen. Derjenige Term, der die höchste Größenordnung für N als Funktion von p erzwingt, bestimmt das asymptotische Gesamtverhalten des parallelen Systems.

Diese Vorgehensweise wird für beide Datenverteilungen durchgeführt, wobei mit der Blockdatenverteilung begonnen wird. Betrachtet man nur den ersten Term in (3.11), besteht die Aufgabe darin, die Gleichung

$$cN\,t_{\mathrm{a}} = \frac{E}{1-E}\,2mt_{\mathrm{w}}\sqrt{N}\,p$$

nach N aufzulösen. Man erhält

$$N = \left(\frac{2mt_{\mathrm{w}}E}{c\,t_{\mathrm{a}}(1-E)}\right)^2 p^2 = \Theta(p^2).$$

Betrachtet man nun alle verbleibenden Terme aus (3.11), so muß man die Gleichung

$$cN\,t_{\mathrm{a}} = \frac{E}{1-E}\Big[2m(t_{\mathrm{s}}+t_{\mathrm{h}})p + 2s(t_{\mathrm{s}}+t_{\mathrm{w}})p\log p + 4st_{\mathrm{h}}p^{3/2}\Big]$$

nach N auflösen. Da keiner der Terme auf der rechten Seite von N abhängt, wird das asymptotische Verhalten durch den letzten Term bestimmt, der die höchste Größenordnung in bezug auf p besitzt. Daher gilt

$$N = \frac{4st_{\mathrm{h}}E}{c\,t_{\mathrm{a}}(1-E)}\,p^{3/2} = \Theta(p^{3/2}). \tag{3.12}$$

Um die Effizienz auf eine festen Wert E einzustellen, muß die Problemgröße N wie $\Theta(p^2)$ erhöht werden, wenn man nur den ersten Term der Overhead–Funktion berücksichtigt. Betrachtet man alle übrigen Terme, so muß N wie $\Theta(p^{3/2})$ anwachsen. Das asymptotische Gesamtverhalten ist daher durch $N = \Theta(p^2)$ bestimmt.

Betrachtet man nun die blockzyklische Datenverteilung, so führt die Betrachtung des ersten Terms von (3.11) zu der Lösung von

$$cN\,t_{\mathrm{a}} = \frac{E}{1-E}\,4mt_{\mathrm{w}}\sqrt{Np}$$

die durch

$$N = \left(\frac{4mt_{\mathrm{w}}E}{c\,t_{\mathrm{a}}(1-E)}\right)^2 p = \Theta(p)$$

gegeben ist. Die Diskussion aller restlichen Terme führt zu dem gleichen Ergebnis wie in (3.12). Daher ist das asymptotische Gesamtverhalten durch $N = \Theta(p^{3/2})$ bestimmt. Der folgende Satz stellt die gewonnenen Ergebnisse für die beiden unterschiedlichen Datenverteilungen zusammen.

Satz 3.1 (Isoeffizienzanalyse). *Ein einzelner Iterationsschritt eines Verfahrens zur Erzeugung von Krylov–Teilräumen, das auf einem Parallelrechner mit zweidimensionaler Torusstruktur implementiert ist, ist ein skalierbares paralleles System. Um die Effizienz bei einem gewünschten Wert E konstant zu halten, muß die Problemgröße wie*

$$N = \begin{cases} \left(\dfrac{2mt_w E}{c\,t_a(1-E)}\right)^2 p^2 & = \Theta(p^2) \qquad \textit{für Blockdatenverteilung,} \\ \dfrac{4st_h E}{c\,t_a(1-E)}\,p^{3/2} & = \Theta(p^{3/2}) \qquad \textit{für blockzyklische Datenverteilung} \end{cases} \tag{3.13}$$

erhöht werden.

Die blockzyklische Datenverteilung ist in dieser Isoeffizienzanalyse asymptotisch skalierbarer als die Blockdatenverteilung. Denn zur Erhaltung einer konstanten Effizienz muß die Problemgröße bei der blockzyklischen Datenverteilung lediglich wie $\Theta(p^{3/2})$ und nicht wie bei der Blockdatenverteilung wie $\Theta(p^2)$ erhöht werden.

3.3 Anforderungen an den parallelen Algorithmenentwurf

Das Ziel der durchgeführten Isoeffizienzanalyse ist, Einblick in das Gesamtverhalten eines Iterationsschrittes einer Methode zur Erzeugung von Krylov–Teilräumen zu geben und nicht etwa detaillierte Laufzeitergebnisse vorherzusagen. Deshalb ist es gerechtfertigt, asymptotisches Verhalten zu betrachten. Der nun folgende Abschnitt zeigt, welche Schlüsse man aus dem Konzept der Isoeffizienz für den Entwurf paralleler iterativer Methoden ziehen kann. Von den beiden untersuchten Datenverteilungen steht hier die blockzyklische Datenverteilung im Vordergrund, da diese asymptotisch bessere Skalierungseigenschaften besitzt. Die einfache Blockdatenverteilung wird lediglich kurz behandelt.

3.3.1 Blockzyklische Datenverteilung

Um die Effizienz bei einem Wert E konstant zu halten, muß nach Satz 3.1 in der blockzyklischen Datenverteilung die Problemgröße wie $\Theta(p^{3/2})$ erhöht werden, die Konstante des führenden Terms beträgt dabei

$$\frac{4st_{\mathrm{h}}E}{c\,t_{\mathrm{a}}(1-E)}.$$

In diesem Ausdruck sind t_{a} und t_{h} durch die Hardware vorgegebene Parameter des zugrundeliegenden Parallelrechners. Die Konstante c ist nach (3.1) durch den schnellsten bekannten sequentiellen Algorithmus definiert und kann ebenfalls nicht beeinflußt werden. Der einzige Ansatzpunkt, der für den Entwickler von parallelen Algorithmen zur Minimierung des Ausdrucks zur Verfügung steht, ist die Größe s. Da man nicht erwarten kann, diese Anzahl der Skalarprodukte s eines Iterationsschrittes im Vergleich zu einem sequentiellen Algorithmus verringern zu können, ist es naheliegend, Kommunikationszeiten von Skalarprodukten einzusparen.

Es wird nun angenommen, daß alle s Skalarprodukte eines Iterationsschrittes unabhängig voneinander berechnet werden können. Man beachte, daß diese Annahme normalerweise *nicht* zutrifft. Beispielsweise sind von den $s = 4$ Skalarprodukten in Alg. 2.2 lediglich zwei Skalarprodukte unabhängig voneinander. Die Berechnung von ρ_{n+1} ist unabhängig von der Berechnung von ξ_{n+1}. Dagegen benötigt man zur Berechnung von ε_n die Kenntnis der Vektoren $\mathbf{p}_n$ und $\mathbf{q}_n$, die wiederum von der Berechnung des Skalarproduktes $\delta_n = \mathbf{w}_n^T\mathbf{v}_n$ abhängen. Nun stellt sich die folgende Frage: Ist es im Konzept der Isoeffizienz von Vorteil, wenn alle s Skalarprodukte unabhängig voneinander berechnet werden können? In diesem Fall können alle s Skalarprodukte gleichzeitig berechnet werden, indem zuerst alle lokalen Skalarprodukte der Teilvektoren gebildet werden und anschließend eine Vektorreduktion ausgeführt wird. Darunter versteht man die komponentenweise Reduktion eines Vektors mit s Komponenten, die jedoch das Kommunikationsmuster einer skalaren Reduktion verwendet, in der die Länge der Nachrichten s beträgt. Vor diesem Hintergrund wird der Ausdruck, der in der oben durchgeführten Isoeffizienzanalyse den Beitrag von s nacheinander ausgeführten Berechnungen von Skalarprodukten beschreibt, nämlich

$$sT_{\mathrm{comm}}^{\mathrm{SP}} = 2s\left[(t_{\mathrm{s}} + t_{\mathrm{w}})\log p + 2t_{\mathrm{h}}\sqrt{p}\,\right],$$

durch den Ausdruck

$$2\left[(t_{\mathrm{s}} + st_{\mathrm{w}})\log p + 2t_{\mathrm{h}}\sqrt{p}\,\right] \tag{3.14}$$

ersetzt. Darin werden (3.3) mit $l = s$ sowie die entsprechenden Näherungen benutzt, die zu (3.4) führen. Setzt man (3.14) statt $sT_{\mathrm{comm}}^{\mathrm{SP}}$ in die Overhead–Funktion (3.10) ein und führt eine Isoeffizienzanalyse durch, so erhält man

$$N = \frac{4t_{\mathrm{h}}E}{c\,t_{\mathrm{a}}(1-E)}\,p^{3/2}.$$

Damit ist der folgende Satz gezeigt.

Satz 3.2 (Unabhängige Skalarprodukte in der blockzyklischen Datenverteilung). *Sind alle s Skalarprodukte eines Iterationsschrittes voneinander unabhängig, so bleibt in der Isoeffizienzanalyse die Größenordnung $N = \Theta(p^{3/2})$ bei der blockzyklischen Datenverteilung erhalten; die Konstante des führenden Terms ist jedoch um den Faktor s kleiner.*

Die Isoeffizienzanalyse zeigt also, daß der Entwurf von parallelen iterativen Verfahren auf die Möglichkeit der unabhängigen Berechnung von Skalarprodukten zielen muß.

Als weiteres Ergebnis der Isoeffizienzanalyse wird darauf hingewiesen, daß die Konstante des führenden Terms nicht von der Anzahl der Matrix-Vektor-Produkte m eines Iterationsschrittes abhängt. Deshalb lohnt sich eine Minimierung der Kommunikationszeiten von Matrix-Vektor-Produkten hinsichtlich der Skalierbarkeit im Konzept der Isoeffizienz bei der blockzyklischen Datenverteilung nicht. Obwohl effiziente Implementierungen diesen Punkt berücksichtigen sollten, ist es vielversprechender, über neue Algorithmen mit unabhängigen Skalarprodukten nachzudenken.

3.3.2 Blockdatenverteilung

Für die Blockdatenverteilung zeigt Satz 3.1 ein asymptotisch schlechteres Verhalten als für die blockzyklische Datenverteilung. Die Konstante des führenden Terms der Blockdatenverteilung enthält den Parameter m, die Anzahl der Matrix-Vektor-Produkte eines Iterationsschrittes. Man ist daher geneigt, dem Parameter m bei der Blockdatenverteilung eine ähnliche Rolle wie dem Parameter s bei der blockzyklischen Datenverteilung zuzuweisen. Und man könnte vermuten, daß in der Blockdatenverteilung die unabhängige Berechnung von Matrix-Vektor-Produkten im Konzept der Isoeffizienz von Vorteil ist. Der folgende Satz zeigt nicht nur das Gegenteil, sondern weist auch einen anderen Nachteil der Blockdatenverteilung nach.

Satz 3.3 (Unabhängige Operationen in der Blockdatenverteilung). *Weder die Unabhängigkeit aller s Skalarprodukte noch die Unabhängigkeit aller m Matrix-Vektor-Produkte eines Iterationsschrittes besitzt in der Isoeffizienzanalyse einen Einfluß auf die Größenordnung oder die Konstante des führenden Terms bei der Blockdatenverteilung.*

Beweis. Unter der Annahme, daß alle m Matrix-Vektor-Produkte eines Iterationsschrittes unabhängig berechnet werden können, kann man in der Isoeffizienzanalyse den Ausdruck

$$2m(t_{\mathrm{s}} + \sqrt{N}t_{\mathrm{w}} + t_{\mathrm{h}})$$

durch den Ausdruck

$$2(t_{\mathrm{s}} + m\sqrt{N}t_{\mathrm{w}} + t_{\mathrm{h}})$$

ersetzen. Benutzt man in der Overhead–Funktion diesen Ausdruck an Stelle von $mT_{\text{comm}}^{\text{MVP}}$, so folgt das gleiche Ergebnis wie in (3.13). Da die Größenordnung $\Theta(p^2)$ durch einen Beitrag der Berechnung des Matrix-Vektor-Produktes bestimmt ist, ändert sich die Isoeffizienzanalyse nicht, wenn man unabhängige Skalarprodukte betrachtet. □

Abschließend sei noch erwähnt, daß man durch Überlappung von Kommunikation mit nützlichen Berechnungen eine Leistungssteigerung paralleler iterativer Verfahren erreicht. Dies geschieht in der Regel durch eine Restrukturierung eines bekannten sequentiellen Algorithmus und ist daher oft einfacher zu realisieren, als einen parallelen Algorithmus neu zu entwerfen [20]. Eine ausführliche Diskussion der Isoeffizienzanalyse mit weiteren Literaturhinweisen gibt [8].

Kapitel 4

Minimierung von Synchronisation

Die zentralen Bestandteile beim Entwurf neuer Krylov–Teilraumverfahren sind die numerisch stabile Erzeugung einer Basis für die Krylov–Teilräume und die Definition der aktuellen Iterierten. Während letzteres auf einem Parallelrechner mit verteiltem Speicher in allen in dieser Arbeit betrachteten Methoden zu keiner Kommunikation der Prozessoren führt, benötigt man zum Aufspannen der Krylov–Teilräume Orthogonalisierungen und Normierungen von Vektoren, deren parallele Berechnung eine besonders aufwendige Kommunikation aller beteiligten Prozessoren erfordert. Aus der Sicht des parallelen Algorithmenentwurfs werden daher diese beiden Bestandteile getrennt voneinander betrachtet, wobei der Definition der aktuellen Iterierten keine Bedeutung beigemessen wird.

Die Ergebnisse des vorherigen Kapitels lassen sich wie folgt zusammenfassen. Die zum Aufspannen der Teilräume notwendigen Skalarprodukte führen zu dem globalen Kommunikationsmuster einer Reduktion. Um eine hohe Skalierbarkeit zu erreichen, muß der Entwurf neuer paralleler Verfahren auf die Unabhängigkeit der skalarproduktartigen Operationen ausgerichtet sein. In diesem Kapitel wird daher gezeigt, wie neue parallele Varianten der in Kapitel 2 in ihrer ursprünglichen Form vorgestellten iterativen Methoden zur Erzeugung von Krylov–Teilräumen hergeleitet werden. Dabei wird die Unabhängigkeit aller Skalarprodukte eines Iterationsschrittes ausgenutzt, um globale Synchronisationspunkte einzusparen. Ein vorgeschobener Abschnitt führt solche globalen Synchronisationspunkte ein. Anschließend werden parallele Versionen des Lanczos–Algorithmus vorgestellt.

4.1 Globale Synchronisationspunkte

Das vorherige Kapitel führt eine Isoeffizienzanalyse eines einzelnen Iterationsschrittes eines Verfahrens zur Erzeugung von Krylov–Teilräumen durch. Dieses Modell dient der Analyse der Skalierbarkeit eines Algorithmus auf einer Parallelrechnerarchitektur. Im vorherigen Kapitel wird dabei eine spezielle Rechnerarchitektur angenommen, in der die Prozessoren in Form eines zweidimensionalen Gitters angeordnet sind, das über die Kanten zu einem Torus verbunden ist. Damit gelten alle Aussagen in den Sätzen 3.1–3.3 nur für die spezielle Verbindungsstruktur des zweidimensionalen Torus. Grundsätzlich lassen sich die getroffenen Aussagen aber auf beliebige Verbindungsstrukturen eines Parallelrechners übertragen. Die Argumentation läuft darauf hinaus, daß die Dünnbesetztheit der Matrix so ausgenutzt wird, daß bei der parallelen Berechnung eines Matrix-Vektor-Produktes Kommunikation nur zwischen wenigen, benachbarten Prozessoren auftritt. Im Gegensatz dazu entsprechen skalarproduktartige Operationen einer Reduktion, die zu einer Kommunikation aller eingesetzten Prozessoren führt. Diese *globale* Kommunikation wird auf Parallelrechnern mit unterschiedlichen Verbindungsstrukturen durch unterschiedliche Kommunikationsalgorithmen realisiert. Eine allgemeine Aussage läßt sich formulieren, wenn man die

Verbindungsstruktur eines Parallelrechners, die auch seine *Topologie* genannt wird, durch einen ungerichteten Graphen repräsentiert. Die Knoten des Graphen entsprechen dabei den Prozessoren, und es existiert genau dann eine Kante von Knoten i nach Knoten j, wenn es eine direkte Verbindungsleitung zwischen Prozessor i und Prozessor j gibt. Eine Kenngröße eines Graphen ist sein *Durchmesser*, welcher als maximale Distanz zwischen zwei beliebigen Knoten definiert ist. Die Anzahl der Kommunikationsschritte, die die Berechnung eines Skalarproduktes auf einem Parallelrechner beliebiger Topologie erfordert, ist mindestens so groß wie der Durchmesser D seines Graphen und beträgt daher $\Omega(D)$. Je nach Topologie ergeben sich daraus unterschiedliche untere Schranken für die Anzahl von Kommunikationsschritten zur Berechnung einer Reduktion auf p Prozessoren. Beispielsweise gelten folgende Aussagen für diese untere Schranken:

$\Omega(1)$	Vollständige Vernetzung,
$\Omega(\log p)$	Hypercube,
$\Omega(p^{1/d})$	d-dimensionales Gitter,
$\Omega(p)$	Ring.

Ein extremes Beispiel ist ein vollständig vernetzter Parallelrechner, bei dem jeder Prozessor mit jedem anderen Prozessor direkt verbunden ist. In dieser Topologie wird eine Reduktion durchgeführt, indem jedem Prozessor in einem einzigen Kommunikationsschritt alle zu reduzierenden Daten verfügbar gemacht werden. Die anschließende assoziative Verknüpfung dieser Teilergebnisse findet ohne Kommunikation statt. Der Kommunikationsaufwand zur Berechnung eines Skalarproduktes ist demnach auf dieser Topologie unabhängig von der Anzahl der Prozessoren, also besonders günstig. Betrachtet man aber Topologien mit zunehmender Größenordnung des Durchmessers, so steigt die Komplexität des Kommunikationsaufwandes zur Berechnung eines Skalarproduktes in der Anzahl der Prozessoren an. Für eine hohe Prozessorzahl p wird der Gesamtkommunikationsaufwand eines Iterationsschrittes daher durch die Skalarprodukte und nicht durch die Matrix-Vektor-Produkte dominiert. Aus diesem Grunde lassen sich die Aussagen des vorherigen Kapitels wie folgt auf beliebige Topologien erweitern: Die Unabhängigkeit aller Skalarprodukte ist auf allen Topologien von Vorteil. Der dadurch erzielte Gewinn nimmt zu, wenn man Topologien mit steigender Komplexität des Durchmessers betrachtet.

Das Ziel dieses Kapitels ist die Bereitstellung von parallelen Varianten von Verfahren zur Erzeugung von Krylov–Teilräumen. Darunter wird die Minimierung von globaler Synchronisation verstanden, die aus der Unabhängigkeit von Reduktionen eines Iterationsschrittes gewonnen wird. Dazu betrachte man das folgende Fragment eines hypothetischen Iterationsverfahrens, in dem Vektoren $\mathbf{v}_n, \mathbf{w}_n \in \mathbb{C}^N$ enthalten sind:

for $n = 1, 2, 3, \ldots$ **do**
- $\delta_n = \mathbf{w}_n^T \mathbf{v}_n$
- $\xi_n = \|\mathbf{w}_n\|_2$
  $\mathbf{v}_n = \delta_n \mathbf{v}_n + \mathbf{w}_n / \xi_n$
- $\rho_n = \|\mathbf{v}_n\|_2$
  $\mathbf{v}_n = \mathbf{v}_n / \rho_n$

enddo

Die Berechnungen von Reduktionen, die zu einer globalen Kommunikation führen, sind durch einen schwarzen Punkt gekennzeichnet. Die Berechnung der ersten beiden Reduktionen in δ_n und ξ_n sind voneinander unabhängig. Anstatt nacheinander zwei Reduktionen auszuführen, die jeweils mit Elementen aus $\mathbb{C}$ operieren, wird durch eine Vektorreduktion Kommunikation eingespart. In dieser Vektorreduktion werden Vektoren aus $\mathbb{C}^2$ jeweils komponentenweise assoziativ verknüpft. Das verwendete Kommunikationsmuster ist dabei identisch mit dem einer skalaren

Reduktion; lediglich die Länge der gesendeten Nachrichten wird verdoppelt. Aufgrund der Datenabhängigkeit kann die Reduktion zur Berechnung von ρ_n nicht gleichzeitig mit den beiden anderen Reduktionen ausgeführt werden. Die folgende Definition präzisiert diesen Sachverhalt.

Definition 4.1. Als einen *globalen Synchronisationspunkt* bezeichnet man diejenige Stelle eines Algorithmus, an der das Ergebnis einer (skalaren oder Vektor-) Reduktion auf allen Prozessoren verfügbar sein muß, um mit Berechnungen fortfahren zu können.

Nach dieser Definition bildet in dem oben beschriebenen Fragment eines Iterationsverfahrens die Vereinigung der beiden Reduktionen zur Berechnung von δ_n und ξ_n einen globalen Synchronisationspunkt. Einen zweiten globalen Synchronisationspunkt stellt die Berechnung von ρ_n dar. Im restlichen Teil dieses Kapitels werden nun Varianten der im vorherigen Kapitel beschriebenen iterativen Verfahren hergeleitet, in denen die Anzahl solcher globalen Synchronisationspunkte im Vergleich zu ihrer ursprünglichen Version minimiert ist.

4.2 Parallele Varianten des Lanczos–Algorithmus

In diesem Abschnitt werden Varianten des Lanczos–Algorithmus eingeführt, die innerhalb eines Iterationsschrittes nur einen einzigen globalen Synchronisationspunkt besitzen. Es wird sowohl die Implementierung mit Drei-Term-Rekursionen als auch die mit gekoppelten Zwei-Term-Rekursionen betrachtet. Die Techniken, mit denen das Ziel der Minimierung von globaler Synchronisation erreicht wird, unterscheiden sich in diesen beiden Implementierungen grundlegend. Während bei der Implementierung mit Drei-Term-Rekursionen eine einfache Restrukturierung der Anweisungen des sequentiellen Algorithmus ausreicht, wird bei der Implementierung mit gekoppelten Zwei-Term-Rekursionen eine andere Vorgehensweise benötigt. Hier wird in die Grundstruktur des ursprünglichen Algorithmus eingegriffen, indem nun veränderte Vektorräume generiert werden. Durch diesen neuen Ansatz in der Entwurfsphase des Algorithmus wird eine parallele Variante des Lanczos–Algorithmus hergeleitet, die in nachfolgenden Kapiteln als zugrundeliegende Technik in neuen Krylov–Teilraumverfahren verwendet wird.

4.2.1 Implementierung mit Drei-Term-Rekursionen

Der auf Drei-Term-Rekursionen basierende Lanczos–Algorithmus besitzt in seiner ursprünglichen Form, die in Alg. 2.1 dargestellt ist, drei globale Synchronisationspunkte pro Iterationsschritt. Der erste ist durch das Skalarprodukt in der Berechnung von α_n bedingt. Dieser Wert α_n fließt in die Berechnung der unskalierten Lanczos–Vektoren $\widetilde{\mathbf{v}}_{n+1}$ und $\widetilde{\mathbf{w}}_{n+1}$ ein. Deren Normen können unabhängig voneinander berechnet werden. Daher bildet die Vereinigung der Berechungen von ρ_{n+1} und ξ_{n+1} den zweiten globalen Synchronisationspunkt. Erst wenn ρ_{n+1} und ξ_{n+1} bekannt sind, können die Lanczos–Vektoren $\mathbf{v}_{n+1}$ und $\mathbf{w}_{n+1}$ berechnet werden. Das Skalarprodukt aus diesen beiden Lanczos–Vektoren bildet in der Berechnung von δ_{n+1} den dritten globalen Synchronisationspunkt.

In einer ersten Variante des Lanczos–Algorithmus wird die Anzahl der globalen Synchronisationspunkte pro Iterationsschritt auf zwei reduziert. Dies geschieht durch eine einfache Restrukturierung des Algorithmus, die die Numerik des Verfahrens nicht ändert. Die Herleitung dieser Variante basiert auf der Beobachtung, daß die Berechnung von γ_{n+1} im nächsten Iterationsschritt erst wieder in den Linearkombinationen der unskalierten Lanczos–Vektoren $\widetilde{\mathbf{v}}_{n+1}$ und $\widetilde{\mathbf{w}}_{n+1}$ benötigt wird. Die Berechnung von γ_{n+1} kann daher bis hinter die Berechnung von α_n verzögert werden. So werden die zwei globalen Synchronisationspunkte bei α_n und δ_{n+1} zu einem einzigen "verschmolzen". Die resultierende Variante mit zwei globalen Synchronisationspunkten pro Itera-

tionsschritt, in der alle zu einem globalen Synchronisationspunkt beitragenden Operationen durch einen schwarzen Punkt gekennzeichnet sind, ist in dem folgenden Algorithmus beschrieben.

ALGORITHMUS 4.1 (LANCZOS–ALGORITHMUS, DREI-TERM-VARIANTE 1)

Wähle $\mathbf{v}_1, \mathbf{w}_1 \in \mathbb{C}^N$ so, daß $\|\mathbf{v}_1\|_2 = \|\mathbf{w}_1\|_2 = 1$ und $\delta_1 = \mathbf{w}_1^T \mathbf{v}_1 \neq 0$
Setze $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$ und $\rho_1 = 0$, $\xi_1 \neq 0$, $\delta_0 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**

- $\alpha_n = \mathbf{w}_n^T \mathbf{A} \mathbf{v}_n / \delta_n$
  $\gamma_n = \xi_n \delta_n / \delta_{n-1}$
  $\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{v}_n - \alpha_n \mathbf{v}_n - \gamma_n \mathbf{v}_{n-1}$
  $\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T \mathbf{w}_n - \alpha_n \mathbf{w}_n - \frac{\gamma_n \rho_n}{\xi_n} \mathbf{w}_{n-1}$
- $\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$
- $\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$
  $\mathbf{v}_{n+1} = \frac{1}{\rho_{n+1}} \widetilde{\mathbf{v}}_{n+1}$
  $\mathbf{w}_{n+1} = \frac{1}{\xi_{n+1}} \widetilde{\mathbf{w}}_{n+1}$
- $\delta_{n+1} = \mathbf{w}_{n+1}^T \mathbf{v}_{n+1}$

enddo

Die Anzahl der globalen Synchronisationspunkte pro Iterationsschritt wird nun in einer zweiten Variante des Lanczos–Algorithmus weiter reduziert. Diese Variante besteht aus nur einem einzigen globalen Synchronisationspunkt. Im Gegensatz zur ersten Variante, in der die Numerik im Vergleich zur ursprünglichen Version unverändert bleibt, werden in der zweiten Variante die gleichen Operationen aber mit veränderten Operanden ausgeführt. Ausgehend von der ersten Variante besteht der Schlüssel zur zweiten Variante in der Verzögerung der Berechnung der Lanczos–Vektoren $\mathbf{v}_{n+1}$ und $\mathbf{w}_{n+1}$. Um diese Verzögerung zu erreichen, wird die Berechnung von α_n mit den unskalierten Lanczos–Vektoren $\widetilde{\mathbf{v}}_{n+1}$ und $\widetilde{\mathbf{w}}_{n+1}$ anstatt mit den normierten Lanczos–Vektoren $\mathbf{v}_{n+1}$ und $\mathbf{w}_{n+1}$ formuliert. Die gleiche Reformulierung wird für die Berechnung von $\delta_{n+1} = \mathbf{w}_{n+1}^T \mathbf{v}_{n+1}$ vorgenommen. Statt δ_{n+1} zu berechnen, wird nun

$$\widetilde{\delta}_{n+1} := \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}$$

berechnet. Aufgrund der Beziehungen $\widetilde{\mathbf{v}}_{n+1} = \rho_{n+1} \mathbf{v}_{n+1}$ und $\widetilde{\mathbf{w}}_{n+1} = \xi_{n+1} \mathbf{w}_{n+1}$ folgt ein Zusammenhang zwischen $\widetilde{\delta}_{n+1}$ und δ_{n+1} in der Form

$$\widetilde{\delta}_{n+1} = \rho_{n+1} \xi_{n+1} \delta_{n+1}. \tag{4.1}$$

Damit werden alle δ_i aus der ersten Variante eliminiert. Für die Berechnung von γ_n gilt dann

$$\gamma_n = \widetilde{\delta}_n \rho_{n-1} \xi_{n-1} / (\widetilde{\delta}_{n-1} \rho_n).$$

Die zweite Variante des Lanczos–Algorithmus mit einem globalen Synchronisationspunkt pro Iterationsschritt ist dann durch den folgenden Prozeß gegeben.

ALGORITHMUS 4.2 (LANCZOS–ALGORITHMUS, DREI-TERM-VARIANTE 2)

Wähle $\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{w}}_1 \in \mathbb{C}^N$ so, daß $\widetilde{\delta}_1 = \widetilde{\mathbf{w}}_1^T \widetilde{\mathbf{v}}_1 \neq 0$
Setze $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$ und $\rho_0 = \xi_0 = 0$, $\rho_1 = \|\widetilde{\mathbf{v}}_1\|_2$, $\xi_1 = \|\widetilde{\mathbf{w}}_1\|_2$, $\widetilde{\delta}_0 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**

- $\alpha_n = \widetilde{\mathbf{w}}_n^T \mathbf{A} \widetilde{\mathbf{v}}_n / \widetilde{\delta}_n$
  $\gamma_n = \widetilde{\delta}_n \rho_{n-1} \xi_{n-1} / (\widetilde{\delta}_{n-1} \rho_n)$
  $\mathbf{v}_n = \frac{1}{\rho_n} \widetilde{\mathbf{v}}_n$
  $\mathbf{w}_n = \frac{1}{\xi_n} \widetilde{\mathbf{w}}_n$
  $\widetilde{\mathbf{v}}_{n+1} = \frac{1}{\rho_n} \mathbf{A} \widetilde{\mathbf{v}}_n - \alpha_n \mathbf{v}_n - \gamma_n \mathbf{v}_{n-1}$
  $\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T \mathbf{w}_n - \alpha_n \mathbf{w}_n - \frac{\gamma_n \rho_n}{\xi_n} \mathbf{w}_{n-1}$
- $\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$
- $\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$
- $\widetilde{\delta}_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}$

enddo

In dieser zweiten Variante des Lanczos–Algorithmus ist die Berechnung von $\widetilde{\mathbf{v}}_{n+1}$ leicht modifiziert. Der erste Term in der Anweisung für $\widetilde{\mathbf{v}}_{n+1}$ lautet nun nicht mehr $\mathbf{A}\mathbf{v}_n$ sondern $\mathbf{A}\widetilde{\mathbf{v}}_n/\rho_n$. Auf diese Weise werden in jedem Iterationsschritt weiterhin zwei Matrix-Vektor-Produkte benötigt, nämlich $\mathbf{A}\widetilde{\mathbf{v}}_n$ und $\mathbf{A}^T\mathbf{w}_n$. Würde man die Formulierung $\mathbf{A}\mathbf{v}_n$ in der Berechnung von $\widetilde{\mathbf{v}}_{n+1}$ beibehalten, so hätte man mit der veränderten Berechung von α_n ein drittes Matrix-Vektor-Produkt $\mathbf{A}\widetilde{\mathbf{v}}_n$ eingeführt.

4.2.2 Implementierung mit gekoppelten Zwei-Term-Rekursionen

In der Implementierung mit gekoppelten Zwei-Term-Rekursionen, deren ursprüngliche Form in Alg. 2.2 abgebildet ist, besteht der Lanczos–Algorithmus aus drei globalen Synchronisationspunkten pro Iterationsschritt. Die Berechnung des Skalarproduktes $\delta_n = \mathbf{w}_n^T \mathbf{v}_n$ bildet den ersten Synchronisationspunkt. Der zweite entsteht durch die Berechnung von $\varepsilon_n = \mathbf{q}_n^T \mathbf{A} \mathbf{p}_n$. Schließlich ist der dritte Synchronisationspunkt durch die Berechnung der beiden Normen der Lanczos–Vektoren gegeben. Mit der gleichen Technik wie im vorherigen Teilabschnitt, nämlich mit der einfachen Restrukturierung von einzelnen Anweisungen, gelingt es, die Anzahl der globalen Synchronisationspunkte pro Iterationsschritt auf zwei zu reduzieren. Dazu wird wiederum die Berechnung der normierten Lanczos–Vektoren verzögert. So wird der Synchronisationspunkt in der Berechnung von δ_n mit dem Synchronisationspunkt, der durch die Berechnung der Normen entsteht, zu einem Synchronisationspunkt kombiniert. Der resultierende Prozeß mit zwei globalen Synchronisationspunkten entsteht analog zur Vorgehensweise bei der Drei-Term-Implementierung durch Elimination aller δ_i mit Hilfe von (4.1) und ist durch folgenden Algorithmus gegeben.

ALGORITHMUS 4.3 (LANCZOS–ALGORITHMUS, ZWEI-TERM-VARIANTE 1)

Wähle $\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{w}}_1 \in \mathbb{C}^N$ so, daß $\widetilde{\delta}_1 = \widetilde{\mathbf{w}}_1^T \widetilde{\mathbf{v}}_1 \neq 0$ und $\rho_1 = \|\widetilde{\mathbf{v}}_1\|_2$, $\xi_1 = \|\widetilde{\mathbf{w}}_1\|_2$
Setze $\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{0}$, $\mathbf{v}_1 = \frac{1}{\rho_1}\widetilde{\mathbf{v}}_1$, $\mathbf{w}_1 = \frac{1}{\xi_1}\widetilde{\mathbf{w}}_1$ und $\varepsilon_0 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**
$\mathbf{p}_n = \mathbf{v}_n - \frac{\widetilde{\delta}_n}{\varepsilon_{n-1}\rho_n}\mathbf{p}_{n-1}$
$\mathbf{q}_n = \mathbf{w}_n - \frac{\widetilde{\delta}_n}{\varepsilon_{n-1}\xi_n}\mathbf{q}_{n-1}$
- $\varepsilon_n = \mathbf{q}_n^T \mathbf{A}\mathbf{p}_n$

$\beta_n = \varepsilon_n \rho_n \xi_n / \widetilde{\delta}_n$
$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \beta_n \mathbf{v}_n$
$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T\mathbf{q}_n - \beta_n \mathbf{w}_n$
- $\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$
- $\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$
- $\widetilde{\delta}_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}$

$\mathbf{v}_{n+1} = \frac{1}{\rho_{n+1}}\widetilde{\mathbf{v}}_{n+1}$
$\mathbf{w}_{n+1} = \frac{1}{\xi_{n+1}}\widetilde{\mathbf{w}}_{n+1}$
enddo

Durch eine weitere Restrukturierung einzelner Anweisungen gelingt es nicht, die zwei noch vorhandenen globalen Synchronisationspunkte zu einem einzigen zusammenzufassen. Dies hat folgenden Grund. Zwar ist die Anzahl der skalarproduktartigen Operationen eines Iterationsschrittes in den Implementierungen mit Drei-Term-Rekursionen und gekoppelten Zwei-Term-Rekursionen gleich, aber bei einer dieser vier skalarproduktartigen Operationen ist die Ursache in beiden Implementierungen unterschiedlich. In der Drei-Term-Implementierung resultiert der Synchronisationspunkt in der Berechnung von α_n aus der $\mathbf{A}$-Bikonjugation der Lanczos–Vektoren $\mathbf{v}_n$ und $\mathbf{w}_n$, während in der gekoppelten Zwei-Term-Implementierung das neu eingeführte Paar der Basisvektoren $\mathbf{p}_n$ und $\mathbf{q}_n$ für einen Synchronisationspunkt in der Berechnung von ε_n verantwortlich zeichnet. Der Synchronisationspunkt in der Berechnung von ε_n ist von den Rekursionen zur Berechnung des Paares $\mathbf{p}_n$, $\mathbf{q}_n$ und des Paares $\widetilde{\mathbf{v}}_{n+1}$, $\widetilde{\mathbf{w}}_{n+1}$ eingeschlossen. Zwischen diesen Größen gibt es die folgenden unmittelbaren Datenabhängigkeiten. Aus dem Paar $\mathbf{p}_n$, $\mathbf{q}_n$ wird zunächst ε_n und darüber das Paar $\widetilde{\mathbf{v}}_{n+1}$, $\widetilde{\mathbf{w}}_{n+1}$ berechnet. Daher kann der Synchronisationspunkt in der Berechnung von ε_n durch eine einfache Restrukturierung nicht mit dem zweiten globalen Synchronisationspunkt, der bei der Berechung von ρ_{n+1}, ξ_{n+1} und $\widetilde{\delta}_{n+1}$ entsteht, kombiniert werden. Abhilfe schafft in diesem Falle nur ein völlig neuer Entwurf der gekoppelten Zwei-Term-Implementierung.

Der neue Ansatz wird aus einer Betrachtung der aufgespannten Räume in Verbindung mit den erwähnten Problemen der Datenabhängigkeiten gewonnen. In der ursprünglichen Version, die in Alg. 2.2 und in leicht modifizierter Form in Alg. 4.3 dargestellt ist, generiert der Lanczos–Algorithmus mit gekoppelten Zwei-Term-Rekursionen ein Paar von Basisvektoren für $\mathcal{K}_n(\mathbf{A}, \mathbf{v}_1)$ und eine weiteres Paar für $\mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1)$. Genauer gilt für den n-ten Schritt der ursprünglichen Formulierung:

$$\mathbf{p}_n \in \mathcal{K}_n(\mathbf{A}, \mathbf{v}_1), \qquad \mathbf{v}_{n+1} \in \mathcal{K}_{n+1}(\mathbf{A}, \mathbf{v}_1), \tag{4.2a}$$

$$\mathbf{q}_n \in \mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1), \qquad \mathbf{w}_{n+1} \in \mathcal{K}_{n+1}(\mathbf{A}^T, \mathbf{w}_1). \tag{4.2b}$$

Das Ziel des neuen Ansatzes ist das Aufheben der oben erwähnten Datenabhängigkeiten, so daß der globale Synchronisationspunkt in der Berechnung von ε_n mit den restlichen Operationen, die globale Synchronisation erfordern, kombiniert werden kann. Die grundsätzliche Idee besteht nun darin, andere Räume aufzuspannen, um die Berechnung von $\varepsilon_n = \mathbf{q}_n^T \mathbf{A} \mathbf{p}_n$ durch andere Anweisungen ersetzen zu können. Da der Lanczos–Algorithmus später als zugrundeliegende Technik in Krylov–Teilraumverfahren zur Lösung von $\mathbf{A}\mathbf{x} = \mathbf{b}$ eingesetzt wird, ist eine Änderung bezüglich der Basisvektoren von $\mathcal{K}_n(\mathbf{A}, \mathbf{v}_1)$ nicht sinnvoll. Damit besteht keine Freiheit in der Wahl von $\mathbf{p}_n$ und es ist naheliegend, die Räume bezüglich $\mathbf{q}_n$ zu verändern. Dazu faßt man ε_n in der Form von $\varepsilon_n = (\mathbf{A}^T \mathbf{q}_n)^T \mathbf{p}_n$ auf und führt eine neue Folge von Vektoren $\widetilde{\mathbf{q}}_n$ aus demjenigen Raum ein, der aus dem durch die $\mathbf{q}_i$ aufgespannten Raum durch Multiplikation mit $\mathbf{A}^T$ hervorgeht. Der n-te Iterationsschritt des neuen Ansatzes ist daher durch die Beziehungen

$$\mathbf{p}_n \in \mathcal{K}_n(\mathbf{A}, \mathbf{v}_1), \qquad \mathbf{v}_{n+1} \in \mathcal{K}_{n+1}(\mathbf{A}, \mathbf{v}_1), \tag{4.3a}$$

$$\widetilde{\mathbf{q}}_n \in \mathbf{A}^T \mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1), \qquad \mathbf{w}_{n+1} \in \mathcal{K}_{n+1}(\mathbf{A}^T, \mathbf{w}_1) \tag{4.3b}$$

charakterisiert. Der neue Ansatz beruht also darauf, die ursprünglichen Vektoren $\mathbf{q}_n$ durch Vektoren $\widetilde{\mathbf{q}}_n$ zu ersetzen, die aus $\mathbf{A}^T \mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1)$ und nicht aus $\mathcal{K}_n(\mathbf{A}^T, \mathbf{w}_1)$ sind. Der Ansatz besteht in der Wahl von $\widetilde{\mathbf{q}}_n = \mathbf{A}^T \mathbf{q}_n$, der mit der $N \times n$ Matrix

$$\widetilde{\mathbf{Q}}_n := [\widetilde{\mathbf{q}}_1 \; \widetilde{\mathbf{q}}_2 \; \cdots \; \widetilde{\mathbf{q}}_n]$$

in Matrixnotation durch

$$\widetilde{\mathbf{Q}}_n = \mathbf{A}^T \mathbf{Q}_n \tag{4.4}$$

ausgedrückt wird. Setzt man (4.4) in die Gleichungen (2.37)–(2.40) ein, die in Satz 2.2 die ursprüngliche gekoppelte Zwei-Term-Implementierung charakterisieren, so folgt für den neuen Ansatz:

$$\mathbf{V}_n = \mathbf{P}_n \mathbf{U}_n, \tag{4.5a}$$

$$\mathbf{A}\mathbf{P}_n = \mathbf{V}_n \mathbf{L}_n + \rho_{n+1} \mathbf{v}_{n+1} \mathbf{e}_n^T, \tag{4.5b}$$

$$\mathbf{A}^T \mathbf{W}_n = \widetilde{\mathbf{Q}}_n \mathbf{E}_n^{-1} \mathbf{L}_n^T \mathbf{D}_n, \tag{4.5c}$$

$$\widetilde{\mathbf{Q}}_n = \mathbf{W}_n \mathbf{D}_n^{-1} \mathbf{U}_n^T \mathbf{E}_n + \xi_{n+1} \mathbf{w}_{n+1} \mathbf{e}_n^T. \tag{4.5d}$$

Darin sind die ersten beiden Gleichungen, die zur Erzeugung von $\mathcal{K}_n(\mathbf{A}, \mathbf{v}_1)$ beitragen, gegenüber der ursprünglichen gekoppelten Zwei-Term-Implementierung unverändert. Dies korrespondiert zu der Gleichheit von (4.2a) und (4.3a). Der Unterschied zwischen (4.2b) und (4.3b) spiegelt sich in den letzten beiden Matrixgleichungen wider, die im Vergleich zur ursprünglichen Zwei-Term-Implementierung verändert sind. Die weitere Herleitung, für die an die spezielle Struktur der bidiagonalen Matrizen

$$\mathbf{L}_n = \begin{bmatrix} \beta_1 & & & \mathbf{0} \\ \rho_2 & \beta_2 & & \\ & \rho_3 & \beta_3 & \\ & & \ddots & \ddots \\ \mathbf{0} & & & \rho_n \quad \beta_n \end{bmatrix} \in \mathbb{C}^{n \times n}$$

und

$$\mathbf{U}_n = \begin{bmatrix} 1 & \frac{\xi_2\delta_2}{\varepsilon_1} & & \\ & 1 & \frac{\xi_3\delta_3}{\varepsilon_2} & \mathbf{0} \\ & & 1 & \ddots \\ & \mathbf{0} & & \ddots & \frac{\xi_n\delta_n}{\varepsilon_{n-1}} \\ & & & & 1 \end{bmatrix} \in \mathbb{C}^{n\times n}$$

erinnert wird, vergleicht in den Gleichungen (4.5) jeweils die n-te Spalte

$$\mathbf{v}_n = \mathbf{p}_n + \frac{\xi_n\delta_n}{\varepsilon_{n-1}}\mathbf{p}_{n-1}, \tag{4.6a}$$

$$\mathbf{A}\mathbf{p}_n = \rho_{n+1}\mathbf{v}_{n+1} + \beta_n\mathbf{v}_n, \tag{4.6b}$$

$$\mathbf{A}^T\mathbf{w}_n = \frac{\beta_n\delta_n}{\varepsilon_n}\widetilde{\mathbf{q}}_n + \frac{\rho_n\delta_n}{\varepsilon_{n-1}}\widetilde{\mathbf{q}}_{n-1}, \tag{4.6c}$$

$$\widetilde{\mathbf{q}}_n = \xi_{n+1}\mathbf{w}_{n+1} + \frac{\varepsilon_n}{\delta_n}\mathbf{w}_n, \tag{4.6d}$$

wobei $\mathbf{p}_0 = \widetilde{\mathbf{q}}_0 = \mathbf{0}$, $\xi_1 = \rho_1 = 0$ und $\varepsilon_0 \neq 0$. Unter Verwendung der Biorthogonalität der Lanczos–Vektoren, aus der

$$\delta_n = \mathbf{w}_n^T\mathbf{v}_n \tag{4.7}$$

folgt, leitet man einen Ausdruck für β_n ab. Dazu multipliziert man (4.6b) mit $\mathbf{w}_n^T$ und erhält

$$\beta_n = \frac{\mathbf{w}_n^T\mathbf{A}\mathbf{p}_n}{\delta_n}.$$

Ersetzt man darin zuerst $\mathbf{p}_n$ durch (4.6a) und anschließend $\mathbf{A}\mathbf{p}_{n-1}$ nach (4.6b), so folgt

$$\begin{aligned} \beta_n &= \frac{\mathbf{w}_n^T\mathbf{A}\mathbf{v}_n}{\delta_n} - \frac{\xi_n}{\varepsilon_{n-1}}\mathbf{w}_n^T\mathbf{A}\mathbf{p}_{n-1} \\ &= \frac{\mathbf{w}_n^T\mathbf{A}\mathbf{v}_n}{\delta_n} - \frac{\xi_n\rho_n}{\varepsilon_{n-1}}\mathbf{w}_n^T\mathbf{v}_n - \frac{\xi_n\beta_{n-1}}{\varepsilon_{n-1}}\mathbf{w}_n^T\mathbf{v}_{n-1} \\ &= \frac{\mathbf{w}_n^T\mathbf{A}\mathbf{v}_n}{\delta_n} - \frac{\rho_n\xi_n\delta_n}{\varepsilon_{n-1}}, \end{aligned} \tag{4.8}$$

wobei $\beta_1 = \mathbf{w}_1^T\mathbf{A}\mathbf{v}_1/\delta_1$. Mit den Setzungen

$$\begin{aligned} \widetilde{\mathbf{v}}_{n+1} &:= \mathbf{A}\mathbf{p}_n - \beta_n\mathbf{v}_n, \\ \widetilde{\mathbf{w}}_{n+1} &:= \widetilde{\mathbf{q}}_n - \beta_n\mathbf{w}_n, \end{aligned}$$

und

$$\varepsilon_n = \beta_n\delta_n \tag{4.9}$$

folgt einerseits eine äquivalente Darstellung der Gleichungen (4.6) zu

$$\mathbf{p}_n = \mathbf{v}_n - \frac{\xi_n\delta_n}{\varepsilon_{n-1}}\mathbf{p}_{n-1}, \tag{4.10a}$$

$$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \beta_n\mathbf{v}_n, \tag{4.10b}$$

$$\widetilde{\mathbf{q}}_n = \mathbf{A}^T\mathbf{w}_n - \frac{\rho_n\delta_n}{\varepsilon_{n-1}}\widetilde{\mathbf{q}}_{n-1}, \tag{4.10c}$$

$$\widetilde{\mathbf{w}}_{n+1} = \widetilde{\mathbf{q}}_n - \beta_n\mathbf{w}_n, \tag{4.10d}$$

und andererseits erreicht man eine Normierung der Lanczos–Vektoren

$$\mathbf{v}_{n+1} = \widetilde{\mathbf{v}}_{n+1}/\rho_{n+1}, \tag{4.11a}$$
$$\mathbf{w}_{n+1} = \widetilde{\mathbf{w}}_{n+1}/\xi_{n+1}, \tag{4.11b}$$

auf Einheitslänge in der euklidischen Norm durch die Wahl der Skalierungsfaktoren

$$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2, \tag{4.12a}$$
$$\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2. \tag{4.12b}$$

Durch korrektes Zusammenfügen von relevanten Gleichungen erhält man eine vorläufige Form des Algorithmus.

ALGORITHMUS 4.4 (LANCZOS–ALGORITHMUS, ZWEI-TERM-VARIANTE 2)

Wähle $\mathbf{v}_1, \mathbf{w}_1 \in \mathbb{C}^N$ so, daß $\|\mathbf{v}_1\|_2 = \|\mathbf{w}_1\|_2 = 1$ und $\mathbf{w}_1^T\mathbf{v}_1 \neq 0$
Setze $\mathbf{p}_0 = \widetilde{\mathbf{q}}_0 = \mathbf{0}$ und $\xi_1 = \rho_1 = 0, \varepsilon_0 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**

- $\delta_n = \mathbf{w}_n^T\mathbf{v}_n$ Gl. (4.7)
- $\beta_n = \mathbf{w}_n^T\mathbf{A}\mathbf{v}_n/\delta_n - \rho_n\xi_n\delta_n/\varepsilon_{n-1}$ Gl. (4.8)

  $\varepsilon_n = \beta_n\delta_n$ Gl. (4.9)

  $\mathbf{p}_n = \mathbf{v}_n - \frac{\xi_n\delta_n}{\varepsilon_{n-1}}\mathbf{p}_{n-1}$ Gl. (4.10a)

  $\widetilde{\mathbf{q}}_n = \mathbf{A}^T\mathbf{w}_n - \frac{\rho_n\delta_n}{\varepsilon_{n-1}}\widetilde{\mathbf{q}}_{n-1}$ Gl. (4.10c)

  $\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \beta_n\mathbf{v}_n$ Gl. (4.10b)

  $\widetilde{\mathbf{w}}_{n+1} = \widetilde{\mathbf{q}}_n - \beta_n\mathbf{w}_n$ Gl. (4.10d)
- $\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$ Gl. (4.12a)
- $\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$ Gl. (4.12b)

  $\mathbf{v}_{n+1} = \frac{1}{\rho_{n+1}}\widetilde{\mathbf{v}}_{n+1}$ Gl. (4.11a)

  $\mathbf{w}_{n+1} = \frac{1}{\xi_{n+1}}\widetilde{\mathbf{w}}_{n+1}$ Gl. (4.11b)

enddo

Diese Formulierung besitzt zwei globale Synchronisationspunkte pro Iterationsschritt. Das Ziel, keinen globalen Synchronisationspunkt zwischen der Berechnung des Paares $\mathbf{p}_n$, $\mathbf{q}_n$ und des Paares $\widetilde{\mathbf{v}}_{n+1}$, $\widetilde{\mathbf{w}}_{n+1}$ vorzufinden, ist aber erreicht. Damit ist die Ursache, die das Kombinieren der zwei in Alg. 4.3 noch vorhandenen Synchronisationspunkte verhindert, beseitigt. Die zwei Synchronisationspunkte, die aus dem neuen Ansatz in Alg. 4.4 noch resultieren, werden durch die Verzögerung der Berechnungen der Lanczos–Vektoren $\mathbf{v}_{n+1}$ und $\mathbf{w}_{n+1}$ zu einem einzigen kombiniert. Anolog zur Vorgehensweise bei der Drei-Term-Implementierung formuliert man dazu die Berechnung von $\mathbf{w}_n^T\mathbf{A}\mathbf{v}_n$ in den unskalierten Lanczos–Vektoren

$$\sigma_n := \widetilde{\mathbf{w}}_n^T\mathbf{A}\widetilde{\mathbf{v}}_n.$$

Außerdem wird δ_n durch

$$\widetilde{\delta}_n := \widetilde{\mathbf{w}}_n^T\widetilde{\mathbf{v}}_n$$

ersetzt. Daraus folgt die Beziehung

$$\delta_n = \widetilde{\delta}_n/(\rho_n\xi_n),$$

mit Hilfe derer man alle δ_i eliminiert. Für die Berechnung von β_n, ε_n und $\mathbf{p}_n$, $\widetilde{\mathbf{q}}_n$ folgt dann

$$\begin{aligned}\beta_n &= \sigma_n/\widetilde{\delta}_n - \widetilde{\delta}_n/\varepsilon_{n-1}\,,\\ \varepsilon_n &= \beta_n\widetilde{\delta}_n/(\rho_n\xi_n)\,,\end{aligned}$$

mit $\beta_1 = \sigma_1/\widetilde{\delta}_1$ und

$$\begin{aligned}\mathbf{p}_n &= \mathbf{v}_n - \frac{\widetilde{\delta}_n}{\varepsilon_{n-1}\rho_n}\mathbf{p}_{n-1}\,,\\ \widetilde{\mathbf{q}}_n &= \mathbf{A}^T\mathbf{w}_n - \frac{\widetilde{\delta}_n}{\varepsilon_{n-1}\xi_n}\widetilde{\mathbf{q}}_{n-1}\,.\end{aligned}$$

Der resultierende Algorithmus, in welchem außerdem $\mathbf{A}^T\mathbf{w}_n$ durch $\mathbf{A}^T\widetilde{\mathbf{w}}_n/\xi_n$ ersetzt ist, verfügt über nur einen einzigen globalen Synchronisationspunkt.

ALGORITHMUS 4.5 (LANCZOS–ALGORITHMUS, ZWEI-TERM-VARIANTE 3)

Wähle $\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{w}}_1 \in \mathbb{C}^N$ so, daß $\widetilde{\delta}_1 = \widetilde{\mathbf{w}}_1^T\widetilde{\mathbf{v}}_1 \neq 0$ und $\rho_1 = \|\widetilde{\mathbf{v}}_1\|_2$, $\xi_1 = \|\widetilde{\mathbf{w}}_1\|_2$
Setze $\mathbf{p}_0 = \widetilde{\mathbf{q}}_0 = \mathbf{0}$ und $\varepsilon_0 \neq 0$
for $n = 1, 2, 3, \ldots$ **do**

- $\widetilde{\delta}_n = \widetilde{\mathbf{w}}_n^T\widetilde{\mathbf{v}}_n$
- $\sigma_n = \widetilde{\mathbf{w}}_n^T\mathbf{A}\widetilde{\mathbf{v}}_n$

$\mathbf{v}_n = \frac{1}{\rho_n}\widetilde{\mathbf{v}}_n$
$\mathbf{w}_n = \frac{1}{\xi_n}\widetilde{\mathbf{w}}_n$
$\mathbf{p}_n = \mathbf{v}_n - \frac{\widetilde{\delta}_n}{\varepsilon_{n-1}\rho_n}\mathbf{p}_{n-1}$
$\widetilde{\mathbf{q}}_n = \frac{1}{\xi_n}\mathbf{A}^T\widetilde{\mathbf{w}}_n - \frac{\widetilde{\delta}_n}{\varepsilon_{n-1}\xi_n}\widetilde{\mathbf{q}}_{n-1}$
$\beta_n = \begin{cases}\sigma_1/\widetilde{\delta}_1 & \textbf{if } n = 1\\ \sigma_n/\widetilde{\delta}_n - \widetilde{\delta}_n/\varepsilon_{n-1} & \textbf{if } n \geq 2\end{cases}$
$\varepsilon_n = \beta_n\widetilde{\delta}_n/(\rho_n\xi_n)$
$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \beta_n\mathbf{v}_n$
$\widetilde{\mathbf{w}}_{n+1} = \widetilde{\mathbf{q}}_n - \beta_n\mathbf{w}_n$

- $\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$
- $\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$

enddo

Da diese Variante in nachfolgenden Kapiteln verwendet wird, stellt der folgenden Satz ihre Eigenschaften zusammen.

Satz 4.1 (Parallele Variante des Lanczos–Algorithmus). *Falls Alg. 4.5 bis zum n-ten Schritt nicht abbricht, bilden die Lanczos–Vektoren $\{\mathbf{v}_i\}_{1\leq i\leq n}$ und $\{\mathbf{w}_i\}_{1\leq i\leq n}$ in exakter Arithmetik ein biorthogonales System, also*

$$\mathbf{W}_n^T\mathbf{V}_n = \mathbf{D}_n\,. \tag{4.13}$$

Außerdem bilden diese Lanczos–Vektoren und die Vektoren $\{\mathbf{p}_i\}_{1\le i\le n}$ *und* $\{\widetilde{\mathbf{q}}_i\}_{1\le i\le n}$ *eine Basis der Vektorräume*

$$\mathcal{K}_n\left(\mathbf{A},\mathbf{v}_1\right) = \operatorname{span}\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n\} = \operatorname{span}\{\mathbf{p}_1,\mathbf{p}_2,\ldots,\mathbf{p}_n\}, \tag{4.14}$$

$$\mathcal{K}_n\left(\mathbf{A}^T,\mathbf{w}_1\right) = \operatorname{span}\{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_n\}, \tag{4.15}$$

$$\mathbf{A}^T\mathcal{K}_n\left(\mathbf{A}^T,\mathbf{w}_1\right) = \operatorname{span}\{\widetilde{\mathbf{q}}_1,\widetilde{\mathbf{q}}_2,\ldots,\widetilde{\mathbf{q}}_n\}, \tag{4.16}$$

und es gelten die Beziehungen

$$\mathbf{V}_n = \mathbf{P}_n\mathbf{U}_n, \tag{4.17}$$

$$\mathbf{A}\mathbf{P}_n = \mathbf{V}_n\mathbf{L}_n + \rho_{n+1}\mathbf{v}_{n+1}\mathbf{e}_n^T, \tag{4.18}$$

$$\mathbf{A}^T\mathbf{W}_n = \widetilde{\mathbf{Q}}_n\mathbf{E}_n^{-1}\mathbf{L}_n^T\mathbf{D}_n, \tag{4.19}$$

$$\widetilde{\mathbf{Q}}_n = \mathbf{W}_n\mathbf{D}_n^{-1}\mathbf{U}_n^T\mathbf{E}_n + \xi_{n+1}\mathbf{w}_{n+1}\mathbf{e}_n^T, \tag{4.20}$$

$$\mathbf{W}_n^T\mathbf{A}\mathbf{V}_n = \mathbf{D}_n\mathbf{L}_n\mathbf{U}_n, \tag{4.21}$$

$$\widetilde{\mathbf{Q}}_n^T\mathbf{P}_n = \mathbf{E}_n. \tag{4.22}$$

Beweis. Im Vergleich zu der ursprünglichen Formulierung des Lanczos–Algorithmus mit gekoppelten Zwei-Term-Rekursionen, deren Eigenschaften in Satz 2.2 zusammengefaßt sind, treten keine Änderungen bezüglich der Größen auf, die zur Erzeugung von $\mathcal{K}_n\left(\mathbf{A},\mathbf{v}_1\right)$ beitragen. Deshalb wird an dieser Stelle auf die Beweise von (4.14), (4.17), (4.18) und (4.21) verzichtet, deren Herleitungen identisch zu denen in Satz 2.2 sind.

Die Biorthogonalität der Lanczos–Vektoren (4.13) wird durch Induktion über n gezeigt. Dabei bezieht sich dieser Beweis auf Alg. 4.4 und nicht auf Alg. 4.5. Da sich diese beiden Algorithmen lediglich durch eine verzögerte Berechnung der Lanczos–Vektoren unterscheiden, gilt die für Alg. 4.4 nachgewiesene Biorthogonalität in exakter Arithmetik ebenfalls für Alg. 4.5.

Die Induktionsvoraussetzung wird durch die Initialisierungsphase von Alg. 4.4 erfüllt. Unter der Annahme, daß die Vektoren $\{\mathbf{v}_i\}_{1\le i\le n}$ und $\{\mathbf{w}_i\}_{1\le i\le n}$ Gleichung (4.13) erfüllen, wird nun die Biorthogonalität der Vektoren $\{\mathbf{v}_i\}_{1\le i\le n+1}$ und $\{\mathbf{w}_i\}_{1\le i\le n+1}$ nachgewiesen. Dazu wird explizit gezeigt, daß

$$\mathbf{w}_{n+1}^T\mathbf{v}_i = 0 \qquad \text{für} \qquad i = 1,2,\ldots,n. \tag{4.23}$$

Aufgrund der Konstruktion in Alg. 4.4 gilt $\mathbf{w}_{n+1}^T\mathbf{v}_{n+1} = \delta_{n+1}$. Die dann noch fehlende Aussage

$$\mathbf{w}_i^T\mathbf{v}_{n+1} = 0 \qquad \text{für} \qquad i = 1,2,\ldots,n,$$

folgt analog zu (4.23) und wird deshalb hier nicht hergeleitet.

Für den Beweis von (4.23) werden einige Vorbemerkungen benötigt. Durch Elimination der $\widetilde{\mathbf{q}}_j$ aus den Anweisungen, die in Alg. 4.4 mit (4.10c) und (4.10d) markiert sind, leitet man die Beziehung

$$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T\mathbf{w}_n - \left(\frac{\rho_n\xi_n\delta_n}{\varepsilon_{n-1}} + \beta_n\right)\mathbf{w}_n - \frac{\rho_n\delta_n\beta_{n-1}}{\varepsilon_{n-1}}\mathbf{w}_{n-1} \tag{4.24}$$

mit $\mathbf{w}_0 = \mathbf{0}$ her. Dabei wird die Gleichung $\widetilde{\mathbf{w}}_n = \xi_n\mathbf{w}_n$ verwendet. Eine analoge Berücksichtigung von $\widetilde{\mathbf{v}}_n = \rho_n\mathbf{v}_n$ eliminiert alle $\mathbf{p}_j$ aus (4.10a) und (4.10b) und führt zu

$$\widetilde{\mathbf{v}}_{i+1} = \mathbf{A}\mathbf{v}_i - \left(\frac{\rho_i\xi_i\delta_i}{\varepsilon_{i-1}} + \beta_i\right)\mathbf{v}_i - \frac{\xi_i\delta_i\beta_{i-1}}{\varepsilon_{i-1}}\mathbf{v}_{i-1}, \qquad i = 1,2,\ldots,n.$$

Multipliziert man diese Gleichung mit $\mathbf{w}_n^T$ und benutzt die Induktionsannahme, so folgt

$$\mathbf{w}_n^T \mathbf{A} \mathbf{v}_i = \begin{cases} 0, & \text{falls } i = 1, 2, \ldots, n-2, \\ \rho_n \delta_n, & \text{falls } i = n-1. \end{cases}$$

Diese Beziehung wird neben der Induktionsannahme verwendet, um aus (4.24) den Zusammenhang

$$\widetilde{\mathbf{w}}_{n+1}^T \mathbf{v}_i = \mathbf{w}_n^T \mathbf{A} \mathbf{v}_i - \left(\frac{\rho_n \xi_n \delta_n}{\varepsilon_{n-1}} + \beta_n \right) \mathbf{w}_n^T \mathbf{v}_i - \frac{\rho_n \delta_n \beta_{n-1}}{\varepsilon_{n-1}} \mathbf{w}_{n-1}^T \mathbf{v}_i, \qquad i = 1, 2, \ldots, n,$$

$$= \begin{cases} 0, & \text{falls } i = 1, 2, \ldots, n-2, \\ \rho_n \delta_n - \dfrac{\rho_n \delta_n \beta_{n-1}}{\varepsilon_{n-1}} \delta_{n-1}, & \text{falls } i = n-1 \\ \mathbf{w}_n^T \mathbf{A} \mathbf{v}_n - \left(\dfrac{\rho_n \xi_n \delta_n}{\varepsilon_{n-1}} + \beta_n \right) \delta_n, & \text{falls } i = n, \end{cases}$$

zu folgern. Um daraus (4.23) nachzuweisen, benutzt man für die Fälle $i = n-1$ und $i = n$ die Anweisungen (4.9) und (4.8).

Die Beziehungen (4.15) und (4.16) gelten aufgrund der Konstruktion der gekoppelten Anweisungen für $\widetilde{\mathbf{q}}_n$ und $\widetilde{\mathbf{w}}_{n+1}$.

Gleichungen (4.19)–(4.20) folgen aus der ursprünglichen gekoppelten Zwei-Term-Implementierung mit dem Ansatz (4.4). Gleichung (4.22) zeigt man unter Verwendung der bereits bewiesenen Beziehungen (4.20), (4.17) und der Biorthogonalität (4.13)

$$\begin{aligned} \widetilde{\mathbf{Q}}_n^T \mathbf{P}_n &= \left(\mathbf{W}_n \mathbf{D}_n^{-1} \mathbf{U}_n^T \mathbf{E}_n + \xi_{n+1} \mathbf{w}_{n+1} \mathbf{e}_n^T \right)^T \mathbf{V}_n \mathbf{U}_n^{-1} \\ &= \mathbf{E}_n \mathbf{U}_n \mathbf{D}_n^{-1} \mathbf{W}_n^T \mathbf{V}_n \mathbf{U}_n^{-1} \\ &= \mathbf{E}_n. \end{aligned}$$

□

Kapitel 5

Spezielle Minimierungsprobleme

Die vorangehenden drei Kapitel bilden einen in sich abgeschlossenen Teil der vorliegenden Arbeit, in dem der Entwurf von iterativen Verfahren zur Erzeugung von Krylov–Teilräumen unter besonderer Berücksichtigung von Parallelisierungsaspekten behandelt wird. Mit den dort entwickelten Verfahren ist es möglich, eine Basis von Krylov–Teilräumen zu generieren. Um Krylov–Teilraumverfahren zur Lösung von linearen Gleichungssystemen herzuleiten, fehlt nun noch eine Angabe darüber, welcher Vektor aus dem aufgespannten Teilraum als aktuelle Approximation an die exakte Lösung ausgewählt wird. Diese Auswahl wird im nächsten Kapitel unter Verwendung unterschiedlicher Minimierungsprobleme getroffen. Genauer gesagt wird dabei die Lösung einer Folge von Minimierungsproblemen benötigt. Diese Minimierungsprobleme besitzen eine spezielle Form und werden außerdem in unterschiedlichen Normen angewendet. Dieses Kapitel übernimmt die Aufgabe, die Form dieser speziellen Minimierungsprobleme vorzustellen sowie Techniken zu deren effizienten Lösung in unterschiedlichen Normen bereitzustellen. Im Anschluß an die wohl in praktischen Anwendungen am häufigsten verwendete euklidische Norm werden 1-Norm und allgemeine p-Normen für $1 < p \leq \infty$ diskutiert.

5.1 Spezielle Form der Minimierungsprobleme

Bevor Minimierungsprobleme einer speziellen Form untersucht werden, wird zunächst das allgemeine Problem betrachtet, zu vorgegebener Matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ mit $m > n$ und vorgegebenem Vektor $\mathbf{b} \in \mathbb{C}^m$ einen Vektor $\mathbf{z} \in \mathbb{C}^n$ mit $\mathbf{A}\mathbf{z} = \mathbf{b}$ zu finden. Da dieses überbestimmte System im allgemeinen keine eindeutige Lösung besitzt, ist es naheliegend, stattdessen die Minimierung des Vektors $\mathbf{b} - \mathbf{A}\mathbf{z}$ in einer geeigneten Norm zu diskutieren. Eine in der numerischen linearen Algebra weit verbreitete Klasse von Vektornormen sind die Hölder- oder p-Normen.

Definition 5.1. Als *p-Norm* eines Vektors $\mathbf{x} = (x_1, x_2, \ldots, x_m)^T \in \mathbb{C}^m$ wird die reellwertige Abbildung

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p}, \qquad 1 \leq p \leq \infty,$$

bezeichnet.

Der Grenzwert von $\|\mathbf{x}\|_p$ für $p \to \infty$ existiert und bildet einen der drei in der Praxis am meisten

verwendeten Spezialfälle:

$$\begin{aligned}\|\mathbf{x}\|_1 &= |x_1| + |x_2| + \cdots + |x_m|,\\ \|\mathbf{x}\|_2 &= \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_m|^2},\\ \|\mathbf{x}\|_\infty &= \max_{1\le i\le m} |x_i|.\end{aligned}$$

Ein p-Norm-Minimierungsproblem in der allgemeinen Form besteht nun darin, die Funktion $f(\mathbf{z}) = \|\mathbf{b} - \mathbf{A}\mathbf{z}\|_p$ für ein vorgegebenes $p \in \mathbb{R}$ zu minimieren. Dabei führen unterschiedliche Wahlen von p sowohl zu unterschiedlichen Lösungen als auch zu unterschiedlichen dort angenommenen Minima. Die analytische Betrachtung des allgemeinen p-Norm-Minimierungsproblems konzentriert sich vor allem darauf, Bedingungen für die Eindeutigkeit der Lösung zu finden. Während die Eindeutigkeit des minimierten Vektors in den Fällen $1 < p < \infty$ durch die strikte Konvexität des genormten Raumes gesichert werden kann, sind in den Fällen $p = 1$ und $p = \infty$ andere Gesichtspunkte zu berücksichtigen. Für $p = \infty$ kann die sogenannte Haar-Bedingung [14, 69] als hinreichende Bedingung für die Eindeutigkeit Anwendung finden. Eine entsprechende "einfache" Bedingung ist für den Fall $p = 1$ unbekannt. Numerische Verfahren zur Lösung von Minimierungsproblemen in der 1-Norm oder der ∞-Norm werden darüber hinaus durch die Tatsache erschwert, daß die Funktion $f(\mathbf{z}) = \|\mathbf{b} - \mathbf{A}\mathbf{z}\|_p$ für diese Werte von p nicht differenzierbar ist. Eine Behandlung von allgemeinen p-Norm-Minimierungsproblemen unter Berücksichtigung von numerischen Techniken zu deren Lösung findet man in [69]. Neuere Arbeiten zur Minimierung in der 1-Norm und ∞-Norm sind in [15, 54, 73] enthalten.

Im Gegensatz zu Problemen der allgemeinen Form stehen im Mittelpunkt dieses Kapitels Minimierungsprobleme eines ganz bestimmten Typs, die in anderen Kapiteln dieser Arbeit Anwendung finden. Diese speziellen Minimierungsprobleme sind von der Form

$$\min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}\|_p, \qquad 1 \le p \le \infty, \tag{5.1}$$

wobei

$$\mathbf{f}_{n+1} = (\varepsilon_1, 0, 0, \ldots, 0)^T \in \mathbb{R}^{n+1}, \qquad \varepsilon_1 > 0,$$

ein reelles Vielfaches des ersten Standardeinheitsvektors des $\mathbb{R}^{n+1}$ und

$$\widehat{\mathbf{H}}_n \in \mathbb{C}^{(n+1)\times n} \qquad \text{mit} \qquad h_{ij} = 0 \quad \text{für} \quad i > j+1$$

eine obere Hessenberg–Matrix bezeichnet. Solange nicht explizit auf ein anderes Szenario hingewiesen wird, besitzt diese Hessenberg–Matrix vollen Rang.

Das Ziel dieses Kapitels ist nicht etwa, die Lösung eines p-Norm-Minimierungsproblems der Form (5.1) für lediglich ein festes $n \in \mathbb{N}$ anzugeben. Vielmehr ist die effiziente Lösung einer Folge von Minimierungsproblemen solchen Typs gesucht, in der die Hessenberg–Matrix $\widehat{\mathbf{H}}_n$ in jedem Schritt um eine Zeile und Spalte erweitert wird. Unter Effizienz wird in diesem Zusammenhang verstanden, daß sich die Lösung eines Problems zur Matrix $\widehat{\mathbf{H}}_n$ aus den Lösungen der unmittelbaren Vorgängerprobleme zu den Matrizen $\widehat{\mathbf{H}}_{n-1}, \widehat{\mathbf{H}}_{n-2}, \ldots, \widehat{\mathbf{H}}_{n-k}$ mit möglichst kleinem k berechnen läßt. Bezeichnet $\mathbf{z}_n$ die Lösung des Minimierungsproblems (5.1), also

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}_n\|_p = \min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}\|_p,$$

dann besteht die Aufgabe darin, eine Rekursionsvorschrift für $\mathbf{z}_n$ von möglichst geringer Tiefe herzuleiten. Die folgenden Abschnitte gliedern diese Aufgabe nach unterschiedlichen Werten von p. Dabei wird in den Fällen $p \neq 2$ für die Hessenberg–Matrix $\widehat{\mathbf{H}}_n$ zusätzlich die spezielle Struktur einer unteren Bidiagonalmatrix angenommen. Der Fall $p = 1$ erlaubt sogar die allgemeinere Struktur einer Tridiagonalmatrix.

5.2 Minimierung in der euklidischen Norm

Die wohl gebräuchlichste Form von p-Norm-Minimierungsproblemen in der numerischen linearen Algebra tritt für den Fall $p = 2$ auf. Dieser wichtige Spezialfall der euklidischen Norm ist Gegenstand zahlreicher Monographien [3, 22, 53]. In dieser Arbeit steht die Minimierung der speziellen Form

$$\min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}\|_2 \tag{5.2}$$

im Vordergrund, die im Vergleich zu den Fällen $p \neq 2$ in (5.1) durch folgende Punkte vereinfacht wird:

- Die Lösung von (5.2) ist eindeutig, weil die Matrix $\widehat{\mathbf{H}}_n$ vollen Rang besitzt.
- Die Funktion $f(\mathbf{z}) = \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}\|_2$ ist differenzierbar.
- Die euklidische Norm ist invariant gegenüber unitären Transformationen.

Die letztgenannte Eigenschaft der euklidischen Norm begründet die Verwendung von unitären Transformationen wie beispielsweise Givens–Rotationen als Standardtechnik zur Lösung von Minimierungsproblemen der Form (5.2). Dabei wird eine geeignete unitäre Transformation $\mathbf{Q}_{n+1}$ gesucht, so daß das Problem

$$\min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{Q}_{n+1}\mathbf{f}_{n+1} - \mathbf{Q}_{n+1}\widehat{\mathbf{H}}_n\mathbf{z}\|_2 \tag{5.3}$$

einfacher zu lösen ist als das ursprüngliche Problem (5.2). Im Anschluß an Techniken, die diese Invarianz der euklidischen Norm ausnutzen, wird ein alternativer Ansatz über die Methode der Normalgleichungen beschrieben.

5.2.1 QR-Zerlegung mit Hilfe von Givens–Rotationen

Die Standardtechnik zur Lösung von (5.2) besteht darin, eine QR-Zerlegung der $(n+1)\times n$ Matrix $\widehat{\mathbf{H}}_n$ zu berechnen. Aufgrund der Hessenberg–Form von $\widehat{\mathbf{H}}_n$ bieten sich aufeinanderfolgende Givens–Rotationen an, die die Elemente der unteren Nebendiagonale nacheinander eliminieren. Eine QR-Zerlegung mittels aufeinanderfolgender Householder–Transformationen benötigt dagegen mehr arithmetische Operationen, weil diese Vorgehensweise jeweils auf die gesamte aktuelle Spalte wirkt. Dadurch wird die Struktur der Matrix $\widehat{\mathbf{H}}_n$, in der im unteren linken Teil bereits Nullen vorhandenen sind, nicht berücksichtigt. Dagegen wirken Givens–Rotationen auf nur jeweils zwei Elemente der aktuellen Spalte, so daß im Verlaufe des Prozesses die untere Nebendiagonale gezielt annulliert wird. Das folgende Schema illustriert die grundsätzliche Vorgehensweise für $n = 4$:

$$\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} \overset{\mathbf{G}_1}{\longmapsto} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} \overset{\mathbf{G}_2}{\longmapsto} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} \overset{\mathbf{G}_3}{\longmapsto} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \end{bmatrix} \overset{\mathbf{G}_4}{\longmapsto} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

In dieser Vorgehensweise wird im i-ten Schritt eine durch die Matrix $\mathbf{G}_i$ beschriebene Givens–Rotation durchgeführt, die das Matrixelement in Zeile $i+1$ und Spalte i annulliert. In formaler

Notation ist $\mathbf{G}_i$ eine $(n+1) \times (n+1)$ Matrix, die sich von der Einheitsmatrix $\mathbf{I}_{n+1}$ nur an den Schnittpunkten der i-ten und $(i+1)$-ten Zeile und Spalte unterscheidet

$$\mathbf{G}_i = \begin{bmatrix} \mathbf{I}_{i-1} & 0 & 0 & 0 \\ 0 & c_i & -\overline{s}_i & 0 \\ 0 & s_i & c_i & 0 \\ 0 & 0 & 0 & \mathbf{I}_{n-i} \end{bmatrix},$$

wobei $c_i \geq 0$ und $s_i \in \mathbb{C}$ geeignet gewählte Größen mit $c_i^2 + |s_i|^2 = 1$ sind. Die unitäre Transformation $\mathbf{Q}_{n+1}$ aus (5.3) ist dann durch das Produkt von n aufeinanderfolgenden Givens–Rotationen

$$\mathbf{Q}_{n+1} = \mathbf{G}_n \mathbf{G}_{n-1} \cdots \mathbf{G}_1 \in \mathbb{C}^{(n+1)\times(n+1)}$$

gegeben. Wegen $\mathbf{Q}_{n+1} = \mathbf{G}_n \mathbf{Q}_n$ ist diese Vorgehensweise leicht von einem Iterationsschritt zum nächsten aufzudatieren. Die Matrix $\mathbf{Q}_{n+1}\widehat{\mathbf{H}}_n$ wird so zu einer oberen Dreiecksmatrix, und die Lösung von (5.3) ist einfach anzugeben. Diese Technik wird in dem folgenden Lemma, das in [28] angegeben ist und das implizit in den Arbeiten [7, 35, 58, 63] enthalten ist, benutzt, um eine Folge von Minimierungsproblemen der Form (5.2) zu lösen. Aus Gründen, die im nächsten Kapitel ersichtlich werden, wird nachfolgend für den Vektor $\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n$ der Begriff *Quasi-Residuum* benutzt.

Lemma 5.1 (Euklidisches Minimierungsproblem mit Hessenberg–Struktur). *Für $n \geq 1$ sei $\widehat{\mathbf{H}}_n$ eine Familie von $(n+1) \times n$ oberen Hessenberg–Matrizen der Form*

$$\widehat{\mathbf{H}}_n = \begin{bmatrix} & \mathbf{H}_n & \\ \mathbf{0}_{n-1}^T & & \varepsilon_{n+1} \end{bmatrix}$$

mit vollem Rang. Dabei ist $\mathbf{H}_n$ diejenige $n \times n$ Matrix, die durch Streichen der letzten Zeile von $\widehat{\mathbf{H}}_n$ entsteht. Es sei $\varepsilon_1 > 0$, und $\mathbf{z}_n \in \mathbb{C}^n$ bezeichne die eindeutige Lösung des euklidischen Minimierungsproblems

$$\tau_n := \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n\|_2 = \min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}\|_2,$$

wobei $\mathbf{f}_{n+1} = \varepsilon_1 \mathbf{e}_1^{(n+1)} \in \mathbb{R}^{n+1}$ und τ_n die Größe des Quasi-Residuums bezeichnet. Außerdem sei $\mathbf{H}_n$ nicht singulär und $\widetilde{\mathbf{z}}_n := \mathbf{H}_n^{-1}\mathbf{f}_n$ gesetzt. Dann gelten die Rekursionen

$$\mathbf{z}_n = (1 - c_n^2) \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix} + c_n^2 \widetilde{\mathbf{z}}_n, \tag{5.4}$$

$$\tau_n = \tau_{n-1} \vartheta_n c_n, \tag{5.5}$$

wobei die Anfangsbedingungen durch $\mathbf{z}_1 = c_1^2 \widetilde{\mathbf{z}}_1$, $\tau_0 = \varepsilon_1$ und die weiteren Größen durch

$$\vartheta_n = \frac{1}{\tau_{n-1}} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n\|_2, \tag{5.6}$$

$$c_n = \frac{1}{\sqrt{1 + \vartheta_n^2}}, \tag{5.7}$$

gegeben sind.

5.2.2 Methode der Normalgleichungen

Im letzten Teilabschnitt wird das Problem (5.2) durch eine QR-Zerlegung der Matrix $\widehat{\mathbf{H}}_n$ mit Hilfe von Givens–Rotationen gelöst. Eine Alternative zur QR-Zerlegung ist die Methode der Normalgleichungen. Es ist keinesfalls einfach zu entscheiden, welche der beiden Methoden für ein gegebenes Problem vorzuziehen ist [38, Sect. 5.3.8]. In diesem Abschnitt wird die Methode der Normalgleichungen implizit in einem Beweis angewendet, um das Problem (5.2) für eine spezielle Form der Matrix $\widehat{\mathbf{H}}_n$ zu lösen. Es wird das Problem

$$\min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}\|_2 \tag{5.8}$$

betrachtet, wobei $\widehat{\mathbf{B}}_n$ eine untere Bidiagonalmatrix bezeichnet. Das folgende Lemma gibt eine effiziente Lösung für eine Folge von Minimierungsproblemen der Form (5.8) an.

Lemma 5.2 (Euklidisches Minimierungsproblem mit Bidiagonalstruktur). *Für $n \geq 1$ sei $\widehat{\mathbf{B}}_n$ eine Familie von $(n+1) \times n$ unteren Bidiagonalmatrizen der Form*

$$\widehat{\mathbf{B}}_n = \begin{bmatrix} \delta_1 & & & \\ \varepsilon_2 & \delta_2 & & \mathbf{0} \\ & \ddots & \ddots & \\ & & \varepsilon_n & \delta_n \\ \mathbf{0} & & & \varepsilon_{n+1} \end{bmatrix}$$

mit vollem Rang. Es sei $\varepsilon_1 > 0$, und $\mathbf{z}_n \in \mathbb{C}^n$ bezeichne die eindeutige Lösung des euklidischen Minimierungsproblems

$$\tau_n := \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}_n\|_2 = \min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}\|_2,$$

wobei $\mathbf{f}_{n+1} = \varepsilon_1 \mathbf{e}_1^{(n+1)} \in \mathbb{R}^{n+1}$ und τ_n die Größe des Quasi-Residuums bezeichnet. Dann gelten die Rekursionen

$$\mathbf{z}_n = \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix} + \mathbf{g}_n, \tag{5.9}$$

$$\mathbf{g}_n = \theta_n \begin{bmatrix} \mathbf{g}_{n-1} \\ 0 \end{bmatrix} + \kappa_n \mathbf{e}_n^{(n)}, \tag{5.10}$$

$$\tau_n = \sqrt{\left| \frac{\tau_{n-1} |\varepsilon_{n+1}|^2}{\lambda_n |\delta_n|^2 + |\varepsilon_{n+1}|^2} \right|^2 + |\varepsilon_{n+1} \kappa_n|^2} \tag{5.11}$$

wobei die Anfangsbedingungen durch $\mathbf{z}_1 = \mathbf{g}_1 = \kappa_1$, $\tau_0 = \varepsilon_1$ und die Koeffizienten durch

$$\theta_n = \frac{|\delta_n|^2 (1 - \lambda_n)}{\lambda_n |\delta_n|^2 + |\varepsilon_{n+1}|^2}, \qquad n \geq 1, \tag{5.12}$$

$$\kappa_n = \frac{-\varepsilon_n \overline{\delta}_n \, \kappa_{n-1}}{\lambda_n |\delta_n|^2 + |\varepsilon_{n+1}|^2}, \qquad n \geq 1, \qquad \kappa_0 = -1, \tag{5.13}$$

mit

$$\lambda_n = \frac{\lambda_{n-1} |\delta_{n-1}|^2}{\lambda_{n-1} |\delta_{n-1}|^2 + |\varepsilon_n|^2}, \qquad n \geq 2, \qquad \lambda_1 = 1, \tag{5.14}$$

gegeben sind.

Beweis. Setzt man zuerst (5.10) in (5.9) ein und ersetzt anschließend $\mathbf{g}_{n-1}$ durch (5.9), so folgt

$$\begin{aligned}\mathbf{z}_n &= \begin{bmatrix}\mathbf{z}_{n-1}\\ 0\end{bmatrix} + \theta_n \begin{bmatrix}\mathbf{g}_{n-1}\\ 0\end{bmatrix} + \kappa_n \mathbf{e}_n^{(n)} \\ &= (1+\theta_n)\begin{bmatrix}\mathbf{z}_{n-1}\\ 0\end{bmatrix} - \theta_n \begin{bmatrix}\mathbf{z}_{n-2}\\ 0\\ 0\end{bmatrix} + \kappa_n \mathbf{e}_n^{(n)}.\end{aligned}$$

Das Lemma aus [65], das mit Hilfe der Methode der Normalgleichungen bewiesen wird, zeigt dann (5.9),(5.10) und (5.12)–(5.14).

Um (5.11) zu zeigen, sei $\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}_n =: (\mu_1, \mu_2, \ldots, \mu_{n+1})^T$. Da das Quasi-Residuum die Gleichung $\widehat{\mathbf{B}}_n^H (\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}_n) = \mathbf{0}$ erfüllt und die letzte Komponente des Quasi-Residuums durch

$$\mu_{n+1} = -\varepsilon_{n+1}\kappa_n$$

gegeben ist, folgt

$$\mu_i = -\frac{\overline{\varepsilon}_{i+1}}{\overline{\delta}_i}\mu_{i+1}, \qquad i = n, n-1, \ldots, 1.$$

Damit kann man leicht die Rekursion

$$\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}_n = -\frac{|\varepsilon_{n+1}|^2 \kappa_n}{\overline{\delta}_n \varepsilon_n \kappa_{n-1}} \begin{bmatrix}\mathbf{f}_n - \widehat{\mathbf{B}}_{n-1}\mathbf{z}_{n-1}\\ 0\end{bmatrix} - \varepsilon_{n+1}\kappa_n \mathbf{e}_{n+1}^{(n+1)}$$

mit $\mathbf{f}_1 - \widehat{\mathbf{B}}_0 \mathbf{z}_0 = -\varepsilon_1 \kappa_0$ zeigen. Mit (5.13) folgt daraus

$$\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}_n = \frac{|\varepsilon_{n+1}|^2}{\lambda_n |\delta_n|^2 + |\varepsilon_{n+1}|^2} \begin{bmatrix}\mathbf{f}_n - \widehat{\mathbf{B}}_{n-1}\mathbf{z}_{n-1}\\ 0\end{bmatrix} - \varepsilon_{n+1}\kappa_n \mathbf{e}_{n+1}^{(n+1)}$$

mit $\mathbf{f}_1 - \widehat{\mathbf{B}}_0 \mathbf{z}_0 = \varepsilon_1$. Diese Gleichung zeigt in Verbindung mit der Definition von τ_n die Rekursion (5.11). □

5.3 Minimierung in der 1-Norm

In diesem Abschnitt wird das Minimierungsproblem (5.1) für den Spezialfall $p = 1$ betrachtet, also

$$\min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}\|_1. \tag{5.15}$$

Im Vergleich zu der im vorherigen Abschnitt betrachteten euklidischen Norm wird das Minimierungsproblem (5.15) in der 1-Norm dadurch erschwert, daß (i) die Lösung nicht eindeutig ist, (ii) die Funktion $f(\mathbf{z}) = \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}\|_1$ nicht differenzierbar ist, und (iii) die 1-Norm unter unitären Transformationen nicht invariant ist. Man beachte, daß die letztgenannte Eigenschaft der 1-Norm die Verwendung von Givens–Rotationen zur Lösung von (5.15) ausschließt. Daher ist es um so erstaunlicher, daß man eine Folge von Minimierungsproblemen der Form (5.15) effizient lösen kann. Effizienz bedeutet hier, daß die Lösungen dieser Folge von Minimierungsproblemen einer Rekursion von geringer Tiefe genügen. Um diese Effizienz zu gewährleisten, wird jedoch eine weitere Forderung an die obere Hessenberg–Matrix $\widehat{\mathbf{H}}_n$ aus (5.15) gestellt. Die Hessenberg–Matrix muß entweder eine untere Bidiagonalmatrix oder aber eine Tridiagonalmatrix sein. Die folgenden Teilabschnitte diskutieren diese beiden Spezialfälle. Abschließend wird nochmals der allgemeine Fall einer oberen Hessenberg–Matrix betrachtet.

5.3.1 Bidiagonalstruktur

In diesem Teilabschnitt wird angenommen, daß die obere Hessenberg–Matrix $\widehat{\mathbf{H}}_n$ aus (5.15) die spezielle Form einer unteren Bidiagonalmatrix besitzt, für die das neue Symbol $\widehat{\mathbf{B}}_n$ eingeführt wird. In diesem Spezialfall kann eine effiziente Lösung einer Folge von Minimierungsproblemen der Form

$$\min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n\mathbf{z}\|_1 \tag{5.16}$$

effizient angegeben werden. Das folgende Lemma zeigt nämlich, daß man zur Berechnung einer Lösung $\mathbf{z}_n$ von (5.16) lediglich die Lösung des Vorgängerproblems $\mathbf{z}_{n-1}$ benötigt.

Lemma 5.3 (1-Norm-Minimierungsproblem mit Bidiagonalstruktur). *Für $n \geq 1$ sei $\widehat{\mathbf{B}}_n$ eine Familie von $(n+1)\times n$ unteren Bidiagonalmatrizen der Form*

$$\widehat{\mathbf{B}}_n = \begin{bmatrix} \delta_1 & & \mathbf{0} & \\ \varepsilon_2 & \delta_2 & & \\ & \ddots & \ddots & \\ \mathbf{0} & & \varepsilon_n & \delta_n \\ & & & \varepsilon_{n+1} \end{bmatrix} = \begin{bmatrix} \widehat{\mathbf{B}}_{n-1} & \mathbf{0}_{n-1} \\ \mathbf{0}_{n-1}^T & \delta_n \\ & \varepsilon_{n+1} \end{bmatrix} = \begin{bmatrix} & \mathbf{B}_n & \\ \mathbf{0}_{n-1}^T & & \varepsilon_{n+1} \end{bmatrix}, \tag{5.17}$$

wobei $\varepsilon_2, \varepsilon_3, \ldots, \varepsilon_n \neq 0$. Dabei ist $\widehat{\mathbf{B}}_{n-1}$ die $n\times(n-1)$ Hauptuntermatrix von $\widehat{\mathbf{B}}_n$, und $\mathbf{B}_n$ ist diejenige $n\times n$ Matrix, die durch Streichen der letzten Zeile von $\widehat{\mathbf{B}}_n$ entsteht. Es sei $\varepsilon_1 > 0$, und $\mathbf{z}_n \in \mathbb{C}^n$ bezeichne eine Lösung des 1-Norm-Minimierungsproblems

$$\tau_n := \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n\mathbf{z}_n\|_1 = \min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n\mathbf{z}\|_1, \tag{5.18}$$

wobei $\mathbf{f}_{n+1} = \varepsilon_1\mathbf{e}_1^{(n+1)} \in \mathbb{R}^{n+1}$ und τ_n die Größe des Quasi-Residuums bezeichnet. Außerdem sei $\mathbf{B}_n$ nicht singulär und $\widetilde{\mathbf{z}}_n := \mathbf{B}_n^{-1}\mathbf{f}_n$ gesetzt. Dann gilt:

(i) Der Hilfsvektor $\widetilde{\mathbf{z}}_n \in \mathbb{C}^n$ wird nur in seiner jeweils letzten Komponente aufdatiert

$$\widetilde{\mathbf{z}}_n = \begin{bmatrix} \widetilde{\mathbf{z}}_{n-1} \\ \nu_n \end{bmatrix}, \tag{5.19}$$

wobei die Anfangsbedingungen durch $\widetilde{\mathbf{z}}_1 = \nu_1$ und dessen letzte Komponente durch

$$\nu_n = \begin{cases} -1, & \text{falls } n = 0, \\ -\dfrac{\varepsilon_n}{\delta_n}\nu_{n-1}, & \text{falls } n \geq 1, \end{cases} \tag{5.20}$$

gegeben ist.

(ii) Eine (spezielle) Lösung $\mathbf{z}_n \in \mathbb{C}^n$ von (5.18) *steht mit $\widetilde{\mathbf{z}}_n$ in dem Zusammenhang*

$$\mathbf{z}_n = \begin{cases} \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix}, & \text{falls } |\varepsilon_{n+1}\nu_n| \geq \tau_{n-1}, \\ \widetilde{\mathbf{z}}_n, & \text{falls } |\varepsilon_{n+1}\nu_n| < \tau_{n-1}, \end{cases} \tag{5.21}$$

und die Größe des Quasi-Residuums genügt der Rekursion

$$\tau_n = \min\{\tau_{n-1}, |\varepsilon_{n+1}\nu_n|\} \qquad \text{mit} \qquad \tau_0 = \varepsilon_1. \tag{5.22}$$

Beweis. Man beachte zuerst, daß $\delta_i \neq 0$ für $i = 1, 2, \ldots, n$, weil $\mathbf{B}_n$ nicht singulär ist.

(i) Die Rekursion (5.19) wird durch Induktion über n gezeigt und macht Gebrauch von der Beobachtung, daß ν_n die letzte Komponente von $\widetilde{\mathbf{z}}_n$ ist. Für $n = 1$ gilt $\widetilde{\mathbf{z}}_1 = \mathbf{B}_1^{-1}\mathbf{f}_1 = \varepsilon_1/\delta_1 = \nu_1$. Unter Verwendung der Induktionsannahme folgt die Beziehung

$$\mathbf{B}_n\widetilde{\mathbf{z}}_n = \begin{bmatrix} \mathbf{B}_{n-1} & \mathbf{0}_{n-1} \\ \mathbf{0}_{n-2}^T \quad \varepsilon_n & \delta_n \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{z}}_{n-1} \\ \nu_n \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{n-1}\widetilde{\mathbf{z}}_{n-1} \\ \varepsilon_n\nu_{n-1} + \delta_n\nu_n \end{bmatrix} = \mathbf{f}_n,$$

wobei die letzte Umformung durch (5.20) impliziert wird.

(ii) Da die Konstruktion von $\mathbf{z}_n$ aus (5.21) unmittelbar $\|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n\mathbf{z}_n\|_1 = \min\{\tau_{n-1}, |\varepsilon_{n+1}\nu_n|\}$ liefert, genügt ein Induktionsbeweis der Aussage

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n\mathbf{z}\|_1 \geq \min\{\tau_{n-1}, |\varepsilon_{n+1}\nu_n|\}.$$

Wegen $\varepsilon_i \neq 0$ für $i = 1, 2, \ldots, n$ gilt $\nu_i \neq 0$ für $i = 1, 2, \ldots, n$. Deshalb entsteht eine Basis des $\mathbb{C}^n$ durch die mit Nullen aufgefüllten Vektoren $\widetilde{\mathbf{z}}_i$, und jedes $\mathbf{z} \in \mathbb{C}^n$ ist von der Form

$$\mathbf{z} = \sum_{i=1}^{n} \alpha_i \begin{bmatrix} \widetilde{\mathbf{z}}_i \\ \mathbf{0}_{n-i} \end{bmatrix}. \tag{5.23}$$

Als eine Vorbemerkung, die durch die Zerlegung von $\widehat{\mathbf{B}}_n$ gemäß (5.17) leicht zu verifizieren ist, wird auf

$$\widehat{\mathbf{B}}_n \begin{bmatrix} \widetilde{\mathbf{z}}_i \\ \mathbf{0}_{n-i} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_i \\ \varepsilon_{i+1}\nu_i \\ \mathbf{0}_{n-i} \end{bmatrix}, \qquad i = 1, 2, \ldots, n, \tag{5.24}$$

hingewiesen. Deshalb führt das Einsetzen von (5.23) in die rechte Seite von (5.18) mit anschließendem Sortieren der Komponenten auf

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n\mathbf{z}\|_1 = \left\| \left(1 - \sum_{i=1}^{n} \alpha_i\right) \mathbf{f}_{n+1} - \sum_{i=1}^{n} \alpha_i\varepsilon_{i+1}\nu_i\mathbf{e}_{i+1}^{(n+1)} \right\|_1, \tag{5.25}$$

wobei $\mathbf{e}_{i+1}^{(n+1)} \in \mathbb{R}^{n+1}$ den Standardeinheitsvektor mit einer 1 an Position $i+1$ bezeichnet. Falls $\alpha_n = 1$, so folgt $\|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n\mathbf{z}\|_1 \geq |\varepsilon_{n+1}\nu_n|$. Für $\alpha_n \neq 1$ ist eine äquivalente Darstellung der rechten Seite von (5.25) durch

$$|1 - \alpha_n| \cdot \left\| \left(1 - \sum_{i=1}^{n-1} \frac{\alpha_i}{1 - \alpha_n}\right) \mathbf{f}_n - \sum_{i=1}^{n-1} \frac{\alpha_i}{1 - \alpha_n}\varepsilon_{i+1}\nu_i\mathbf{e}_{i+1}^{(n)} \right\|_1 + |\alpha_n\varepsilon_{n+1}\nu_n|$$

gegeben. Mit der Induktionsannahme folgen die Ungleichungen:

$$\begin{aligned} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n\mathbf{z}\|_1 &\geq |1 - \alpha_n|\tau_{n-1} + |\alpha_n\varepsilon_{n+1}\nu_n| \\ &\geq \min\{\tau_{n-1}, |\varepsilon_{n+1}\nu_n|\} \cdot (|1 - \alpha_n| + |\alpha_n|) \\ &\geq \min\{\tau_{n-1}, |\varepsilon_{n+1}\nu_n|\}, \end{aligned}$$

wobei die letzte Ungleichung aus der unmittelbaren Folgerung der Dreiecksungleichung, nämlich $|1 - \alpha_n| \geq 1 - |\alpha_n|$, resultiert. Der Induktionsanfang ist durch

$$\|\mathbf{f}_2 - \widehat{\mathbf{B}}_1\mathbf{z}\|_1 = |\varepsilon_1 - \delta_1 z| + |\varepsilon_2 z| \geq \min\{\varepsilon_1, |\varepsilon_2\varepsilon_1/\delta_1|\}$$

gegeben. □

5.3.2 Tridiagonalstruktur

Im vorangehenden Teilabschnitt wird eine Folge von 1-Norm-Minimierungsproblemen mit Bidiagonalstruktur effizient gelöst. Es wird nun gezeigt, daß man eine effiziente Lösung ebenfalls in dem allgemeineren Fall einer Tridiagonalmatrix $\widehat{\mathbf{T}}_n$ angeben kann. Dazu muß eine Folge von Minimierungsproblemen der Form

$$\min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{T}}_n \mathbf{z}\|_1$$

betrachtet werden. Das folgende Lemma gibt eine effiziente Lösung dieses Problems an.

Lemma 5.4 (1-Norm-Minimierungsproblem mit Tridiagonalstruktur). *Für $n \geq 1$ sei $\widehat{\mathbf{T}}_n$ eine Familie von $(n+1)\times n$ Tridiagonalmatrizen der Form*

$$\widehat{\mathbf{T}}_n = \begin{bmatrix} \delta_1 & \gamma_2 & & & \\ \varepsilon_2 & \delta_2 & \gamma_3 & & \mathbf{0} \\ & \varepsilon_3 & \ddots & \ddots & \\ & & \ddots & & \gamma_n \\ & \mathbf{0} & & \varepsilon_n & \delta_n \\ & & & & \varepsilon_{n+1} \end{bmatrix} = \begin{bmatrix} & \mathbf{T}_n & \\ \mathbf{0}_{n-1}^T & & \varepsilon_{n+1} \end{bmatrix},$$

wobei $\varepsilon_2, \varepsilon_3, \ldots, \varepsilon_n \neq 0$. Dabei ist $\mathbf{T}_n$ diejenige $n\times n$ Matrix, die durch Streichen der letzten Zeile von $\widehat{\mathbf{T}}_n$ entsteht. Es sei $\varepsilon_1 > 0$, und $\mathbf{z}_n \in \mathbb{C}^n$ bezeichne eine Lösung des 1-Norm-Minimierungsproblems

$$\tau_n := \|\mathbf{f}_{n+1} - \widehat{\mathbf{T}}_n \mathbf{z}_n\|_1 = \min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{T}}_n \mathbf{z}\|_1, \tag{5.26}$$

wobei $\mathbf{f}_{n+1} = \varepsilon_1 \mathbf{e}_1^{(n+1)} \in \mathbb{R}^{n+1}$ und τ_n die Größe des Quasi-Residuums bezeichnet. Außerdem sei $\mathbf{T}_n$ nicht singulär und $\widetilde{\mathbf{z}}_n := \mathbf{T}_n^{-1}\mathbf{f}_n$ gesetzt. Dann gilt:

(i) Der Hilfsvektor $\widetilde{\mathbf{z}}_n \in \mathbb{C}^n$ erfüllt die Rekursionen

$$\widetilde{\mathbf{z}}_n = \begin{bmatrix} \widetilde{\mathbf{z}}_{n-1} \\ 0 \end{bmatrix} - \gamma_n\nu_n \begin{bmatrix} \widetilde{\mathbf{y}}_{n-1} \\ 0 \end{bmatrix} + \nu_n \mathbf{e}_n^{(n)}, \tag{5.27}$$

$$\widetilde{\mathbf{y}}_n = -\gamma_n\kappa_n \begin{bmatrix} \widetilde{\mathbf{y}}_{n-1} \\ 0 \end{bmatrix} + \kappa_n \mathbf{e}_n^{(n)}, \tag{5.28}$$

wobei die Anfangsbedingungen durch $\widetilde{\mathbf{z}}_1 = \varepsilon_1/\delta_1$, $\widetilde{\mathbf{y}}_1 = 1/\delta_1$ und die letzten Komponenten von $\widetilde{\mathbf{z}}_n$ und $\widetilde{\mathbf{y}}_n$ durch

$$\nu_n = -\varepsilon_n\kappa_n\nu_{n-1}, \qquad n \geq 1, \qquad \nu_0 = -1, \tag{5.29}$$

$$\kappa_n = \frac{1}{\delta_n - \varepsilon_n\gamma_n\kappa_{n-1}}, \qquad n \geq 1, \qquad \kappa_0 = 0, \tag{5.30}$$

gegeben sind.

(ii) Eine (spezielle) Lösung $\mathbf{z}_n \in \mathbb{C}^n$ von (5.26) *steht mit $\widetilde{\mathbf{z}}_n$ in dem Zusammenhang*

$$\mathbf{z}_n = \begin{cases} \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix}, & \textit{falls } |\varepsilon_{n+1}\nu_n| \geq \tau_{n-1}, \\ \widetilde{\mathbf{z}}_n, & \textit{falls } |\varepsilon_{n+1}\nu_n| < \tau_{n-1}, \end{cases} \tag{5.31}$$

und die Größe des Quasi-Residuums genügt der Rekursion

$$\tau_n = \min\{\tau_{n-1}, |\varepsilon_{n+1}\nu_n|\} \qquad \textit{mit} \qquad \tau_0 = \varepsilon_1. \tag{5.32}$$

Beweis. (i) Um die Rekursionen (5.27) und (5.28) nachzuweisen, wird zunächst durch Induktion über n die Hilfsbehauptung $\mathbf{T}_n\widetilde{\mathbf{y}}_n = \mathbf{e}_n^{(n)}$ gezeigt. Für $n = 1$ gilt $\mathbf{T}_1\widetilde{\mathbf{y}}_1 = \delta_1/\delta_1 = 1$. Durch Einsetzen von (5.28) und unter Verwendung der Induktionsannahme folgt

$$\begin{aligned}
\mathbf{T}_n\widetilde{\mathbf{y}}_n &= \begin{bmatrix} \mathbf{T}_{n-1} & \mathbf{0}_{n-2} & \\ \mathbf{0}_{n-2}^T & \varepsilon_n & \delta_n \end{bmatrix} \left(-\gamma_n\kappa_n \begin{bmatrix} \widetilde{\mathbf{y}}_{n-1} \\ 0 \end{bmatrix} + \kappa_n \mathbf{e}_n^{(n)} \right) \\
&= -\gamma_n\kappa_n \begin{bmatrix} \mathbf{T}_{n-1}\widetilde{\mathbf{y}}_{n-1} \\ \varepsilon_n\kappa_{n-1} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n-2} \\ \gamma_n\kappa_n \\ \delta_n\kappa_n \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{0}_{n-2} \\ -\gamma_n\kappa_n + \gamma_n\kappa_n \\ -\gamma_n\kappa_n\varepsilon_n\kappa_{n-1} + \delta_n\kappa_n \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n-1} \\ 1 \end{bmatrix},
\end{aligned}$$

wobei die letzte Umformung durch (5.30) impliziert wird.

Ebenfalls durch Induktion über n wird nun $\mathbf{T}_n\widetilde{\mathbf{z}}_n = \mathbf{f}_n$ gezeigt. Der Induktionsanfang ist durch $\mathbf{T}_1\widetilde{\mathbf{z}}_1 = \delta_1\varepsilon_1/\delta_1 = \varepsilon_1$ gegeben. Durch Einsetzen von (5.27) und unter Verwendung der Induktionsannahme und der Hilfsbehauptung folgt

$$\begin{aligned}
\mathbf{T}_n\widetilde{\mathbf{z}}_n &= \begin{bmatrix} \mathbf{T}_{n-1} & \mathbf{0}_{n-2} & \\ \mathbf{0}_{n-2}^T & \varepsilon_n & \delta_n \end{bmatrix} \left(\begin{bmatrix} \widetilde{\mathbf{z}}_{n-1} \\ 0 \end{bmatrix} - \gamma_n\nu_n \begin{bmatrix} \widetilde{\mathbf{y}}_{n-1} \\ 0 \end{bmatrix} + \nu_n \mathbf{e}_n^{(n)} \right) \\
&= \begin{bmatrix} \mathbf{T}_{n-1}\widetilde{\mathbf{z}}_{n-1} \\ \varepsilon_n\nu_{n-1} \end{bmatrix} - \gamma_n\nu_n \begin{bmatrix} \mathbf{T}_{n-1}\widetilde{\mathbf{y}}_{n-1} \\ \varepsilon_n\kappa_{n-1} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n-2} \\ \gamma_n\nu_n \\ \delta_n\nu_n \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{f}_{n-1} \\ \varepsilon_n\nu_{n-1} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n-2} \\ -\gamma_n\nu_n + \gamma_n\nu_n \\ -\gamma_n\nu_n\varepsilon_n\kappa_{n-1} + \delta_n\nu_n \end{bmatrix} = \mathbf{f}_n,
\end{aligned}$$

wobei die letzte Umformung durch (5.29) und (5.30) impliziert wird.

(ii) Dieser Teil des Lemmas wird analog zum Beweis (ii) aus Lemma 5.3 gezeigt. □

Die Analogie der Beweise für die speziellen 1-Norm-Minimierungsprobleme mit Bidiagonal- und Tridiagonalstruktur wird im folgenden Teilabschnitt aufgenommen.

5.3.3 Obere Hessenberg–Struktur

In den vorangehenden beiden Teilabschnitten werden Folgen von 1-Norm-Minimierungsproblemen der Form

$$\min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}\|_1 \tag{5.33}$$

für spezielle obere Hessenberg–Matrizen $\widehat{\mathbf{H}}_n$ gelöst. Für bi- und tridiagonale Matrizen werden effiziente Vorgehensweisen zur Lösung der Minimierungsprobleme angegeben. In beiden Spezialfällen machen die angegebenen Lösungen Gebrauch von einer Hilsfolge von Vektoren $\widetilde{\mathbf{z}}_n \in \mathbb{C}^n$,

die auf eine spezielle Weise definiert sind. Sie sind die Lösungen eines nicht-singulären linearen Gleichungssystems, dessen Koeffizientenmatrix durch Streichen der letzten Zeile von $\widehat{\mathbf{H}}_n$ entsteht. Deshalb ist es naheliegend, die Lösung einer Folge von Minimierungsproblemen der Form (5.33) mit allgemeiner oberer Hessenberg–Matrix ebenfalls unter der Zuhilfenahme dieser Vektoren $\widetilde{\mathbf{z}}_n$ zu konstruieren. Das folgende Lemma zeigt, daß diese Vorgehensweise prinzipiell zum Ziel führt.

Lemma 5.5 (1-Norm-Minimierungsproblem mit Hessenberg–Struktur). *Für $n \geq 1$ sei $\widehat{\mathbf{H}}_n$ eine Familie von $(n+1) \times n$ oberen Hessenberg–Matrizen der Form*

$$\widehat{\mathbf{H}}_n = \begin{bmatrix} & \mathbf{H}_n & \\ \mathbf{0}_{n-1}^T & & \varepsilon_{n+1} \end{bmatrix},$$

wobei für die unteren Nebendiagonalelemente $\varepsilon_2, \varepsilon_3, \ldots, \varepsilon_n \neq 0$ gilt. Dabei ist $\mathbf{H}_n$ diejenige $n \times n$ Matrix, die durch Streichen der letzten Zeile von $\widehat{\mathbf{H}}_n$ entsteht. Es sei $\varepsilon_1 > 0$, und $\mathbf{z}_n \in \mathbb{C}^n$ bezeichne eine Lösung des 1-Norm-Minimierungsproblems

$$\tau_n := \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n\|_1 = \min_{\mathbf{z} \in \mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}\|_1, \tag{5.34}$$

wobei $\mathbf{f}_{n+1} = \varepsilon_1 \mathbf{e}_1^{(n+1)} \in \mathbb{R}^{n+1}$ und τ_n die Größe des Quasi-Residuums bezeichnet. Außerdem sei $\mathbf{H}_n$ nicht singulär und $\widetilde{\mathbf{z}}_n := \mathbf{H}_n^{-1} \mathbf{f}_n$ mit letzter Komponente $\nu_n := \widetilde{\mathbf{z}}_n^T \mathbf{e}_n^{(n)}$ gesetzt.

Dann steht eine (spezielle) Lösung $\mathbf{z}_n \in \mathbb{C}^n$ von (5.34) *mit dem Hilfsvektor $\widetilde{\mathbf{z}}_n$ in dem Zusammenhang*

$$\mathbf{z}_n = \begin{cases} \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix}, & \textit{falls } |\varepsilon_{n+1}\nu_n| \geq \tau_{n-1}, \\ \widetilde{\mathbf{z}}_n, & \textit{falls } |\varepsilon_{n+1}\nu_n| < \tau_{n-1}, \end{cases} \tag{5.35}$$

und die Größe des Quasi-Residuums genügt der Rekursion

$$\tau_n = \min\{\tau_{n-1}, |\varepsilon_{n+1}\nu_n|\} \qquad \textit{mit} \qquad \tau_0 = \varepsilon_1. \tag{5.36}$$

Beweis. Der entscheidende Punkt in den Beweisen der Lemmata 5.3 und 5.4, in denen Bidiagonal- und Tridiagonalstrukturen behandelt werden, ist der Übergang zu der Basis des $\mathbb{C}^n$ der Form (5.23). Eine genaue Analyse der Beweise zeigt, daß die Struktur der Matrix $\widehat{\mathbf{B}}_n$ in Lemma 5.3 lediglich in Gleichung (5.24) eingeht. Diese Gleichung gilt jedoch ebenfalls für allgemeine obere Hessenberg–Matrizen $\widehat{\mathbf{H}}_n$. Der restliche Beweis ist dann identisch mit den Beweisen der Lemmata 5.3 und 5.4. □

Dieses Lemma gibt eine Lösung $\mathbf{z}_n$ des Minimierungsproblems (5.34) in Abhängigkeit der Hilfsvektoren $\widetilde{\mathbf{z}}_n$ an. Es läßt allerdings offen, wie man die Folge der Hilfsvektoren $\widetilde{\mathbf{z}}_n$ effizient berechnet.

Im folgenden Kapitel werden nicht nur die Lösungen von speziellen Minimierungsproblemen in unterschiedlichen p-Normen benötigt. Darüber hinaus wird im Fall $p = 1$ ein weiterer Aspekt beleuchtet, der im folgenden Korollar vorbereitet wird.

Korollar 5.1. *Unter den Voraussetzungen von Lemma 5.5 sei $1 \leq p \leq \infty$ und $\mathbf{g}_n \in \mathbb{C}^N$ ein Vektor, der mit einer $N \times (n+1)$ Matrix*

$$\mathbf{V}_{n+1} = [\mathbf{v}_1 \; \mathbf{v}_2 \; \cdots \; \mathbf{v}_{n+1}]$$

durch

$$\mathbf{g}_n = \mathbf{V}_{n+1}(\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n) \tag{5.37}$$

definiert ist. Falls die Spaltenvektoren von $\mathbf{V}_{n+1}$ *zu*

$$\|\mathbf{v}_i\|_p = 1, \qquad i = 1, 2, \ldots, n+1,$$

normiert sind, gilt

$$\|\mathbf{g}_n\|_p = \tau_n.$$

Beweis. Das nachfolgend benutzte Symbol $*_n$ bezeichne einen Vektor aus $\mathbb{C}^n$ mit beliebigen Einträgen. Durch Einsetzen von (5.35) in (5.37) und Verwendung der Struktur von $\widehat{\mathbf{H}}_n$ folgt

$$\begin{aligned}
\mathbf{g}_n &= \begin{cases} \mathbf{V}_{n+1}\left(\mathbf{f}_{n+1} - \begin{bmatrix} \widehat{\mathbf{H}}_{n-1} & *_n \\ \mathbf{0}_{n-1}^T & \varepsilon_{n+1} \end{bmatrix} \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix}\right), & \text{falls } |\varepsilon_{n+1}\nu_n| \geq \tau_{n-1}, \\ \mathbf{V}_{n+1}\left(\mathbf{f}_{n+1} - \begin{bmatrix} \mathbf{H}_n \\ \mathbf{0}_{n-1}^T \quad \varepsilon_{n+1} \end{bmatrix} \widetilde{\mathbf{z}}_n\right), & \text{falls } |\varepsilon_{n+1}\nu_n| < \tau_{n-1}, \end{cases} \\
&= \begin{cases} \mathbf{V}_n(\mathbf{f}_n - \widehat{\mathbf{H}}_{n-1}\mathbf{z}_{n-1}), & \text{falls } |\varepsilon_{n+1}\nu_n| \geq \tau_{n-1}, \\ \mathbf{V}_{n+1}\left(\mathbf{f}_{n+1} - \begin{bmatrix} \mathbf{f}_n \\ \varepsilon_{n+1}\nu_n \end{bmatrix}\right), & \text{falls } |\varepsilon_{n+1}\nu_n| < \tau_{n-1}, \end{cases} \\
&= \begin{cases} \mathbf{g}_{n-1}, & \text{falls } |\varepsilon_{n+1}\nu_n| \geq \tau_{n-1}, \\ -\varepsilon_{n+1}\nu_n \mathbf{v}_{n+1}, & \text{falls } |\varepsilon_{n+1}\nu_n| < \tau_{n-1}. \end{cases}
\end{aligned}$$

Wegen $\tau_0 = \varepsilon_1$ and $\mathbf{g}_0 = \varepsilon_1 \mathbf{v}_1$ folgt das Korollar induktiv durch Bilden der p-Normen auf beiden Seiten der letzten Gleichung. □

5.4 Minimierung in der *p*-Norm für $1 < p \leq \infty$

Das Minimierungsproblem (5.1) wird in diesem Abschnitt für die Fälle $1 < p \leq \infty$ untersucht. Der Spezialfall $p = 2$ wird bereits in Abschnitt 5.2 betrachtet und wird daher hier nicht gesondert behandelt. Außerdem wird die obere Hessenberg–Matrix $\widehat{\mathbf{H}}_n$ aus (5.1) auf die spezielle Struktur einer unteren Bidiagonalmatrix $\widehat{\mathbf{B}}_n$ eingeschränkt. Dieser Abschnitt befaßt sich also mit dem p-Norm-Minimierungsproblem

$$\min_{\mathbf{z} \in \mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}\|_p, \qquad 1 < p \leq \infty. \tag{5.38}$$

In [64, Korollar 3.5.1] wird nachgewiesen, daß eine Folge von p-Norm-Minimierungsproblemen der Form (5.38) mit Hilfe von Rekursionen von geringer Tiefe wie folgt gelöst werden kann.

Lemma 5.6 (*p*-Norm-Minimierungsproblem mit Bidiagonalstruktur). *Für $1 < p \leq \infty$ und $n \geq 1$ sei $\widehat{\mathbf{B}}_n$ eine Familie von $(n+1) \times n$ unteren Bidiagonalmatrizen der Form*

$$\widehat{\mathbf{B}}_n = \begin{bmatrix} \delta_1 & & & \mathbf{0} \\ \varepsilon_2 & \delta_2 & & \\ & \ddots & \ddots & \\ & & \varepsilon_n & \delta_n \\ \mathbf{0} & & & \varepsilon_{n+1} \end{bmatrix},$$

wobei $\varepsilon_2, \varepsilon_3, \ldots, \varepsilon_n \neq 0$ und $\delta_1, \delta_2, \ldots, \delta_n \neq 0$. Es sei $\varepsilon_1 > 0$, und $\mathbf{z}_n \in \mathbb{C}^n$ bezeichne die eindeutige Lösung des p-Norm-Minimierungsproblems

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}_n\|_p = \min_{\mathbf{z} \in \mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}\|_p,$$

wobei $\mathbf{f}_{n+1} = \varepsilon_1 \mathbf{e}_1^{(n+1)} \in \mathbb{R}^{n+1}$. Dann gelten die Rekursionen

$$\mathbf{z}_n = \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix} + \mathbf{g}_n, \tag{5.39}$$

$$\mathbf{g}_n = \sigma_n \begin{bmatrix} \mathbf{g}_{n-1} \\ 0 \end{bmatrix} + \kappa_n \mathbf{e}_n^{(n)}, \tag{5.40}$$

wobei die Anfangsbedingungen durch $\mathbf{z}_1 = \mathbf{g}_1 = \kappa_1$ und die Koeffizienten durch

$$\sigma_n = \frac{1 - \lambda_n}{\lambda_n + \vartheta_n}, \qquad n \geq 1, \tag{5.41}$$

$$\kappa_n = \begin{cases} -1, & \text{falls } n = 0, \\ -\dfrac{\varepsilon_n}{\delta_n} \cdot \dfrac{\kappa_{n-1}}{\lambda_n + \vartheta_n}, & \text{falls } n \geq 1, \end{cases} \tag{5.42}$$

mit

$$\lambda_n = \begin{cases} 1, & \text{falls } n = 1, \\ \dfrac{\lambda_{n-1}}{\lambda_{n-1} + \vartheta_{n-1}}, & \text{falls } n \geq 2, \end{cases} \tag{5.43}$$

$$\vartheta_n = \left| \frac{\varepsilon_{n+1}}{\delta_n} \right|^q, \qquad n \geq 1, \tag{5.44}$$

wobei

$$q := \begin{cases} \dfrac{p}{p-1}, & \text{falls } 1 < p < \infty, \\ 1, & \text{falls } p = \infty, \end{cases} \tag{5.45}$$

gegeben sind.

Das folgende Korollar weist nach, daß ein spezielles gewichtetes euklidisches Minimierungsproblem die gleiche Lösung wie das p-Norm-Minimierungsproblem des vorangehenden Lemmas besitzt.

Korollar 5.2 (Simulation des *p*-Norm-Minimierungsproblems). *Für $1 < p \leq \infty$ und $n \geq 1$ sei $\widehat{\mathbf{B}}_n$ eine Familie von $(n+1) \times n$ unteren Bidiagonalmatrizen der Form*

$$\widehat{\mathbf{B}}_n = \begin{bmatrix} \delta_1 & & \mathbf{0} \\ \varepsilon_2 & \delta_2 & \\ & \ddots & \ddots \\ & & \varepsilon_n & \delta_n \\ \mathbf{0} & & & \varepsilon_{n+1} \end{bmatrix},$$

wobei $\varepsilon_2, \varepsilon_3, \ldots, \varepsilon_n \neq 0$ und $\delta_1, \delta_2, \ldots, \delta_n \neq 0$. Außerdem sei

$$\Omega_{n+1} := \operatorname{diag}(\omega_1, \omega_2, \ldots, \omega_{n+1})$$

eine $(n+1) \times (n+1)$ Skalierungsmatrix mit Gewichten

$$\omega_{j+1} = \begin{cases} \omega_j \left| \dfrac{\varepsilon_{j+1}}{\delta_j} \right|^{(2-p)/(2p-2)}, & \text{falls } 1 < p < \infty, \\ \omega_j \left| \dfrac{\varepsilon_{j+1}}{\delta_j} \right|^{-1/2}, & \text{falls } p = \infty, \end{cases} \qquad j = 1, 2, \ldots, n, \qquad \omega_1 = 1. \quad (5.46)$$

Es sei $\varepsilon_1 > 0$, und $\mathbf{z}_n \in \mathbb{C}^n$ bezeichne die eindeutige Lösung des gewichteten euklidischen Minimierungsproblems

$$\|\omega_1 \mathbf{f}_{n+1} - \Omega_{n+1} \widehat{\mathbf{B}}_n \mathbf{z}_n\|_2 = \min_{\mathbf{z} \in \mathbb{C}^n} \|\omega_1 \mathbf{f}_{n+1} - \Omega_{n+1} \widehat{\mathbf{B}}_n \mathbf{z}\|_2,$$

wobei $\mathbf{f}_{n+1} = \varepsilon_1 \mathbf{e}_1^{(n+1)} \in \mathbb{R}^{n+1}$. Dann gilt

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}_n\|_p = \min_{\mathbf{z} \in \mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}\|_p.$$

Beweis. Für $p = 2$ liefert die Anwendung von Lemma 5.6 auf das euklidische Minimierungsproblem

$$\min_{\mathbf{z} \in \mathbb{C}^n} \|\omega_1 \mathbf{f}_{n+1} - \Omega_{n+1} \widehat{\mathbf{B}}_n \mathbf{z}\|_2$$

eine Darstellung der Lösung $\mathbf{z}_n$ in der Form der Rekursionen (5.39)–(5.40), deren Koeffizienten σ_n und κ_n von den Einträgen der Matrix $\Omega_{n+1} \widehat{\mathbf{B}}_n$ abhängen. Speziell gilt

$$\vartheta_n = \left| \frac{\omega_{n+1} \varepsilon_{n+1}}{\omega_n \delta_n} \right|^2, \qquad n \geq 1.$$

Setzt man darin die Gewichte (5.46) für $1 < p < \infty$ ein, so gilt

$$\vartheta_n = \left| \frac{\varepsilon_{n+1}}{\delta_n} \right|^{2(2-p)/(2p-2)+2} = \left| \frac{\varepsilon_{n+1}}{\delta_n} \right|^{p/(p-1)}.$$

Da dieses Ergebnis mit (5.44) übereinstimmt und außerdem eine analoge Rechnung für κ_n wegen $\omega_1 = 1$ die Gleichung (5.42) liefert, genügen die Lösungen $\mathbf{z}_n$ des gewichteten euklidischen Minimierungsproblems den gleichen Rekursionen wie die Lösungen des p-Norm-Minimierungsproblem aus Lemma 5.6. Der Fall $p = \infty$ folgt analog. □

Nach diesem Korollar kann man bei Vorgabe eines beliebigen $1 < p \leq \infty$ ein gemäß (5.46) speziell gewichtetes euklidisches Minimierungsproblem benutzen, um das entsprechende ungewichtete p-Norm-Minimierungsproblem zu "simulieren". Umgekehrt zeigt das folgende Korollar die Simulation eines euklidischen Minimierungsproblems durch ein gewichtetes p-Norm-Minimierungsproblem.

Korollar 5.3 (Simulation des euklidischen Minimierungsproblems). *Für $1 < p \leq \infty$ und $n \geq 1$ sei $\widehat{\mathbf{B}}_n$ eine Familie von $(n+1) \times n$ unteren Bidiagonalmatrizen der Form*

$$\widehat{\mathbf{B}}_n = \begin{bmatrix} \delta_1 & & & \\ \varepsilon_2 & \delta_2 & & \mathbf{0} \\ & \ddots & \ddots & \\ & & \varepsilon_n & \delta_n \\ \mathbf{0} & & & \varepsilon_{n+1} \end{bmatrix},$$

wobei $\varepsilon_2, \varepsilon_3, \ldots, \varepsilon_n \neq 0$ und $\delta_1, \delta_2, \ldots, \delta_n \neq 0$. Außerdem sei

$$\Omega_{n+1} := \operatorname{diag}(\omega_1, \omega_2, \ldots, \omega_{n+1})$$

eine $(n+1) \times (n+1)$ Skalierungsmatrix mit Gewichten

$$\omega_{j+1} = \begin{cases} \omega_j \left| \dfrac{\varepsilon_{j+1}}{\delta_j} \right|^{(p-2)/p}, & \text{falls } 1 < p < \infty, \\ \omega_j \left| \dfrac{\varepsilon_{j+1}}{\delta_j} \right|, & \text{falls } p = \infty, \end{cases} \qquad j = 1, 2, \ldots, n, \qquad \omega_1 = 1.$$

Es sei $\varepsilon_1 > 0$, und $\mathbf{z}_n \in \mathbb{C}^n$ bezeichne die eindeutige Lösung des gewichteten p-Norm-Minimierungsproblems

$$\|\omega_1 \mathbf{f}_{n+1} - \Omega_{n+1} \widehat{\mathbf{B}}_n \mathbf{z}_n\|_p = \min_{\mathbf{z} \in \mathbb{C}^n} \|\omega_1 \mathbf{f}_{n+1} - \Omega_{n+1} \widehat{\mathbf{B}}_n \mathbf{z}\|_p,$$

wobei $\mathbf{f}_{n+1} = \varepsilon_1 \mathbf{e}_1^{(n+1)} \in \mathbb{R}^{n+1}$. Dann gilt

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}_n\|_2 = \min_{\mathbf{z} \in \mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{B}}_n \mathbf{z}\|_2.$$

Beweis. Der Beweis wird analog zu dem Beweis von Korollar 5.2 durch Rückführung auf Lemma 5.6 gezeigt. □

Kapitel 6

Entwurf von Krylov–Teilraumverfahren

Konzeptionell besteht der Entwurf von Krylov–Teilraumverfahren aus einer geeigneten Konstruktion einer Basis der zugrundeliegenden Vektorräume und der anschließenden Festlegung eines speziellen Vektors aus diesen aufgespannten Räumen, der die aktuelle Approximation an die exakte Lösung des Gleichungssystems darstellt. Während das Aufspannen der Vektorräume in den Kapiteln 2–4 behandelt wird, dient dieses Kapitel der Festlegung der aktuellen Iterierten. Damit wird der Entwurf neuer Krylov–Teilraumverfahren auf konzeptioneller Ebene abgeschlossen. Es bleibt dann noch zu zeigen, wie tatsächliche Implementierungen der neudefinierten Verfahren hergeleitet werden können. Dazu bedient sich dieses Kapitel der Techniken, die im vorangehenden Kapitel bereitgestellt werden. Auf diese Art und Weise werden eine Reihe von neuen iterativen Verfahren zur Lösung von linearen Gleichungssystemen vorgestellt. Dabei werden sowohl neue parallele Varianten bekannter Iterationsverfahren als auch neue (sequentielle) iterative Methoden eingeführt. Von Teilen dieser neuen sequentiellen Methoden werden außerdem parallele Varianten hergeleitet.

Der Aufbau dieses Kapitels ist an den des Kapitels 2, in dem die Erzeugung von Krylov–Teilräumen behandelt wird, angelehnt. Nach einer einführenden Darstellung der grundsätzlichen Vorgehensweise zur Festlegung der Iterierten werden die in Kapitel 2 vorgestellten Methoden in der gleichen Reihenfolge als zugrundeliegende Technik zur Herleitung von Krylov–Teilraumverfahren verwendet. Zu Referenzzwecken werden bekannte Verfahren kurz erwähnt. Der Schwerpunkt liegt jedoch in der Herleitung neuer iterativer Methoden. Nacheinander werden Verfahren betrachtet, die zum Aufspannen der Teilräume den Lanczos–Algorithmus, CGS und Bi-CGSTAB benutzen.

6.1 Festlegung der aktuellen Iterierten

Neben dem Aufspannen der Krylov–Teilräume besteht die zweite Aufgabe beim Entwurf von Krylov–Teilraumverfahren darin, einen speziellen Vektor $\mathbf{x}_n$ als aktuelle Iterierte zu definieren. Diese Iterierte ist von der Form

$$\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K}_n\left(\mathbf{A}, \mathbf{r}_0\right). \tag{6.1}$$

Unter der Annahme, daß ein geeignetes Verfahren zur Konstruktion von Basisvektoren $\mathbf{v}_i \in \mathbb{C}^N$ mit

$$\operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} = \mathcal{K}_n\left(\mathbf{A}, \mathbf{r}_0\right) \tag{6.2}$$

zur Verfügung steht, kann man (6.1) mit der $N \times n$ Matrix

$$\mathbf{V}_n := [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_n]$$

in der Form

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n \mathbf{z} \qquad \text{für ein} \qquad \mathbf{z} \in \mathbb{C}^n \tag{6.3}$$

angeben. Darin bezeichnet $\mathbf{z} \in \mathbb{C}^n$ einen freien Parametervektor, der zur Festlegung der Iterierten $\mathbf{x}_n$ dient. Unterschiedliche Wahlen von $\mathbf{z}$ führen zu unterschiedlichen iterativen Methoden, denen dann allerdings der gleiche Prozeß zum Aufspannen der Krylov–Teilräume zugrundeliegt. Um die in dieser Arbeit verwendeten unterschiedlichen Wahlen von $\mathbf{z}$ besser einordnen zu können, ist eine kurze Betrachtung von iterativen Methoden hinsichtlich der folgenden Klassifikation sinnvoll.

Krylov–Teilraumverfahren zur Lösung von linearen Gleichungssystemen mit allgemeiner nichthermitescher Koeffizientenmatrix lassen sich nach dem Faber–Manteuffel–Theorem [21] in zwei Klassen einteilen. Vereinfacht ausgedrückt zeigt dieses Theorem, daß es keine CG-ähnlichen Verfahren für allgemeine nicht-hermitesche Systeme gibt, die über die zugrundeliegenden Krylov–Teilräume in einer Norm minimieren und gleichzeitig mit kurzen Rekursionen implementiert werden können. Dabei versteht man unter dem Begriff *kurze Rekursionen* solche Verfahren zur Lösung von Systemen der Ordnung N, deren Speicher- und Rechenaufwand pro Iteration neben den benötigten Matrix-Vektor-Produkten $\Theta(N)$ beträgt. Im Gegensatz dazu bezeichnet man Verfahren, deren n-te Iteration zusätzlich zu den Matrix-Vektor-Produkten einen Speicher- und Rechenbedarf von $\Theta(nN)$ besitzt, als Verfahren mit *langen Rekursionen*. Ein bekanntes Beispiel für ein Verfahren mit langen Rekursionen ist GMRES, dessen Speicher- und Rechenbedarf linear mit der Anzahl der Iterationen ansteigt. GMRES minimiert in jedem Iterationsschritt die euklidische Norm des Residuums. Diese wünschenswerte Minimierungseigenschaft wird nach dem Faber–Manteuffel–Theorem aber mit langen Rekursionen bezahlt, die in der Praxis einen Neustart des Verfahrens nach einer vorgegebenen maximalen Iterationsanzahl erfordert. Typischerweise wird dabei der letzte Vektor einer GMRES–Sequenz als Startvektor der nächsten GMRES–Sequenz gewählt. Unglücklicherweise geht dadurch die Information, die in den bereits berechneten Krylov–Teilräumen enthalten ist, verloren, so daß das Konvergenzverhalten negativ beeinflußt wird. Verfahren mit langen Rekursionen werden in der vorliegenden Arbeit nicht weiter betrachtet. Stattdessen werden ausschließlich Verfahren mit kurzen Rekursionen hergeleitet, deren Speicher- und Rechenaufwand in jeder Iteration konstant ist. Nach dem Faber–Manteuffel–Theorem basieren solche Verfahren zwar auf suboptimalen Minimierungseigenschaften; sie sind aber aufgrund ihrer kurzen Rekursionen in der Praxis vorteilhaft einsetzbar.

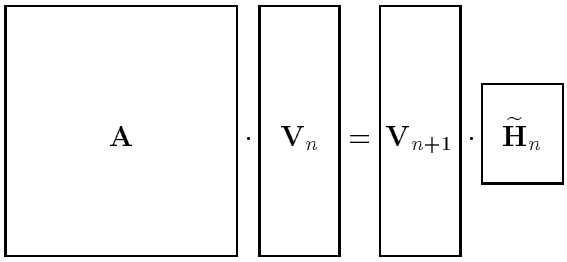
Die Festlegung der aktuellen Iterierten $\mathbf{x}_n$ eines Krylov–Teilraumverfahrens erfolgt durch eine spezielle Wahl des freien Parametervektors $\mathbf{z}$ in (6.3). Diese Wahl von $\mathbf{z}$ ist nun so vorzunehmen, daß die resultierenden Verfahren mit kurzen Rekursionen implementiert werden können. Das Ziel dabei ist, das zugehörige Residuum

$$\mathbf{r}_n = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_n \mathbf{z} \qquad \text{für ein} \qquad \mathbf{z} \in \mathbb{C}^n \tag{6.4}$$

gegen den Nullvektor zu treiben. Da die hier betrachteten iterativen Methoden Projektionen auf den Teilraum $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ darstellen und die Spaltenvektoren der Matrix $\mathbf{V}_n$ nach (6.2) eine Basis dieses Raumes bilden, genügen alle Verfahren zur Erzeugung von Krylov–Teilräumen einer Beziehung der Form

$$\mathbf{A}\mathbf{V}_n = \mathbf{V}_{n+1}\widetilde{\mathbf{H}}_n, \tag{6.5}$$

wobei $\widetilde{\mathbf{H}}_n$ eine $(n+1)\times n$ obere Hessenberg–Matrix bezeichnet. Diese Gleichungen sind in Kapitel 2 für unterschiedliche Verfahren zur Erzeugung von Krylov–Teilräumen in den entsprechenden Sätzen gesammelt. Die Wirkung der Matrix $\mathbf{A}$ auf $\mathbf{V}_n$ läßt sich schematisch durch



wiedergeben und zeigt die Elimination der "großen" Matrix $\mathbf{A}$. Mit der Gleichung (6.5) kann deshalb in (6.4) die Koeffizientenmatrix $\mathbf{A}$ aus der Darstellung des Residuums eliminiert werden

$$\mathbf{r}_n = \mathbf{V}_{n+1}(\rho_0\mathbf{e}_1^{(n+1)} - \widetilde{\mathbf{H}}_n\mathbf{z}) \qquad \text{für ein} \qquad \mathbf{z}\in\mathbb{C}^n, \tag{6.6}$$

wobei für den ersten Basisvektor die Beziehung

$$\mathbf{v}_1 = \mathbf{r}_0/\rho_0 \tag{6.7}$$

gelten muß. Die "optimale" Wahl von $\mathbf{z}$ in (6.6) würde das Residuum $\mathbf{r}_n$ in einer Norm minimieren. Nach dem oben erwähnten Faber–Manteuffel–Theorem führt eine solche Wahl von $\mathbf{z}$ in den resultierenden Verfahren, die für allgemeine Koeffizientenmatrizen anwendbar sind, zu langen Rekursionen. Dagegen bieten sogenannte quasi-minimale Residuenansätze, die lediglich den zweiten Faktor in der Darstellung (6.6) des Residuums minimieren, den Vorteil von kurzen Rekursionen.

Gewöhnlich werden solche Ansätze unter Verwendung einer reellen Skalierungsmatrix

$$\Omega_{n+1} = \operatorname{diag}(\omega_1,\omega_2,\ldots,\omega_{n+1}), \qquad \omega_k > 0, \qquad k = 1,2,\ldots,n+1, \tag{6.8}$$

formuliert. Damit ergibt sich (6.6) zu

$$\mathbf{r}_n = \mathbf{V}_{n+1}\Omega_{n+1}^{-1}(\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}) \qquad \text{für ein} \qquad \mathbf{z}\in\mathbb{C}^n \tag{6.9}$$

mit

$$\mathbf{f}_{n+1} = \omega_1\rho_0\mathbf{e}_1^{(n+1)} \qquad \text{und} \qquad \widehat{\mathbf{H}}_n = \Omega_{n+1}\widetilde{\mathbf{H}}_n.$$

Die quasi-minimalen Residuenansätze benutzen die Darstellung des Residuums in der Form (6.9). Anstelle der Wahl von $\mathbf{z}$ in (6.9), so daß $\mathbf{r}_n$ in einer Norm minimiert wird, erfolgt die Bestimmung von $\mathbf{z}\in\mathbb{C}^n$ durch das p-Norm-Minimierungsproblem

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}_n\|_p = \min_{\mathbf{z}\in\mathbb{C}^n}\|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}\|_p, \tag{6.10}$$

welches lediglich den zweiten Faktor in der Darstellung (6.9) des Residuums berücksichtigt. Mit der Lösung $\mathbf{z}_n$ dieses Minimierungsproblems werden dann nach (6.3) die Iterierten zu

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n\mathbf{z}_n$$

definiert. Der nach dem Minimierungsprozeß entstandene Vektor

$$\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}_n$$

wird als *Quasi-Residuum* bezeichnet. Seine Komponenten stellen die Koeffizienten des Residuums $\mathbf{r}_n$ bezüglich der Basisvektoren $\mathbf{v}_i$ dar. Der Ansatz der quasi-minimalen Residuen ist ursprünglich von Freund [25, 28, 35, 36] für den Fall $p = 2$ eingeführt worden. Dabei ist zu beachten, daß dieser Ansatz noch von der Wahl der Skalierungsmatrix Ω_{n+1} abhängt. Die Standardwahl im Fall $p = 2$ besteht darin,

$$\omega_k = \|\mathbf{v}_k\|_2, \qquad k = 1, 2, \ldots, n+1, \tag{6.11}$$

zu setzen, so daß alle Spalten der Matrix $\mathbf{V}_{n+1}\Omega_{n+1}^{-1}$ gleich behandelt werden und in der euklidischen Norm auf Einheitslänge skaliert sind. Mitunter sind allerdings andere Gewichte ω_k erforderlich [37]. In [27] wird unter der Voraussetzung (6.11) die Beziehung

$$\|\mathbf{r}_n\|_2 \leq \sqrt{n+1}\, \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n\|_2 \tag{6.12}$$

hergeleitet, mit Hilfe derer man Konvergenzbeweise führt [26]. Diese obere Schranke für die Norm des Residuums ist ein mächtiges Werkzeug, das bei der Implementierung eines Abbruchkriteriums benutzt werden sollte. Die euklidische Norm der Quasi-Residuen $\|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n\|_2$ ist im Verlaufe des iterativen Prozesses bei den ursprünglichen Verfahren mit $p = 2$ ohne zusätzliche Kosten verfügbar.

Um Verfahren mit kurzen Rekursionen herzuleiten, müssen die Minimierungsprobleme der Form (6.10) effizient berechnet werden. Dazu stellt das vorangehende Kapitel Techniken für unterschiedliche Werte von p zur Verfügung. Hinsichtlich der Konvergenzgeschwindigkeit der resultierenden Verfahren stellt sich unmittelbar die Frage nach dem "besten" p. Diese Frage wird in Abschnitt 5.4 des vorangehenden Kapitels für den speziellen Fall einer Bidiagonalmatrix teilweise beantwortet. Für Bidiagonalmatrizen lassen sich mit Hilfe von Lemma 5.6 Verfahren mit kurzen Rekursionen herleiten. Beispielsweise wird dieses Lemma in [64] auf den Lanczos–Algorithmus mit gekoppelten Zwei-Term-Rekursionen angewendet. Durch die Minimierung der Quasi-Residuen in der p-Norm entsteht eine Klasse von iterativen Verfahren mit Parameter $1 \leq p \leq \infty$. Die Korollare 5.2 und 5.3 zeigen nun, daß sich die Fälle $p \neq 1$ lediglich in der Normierung der Basisvektoren $\mathbf{v}_i$ unterscheiden. Durch spezielle gewichtete Minimierungen der Quasi-Residuen in der euklidischen Norm lassen sich ungewichtete Minimierungen der Quasi-Residuen in allen p-Normen darstellen. Umgekehrt kann eine speziell gewichtete Minimierung der Quasi-Residuen in der p-Norm durch eine ungewichtete euklidische Minimierung der Quasi-Residuen simuliert werden. In diesem Sinne sind die Fälle $p \neq 1$ bereits in dem ursprünglichen Fall $p = 2$ enthalten. Aus diesem Grunde wird in der restlichen Arbeit die Klasse der Fälle $1 < p \leq \infty$ nicht näher betrachtet. Stattdessen stehen die beiden Fälle $p = 1$ und $p = 2$ im Vordergrund. Für den Fall $p = 1$ wird die Verfügbarkeit eines kostengünstigen Abbruchkriteriums in der Form

$$\|\mathbf{r}_n\|_2 = \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n\|_1$$

gezeigt. Da die Norm der Quasi-Residuen $\|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}_n\|_1$ während des iterativen Prozesses ohne zusätzliche Kosten bereitsteht, wird das bereits mächtige Werkzeug aus (6.12) noch weiter verbessert.

Zusammenfassend gilt, daß die grundlegende Vorgehensweise bei der Festlegung einer Iterierten in der folgenden Weise stattfindet. Den Ausgangspunkt bildet eine Darstellung des Residuums in der Form von (6.9) mit freiem Parametervektor $\mathbf{z} \in \mathbb{C}^n$. Dabei führen unterschiedliche Prozesse zum Aufspannen der Krylov–Teilräume, die in den folgenden Abschnitten jeweils separat betrachtet werden, zu unterschiedlichen Matrizen $\mathbf{V}_{n+1}$ und $\widehat{\mathbf{H}}_n$. Die zweite Möglichkeit, auf die der Entwickler eines Krylov–Teilraumverfahrens Einfluß besitzt, bezieht sich auf die Wahl des freien Parametervektors $\mathbf{z}$ und erfolgt jeweils innerhalb der einzelnen Abschnitte.

6.2 Lanczos–basierte Verfahren

Die enge Verwandtschaft der Methoden CGS und Bi-CGSTAB mit dem Lanczos–Algorithmus ist in Kapitel 2 erwähnt. Da nach Abschnitt 2.5.1 die Residualpolynome von CGS und Bi-CGSTAB unter Verwendung des Residualpolynoms von BCG definiert sind, basieren sowohl CGS als auch Bi-CGSTAB auf BCG. In dieser Arbeit wird jedoch der Begriff "Lanczos–basiert" in einem strengeren Sinne verstanden. Danach sind Lanczos–basierte Verfahren nur solche Verfahren, die tatsächlich den Lanczos–Algorithmus zur Erzeugung ihrer Krylov–Teilräume verwenden. In diesem Abschnitt werden Lanczos–basierte Verfahren eingeführt, die mit Drei-Term-Rekursionen und gekoppelten Zwei-Term-Rekursionen implementiert sind. Die nächsten beiden Abschnitte befassen sich anschließend mit CGS–basierten und Bi-CGSTAB–basierten Verfahren.

6.2.1 Drei-Term-Rekursionen

Die Implementierung des Lanczos–Algorithmus mit Drei-Term-Rekursionen ist in Kapitel 2 in dem Teilabschnitt 2.3.2 beschrieben. Satz 2.1 faßt die Eigenschaften dieser Implementierung zusammen. Danach bilden die Spaltenvektoren der Matrix $\mathbf{V}_n$ eine Basis des $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$, falls der Lanczos–Algorithmus mit einem ersten Basisvektor $\mathbf{v}_1$ gemäß (6.7) gestartet wird. Außerdem gilt

$$\mathbf{A}\mathbf{V}_n = \mathbf{V}_{n+1}\widetilde{\mathbf{T}}_n \tag{6.13}$$

mit $\widetilde{\mathbf{T}}_n = \begin{bmatrix} \mathbf{T}_n \\ \rho_{n+1}\mathbf{e}_n^T \end{bmatrix}$ und der tridiagonalen Matrix $\mathbf{T}_n$ aus (2.9). Daher gilt als Ausgangspunkt für diesen Teilabschnitt die Beziehung

$$\mathbf{r}_n = \mathbf{V}_{n+1}\Omega_{n+1}^{-1}(\mathbf{f}_{n+1} - \widehat{\mathbf{T}}_n\mathbf{z}) \qquad \text{für ein} \qquad \mathbf{z} \in \mathbb{C}^n$$

mit

$$\mathbf{f}_{n+1} = \omega_1\rho_0\mathbf{e}_1^{(n+1)} \qquad \text{und} \qquad \widehat{\mathbf{T}}_n = \Omega_{n+1}\widetilde{\mathbf{T}}_n.$$

Euklidische Minimierung der Quasi-Residuen

In [35] wird der Lanczos–Algorithmus mit Drei-Term-Rekursionen als zugrundeliegender Prozeß bei der Methode der quasi-minimalen Residuen (**Q**uasi-**M**inimal **R**esidual, QMR) verwendet. Die QMR–Iterierten sind durch

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n\mathbf{z}_n \tag{6.14}$$

definiert, wobei $\mathbf{z}_n$ durch die eindeutige Lösung des euklidischen Minimierungsproblems

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{T}}_n\mathbf{z}_n\|_2 = \min_{\mathbf{z}\in\mathbb{C}^n}\|\mathbf{f}_{n+1} - \widehat{\mathbf{T}}_n\mathbf{z}\|_2$$

bestimmt ist. Dieses Minimierungsproblem wird in [35] durch die QR-Zerlegung von $\widehat{\mathbf{T}}_n$ mit Hilfe von Givens–Rotationen gelöst. Dies entspricht einer Anwendung von Lemma 5.1 und führt in QMR, das Alg. 2.1 als zugrundeliegenden Prozeß benutzt, zu kurzen Rekursionen.

In der gleichen Weise läßt sich mit Lemma 5.1 und Alg. 4.2 eine parallele QMR–Variante mit einem globalen Synchronisationspunkt pro Iterationsschritt herleiten. In Kapitel 4 wird gezeigt, daß sich die ursprüngliche Version des Lanczos–Prozesses aus Alg. 2.1 von der parallelen Variante aus Alg. 4.2 nur durch eine Restrukturierung einzelner Anweisungen unterscheidet. Deshalb kann auch eine parallele Variante von QMR durch Restrukturierung aus dem ursprünglichen QMR hergeleitet werden. Aus diesem Grunde werden diese beiden QMR–Implementierungen nicht betrachtet.

1-Norm-Minimierung der Quasi-Residuen

Anstatt den Vektor $\mathbf{z}_n$ in (6.14) durch die euklidische Minimierung der Quasi-Residuen festzulegen, wird in diesem Teilabschnitt eine andere Vorgehensweise durchgeführt. Die Iterierten der neuen Methode sind weiterhin von der Form (6.14). Dabei wird der Vektor $\mathbf{z}_n$ nun aber als eine Lösung des 1-Norm-Minimierungsproblems

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{T}}_n \mathbf{z}_n\|_1 = \min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{T}}_n \mathbf{z}\|_1 \tag{6.15}$$

bestimmt. Im vorangehenden Kapitel wird erwähnt, daß eine solche Lösung nicht notwendig eindeutig ist. Da die Matrix $\widehat{\mathbf{T}}_n$ Tridiagonalstruktur besitzt, ist Lemma 5.4 zur Lösung dieses Problems anwendbar. Dieses Lemma konstruiert eine spezielle Lösung von (6.15) in effizienter Weise. Die resultierende Methode der **1**-Norm **Q**uasi-**M**inimalen **R**esiduen (QMR_1) mit Drei-Term-Rekursionen besitzt kurze Rekursionen.

Die Herleitung von QMR_1 benutzt eine Folge von Hilfsiterierten, die durch

$$\widetilde{\mathbf{x}}_n = \mathbf{x}_0 + \mathbf{V}_n \widetilde{\mathbf{z}}_n \qquad \text{mit} \qquad \widetilde{\mathbf{z}}_n = (\Omega_n \mathbf{T}_n)^{-1} \mathbf{f}_n \tag{6.16}$$

definiert sind. Eine Anwendung von Lemma 5.4 auf das spezielle Minimierungsproblem (6.15) liefert wegen (5.27) eine Rekursion für die Hilfsiterierten in der Form

$$\begin{aligned} \widetilde{\mathbf{x}}_n &= \mathbf{x}_0 + \mathbf{V}_n \begin{bmatrix} \widetilde{\mathbf{z}}_{n-1} \\ 0 \end{bmatrix} - \omega_{n-1}\gamma_n\nu_n \mathbf{V}_n \begin{bmatrix} \widetilde{\mathbf{y}}_{n-1} \\ 0 \end{bmatrix} + \nu_n \mathbf{V}_n \mathbf{e}_n^{(n)} \\ &= \widetilde{\mathbf{x}}_{n-1} - \omega_{n-1}\gamma_n\nu_n \mathbf{y}_{n-1} + \nu_n \mathbf{v}_n, \end{aligned} \tag{6.17}$$

wobei $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$ und $\mathbf{y}_{n-1} := \mathbf{V}_{n-1}\widetilde{\mathbf{y}}_{n-1}$ eingeführt wird. Durch Multiplikation mit $\mathbf{V}_n$ folgt aus (5.28) eine rekursive Darstellung für $\mathbf{y}_n$ in der Form

$$\mathbf{y}_n = -\omega_{n-1}\gamma_n\kappa_n \mathbf{y}_{n-1} + \kappa_n \mathbf{v}_n, \tag{6.18}$$

mit $\mathbf{y}_0 = \mathbf{0}$. Die Koeffizienten ν_n und κ_n in (6.17) und (6.18) folgen aus (5.29) und (5.30) in der Form

$$\nu_n = -\omega_n \rho_n \kappa_n \nu_{n-1}, \qquad n \geq 1, \qquad \nu_0 = -1, \tag{6.19}$$

$$\kappa_n = \frac{1}{\omega_n \alpha_n - \omega_n \omega_{n-1} \rho_n \gamma_n \kappa_{n-1}}, \qquad n \geq 1, \qquad \kappa_0 = 0, \tag{6.20}$$

wobei $\rho_1 := \rho_0$. Benutzt man die Definition der Hilfsiterierten aus (6.16), dann folgt aus (6.14) durch Einsetzen von (5.31) eine Rekursion für die QMR_1–Iterierten

$$\mathbf{x}_n = \begin{cases} \mathbf{x}_0 + \mathbf{V}_n \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix} = \mathbf{x}_{n-1}, & \text{falls } \omega_{n+1}|\rho_{n+1}\nu_n| \geq \tau_{n-1}, \\ \mathbf{x}_0 + \mathbf{V}_n \widetilde{\mathbf{z}}_n = \widetilde{\mathbf{x}}_n, & \text{falls } \omega_{n+1}|\rho_{n+1}\nu_n| < \tau_{n-1}. \end{cases} \tag{6.21}$$

Darin ist nach (5.32) die Größe des Quasi-Residuums durch

$$\tau_n := \|\mathbf{f}_n - \widehat{\mathbf{T}}_n \mathbf{z}_n\|_1 = \min\{\tau_{n-1}, \omega_{n+1}|\rho_{n+1}\nu_n|\} \tag{6.22}$$

mit $\tau_0 = \omega_1 \rho_0$ gegeben. Eine Implementierung von QMR_1 mit Drei-Term-Rekursionen folgt unter Verwendung von Alg. 2.1 zusammen mit den Gleichungen (6.17)–(6.22).

ALGORITHMUS 6.1 (QMR$_1$ MIT DREI-TERM-REKURSIONEN)

Wähle $\mathbf{x}_0 \in \mathbb{C}^N$ und setze $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\rho_0 = \rho_1 = \|\mathbf{r}_0\|_2$, $\mathbf{v}_1 = \mathbf{w}_1 = \mathbf{r}_0/\rho_0$
Setze $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{y}_0 = \mathbf{0}$, $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$ und $\gamma_1 = 0$, $\xi_1 \neq 0$, $\delta_1 = \mathbf{w}_1^T\mathbf{v}_1 \neq 0$
Wähle ω_1 (zum Beispiel nach (6.24)) und setze $\kappa_0 = 0$, $\nu_0 = -1$, $\tau_0 = \omega_1\rho_0$, $\omega_0 = 0$
for $n = 1, 2, 3, \ldots$ **do**

$$\alpha_n = \mathbf{w}_n^T\mathbf{A}\mathbf{v}_n/\delta_n$$
$$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{v}_n - \alpha_n\mathbf{v}_n - \gamma_n\mathbf{v}_{n-1}$$
$$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T\mathbf{w}_n - \alpha_n\mathbf{w}_n - \tfrac{\gamma_n\rho_n}{\xi_n}\mathbf{w}_{n-1}$$
$$\kappa_n = 1/(\omega_n\alpha_n - \omega_n\omega_{n-1}\rho_n\gamma_n\kappa_{n-1})$$
$$\nu_n = -\omega_n\rho_n\kappa_n\nu_{n-1}$$
$$\widetilde{\mathbf{x}}_n = \widetilde{\mathbf{x}}_{n-1} - \omega_{n-1}\gamma_n\nu_n\mathbf{y}_{n-1} + \nu_n\mathbf{v}_n$$
$$\mathbf{y}_n = -\omega_{n-1}\gamma_n\kappa_n\mathbf{y}_{n-1} + \kappa_n\mathbf{v}_n$$
$$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$$
$$\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$$

Wähle ω_{n+1} (zum Beispiel nach (6.24))

$$\mathbf{x}_n = \begin{cases} \mathbf{x}_{n-1} & \textbf{if } \omega_{n+1}|\rho_{n+1}\nu_n| \geq \tau_{n-1} \\ \widetilde{\mathbf{x}}_n & \textbf{if } \omega_{n+1}|\rho_{n+1}\nu_n| < \tau_{n-1} \end{cases}$$

$$\tau_n = \min\{\tau_{n-1}, \omega_{n+1}|\rho_{n+1}\nu_n|\}$$

stop, falls $\mathbf{x}_n$ konvergiert hat

$$\mathbf{v}_{n+1} = \tfrac{1}{\rho_{n+1}}\widetilde{\mathbf{v}}_{n+1}$$
$$\mathbf{w}_{n+1} = \tfrac{1}{\xi_{n+1}}\widetilde{\mathbf{w}}_{n+1}$$
$$\delta_{n+1} = \mathbf{w}_{n+1}^T\mathbf{v}_{n+1}$$
$$\gamma_{n+1} = \xi_{n+1}\delta_{n+1}/\delta_n$$

enddo

In diesem Algorithmus sind die Gewichte ω_k frei wählbar. Ein kostengünstiges Abbruchkriterium für den iterativen Prozeß erhält man durch Anwendung von Korollar 5.1, nach dem die Spalten der Matrix $\mathbf{V}_{n+1}\Omega_{n+1}^{-1}$ in einer beliebigen p-Norm auf Einheitslänge zu normieren sind. Dabei ist $1 \leq p \leq \infty$ ein vom Benutzer gewählter Parameter, der eine Angabe der QMR$_1$–Residuen in jeder gewählten p-Norm liefert. Die angesprochene Normierung der Spalten von $\mathbf{V}_{n+1}\Omega_{n+1}^{-1}$ erreicht man beispielsweise durch die Modifikation des zugrundeliegenden Lanczos–Prozesses in der Form

$$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_p \qquad \text{und} \qquad \xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_p. \tag{6.23}$$

Daraus resultiert eine Normierung der Lanczos–Vektoren, und damit der Spalten von $\mathbf{V}_{n+1}$, auf p-Norm-Einheitslänge, so daß die Gewichte zu

$$\omega_k = 1, \qquad k = 1, 2, \ldots, n+1, \tag{6.24}$$

gewählt werden können. Der folgenden Satz beschreibt diese Situation.

Satz 6.1 (QMR$_1$ mit Drei-Term-Rekursionen). *Falls man die Modifikationen* (6.23) *im zugrundeliegenden Lanczos–Algorithmus vornimmt und die Gewichte nach* (6.24) *wählt, fällt die Norm der QMR$_1$–Residuen (schwach) monoton, und es gilt*

$$\|\mathbf{r}_n\|_p = \tau_n, \qquad n \geq 1, \tag{6.25}$$

wobei $1 \leq p \leq \infty$ *und* τ_n *die Größe der Quasi-Residuen nach* (6.22) *bezeichnet.*

Beweis. Der Satz ist eine unmittelbare Folgerung aus Korollar 5.1. Die Monotonie ist durch (5.22) gegeben. □

Eine parallele Variante von QMR$_1$ mit Drei-Term-Rekursionen erhält man durch Austauschen des zugrundeliegenden Lanczos–Algorithmus, indem man nicht die ursprüngliche Form aus Alg. 2.1 sondern die parallele Variante aus Alg. 4.2 verwendet.

6.2.2 Gekoppelte Zwei-Term-Rekursionen

Der Lanczos–Algorithmus besteht in seiner klassischen Form aus Drei-Term-Rekursionen. Eine andere Implementierung, die in Kapitel 2 in dem Teilabschnitt 2.3.3 beschrieben ist, basiert auf gekoppelten Zwei-Term-Rekursionen. Die Eigenschaften dieser Implementierung sind in Satz 2.2 zusammengefaßt. Danach bilden die durch Alg. 2.2 generierten Spaltenvektoren der Matrix $\mathbf{P}_n$ eine Basis des $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$, falls der Prozeß mit $\mathbf{p}_1 = \mathbf{v}_1 = \mathbf{r}_0/\rho_0$ gestartet wird, und es gilt

$$\mathbf{A}\mathbf{P}_n = \mathbf{V}_{n+1}\widetilde{\mathbf{L}}_n \tag{6.26}$$

mit $\widetilde{\mathbf{L}}_n = \begin{bmatrix} \mathbf{L}_n \\ \rho_{n+1}\mathbf{e}_n^T \end{bmatrix}$ und der bidiagonalen Matrix $\mathbf{L}_n$ aus (2.23). Deshalb ist der Ausgangspunkt dieses Teilabschnittes die Beziehung

$$\mathbf{r}_n = \mathbf{V}_{n+1}\Omega_{n+1}^{-1}(\mathbf{f}_{n+1} - \widehat{\mathbf{L}}_n\mathbf{z}) \qquad \text{für ein} \qquad \mathbf{z} \in \mathbb{C}^n$$

mit

$$\mathbf{f}_{n+1} = \omega_1\rho_0\mathbf{e}_1^{(n+1)} \qquad \text{und} \qquad \widehat{\mathbf{L}}_n = \Omega_{n+1}\widetilde{\mathbf{L}}_n.$$

Euklidische Minimierung der Quasi-Residuen

Die Methode der quasi-minimalen Residuen (**Q**uasi-**M**inimal **R**esidual, QMR) [36] definiert die Iterierten

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{P}_n\mathbf{z}_n \tag{6.27}$$

durch die Minimierung der Quasi-Residuen in der euklidischen Norm. Dabei ist $\mathbf{z}_n$ die eindeutige Lösung des Minimierungsproblems

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{L}}_n\mathbf{z}_n\|_2 = \min_{\mathbf{z}\in\mathbb{C}^n}\|\mathbf{f}_{n+1} - \widehat{\mathbf{L}}_n\mathbf{z}\|_2.$$

Da $\widehat{\mathbf{L}}_n$ eine bidiagonale Matrix ist, kann zur Lösung dieses Minimierungsproblems neben Lemma 5.1, das für allgemeine obere Hessenberg–Matrizen anwendbar ist, auch Lemma 5.2 benutzt werden. In der Originalarbeit [36] wird die in Lemma 5.1 enthaltene Technik der QR-Zerlegung von $\widehat{\mathbf{L}}_n$ verwendet. Eine neue Implementierung, die Lemma 5.2 anwendet, ist in [65] skizziert. Diese beiden Implementierungen, die zum Aufspannen der Teilräume jeweils den Lanczos–Algorithmus mit gekoppelten Zwei-Term-Rekursionen in seiner ursprünglichen Form aus Alg. 2.2

benutzen, werden hier nicht betrachtet. Stattdessen wird eine Implementierung hergeleitet, der die parallele Variante aus Alg. 4.5 zugrundeliegt. Die Eigenschaften dieser Variante des Lanczos–Algorithmus sind in Satz 4.1 zusammengestellt. Nach diesem Satz gilt weiterhin die Tatsache, daß die Spalten von $\mathbf{P}_n$ eine Basis des $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ bilden und daß $\mathbf{P}_n$ die Gleichung (6.26) erfüllt. Daher kann für Alg. 4.5 die gleiche Diskussion wie für Alg. 2.2 geführt werden. Die Anwendung von Lemma 5.1 führt zu einer neuen parallelen QMR–Variante, die die bekannte Technik der QR-Zerlegung von $\widehat{\mathbf{L}}_n$ zur Minimierung der Quasi-Residuen benutzt. Die Herleitung dieser neuen Variante ist offensichtlich und wird hier nicht betrachtet. Stattdessen wird eine neue parallele Variante von QMR unter Verwendung von Lemma 5.2 hergeleitet.

Setzt man in (6.27) die Rekursion (5.9) für den Vektor $\mathbf{z}_n$ ein, so folgt für die QMR–Iterierten

$$\begin{aligned} \mathbf{x}_n &= \mathbf{x}_0 + \mathbf{P}_{n-1}\mathbf{z}_{n-1} + \mathbf{P}_n\mathbf{g}_n \\ &= \mathbf{x}_{n-1} + \mathbf{d}_n, \end{aligned} \tag{6.28}$$

wobei $\mathbf{d}_n := \mathbf{P}_n\mathbf{g}_n$ eingeführt wird. Aus (5.10) erhält man durch Multiplikation mit $\mathbf{P}_n$ eine Rekursion für diesen Vektor $\mathbf{d}_n$ in der Form

$$\mathbf{d}_n = \theta_n \mathbf{d}_{n-1} + \kappa_n \mathbf{p}_n, \tag{6.29}$$

wobei $\mathbf{d}_0 = \mathbf{0}$ und die Koeffizienten θ_n und κ_n aus den Rekursionen (5.12)–(5.14) folgen

$$\theta_n = \frac{\omega_n^2 |\beta_n|^2 (1-\lambda_n)}{\lambda_n \omega_n^2 |\beta_n|^2 + \omega_{n+1}^2 |\rho_{n+1}|^2}, \qquad n \geq 1, \tag{6.30}$$

$$\kappa_n = \frac{-\omega_n^2 \rho_n \overline{\beta}_n \kappa_{n-1}}{\lambda_n \omega_n^2 |\beta_n|^2 + \omega_{n+1}^2 |\rho_{n+1}|^2}, \qquad n \geq 1, \qquad \kappa_0 = -1, \tag{6.31}$$

$$\lambda_{n+1} = \frac{\lambda_n \omega_n^2 |\beta_n|^2}{\lambda_n \omega_n^2 |\beta_n|^2 + \omega_{n+1}^2 |\rho_{n+1}|^2}, \qquad n \geq 1, \qquad \lambda_1 = 1, \tag{6.32}$$

mit $\rho_1 := \rho_0$.

Abschließend wird die Wahl der Gewichte ω_k diskutiert. Um die Spalten der Matrix $\mathbf{V}_{n+1}\Omega_{n+1}^{-1}$ in der euklidischen Norm auf Einheitslänge zu normieren, bietet sich die Wahl

$$\omega_k = 1, \qquad k = 1, 2, \ldots, n+1, \tag{6.33}$$

an, da der zugrundeliegende Lanczos–Algorithmus bereits die Spalten von $\mathbf{V}_{n+1}$ entsprechend normiert. Mit dieser Wahl der Gewichte erhält man nach (6.12) unmittelbar eine obere Schranke für die euklidische Norm des Residuums. Darin ist noch die Größe der Quasi-Residuen zu bestimmen. Eine Anwendung von Lemma 5.2 führt nach (5.11) zu der Beziehung

$$\tau_n = \sqrt{\left| \frac{\tau_{n-1} \omega_{n+1}^2 |\rho_{n+1}|^2}{\lambda_n \omega_n^2 |\beta_n|^2 + \omega_{n+1}^2 |\rho_{n+1}|^2} \right|^2 + \omega_{n+1}^2 |\kappa_n \rho_{n+1}|^2} \tag{6.34}$$

mit $\tau_0 = \omega_1 \rho_0$, die im Verlaufe des Iterationsverfahrens mühelos aufzudatieren ist. Der resultierende Prozeß folgt dann, indem man Alg. 4.5 mit den Gleichungen (6.28)–(6.34) zusammenfügt.

ALGORITHMUS 6.2 (QMR MIT ALG. 4.5 UND LEMMA 5.2)

Wähle $\mathbf{x}_0 \in \mathbb{C}^N$ und setze $\widetilde{\mathbf{v}}_1 = \widetilde{\mathbf{w}}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 = \widetilde{\mathbf{q}}_0 = \mathbf{d}_0 = \mathbf{0}$
Setze $\widetilde{\delta}_1 = \widetilde{\mathbf{w}}_1^T \widetilde{\mathbf{v}}_1$, $\rho_0 = \rho_1 = \|\widetilde{\mathbf{v}}_1\|_2$, $\xi_1 = \|\widetilde{\mathbf{w}}_1\|_2$, $\sigma_1 = \widetilde{\mathbf{w}}_1^T \mathbf{A} \widetilde{\mathbf{v}}_1$, $\varepsilon_0 \neq 0$
Wähle ω_1 (zum Beispiel nach (6.33)) und setze $\lambda_1 = 1$, $\kappa_0 = -1$, $\tau_0 = \omega_1 \rho_0$
for $n = 1, 2, 3, \ldots$ **do**

$\mathbf{v}_n = \frac{1}{\rho_n} \widetilde{\mathbf{v}}_n$
$\mathbf{w}_n = \frac{1}{\xi_n} \widetilde{\mathbf{w}}_n$
$\mathbf{p}_n = \mathbf{v}_n - \frac{\widetilde{\delta}_n}{\varepsilon_{n-1} \rho_n} \mathbf{p}_{n-1}$
$\widetilde{\mathbf{q}}_n = \frac{1}{\xi_n} \mathbf{A}^T \widetilde{\mathbf{w}}_n - \frac{\widetilde{\delta}_n}{\varepsilon_{n-1} \xi_n} \widetilde{\mathbf{q}}_{n-1}$
$\beta_n = \begin{cases} \sigma_1 / \widetilde{\delta}_1 & \textbf{if } n = 1 \\ \sigma_n / \widetilde{\delta}_n - \widetilde{\delta}_n / \varepsilon_{n-1} & \textbf{if } n \geq 2 \end{cases}$
$\varepsilon_n = \beta_n \widetilde{\delta}_n / (\rho_n \xi_n)$
$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \beta_n \mathbf{v}_n$
$\widetilde{\mathbf{w}}_{n+1} = \widetilde{\mathbf{q}}_n - \beta_n \mathbf{w}_n$

- $\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$
- $\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$
- $\widetilde{\delta}_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}$
- $\sigma_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T \mathbf{A} \widetilde{\mathbf{v}}_{n+1}$

Wähle ω_{n+1} (zum Beispiel nach (6.33))
$\theta_n = \omega_n^2 |\beta_n|^2 (1 - \lambda_n) / \left(\lambda_n \omega_n^2 |\beta_n|^2 + \omega_{n+1}^2 |\rho_{n+1}|^2 \right)$
$\kappa_n = -\omega_n^2 \rho_n \overline{\beta}_n \, \kappa_{n-1} / \left(\lambda_n \omega_n^2 |\beta_n|^2 + \omega_{n+1}^2 |\rho_{n+1}|^2 \right)$
$\lambda_{n+1} = \lambda_n \omega_n^2 |\beta_n|^2 / \left(\lambda_n \omega_n^2 |\beta_n|^2 + \omega_{n+1}^2 |\rho_{n+1}|^2 \right)$
$\tau_n = \sqrt{|\tau_{n-1} \omega_{n+1}^2 |\rho_{n+1}|^2 / (\lambda_n \omega_n^2 |\beta_n|^2 + \omega_{n+1}^2 |\rho_{n+1}|^2)|^2 + \omega_{n+1}^2 |\kappa_n \rho_{n+1}|^2}$
$\mathbf{d}_n = \theta_n \mathbf{d}_{n-1} + \kappa_n \mathbf{p}_n$
$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{d}_n$
stop, falls $\mathbf{x}_n$ konvergiert hat

enddo

Will man die Wahl der Gewichte nicht nach (6.33) vornehmen und damit von der Standardvorgehensweise der Normierung der Spalten der Matrix $\mathbf{V}_{n+1} \Omega_{n+1}^{-1}$ auf Einheitslänge in der euklidischen Norm abweichen, so kann man als Abbruchkriterium die obere Schranke aus (6.12) nicht unmittelbar anwenden. In diesem Falle empfiehlt sich die Herleitung einer modifizierten Schranke, die weiterhin die Größe des Quasi-Residuums enthält, so daß (6.34) anwendbar bleibt.

1-Norm-Minimierung der Quasi-Residuen

Anstelle der Wahl des freien Parametervektors $\mathbf{z}$ durch die Minimierung der Quasi-Residuen in der euklidischen Norm, wird hier eine andere Wahl von $\mathbf{z}$ getroffen, um eine neue Methode zu definieren. Dazu wird der Vektor $\mathbf{z}_n$ in der Darstellung der Iterierten (6.27) durch eine Lösung des

1-Norm-Minimierungsproblems

$$\|\mathbf{f}_{n+1} - \widehat{\mathbf{L}}_n \mathbf{z}_n\|_1 = \min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{L}}_n \mathbf{z}\|_1 \tag{6.35}$$

definiert. Eine Lösung dieses Problems ist nicht notwendig eindeutig. In Lemma 5.3 wird jedoch gezeigt, wie eine spezielle Lösung effizient angegeben werden kann. Die resultierende Methode wird QMR_1 mit gekoppelten Zwei-Term-Rekursionen genannt.

Die Herleitung dieser Methode greift auf eine Folge von Hilfsiterierten

$$\widetilde{\mathbf{x}}_n = \mathbf{x}_0 + \mathbf{P}_n \widetilde{\mathbf{z}}_n \qquad \text{mit} \qquad \widetilde{\mathbf{z}}_n = (\Omega_n \mathbf{L}_n)^{-1} \mathbf{f}_n \tag{6.36}$$

zurück. Aufgrund der bidiagonalen Struktur von $\mathbf{L}_n$ ist Lemma 5.3 zur Lösung des Problems (6.35) anwendbar. Nach (5.19) genügen die Hilfsiterierten der Rekursion

$$\widetilde{\mathbf{x}}_n = \mathbf{x}_0 + \mathbf{P}_n \begin{bmatrix} \widetilde{\mathbf{z}}_{n-1} \\ \nu_n \end{bmatrix} = \widetilde{\mathbf{x}}_{n-1} + \nu_n \mathbf{p}_n, \tag{6.37}$$

wobei $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$. Darin ist ν_n die letzte Komponente des Vektors $\widetilde{\mathbf{z}}_n = (\Omega_n \mathbf{L}_n)^{-1} \mathbf{f}_n$. Wegen (6.8) und der Tatsache, daß die Matrix $\mathbf{L}_n$ aus (2.23) vollen Rang besitzt, ist $\Omega_n \mathbf{L}_n$ nicht singulär. Eine Anwendung von Lemma (5.3), wobei in (5.20) die skalaren Größen

$$\delta_n = \omega_n \beta_n \qquad \text{und} \qquad \varepsilon_{n+1} = \omega_{n+1} \rho_{n+1} \qquad n \geq 1,$$

eingesetzt werden, liefert

$$\nu_n = \begin{cases} -1, & \text{falls } n = 0, \\ -\dfrac{\rho_n}{\beta_n} \nu_{n-1}, & \text{falls } n \geq 1, \end{cases} \tag{6.38}$$

mit $\rho_1 := \rho_0$. Daraus folgt die Beziehung

$$|\varepsilon_{n+1} \nu_n| = \omega_{n+1} |\rho_{n+1} \nu_n|,$$

die in Lemma 5.3 (ii) bei der Fallunterscheidung in (5.21) benötigt wird. Danach folgt für die QMR_1–Iterierten aus (6.27)

$$\mathbf{x}_n = \begin{cases} \mathbf{x}_0 + \mathbf{P}_n \begin{bmatrix} \mathbf{z}_{n-1} \\ 0 \end{bmatrix} = \mathbf{x}_{n-1}, & \text{falls } \omega_{n+1} |\rho_{n+1} \nu_n| \geq \tau_{n-1}, \\ \mathbf{x}_0 + \mathbf{P}_n \widetilde{\mathbf{z}}_n = \widetilde{\mathbf{x}}_n, & \text{falls } \omega_{n+1} |\rho_{n+1} \nu_n| < \tau_{n-1}, \end{cases} \tag{6.39}$$

wobei (6.36) verwendet wird. Außerdem ist darin nach (5.22) die Größe des Quasi-Residuums durch

$$\tau_n := \|\mathbf{f}_n - \widehat{\mathbf{L}}_n \mathbf{z}_n\|_1 = \min\{\tau_{n-1}, \omega_{n+1} |\rho_{n+1} \nu_n|\} \tag{6.40}$$

mit $\tau_0 = \omega_1 \rho_0$ gegeben. Der resultierende Prozeß folgt durch Hinzufügen von Gleichungen (6.37)–(6.40) zu Alg. 2.2.

ALGORITHMUS 6.3 (QMR_1 MIT GEKOPPELTEN ZWEI-TERM-REKURSIONEN)

Wähle $\mathbf{x}_0 \in \mathbb{C}^N$ und setze $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\rho_0 = \rho_1 = \|\mathbf{r}_0\|_2$, $\mathbf{v}_1 = \mathbf{w}_1 = \mathbf{r}_0/\rho_0$
Setze $\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{0}$, $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$ und $\xi_1 = 0$, $\varepsilon_0 \neq 0$, $\nu_0 = -1$, $\tau_0 = \omega_1\rho_0$
for $n = 1, 2, 3, \ldots$ **do**
$\delta_n = \mathbf{w}_n^T\mathbf{v}_n$
$\mathbf{p}_n = \mathbf{v}_n - \frac{\xi_n\delta_n}{\varepsilon_{n-1}}\mathbf{p}_{n-1}$
$\mathbf{q}_n = \mathbf{w}_n - \frac{\rho_n\delta_n}{\varepsilon_{n-1}}\mathbf{q}_{n-1}$
$\varepsilon_n = \mathbf{q}_n^T\mathbf{A}\mathbf{p}_n$
$\beta_n = \varepsilon_n/\delta_n$
$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \beta_n\mathbf{v}_n$
$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T\mathbf{q}_n - \beta_n\mathbf{w}_n$
$\nu_n = -\rho_n\nu_{n-1}/\beta_n$
$\widetilde{\mathbf{x}}_n = \widetilde{\mathbf{x}}_{n-1} + \nu_n\mathbf{p}_n$
$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_2$
$\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_2$
Wähle ω_{n+1} (zum Beispiel nach (6.33))
$$\mathbf{x}_n = \begin{cases} \mathbf{x}_{n-1} & \textbf{if } \omega_{n+1}|\rho_{n+1}\nu_n| \geq \tau_{n-1} \\ \widetilde{\mathbf{x}}_n & \textbf{if } \omega_{n+1}|\rho_{n+1}\nu_n| < \tau_{n-1} \end{cases}$$
$\tau_n = \min\{\tau_{n-1}, \omega_{n+1}|\rho_{n+1}\nu_n|\}$
stop, falls $\mathbf{x}_n$ konvergiert hat
$\mathbf{v}_{n+1} = \frac{1}{\rho_{n+1}}\widetilde{\mathbf{v}}_{n+1}$
$\mathbf{w}_{n+1} = \frac{1}{\xi_{n+1}}\widetilde{\mathbf{w}}_{n+1}$
enddo

Ein kostengünstiges Abbruchkriterium gewinnt man aus Korollar 5.1. Danach sind die Spalten von $\mathbf{V}_{n+1}\Omega_{n+1}^{-1}$ in der p-Norm auf Einheitslänge zu normieren. Dies erreicht man beispielsweise durch die Wahl der Gewichte (6.33) und die Modifikation des zugrundeliegenden Lanczos–Algorithmus in der Form

$$\rho_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|_p \qquad \text{und} \qquad \xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|_p. \tag{6.41}$$

Damit gilt der folgende Satz.

Satz 6.2 (QMR_1 mit gekoppelten Zwei-Term-Rekursionen). *Falls man die Modifikationen* (6.41) *im zugrundeliegenden Lanczos–Algorithmus vornimmt und die Gewichte nach* (6.33) *wählt, fällt die Norm der QMR_1–Residuen (schwach) monoton, und es gilt*

$$\|\mathbf{r}_n\|_p = \tau_n, \qquad n \geq 1, \tag{6.42}$$

wobei $1 \leq p \leq \infty$ und τ_n die Größe der Quasi-Residuen nach (6.40) *bezeichnet.*

Beweis. Der Satz folgt unmittelbar aus Korollar 5.1. □

Mit der gleichen Vorgehensweise erhält man unter Verwendung der parallelen Variante des Lanczos–Algorithmus, die in Alg. 4.5 notiert ist, eine parallele Variante von QMR_1 mit gekoppelten Zwei-Term-Rekursionen. Da die Herleitung analog zu der hier vorgestellten ist, wird auf diese parallele Variante nicht näher eingegangen.

Die Beziehung zwischen QMR und BCG ist bereits in [35, Sect. 5] enthalten. Daraus folgt [11, 17, 29], daß die Folge der Hilfsiterierten mit den BCG–Iterierten übereinstimmen

$$\widetilde{\mathbf{x}}_n = \mathbf{x}_n^{\text{BCG}}.$$

Damit entspricht QMR_1 mit gekoppelten Zwei-Term-Rekursionen der Vorgehensweise, jeweils die beste BCG–Iterierte als neue Approximation auszuwählen.

Eine parallele Variante von BCG mit nur einem globalen Synchronisationspunkt in jedem Iterationsschritt wird in [11, Alg. 1] hergeleitet. Diese BCG–Variante verwendet den Lanczos–Algorithmus aus [10, Alg. 3], der im Vergleich zu Alg. 4.5 gleiche Parallelisierungseigenschaften aber in endlicher Arithmetik ein unterschiedliches numerisches Verhalten aufweist. Eine analoge Vorgehensweise wie in [11] erlaubt die Herleitung einer auf Alg. 4.5 aufbauenden parallelen BCG–Variante, auf die im Rahmen dieser Arbeit jedoch verzichtet wird. In den in nachfolgenden Teilen dieser Arbeit durchgeführten numerischen Experimenten werden dagegen die auf der parallelen Variante des Lanczos–Algorithmus [10, Alg. 3] aufbauenden Versionen von BCG [11] und QMR [9] zu Vergleichszwecken herangezogen.

6.3 CGS–basierte Verfahren

Während im letzten Abschnitt neue Verfahren auf der Basis des Lanczos–Algorithmus hergeleitet werden, befaßt sich dieser Abschnitt mit CGS als zugrundeliegendem Verfahren zur Erzeugung der Krylov–Teilräume. Nach Satz 2.3 bilden die von CGS erzeugten Spaltenvektoren der Matrix $\mathbf{Y}_m$ eine Basis des $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$ und es gilt

$$\mathbf{A}\mathbf{Y}_m = \mathbf{W}_{m+1}\widetilde{\mathbf{B}}_m$$

mit $\widetilde{\mathbf{B}}_m$ aus (2.47). Daher ist der Ausgangspunkt in diesem Abschnitt die Darstellung

$$\mathbf{r}_m = \mathbf{W}_{m+1}\Omega_{m+1}^{-1}(\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m\mathbf{z}) \qquad \text{für ein} \qquad \mathbf{z} \in \mathbb{C}^m \tag{6.43}$$

mit

$$\mathbf{f}_{m+1} = \omega_1\mathbf{e}_1^{(m+1)} \qquad \text{und} \qquad \widehat{\mathbf{B}}_m = \Omega_{m+1}\widetilde{\mathbf{B}}_m. \tag{6.44}$$

Man beachte, daß man zur Herleitung von Gleichung (6.43) die Initialisierung $\mathbf{w}_1 = \mathbf{r}_0$ benötigt, die durch (2.48) gewährleistet wird.

6.3.1 Euklidische Minimierung der Quasi-Residuen

Das Verfahren der transpositionsfreien quasi-minimalen Residuen (**T**ranspose-**F**ree **Q**uasi-**M**inimal **R**esidual, TFQMR) [28] bestimmt den freien Parametervektor $\mathbf{z}$ in (6.43) über die Minimierung des Quasi-Residuums in der euklidischen Norm. Die TFQMR–Iterierten sind durch

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{Y}_m\mathbf{z}_m \tag{6.45}$$

definiert, wobei $\mathbf{z}_m \in \mathbb{C}^m$ die eindeutige Lösung des euklidischen Minimierungsproblems

$$\|\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m\mathbf{z}_m\|_2 = \min_{\mathbf{z}\in\mathbb{C}^m}\|\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m\mathbf{z}\|_2 \tag{6.46}$$

bezeichnet. Benutzt man die Technik aus Lemma 5.1, erhält man die in [28, Alg. 5.1] beschriebene Implementierung von TFQMR mit den Gewichten

$$\omega_k = \|\mathbf{w}_k\|_2, \qquad k = 1, 2, \ldots, m+1, \tag{6.47}$$

die hier nicht näher betrachtet wird. An dieser Stelle wird eine neue Implementierung von TFQMR vorgestellt, die auf Lemma 5.2 beruht.

Da die Matrix $\widehat{\mathbf{B}}_m$ aus (6.43) nach (6.44), (6.8) und (2.47) bidiagonale Struktur besitzt, ist Lemma 5.2 anwendbar. Setzt man in (6.45) die Rekursion (5.9) ein, so folgt

$$\begin{aligned} \mathbf{x}_m &= \mathbf{x}_0 + \mathbf{Y}_{m-1}\mathbf{z}_{m-1} + \mathbf{Y}_m\mathbf{g}_m \\ &= \mathbf{x}_{m-1} + \mathbf{d}_m, \end{aligned} \tag{6.48}$$

wobei $\mathbf{d}_m := \mathbf{Y}_m\mathbf{g}_m$ eingeführt wird. Mit Hilfe der Rekursion (5.10) erhält man eine Vorschrift für diesen Vektor $\mathbf{d}_m$ in der Form

$$\mathbf{d}_m = \theta_m\mathbf{d}_{m-1} + \kappa_m\mathbf{y}_m, \tag{6.49}$$

wobei $\mathbf{d}_0 = \mathbf{0}$ und die Koeffizienten θ_m und κ_m aus den Rekursionen (5.12)–(5.14) folgen

$$\theta_m = \frac{\omega_m^2\,(1-\lambda_m)}{\lambda_m\omega_m^2 + \omega_{m+1}^2}, \qquad m \geq 1, \tag{6.50}$$

$$\kappa_m = \frac{\omega_m^2\,\kappa_{m-1}}{\lambda_m\omega_m^2 + \omega_{m+1}^2} \cdot \frac{\alpha_{\lfloor (m-1)/2 \rfloor}}{\alpha_{\lfloor (m-2)/2 \rfloor}}, \qquad m \geq 1, \qquad \kappa_0 = \alpha_{-1}, \tag{6.51}$$

$$\lambda_{m+1} = \frac{\lambda_m\omega_m^2}{\lambda_m\omega_m^2 + \omega_{m+1}^2}, \qquad m \geq 1, \qquad \lambda_1 = 1. \tag{6.52}$$

Eine gute Unterstützung für die Entscheidung, ob das Verfahren im Sinne einer vorgegebenen Schranke konvergiert oder nicht, liefert die obere Schranke aus (6.12). Darin ist die Größe der aktuellen Quasi-Residuen durch Anwendung von (5.11) ohne zusätzliche Kosten verfügbar. Für TFQMR folgt die Rekursion für die Größe der Quasi-Residuen in der Form

$$\tau_m = \sqrt{\left|\frac{\tau_{m-1}\omega_{m+1}^2}{\lambda_m\omega_m^2 + \omega_{m+1}^2}\right|^2 + \left|\frac{\kappa_m\omega_{m+1}}{\alpha_{\lfloor (m-1)/2 \rfloor}}\right|^2} \tag{6.53}$$

mit $\tau_0 = \omega_1$. Benutzt man als Abbruchkriterium die Schranke aus (6.12) in Verbindung mit (6.53), muß man die Wahl der Gewichte gemäß (6.47) durchführen. Bei einer anderen Wahl der Gewichte muß die Schranke entsprechend angepaßt werden. Der resultierende Prozeß folgt dann mit Hilfe von Alg. 2.4 und den Gleichungen (6.48)–(6.53).

ALGORITHMUS 6.4 (TFQMR MIT ALG. 2.4 UND LEMMA 5.2)

Wähle $\mathbf{x}_0 \in \mathbb{C}^N$ und setze $\mathbf{w}_1 = \mathbf{y}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v}_0 = \mathbf{A}\mathbf{y}_1$, $\mathbf{d}_0 = \mathbf{0}$
Wähle $\tilde{\mathbf{r}}_0 \in \mathbb{C}^N$ so, daß $\rho_0 = \tilde{\mathbf{r}}_0^T \mathbf{r}_0 \neq 0$
Wähle ω_1 (zum Beispiel nach (6.47)) und setze $\lambda_1 = 1$, $\kappa_0 = \alpha_{-1}$, $\tau_0 = \omega_1$
for $n = 1, 2, 3, \ldots$ **do**
$\sigma_{n-1} = \tilde{\mathbf{r}}_0^T \mathbf{v}_{n-1}$
$\alpha_{n-1} = \rho_{n-1} / \sigma_{n-1}$
$\mathbf{y}_{2n} = \mathbf{y}_{2n-1} - \alpha_{n-1} \mathbf{v}_{n-1}$
for $m = 2n-1, 2n$ **do**
$\mathbf{w}_{m+1} = \mathbf{w}_m - \alpha_{n-1} \mathbf{A}\mathbf{y}_m$
Wähle ω_{m+1} (zum Beispiel nach (6.47))
$\theta_m = \omega_m^2 \, (1 - \lambda_m) / (\lambda_m \omega_m^2 + \omega_{m+1}^2)$
$\kappa_m = \omega_m^2 \, \kappa_{m-1} \alpha_{n-1} / \left(\alpha_{\lfloor (m-2)/2 \rfloor} \cdot (\lambda_m \omega_m^2 + \omega_{m+1}^2) \right)$
$\lambda_{m+1} = \lambda_m \omega_m^2 / (\lambda_m \omega_m^2 + \omega_{m+1}^2)$
$\tau_m = \sqrt{|\tau_{m-1} \omega_{m+1}^2 / (\lambda_m \omega_m^2 + \omega_{m+1}^2)|^2 + |\kappa_m \omega_{m+1} / \alpha_{n-1}|^2}$
$\mathbf{d}_m = \theta_m \mathbf{d}_{m-1} + \kappa_m \mathbf{y}_m$
$\mathbf{x}_m = \mathbf{x}_{m-1} + \mathbf{d}_m$
stop, falls $\mathbf{x}_m$ konvergiert hat
enddo
$\rho_n = \tilde{\mathbf{r}}_0^T \mathbf{w}_{2n+1}$
$\beta_n = \rho_n / \rho_{n-1}$
$\mathbf{y}_{2n+1} = \mathbf{w}_{2n+1} + \beta_n \mathbf{y}_{2n}$
$\mathbf{v}_n = \mathbf{A}\mathbf{y}_{2n+1} + \beta_n \left(\mathbf{A}\mathbf{y}_{2n} + \beta_n \mathbf{v}_{n-1} \right)$
enddo

6.3.2 1-Norm-Minimierung der Quasi-Residuen

Neben der Wahl von $\mathbf{z}$, die durch (6.46) vorgenommen wird und die zu dem Verfahren TFQMR führt, wird nun eine neue CGS–basierte Methode definiert. Diese Methode wählt den Vektor $\mathbf{z}_m$ in (6.45) als eine Lösung des 1-Norm-Minimierungsproblems

$$\|\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m \mathbf{z}_m\|_1 = \min_{\mathbf{z} \in \mathbb{C}^m} \|\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m \mathbf{z}\|_1. \tag{6.54}$$

Es wird darauf hingewiesen, daß eine Lösung $\mathbf{z}_m$ nicht notwendig eindeutig bestimmt ist. Lemma 5.3 gibt jedoch eine konstruktive Lösung an, die nun zur Herleitung der neuen Methode der **1**-Norm **T**ranspositions**F**reien **Q**uasi-**M**inimalen **R**esiduen (TFQMR$_1$) benutzt wird.

Zur Herleitung von TFQMR$_1$ wird eine Folge von Hilfsiterierten benötigt, die durch

$$\widetilde{\mathbf{x}}_m = \mathbf{x}_0 + \mathbf{Y}_m \widetilde{\mathbf{z}}_m \qquad \text{mit} \qquad \widetilde{\mathbf{z}}_m = \mathbf{B}_m^{-1} \mathbf{f}_m \tag{6.55}$$

definiert sind. Dabei bezeichnet $\mathbf{B}_m$ diejenige $m \times m$ untere Bidiagonalmatrix, die durch Streichen der letzten Zeile von $\widehat{\mathbf{B}}_m$ entsteht. Lemma 5.3 wird nun auf das spezielle Minimierungsproblem (6.54) angewendet. Wegen (5.19) genügen die Hilfsiterierten aus (6.55) der Rekursion

$$\widetilde{\mathbf{x}}_m = \mathbf{x}_0 + \mathbf{Y}_m \begin{bmatrix} \widetilde{\mathbf{z}}_{m-1} \\ \nu_m \end{bmatrix} = \widetilde{\mathbf{x}}_{m-1} + \nu_m \mathbf{y}_m, \tag{6.56}$$

wobei $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$. Darin ist ν_m die letzte Komponente des Vektors $\widetilde{\mathbf{z}}_m = \mathbf{B}_m^{-1}\mathbf{f}_m$, die durch eine der folgenden beiden Vorgehensweisen hergeleitet wird.

Da die Matrix $\mathbf{B}_m$ durch Streichen der letzten Zeile von $\widehat{\mathbf{B}}_m$ entsteht, zeigen die Gleichungen (6.44), (6.8) und (2.47), daß $\mathbf{B}_m$ nicht singulär ist und

$$\widetilde{\mathbf{z}}_m = (\alpha_0, \alpha_0, \alpha_1, \alpha_1, \ldots, \alpha_{\lfloor (m-1)/2 \rfloor})^T.$$

Die letzte Komponente dieses Vektors liefert die Beziehung

$$\nu_m = \alpha_{\lfloor (m-1)/2 \rfloor}, \tag{6.57}$$

womit aus (6.56) die Rekursion

$$\widetilde{\mathbf{x}}_m = \widetilde{\mathbf{x}}_{m-1} + \alpha_{\lfloor (m-1)/2 \rfloor} \mathbf{y}_m \tag{6.58}$$

folgt. Eine alternative Herleitung von ν_m wendet explizit Lemma 5.3 an, wobei in (5.20) die skalaren Größen

$$\delta_m = \frac{\omega_m}{\alpha_{\lfloor (m-1)/2 \rfloor}} \qquad \text{und} \qquad \varepsilon_{m+1} = \frac{-\omega_{m+1}}{\alpha_{\lfloor (m-1)/2 \rfloor}}, \qquad m \geq 1,$$

eingesetzt werden. Auf diese Weise gelangt man ebenso zu (6.57). Man beachte, daß daraus die Beziehung

$$|\varepsilon_{m+1}\nu_m| = \omega_{m+1}$$

folgt, die man bei der Anwendung von Lemma 5.3 (ii) zur Fallunterscheidung in (5.21) benötigt. Danach genügen die TFQMR$_1$–Iterierten aus (6.45) der Rekursion

$$\mathbf{x}_m = \begin{cases} \mathbf{x}_0 + \mathbf{Y}_m \begin{bmatrix} \mathbf{z}_{m-1} \\ 0 \end{bmatrix} = \mathbf{x}_{m-1}, & \text{falls } \omega_{m+1} \geq \tau_{m-1}, \\ \mathbf{x}_0 + \mathbf{Y}_m \widetilde{\mathbf{z}}_m = \widetilde{\mathbf{x}}_m, & \text{falls } \omega_{m+1} < \tau_{m-1}, \end{cases} \tag{6.59}$$

wobei (6.55) verwendet wird. Darin ist nach (5.22) die Größe des Quasi-Residuums durch

$$\tau_m := \|\mathbf{f}_m - \widehat{\mathbf{B}}_m \mathbf{z}_m\|_1 = \min\{\tau_{m-1}, \omega_{m+1}\} \tag{6.60}$$

mit $\tau_0 = \omega_1$ gegeben. Die resultierende Implementierung von TFQMR$_1$ folgt unter Verwendung von Alg. 2.4 zusammen mit den Gleichungen (6.58)–(6.60).

ALGORITHMUS 6.5 (TFQMR$_1$ MIT BELIEBIGEN GEWICHTEN ω_k)

Wähle $\mathbf{x}_0 \in \mathbb{C}^N$ und setze $\mathbf{w}_1 = \mathbf{y}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v}_0 = \mathbf{A}\mathbf{y}_1$
Wähle $\tilde{\mathbf{r}}_0 \in \mathbb{C}^N$ so, daß $\rho_0 = \tilde{\mathbf{r}}_0^T \mathbf{r}_0 \neq 0$ und setze $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$, $\tau_0 = \omega_1$
for $n = 1, 2, 3, \ldots$ **do**
  $\sigma_{n-1} = \tilde{\mathbf{r}}_0^T \mathbf{v}_{n-1}$
  $\alpha_{n-1} = \rho_{n-1} / \sigma_{n-1}$
  $\mathbf{y}_{2n} = \mathbf{y}_{2n-1} - \alpha_{n-1} \mathbf{v}_{n-1}$
  for $m = 2n-1, 2n$ **do**
    $\mathbf{w}_{m+1} = \mathbf{w}_m - \alpha_{n-1} \mathbf{A}\mathbf{y}_m$
    Wähle ω_{m+1} (zum Beispiel nach (6.61))
    $\widetilde{\mathbf{x}}_m = \widetilde{\mathbf{x}}_{m-1} + \alpha_{n-1} \mathbf{y}_m$
    $\mathbf{x}_m = \begin{cases} \mathbf{x}_{m-1} & \textbf{if } \omega_{m+1} \geq \tau_{m-1} \\ \widetilde{\mathbf{x}}_m & \textbf{if } \omega_{m+1} < \tau_{m-1} \end{cases}$
    $\tau_m = \min\{\tau_{m-1}, \omega_{m+1}\}$
    stop, falls $\mathbf{x}_m$ konvergiert hat
  enddo
  $\rho_n = \tilde{\mathbf{r}}_0^T \mathbf{w}_{2n+1}$
  $\beta_n = \rho_n / \rho_{n-1}$
  $\mathbf{y}_{2n+1} = \mathbf{w}_{2n+1} + \beta_n \mathbf{y}_{2n}$
  $\mathbf{v}_n = \mathbf{A}\mathbf{y}_{2n+1} + \beta_n \left(\mathbf{A}\mathbf{y}_{2n} + \beta_n \mathbf{v}_{n-1}\right)$
enddo

Dieser Algorithmus läßt die Wahl der Gewichte ω_k offen. Dieser Freiheitsgrad wird jetzt benutzt, um ein kostengünstiges Abbruchkriterium zu erhalten. Abbruchkriterien basieren oft auf einer Norm des aktuellen Residuums. In Anbetracht von (6.43) suggeriert Korollar 5.1 die Strategie

$$\omega_k = \|\mathbf{w}_k\|_p, \qquad k = 1, 2, \ldots, m+1. \tag{6.61}$$

Denn dadurch werden die Spalten der Matrix $\mathbf{W}_{m+1} \Omega_{m+1}^{-1}$ in der p-Norm auf Einheitslänge normiert. Hier ist $1 \leq p \leq \infty$ ein vom Benutzer gewählter Parameter, der es erlaubt, die Größe der TFQMR$_1$–Residuen in jeder beliebigen p-Norm unmittelbar anzugeben. Dieser Sachverhalt wird in dem folgenden Satz beschrieben.

Satz 6.3 (TFQMR$_1$). *Falls zur Normierung der Vektoren $\mathbf{w}_k$ des zugrundeliegenden CGS–Prozesses die Strategie* (6.61) *benutzt wird, fällt die Norm der TFQMR$_1$–Residuen (schwach) monoton, und es gilt*

$$\|\mathbf{r}_m\|_p = \tau_m, \qquad m \geq 1, \tag{6.62}$$

wobei $1 \leq p \leq \infty$ und τ_m die Größe der Quasi-Residuen nach (6.60) *bezeichnet.*

Beweis. Der Satz ist eine unmittelbare Folgerung aus Korollar 5.1. Die Monotonie ist durch (5.22) gegeben. □

Abschließend sei darauf hingewiesen, daß man TFQMR$_1$ wie folgt interpretieren kann: Man nehme als aktuelle Iterierte $\mathbf{x}_m$ jeweils die "beste" Hilfsiterierte $\widetilde{\mathbf{x}}_m$. Die Beziehungen

$$\widetilde{\mathbf{x}}_{2n} = \mathbf{x}_n^{\mathrm{CGS}} \qquad \text{und} \qquad \mathbf{w}_{2n+1} = \mathbf{r}_n^{\mathrm{CGS}},$$

die in [28] gezeigt werden, zeigen die enge Verwandschaft von TFQMR$_1$ und CGS.

6.4 Bi-CGSTAB–basierte Verfahren

Neben dem Lanczos–Algorithmus und CGS wird in dem Kapitel, das die Erzeugung von Krylov–Teilräumen behandelt, das Verfahren Bi-CGSTAB vorgestellt. Nach Satz 2.4 bilden die von Alg. 2.6 generierten Spaltenvektoren der Matrix $\mathbf{Y}_m$ eine Basis des $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$. Für diese Matrix weist dieser Satz die Beziehung

$$\mathbf{A}\mathbf{Y}_m = \mathbf{W}_{m+1}\widetilde{\mathbf{B}}_m \tag{6.63}$$

mit der Bidiagonalmatrix $\widetilde{\mathbf{B}}_m$ aus (2.58) nach. Da in der Initialisierung wegen (2.54) die Anweisung $\mathbf{w}_1 = \mathbf{r}_0$ enthalten ist, folgt als Ausgangspunkt in diesem Abschnitt die Darstellung

$$\mathbf{r}_m = \mathbf{W}_{m+1}\Omega_{m+1}^{-1}(\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m\mathbf{z}) \qquad \text{für ein} \qquad \mathbf{z} \in \mathbb{R}^{\,m}$$

mit

$$\mathbf{f}_{m+1} = \omega_1\mathbf{e}_1^{(m+1)} \qquad \text{und} \qquad \widehat{\mathbf{B}}_m = \Omega_{m+1}\widetilde{\mathbf{B}}_m. \tag{6.64}$$

6.4.1 Euklidische Minimierung der Quasi-Residuen

Über eine Minimierung des Quasi-Residuums wird in [13] ein Verfahren mit Iterierten

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{Y}_m\mathbf{z}_m \tag{6.65}$$

definiert, wobei $\mathbf{z}_m \in \mathbb{R}^{\,m}$ die eindeutige Lösung des euklidischen Minimierungsproblems

$$\|\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m\mathbf{z}_m\|_2 = \min_{\mathbf{z}\in\mathbb{R}^{\,m}} \|\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m\mathbf{z}\|_2$$

bezeichnet. Zur Lösung dieses Minimierungsproblems werden die in Lemma 5.1 enthaltenen Techniken der QR-Zerlegung von $\widehat{\mathbf{B}}_m$ mit Hilfe von Givens–Rotationen benutzt. Das resultierende Verfahren, das QMRCGSTAB genannt wird, verwendet die Gewichte

$$\omega_k = \|\mathbf{w}_k\|_2, \qquad k = 1, 2, \ldots, m+1. \tag{6.66}$$

Hier wird eine neue Implementierung dieses Verfahrens vorgestellt, die die bidiagonale Struktur der Matrix $\widehat{\mathbf{B}}_m$ ausnutzt. Diese Struktur erlaubt die Anwendung von Lemma 5.2. Analog zur Vorgehensweise in dem vorangehenden Abschnitt erhält man mit $\mathbf{d}_m := \mathbf{Y}_m\mathbf{g}_m$ aus (5.9)–(5.11) die Beziehungen

$$\mathbf{x}_m = \mathbf{x}_{m-1} + \mathbf{d}_m, \tag{6.67}$$

$$\mathbf{d}_m = \theta_m\mathbf{d}_{m-1} + \kappa_m\mathbf{y}_m, \tag{6.68}$$

$$\tau_m = \sqrt{\left|\frac{\tau_{m-1}\omega_{m+1}^2}{\lambda_m\omega_m^2 + \omega_{m+1}^2}\right|^2 + \left|\frac{\kappa_m\omega_{m+1}}{\sigma_m}\right|^2} \tag{6.69}$$

mit $\mathbf{d}_0 = \mathbf{0}$ und $\tau_0 = \omega_1$. Die Koeffizienten θ_m und κ_m folgen aus den Rekursionen (5.12)–(5.14)

$$\theta_m = \frac{\omega_m^2\,(1-\lambda_m)}{\lambda_m\omega_m^2 + \omega_{m+1}^2}, \qquad m \geq 1, \tag{6.70}$$

$$\kappa_m = \frac{\omega_m^2\,\kappa_{m-1}}{\lambda_m\omega_m^2 + \omega_{m+1}^2} \cdot \frac{\sigma_m}{\sigma_{m-1}}, \qquad m \geq 1, \qquad \kappa_0 = \sigma_0, \tag{6.71}$$

$$\lambda_{m+1} = \frac{\lambda_m\omega_m^2}{\lambda_m\omega_m^2 + \omega_{m+1}^2}, \qquad m \geq 1, \qquad \lambda_1 = 1. \tag{6.72}$$

Wählt man die Gewichte nach (6.66), so erhält man durch (6.12) und (6.69) ein kostengünstiges Abbruchkriterium. Verwendet man Alg. 2.6 als zugrundeliegenden Prozeß zur Erzeugung der Krylov–Teilräume, so erhält man das resultierenden Verfahren mit den Gleichungen (6.67)–(6.72).

ALGORITHMUS 6.6 (QMRCGSTAB MIT ALG. 2.6 UND LEMMA 5.2)

Wähle $\mathbf{x}_0 \in \mathbb{R}^N$ und setze $\mathbf{w}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{y}_{-1} = \mathbf{d}_0 = \mathbf{0}$
Wähle $\tilde{\mathbf{r}}_0 \in \mathbb{R}^N$ so, daß $\tilde{\mathbf{r}}_0^T\mathbf{r}_0 \neq 0$ und setze $\rho_0 = \sigma_{-1} = \sigma_0 = 1$
Wähle ω_1 (zum Beispiel nach (6.66)) und setze $\lambda_1 = 1$, $\kappa_0 = \sigma_0$, $\tau_0 = \omega_1$
for $n = 1, 2, 3, \ldots$ **do**
  $\rho_n = \tilde{\mathbf{r}}_0^T\mathbf{w}_{2n-1}$
  $\beta_n = (\rho_n/\rho_{n-1})(\sigma_{2n-3}/\sigma_{2n-2})$
  $\mathbf{y}_{2n-1} = \mathbf{w}_{2n-1} + \beta_n(\mathbf{y}_{2n-3} - \sigma_{2n-2}\mathbf{A}\mathbf{y}_{2n-3})$
  for $m = 2n-1, 2n$ **do**
    if $m = 2n$ **then** $\mathbf{y}_m = \mathbf{w}_m$ **endif**

$$\sigma_m = \begin{cases} \rho_n/\tilde{\mathbf{r}}_0^T\mathbf{A}\mathbf{y}_m & \textbf{if } m = 2n-1 \\ (\mathbf{A}\mathbf{y}_m)^T\mathbf{w}_m/(\mathbf{A}\mathbf{y}_m)^T\mathbf{A}\mathbf{y}_m & \textbf{if } m = 2n \end{cases}$$

    $\mathbf{w}_{m+1} = \mathbf{w}_m - \sigma_m\mathbf{A}\mathbf{y}_m$
    Wähle ω_{m+1} (zum Beispiel nach (6.66))
    $\theta_m = \omega_m^2\,(1-\lambda_m)/(\lambda_m\omega_m^2 + \omega_{m+1}^2)$
    $\kappa_m = \omega_m^2\,\kappa_{m-1}\sigma_m/\left(\sigma_{m-1}\cdot(\lambda_m\omega_m^2 + \omega_{m+1}^2)\right)$
    $\lambda_{m+1} = \lambda_m\omega_m^2/(\lambda_m\omega_m^2 + \omega_{m+1}^2)$
    $\tau_m = \sqrt{|\tau_{m-1}\omega_{m+1}^2/(\lambda_m\omega_m^2 + \omega_{m+1}^2)|^2 + |\kappa_m\omega_{m+1}/\sigma_m|^2}$
    $\mathbf{d}_m = \theta_m\mathbf{d}_{m-1} + \kappa_m\mathbf{y}_m$
    $\mathbf{x}_m = \mathbf{x}_{m-1} + \mathbf{d}_m$
    stop, falls $\mathbf{x}_m$ konvergiert hat
  enddo
enddo

6.4.2 1-Norm-Minimierung der Quasi-Residuen

Anstelle der Minimierung des Quasi-Residuums in der euklidischen Norm wird nun eine neue Methode durch Minimierung des Quasi-Residuums in der 1-Norm definiert. Die Iterierten dieser neuen Methode sind weiterhin von der Form (6.65), wobei der Vektor $\mathbf{z}_m$ nun eine Lösung des 1-Norm-Minimierungsproblems

$$\|\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m \mathbf{z}_m\|_1 = \min_{\mathbf{z}\in\mathbb{R}^m} \|\mathbf{f}_{m+1} - \widehat{\mathbf{B}}_m \mathbf{z}\|_1$$

darstellt. Eine Lösung $\mathbf{z}_m$ ist nicht notwendig eindeutig bestimmt. Eine spezielle Lösung wird durch Lemma 5.3 angegeben. Die Herleitung dieser so entstehenden Methode, die QMRCGSTAB$_1$ genannt wird, verwendet wiederum eine Folge von Hilfsiterierten. Diese Hilfsiterierten sind durch

$$\widetilde{\mathbf{x}}_m = \mathbf{x}_0 + \mathbf{Y}_m \widetilde{\mathbf{z}}_m \qquad \text{mit} \qquad \widetilde{\mathbf{z}}_m = \mathbf{B}_m^{-1} \mathbf{f}_m$$

definiert, wobei $\mathbf{B}_m$ diejenige $m \times m$ untere Bidiagonalmatrix bezeichnet, die durch Streichen der letzten Zeile von $\widehat{\mathbf{B}}_m$ entsteht. Eine Rekursion für die Hilfsiterierten erhält man durch Anwendung von (5.19) aus Lemma 5.3

$$\widetilde{\mathbf{x}}_m = \widetilde{\mathbf{x}}_{m-1} + \nu_m \mathbf{y}_m, \tag{6.73}$$

wobei $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$. Darin ist ν_m die letzte Komponente des Vektors $\widetilde{\mathbf{z}}_m = \mathbf{B}_m^{-1}\mathbf{f}_m$. Die Matrix $\mathbf{B}_m$ ist nach (6.64), (6.8) und (2.58) nicht singulär. Es ist leicht nachzuweisen, daß

$$\widetilde{\mathbf{z}}_m = (\sigma_1, \sigma_2, \ldots, \sigma_m)^T.$$

Die letzte Komponente dieses Vektors zeigt die Beziehung

$$\nu_m = \sigma_m,$$

womit die Rekursion (6.73) der Hilfsiterierten durch

$$\widetilde{\mathbf{x}}_m = \widetilde{\mathbf{x}}_{m-1} + \sigma_m \mathbf{y}_m \tag{6.74}$$

gegeben ist. Die Anwendung von Lemma 5.3 (ii) liefert eine Rekursion für die QMRCGSTAB$_1$–Iterierten in der Form

$$\mathbf{x}_m = \begin{cases} \mathbf{x}_{m-1}, & \text{falls } \omega_{m+1} \geq \tau_{m-1}, \\ \widetilde{\mathbf{x}}_m, & \text{falls } \omega_{m+1} < \tau_{m-1}, \end{cases} \tag{6.75}$$

wobei

$$\tau_m := \|\mathbf{f}_m - \widehat{\mathbf{B}}_m \mathbf{z}_m\|_1 = \min\{\tau_{m-1}, \omega_{m+1}\} \tag{6.76}$$

mit $\tau_0 = \omega_1$. Der vollständige Prozeß folgt durch Zusammenfügen von Alg. 2.6 mit den Gleichungen (6.74)–(6.76).

ALGORITHMUS 6.7 (QMRCGSTAB$_1$ MIT BELIEBIGEN GEWICHTEN ω_k)

Wähle $\mathbf{x}_0 \in \mathbb{R}^N$ und setze $\mathbf{w}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{y}_{-1} = \mathbf{0}$, $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$
Wähle $\tilde{\mathbf{r}}_0 \in \mathbb{R}^N$ so, daß $\tilde{\mathbf{r}}_0^T \mathbf{r}_0 \neq 0$ und setze $\rho_0 = \sigma_{-1} = \sigma_0 = 1$, $\tau_0 = \omega_1$
for $n = 1, 2, 3, \ldots$ **do**
$\rho_n = \tilde{\mathbf{r}}_0^T \mathbf{w}_{2n-1}$
$\beta_n = (\rho_n / \rho_{n-1})(\sigma_{2n-3} / \sigma_{2n-2})$
$\mathbf{y}_{2n-1} = \mathbf{w}_{2n-1} + \beta_n (\mathbf{y}_{2n-3} - \sigma_{2n-2} \mathbf{A}\mathbf{y}_{2n-3})$
for $m = 2n-1, 2n$ **do**
if $m = 2n$ **then** $\mathbf{y}_m = \mathbf{w}_m$ **endif**

$$\sigma_m = \begin{cases} \rho_n / \tilde{\mathbf{r}}_0^T \mathbf{A}\mathbf{y}_m & \textbf{if } m = 2n-1 \\ (\mathbf{A}\mathbf{y}_m)^T \mathbf{w}_m / (\mathbf{A}\mathbf{y}_m)^T \mathbf{A}\mathbf{y}_m & \textbf{if } m = 2n \end{cases}$$

$\mathbf{w}_{m+1} = \mathbf{w}_m - \sigma_m \mathbf{A}\mathbf{y}_m$
Wähle ω_{m+1} (zum Beispiel nach (6.61))
$\widetilde{\mathbf{x}}_m = \widetilde{\mathbf{x}}_{m-1} + \sigma_m \mathbf{y}_m$

$$\mathbf{x}_m = \begin{cases} \mathbf{x}_{m-1} & \textbf{if } \omega_{m+1} \geq \tau_{m-1} \\ \widetilde{\mathbf{x}}_m & \textbf{if } \omega_{m+1} < \tau_{m-1} \end{cases}$$

$\tau_m = \min\{\tau_{m-1}, \omega_{m+1}\}$
stop, falls $\mathbf{x}_m$ konvergiert hat
enddo
enddo

Analog zum vorangehenden Abschnitt liefert die Wahl der Gewichte auch hier ein kostengünstiges Abbruchkriterium.

Satz 6.4 (QMRCGSTAB$_1$). *Falls zur Normierung der $\mathbf{w}_k$ des zugrundeliegenden Bi-CGSTAB–Prozesses die Strategie* (6.61) *benutzt wird, fällt die Norm der QMRCGSTAB$_1$–Residuen (schwach) monoton, und es gilt*

$$\|\mathbf{r}_m\|_p = \tau_m, \qquad m \geq 1, \tag{6.77}$$

wobei $1 \leq p \leq \infty$ und τ_m die Größe der Quasi-Residuen nach (6.76) *bezeichnet.*

Beweis. Wiederum folgt der Satz unmittelbar aus Korollar 5.1. □

Kapitel 7

Numerische Experimente

Das Verhalten der in den vorherigen Kapiteln hergeleiteten iterativen Verfahren wird in diesem Kapitel anhand einiger ausgewählter numerischer Experimente beschrieben. Dabei werden die Methoden sowohl untereinander als auch mit anderen in der Literatur enthaltenen Methoden verglichen. Nach einem Abschnitt, der die allgemeinen Rahmenbedingungen für die Implementierungen beschreibt, werden numerische Experimente diskutiert.

7.1 Rahmenbedingungen der Implementierungen

Die numerischen Experimente werden auf verschiedenen am Zentralinstitut für Angewandte Mathematik des Forschungszentrums Jülich installierten massiv-parallelen Rechnersystemen durchgeführt. Für die seriellen Untersuchungen des Konvergenzverhaltens wird ausschließlich ein einzelner Rechenknoten des parallelen Systems Intel Paragon XP/S 10 verwendet. Dieses System mit verteiltem Speicher verbindet bis zu 140 Prozessoren in der Form eines zweidimensionalen Gitters miteinander. Für die parallelen Untersuchungen wird neben dem System Intel Paragon XP/S 10 zusätzlich das System CRAY T3E mit der Topologie eines dreidimensionalen Torus mit bis zu 512 Prozessoren benutzt. Wichtige Kenngrößen dieser beiden Systeme sind in Tab. 7.1 zusammengestellt. Während für die Kommunikation der Prozessoren untereinander bei der Implementierung auf Intel Paragon XP/S 10 die vom Hersteller bereitgestellte NX-Bibliothek benutzt wird, verwendet die parallele Implementierung auf CRAY T3E eine herstellerunabhängige Kommunikation auf der Basis von MPI.

Alle Berechnungen werden mit doppelter Genauigkeit unter Verwendung des FORTRAN–Compilers der Firma THE PORTLAND GROUP, INC., Release 5.0.3, durchgeführt. In allen Experimenten wird $\mathbf{x}_0 = \mathbf{0}$ als Startvektor gewählt. Bei den Verfahren, die CGS oder Bi-CGSTAB zur Erzeugung ihrer zugrundeliegenden Krylov–Teilräume verwenden, wird ein zweiter Startvek-

Tabelle 7.1: Wichtige Kenngrößen der verwendeten Parallelrechnersysteme

Kenngröße	*Intel Paragon XP/S 10*	*CRAY T3E*
Anzahl Prozessoren	140 Rechenknoten	512 Rechenknoten
Topologie	2D Gitter	3D Torus
Maximale Leistung	10.5 Gigaflops	307.2 Gigaflops
Verteilter Speicher	4.5 GB (32 MB / Knoten)	64 GB (128 MB / Knoten)
Betriebssystem	Paragon OSF/1 AD	UNICOS/mk

tor $\tilde{\mathbf{r}}_0$ benötigt. Dieser zweite Startvektor wird zufällig mit Hilfe einer Normalverteilung mit Mittelwert 0.0 und Standardabweichung 1.0 erzeugt und ist innerhalb eines Beispieles für alle betreffenden Verfahren identisch. Soweit nicht explizit auf andere Größen hingewiesen wird, werden in allen Abbildungen dieses Kapitels jeweils die euklidischen Normen der tatsächlichen Residuen dargestellt. Genauer wird $\log_{10}(\|\mathbf{r}_n\|_2/\|\mathbf{r}_0\|_2)$ als Funktion der dazu aufgewendeten Matrix-Vektor-Produkte gezeigt. Dabei wird das zur Berechnung des tatsächlichen Residuums $\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$ benötigte zusätzliche Matrix-Vektor-Produkt nicht berücksichtigt. Für die CGS– oder Bi-CGSTAB–basierten Verfahren entspricht diese Anzahl von Matrix-Vektor-Produkten der Anzahl der Iterationen n. Dagegen ist diese Anzahl der Matrix-Vektor-Produkte bei den auf dem Lanczos–Algorithmus basierenden Verfahren doppelt so groß wie die aktuelle Iterationsanzahl. Da der Gesamtaufwand der betrachteten iterativen Verfahren im Wesentlichen durch den Aufwand der anfallenden Matrix-Vektor-Produkte bestimmt ist, erlaubt dieser Darstellung einen "fairen" Vergleich der Methoden untereinander. Außerdem nutzen alle parallelen Implementierungen des Lanczos–Algorithmus und der darauf basierenden Methoden die Unabhängigkeit der zwei in einem Iterationsschritt benötigten Matrix-Vektor-Produkte aus, indem diese beiden Operationen simultan berechnet werden. Um die Unterschiede der einzelnen Verfahren zu verdeutlichen, wird keine Vorkonditionierung verwendet.

Für die numerischen Experimente wird die dreidimensionale partielle Differentialgleichung

$$Lu = f \qquad \text{in} \quad \Omega = (0,1) \times (0,1) \times (0,1) \tag{7.1}$$

mit Dirichlet Randbedingungen $u = 0$ auf $\partial\Omega$ zugrundegelegt, wobei Lu und f in den einzelnen Experimenten unterschiedlich gesetzt werden. Zur Diskretisierung von (7.1) werden zentrierte Differenzen zweiter Ordnung verwendet. Um die Daten bei der parallelen Implementierung auf die Prozessoren zu verteilen, wird Ω in kubische Teilgebiete der gleichen Größe aufgeteilt. Jedem Prozessor werden dann die zu einem Teilgebiet korrespondierenden Daten zugewiesen. Aufgrund der lokalen Struktur des Diskretisierungsschemas muß deshalb ein Prozessor zur Berechnung eines Matrix-Vektor-Produktes mit höchstens sechs benachbarten Prozessoren kommunizieren.

7.2 Analyse der iterativen Verfahren

Anhand einiger ausgewählter Beispiele werden die in der vorliegenden Arbeit hergeleiteten Verfahren nun sowohl hinsichtlich serieller als auch paralleler Gesichtspunkte diskutiert.

Experiment 7.1. Das erste Beispiel stammt aus [68] und ist eine Diskretisierung von (7.1) mit

$$Lu = -\Delta u + 1000\frac{\partial u}{\partial x}.$$

Dabei wird die rechte Seite f so gewählt, daß die Lösung der Differentialgleichung durch

$$u(x,y,z) = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$$

gegeben ist. Für die oben skizzierte Diskretisierung wird ein $22 \times 22 \times 22$ Gitter mit einer Schrittweite von $1/23$ in jeder Dimension zugrundegelegt. Diese Diskretisierung führt auf ein lineares Gleichungssystem mit $10\,648$ Unbekannten und einer Koeffizientenmatrix mit $71\,632$ Nichtnull-Elementen. Die betrachteten Verfahren werden abbgebrochen, sobald die Ungleichung

$$\frac{\|\mathbf{r}_n\|_2}{\|\mathbf{r}_0\|_2} \leq 10^{-10}$$

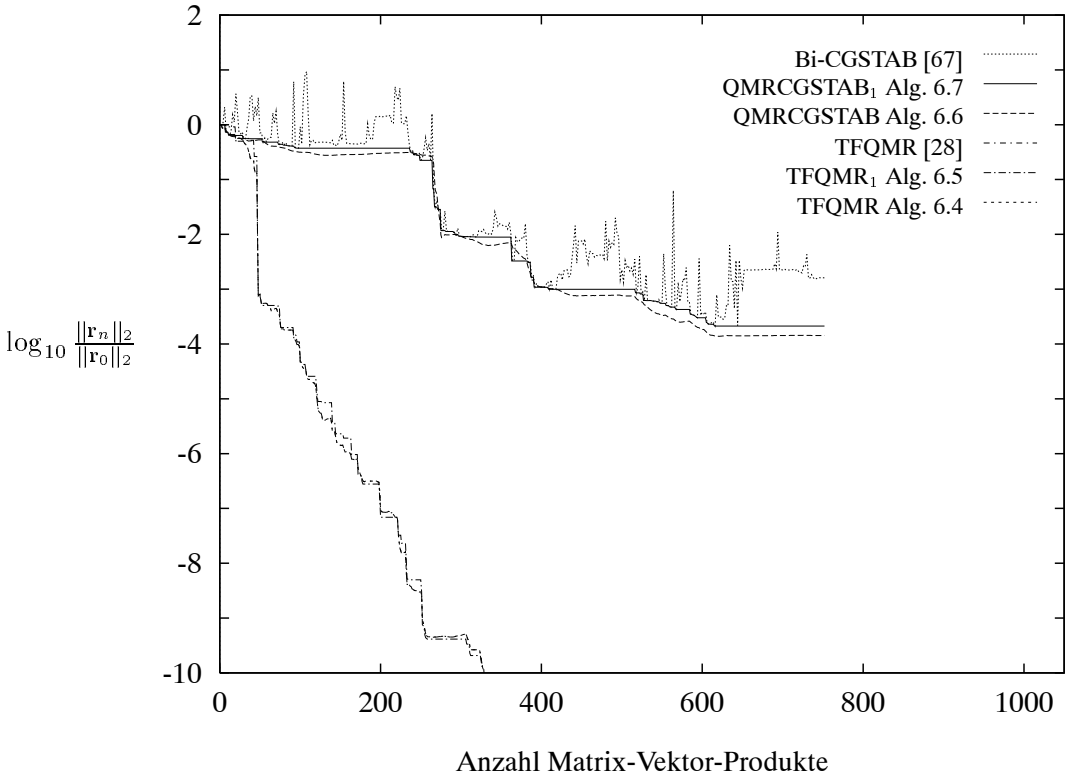


Abbildung 7.1: Konvergenzverläufe aus Exp. 7.1 für Bi-CGSTAB– und CGS–basierte Verfahren

erfüllt ist. Die seriellen Ergebnisse diese Beispiels mit unterschiedlichen iterativen Verfahren sind in den nächsten drei Abbildungen zusammengetragen. Abbildung 7.1 zeigt zunächst solche Verfahren, die zum Aufspannen ihrer zugrundeliegenden Krylov–Teilräume Bi-CGSTAB oder CGS benutzen. Die Aufspaltung der abgebildeten Kurven in zwei grobe Klassen von Konvergenzverläufen zeigt deutlich, daß das Konvergenzverhalten einer iterativen Methode wesentlich durch das zugrundeliegende Verfahren zur Erzeugung der Krylov–Teilräume bestimmt wird. Um diese Tatsache für die Bi-CGSTAB–basierten Verfahren zu verdeutlichen, ist Bi-CGSTAB in dieser Abbildung mitaufgeführt. Da Bi-CGSTAB nach 752 Matrix-Vektor-Produkten abbricht, erfolgt bei QMRCGSTAB aus Alg. 6.6 und auch bei QMRCGSTAB$_1$ aus Alg. 6.7 zu diesem Zeitpunkt ebenfalls ein Abbruch. Dagegen brechen die CGS–basierten Verfahren TFQMR und TFQMR$_1$ in diesem Beispiel nicht ab. Für TFQMR ist neben der ursprünglichen Version [28, Alg. 5.1] auch die Implementierung aus Alg. 6.4 dargestellt. Diese beiden Versionen unterscheiden sich in den Techniken, die zur Lösung der auftretenden euklidischen Minimierungsprobleme angewendet werden. Während die Originalarbeit [28] dazu Givens–Rotationen verwendet, benutzt Alg. 6.4 das in dieser Arbeit vorgestellte Lemma 5.2, das mit Hilfe der Methode der Normalgleichungen bewiesen wird. Das Konvergenzverhalten dieser beiden Implementierungen ist in der logarithmischen Darstellung nicht zu unterscheiden. Die Minimierung der auftretenden Minimierungsprobleme in der 1-Norm statt in der euklidischen Norm führt zu TFQMR$_1$ aus Alg. 6.5, das einen zu TFQMR ähnlichen Konvergenzverlauf aufweist. Zusammenfassend gilt, daß alle CGS–basierten Verfahren schnell konvergieren, während alle Bi-CGSTAB–basierten Verfahren einen nur langsamen Abfall in der Residuumsnorm zeigen und schließlich sogar abbrechen.

In Abb. 7.2 sind die Ergebnisse derjenigen Verfahren dargestellt, die zum Aufspannen ihrer Krylov–Teilräume den Lanczos–Algorithmus benutzen. In dieser Abbildung ist das Verhalten von sechs Algorithmen dargestellt, von denen jeweils zwei in der logarithmischen Darstellung

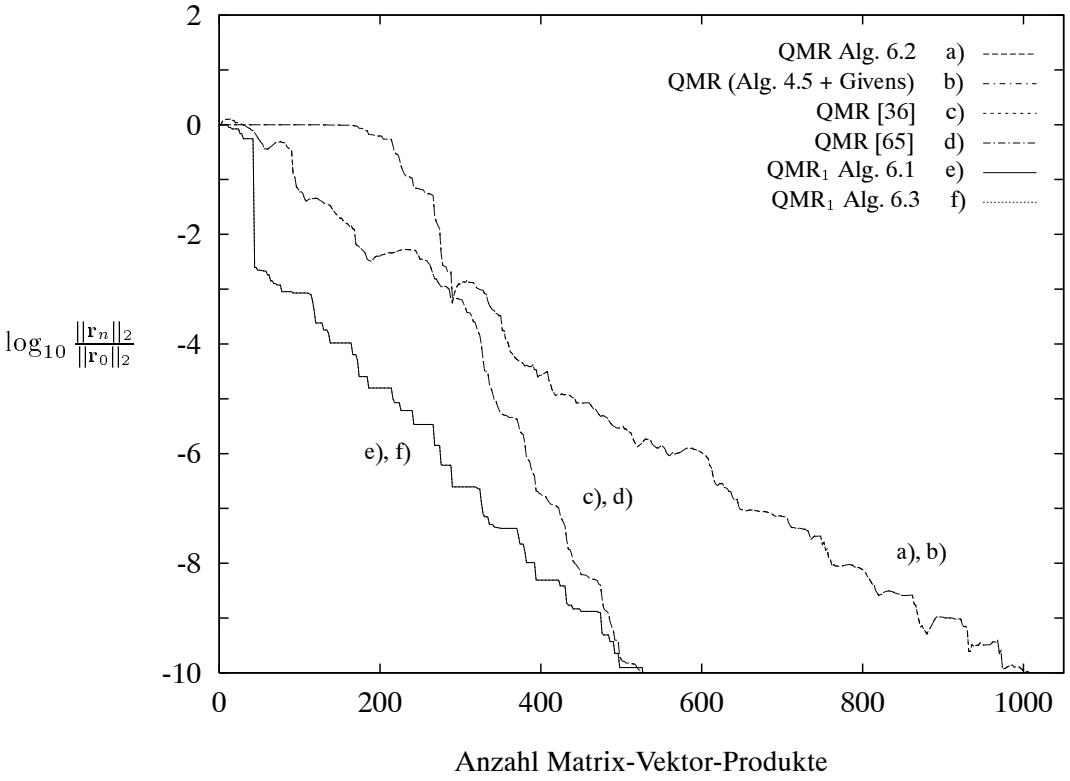


Abbildung 7.2: Konvergenzverläufe aus Exp. 7.1 für Lanczos–basierte Verfahren

zusammenfallen, so daß insgesamt nur drei Kurven sichtbar sind. Die parallele QMR–Variante aus Alg. 6.2 konvergiert in diesem Beispiel deutlich langsamer als die ursprüngliche Version von QMR [36, Alg. 7.1]. Diese beiden Algorithmen unterscheiden sich in zwei grundsätzlichen Aspekten voneinander. Einerseits verwenden diese beiden Algorithmen unterschiedliche Varianten des Lanczos–Algorithmus zum Aufspannen der Krylov–Teilräume und andererseits werden die auftretenden euklidischen Minimierungsprobleme mit Hilfe unterschiedlicher Techniken gelöst. Um festzustellen, welche dieser beiden Einflußfaktoren für das unterschiedliche Konvergenzverhalten verantwortlich ist, werden die folgenden vier QMR–Algorithmen betrachtet. Neben der ursprünglichen Version [36, Alg. 7.1], die den Lanczos–Algorithmus mit gekoppelten Zwei-Term-Rekursionen aus Alg. 2.2 und die Technik der Givens–Rotationen aus Lemma 5.1 verwendet, wird eine Variante [65] mit dem gleichen zugrundeliegenden Lanczos–Algorithmus aber mit der Technik der Normalgleichungen aus Lemma 5.2 untersucht. Zwei weitere QMR–Implementierungen verwenden die parallele Variante des Lanczos–Algorithmus aus Alg. 4.5. Sie unterscheiden sich dadurch, daß die auftretenden euklidischen Minimierungsprobleme in Alg. 6.2 durch die Technik der Normalgleichungen und in einem weiteren Algorithmus, der in dieser Abbildung mit b) gekennzeichnet ist, durch die Technik der Givens–Rotationen gelöst werden. In der logarithmischen Darstellung sind keine Unterschiede in dem Konvergenzverhalten zu beobachten, wenn man für einen der beiden zugrundeliegenden Lanczos–Algorithmen die Technik zur Lösung der auftretenden Minimierungsprobleme variiert. Bei beiden zugrundeliegenden Lanczos–Algorithmen fallen die Kurven der Implementierungen, die Givens–Rotationen bzw. Normalgleichungen verwenden, zusammen. Dieses Verhalten wird auch in anderen Beispielen beobachtet und weist darauf hin, daß die numerischen Eigenschaften der Lemmata 5.1 und 5.2 ähnlich sind. Das in diesem Beispiel im Vergleich zur ursprünglichen QMR–Version [36] deutlich langsamere Konvergenzverhalten der parallelen Variante ist demnach auf den veränderten zugrundeliegenden Lanczos–Algorithmus

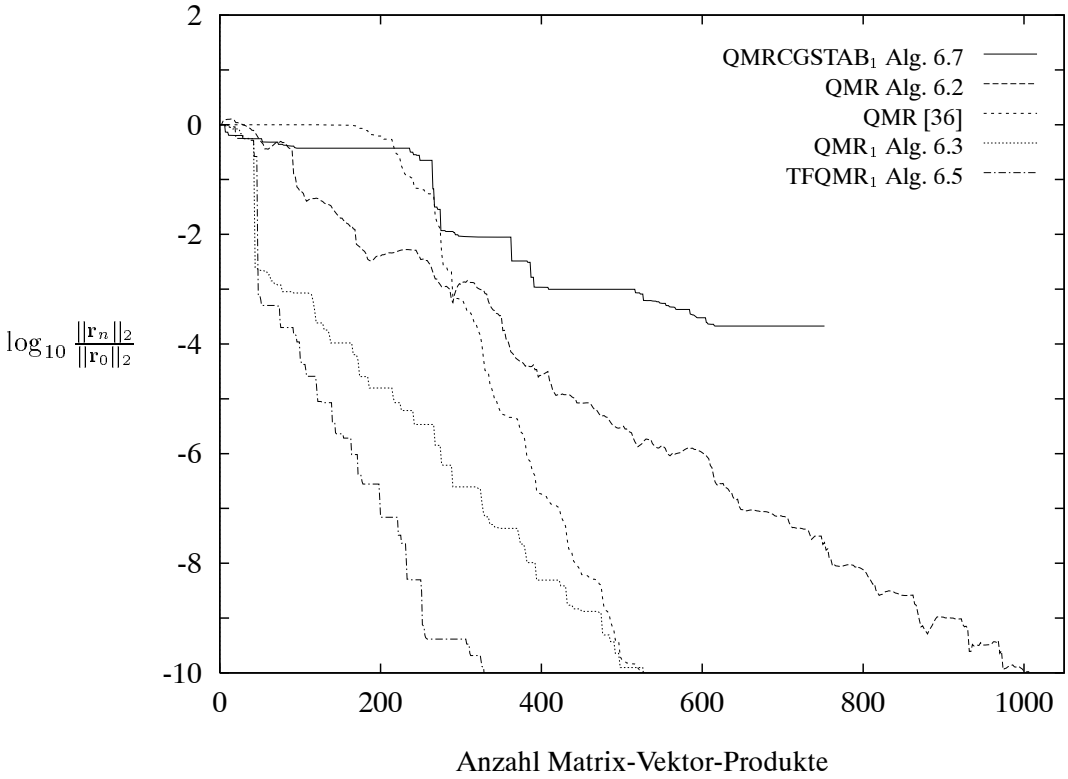


Abbildung 7.3: Konvergenzverläufe aus Exp. 7.1 im Überblick

zurückzuführen. Das nächste Experiment zeigt jedoch, daß dieses ungünstige Verhalten keinesfalls allgemeingültig ist.

Abbildung 7.2 enthält eine weitere Kurve, die den in der logarithmischen Darstellung gemeinsamen Konvergenzverlauf von QMR$_1$ basierend auf Drei-Term-Rekursionen aus Alg. 6.1 und QMR$_1$ basierend auf gekoppelten Zwei-Term-Rekursionen aus Alg. 6.3 zeigt. Beide 1-Norm-Varianten fallen in diesem Beispiel zusammen und konvergieren schneller als die ursprüngliche QMR–Version [36].

In Abb. 7.3 sind keine weiteren Informationen enthalten; sie vergleicht ausgewählte Methoden aus den beiden letzten Abbildungen, um für dieses Beispiel einen abschließenden Vergleich aller Methoden vorzunehmen. QMRCGSTAB$_1$ als Vertreter der Bi-CGSTAB–basierten Verfahren bricht ab, während die parallele QMR–Variante aus Alg. 6.2 langsam konvergiert. Die ursprüngliche QMR–Version [36] weist ein deutlich schnelleres Konvergenzverhalten auf, ist jedoch den 1-Norm QMR–Varianten unterlegen. Alle CGS–basierten Verfahren wie beispielsweise TFQMR$_1$ zeigen in diesem Beispiel das beste Konvergenzverhalten.

Experiment 7.2. Aus [35] ist das zweite Beispiel entnommen, wobei in (7.1) der Ausdruck

$$Lu = -\Delta u + 40\left(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y} + z\frac{\partial u}{\partial z}\right) - 250u \tag{7.2}$$

benutzt wird. Die rechte Seite f ist hier so gewählt, daß für die Lösung der Differentialgleichung

$$u \equiv 1 \tag{7.3}$$

gilt. Zur Diskretisierung wird wiederum ein $22 \times 22 \times 22$ Gitter mit einer Schrittweite von $1/23$ in jeder Dimension verwendet. Die Koeffizientenmatrix des resultierenden Systems besitzt die

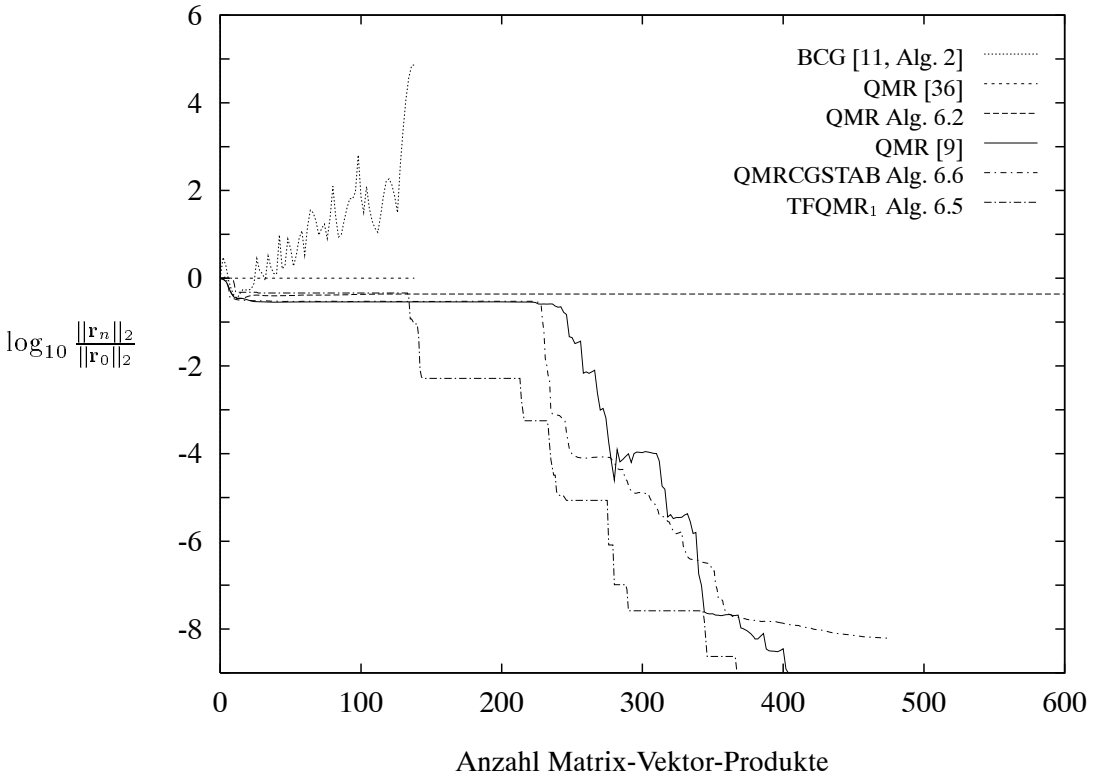


Abbildung 7.4: Konvergenzverläufe aus Exp. 7.2 im Überblick

Ordung $10\,648$ und verfügt über $71\,632$ Nichtnull-Elemente. Als Abbruchkriterium wird

$$\frac{\|\mathbf{r}_n\|_2}{\|\mathbf{r}_0\|_2} \leq 10^{-9}$$

verwendet. In Abb. 7.4 sind die seriellen Ergebnisse für dieses Beispiel dargestellt. Da der Lanczos–Algorithmus in seiner ursprünglichen Form aus Alg. 2.2 in diesem Beispiel nach 138 Matrix-Vektor-Produkten abbricht, brechen zu diesem Zeitpunkt auch die darauf basierenden Verfahren ab. Dazu gehören eine reskalierte BCG–Verison [11], QMR in seiner ursprünglichen Form [36, Alg. 7.1] und eine in dieser Abbildung nicht dargestellte QMR–Version, die an Stelle von Givens–Rotationen die Technik der Normalgleichungen anwendet. Die parallele QMR–Variante aus Alg. 6.2 bricht zwar nicht ab, aber stagniert bei $\|\mathbf{r}_n\|_2/\|\mathbf{r}_0\|_2 \approx 0.436$. Eine andere parallele QMR–Variante [9] dagegen konvergiert. Alle Bi-CGSTAB–basierten Verfahren, von denen QMRCGSTAB aus Alg. 6.6 abbgebildet ist, verhalten sich untereinander ähnlich und brechen nach 476 Matrix-Vektor-Produkten ab. Die Konvergenzverläufe der CGS–basierten Verfahren, die den günstigsten Verlauf aufweisen, sind wiederum untereinander ähnlich. Stellvertretend für diese Verfahren ist TFQMR$_1$ dargestellt. Hier nicht abgebildet sind die QMR$_1$–Implementierungen aus Alg. 6.1 und Alg. 6.3, die in diesem Beispiel nach 142 und 134 Matrix-Vektor-Produkten abbrechen.

Experiment 7.3. Das dritte Beispiel ist eine Diskretisierung von (7.1) mit

$$Lu = -\Delta u - 20\left(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y} + z\frac{\partial u}{\partial z}\right) \tag{7.4}$$

und einer Wahl der rechten Seite f so, daß die Lösung der Differentialgleichung durch

$$u(x,y,z) = 2\sin(4\pi x)\sin(6\pi y)\sin(4\pi z) \tag{7.5}$$

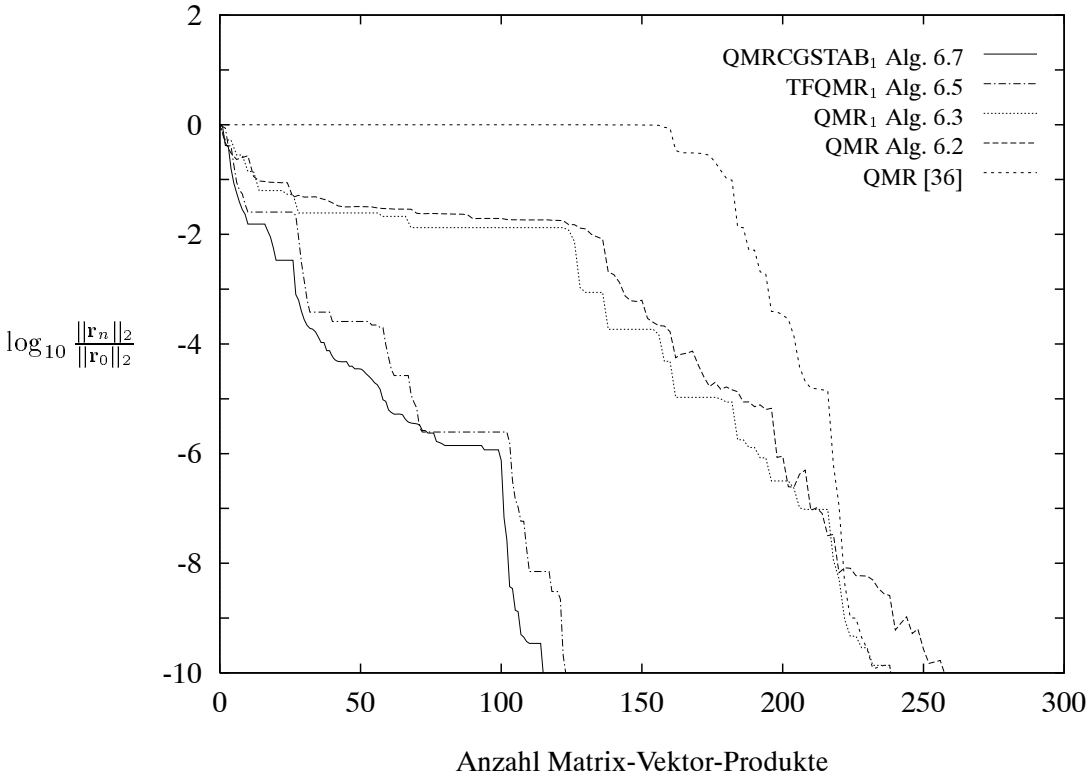


Abbildung 7.5: Konvergenzverläufe aus Exp. 7.3 im Überblick

gegeben ist. Die numerische Lösung basiert auf einem $25 \times 25 \times 25$ Gitter mit einer Schrittweite von $1/26$ in jeder Dimension. Diese Vorgehensweise führt auf ein Gleichungssystem mit $15\,625$ Unbekannten und $105\,625$ Nichtnull-Elementen. Sobald die Ungleichung

$$\frac{\|\mathbf{r}_n\|_2}{\|\mathbf{r}_0\|_2} \leq 10^{-10} \tag{7.6}$$

erfüllt ist, werden die Verfahren abbgebrochen. Die seriellen Ergebnisse dieses Beispiels sind in Abb. 7.5 dargestellt. Während sich die Bi-CGSTAB–basierten Verfahren in den vorangehenden Experimenten als besonders ungünstig erweisen, stellen sich diese Verfahren in diesem Beispiel als die am schnellsten konvergierende Verfahren heraus. Stellvertretend zeigt die Abbildung deren Vertreter QMRCGSTAB$_1$ aus Alg. 6.7. Das Verfahren TFQMR$_1$ aus Alg. 6.5 weist genau wie die anderen hier nicht dargestellten CGS–basierten Verfahren einen ähnlich günstigen Konvergenzverlauf auf. QMR$_1$ mit gekoppelten Zwei-Term-Rekursionen aus Alg. 6.3 zeigt einen zu der parallelen QMR–Variante aus Alg. 6.2 vergleichbaren Konvergenzverlauf. Insbesondere für geringe Genauigkeiten ist das Konvergenzverhalten dieser parallelen Variante günstiger als die der ursprünglichen QMR–Version [36, Alg.7.1].

Um den Fehler $\mathbf{x} - \mathbf{A}^{-1}\mathbf{b}$ zu verkleinern, wird aufgrund der hochfrequenten Lösung ein feineres Gitter benötigt, dessen Daten nicht länger durch einen einzelnen Prozessor gespeichert werden können. Daher wird nun zur Diskretisierung von (7.4) mit gleicher Lösung (7.5) ein $48 \times 48 \times 48$ Gitter mit einer Schrittweite von $1/49$ in jeder Dimension verwendet. Das resultierende Gleichungssystem mit der Ordnung $110\,592$ besitzt $760\,320$ Nichtnull-Elemente und wird mit unterschiedlichen Prozessorzahlen bearbeitet. Als Abbruchkriterium wird weiterhin (7.6) angewendet. Da die bisher beschriebenen Untersuchungen markante Unterschiede im Konvergenzverhalten der ursprünglichen QMR–Version [36] und der parallelen Variante aus Alg. 6.2 zeigen, stellt sich die

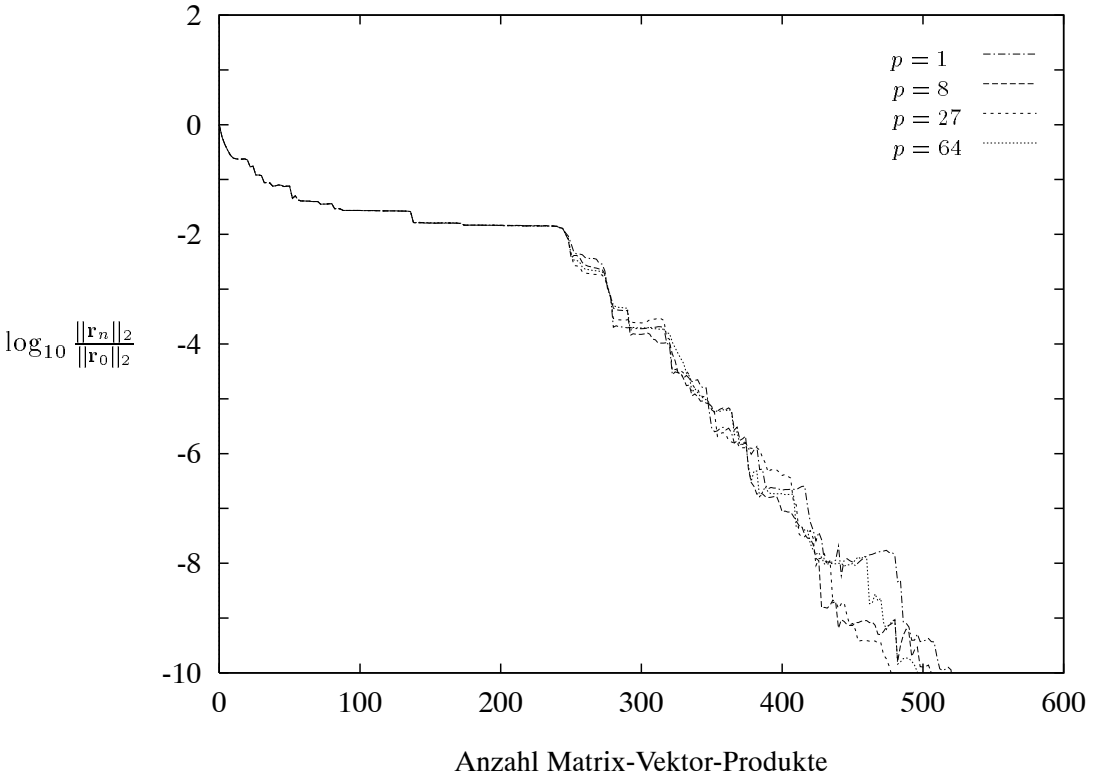


Abbildung 7.6: Konvergenzverläufe aus Exp. 7.3 in Abhängigkeit der Prozessoranzahl p für die die parallele Variante aus Alg. 6.2

Frage, ob sich diese beiden Versionen ebenfalls unterschiedlich verhalten, wenn bei Ausführung auf einem Parallelrechner die Prozessoranzahl p variiert wird. Vorab wird darauf hingewiesen, daß das Konvergenzverhalten für dieses feinere Diskretisierungsgitter ähnlich zu dem Abb. 7.5 zugrundeliegenden gröberen Gitter ist. Die parallele QMR–Varainte besitzt nämlich in diesem Beispiel besonders für geringe Genauigkeiten einen im Vergleich zur originalen QMR–Version günstigeren Konvergenzverlauf. Bei der originalen QMR–Version treten im Konvergenzverhalten nur sehr geringfügige Abweichungen für unterschiedliche Prozessorzahlen p auf, die in der logarithmischen Darstellung kaum wahrnehmbar sind. Daher wird auf eine Darstellung der entsprechenden Kurve in Diagrammform verzichtet. Die Ergebnisse der parallelen Variante zeigt Abb. 7.6. Die parallele QMR–Variante zeigt mit unterschiedlicher Prozessoranzahl ebenfalls nur eine schwache Abweichung im Konvergenzverlauf. Während für Genauigkeiten von $\log_{10}(\|\mathbf{r}_n\|_2/\|\mathbf{r}_0\|_2) > 10^{-2}$ keine Unterschiede in der Residuumsnorm auftreten, sind für höhere Genauigkeiten geringe Abweichungen der Kurven für unterschiedliche Prozessoranzahlen festzustellen. Ab einer Genauigkeit von $\log_{10}(\|\mathbf{r}_n\|_2/\|\mathbf{r}_0\|_2) < 10^{-8}$ sind diese Unterschiede etwas deutlicher.

Wie die vorangehenden numerischen Beispiele belegen, ist das Konvergenzverhalten der Methoden unterschiedlich und außerdem stark von der Wahl des Beispiels abhängig. Um diese Einflußfaktoren von den Untersuchungen hinsichtlich der Skalierbarkeit der Algorithmen auf parallelen Rechnersystemen zu entkoppeln, wird die folgende Vorgehensweise durchgeführt. Indem eine feste auszuführende Iterationsanzahl vorgegeben wird, wird von dem Konvergenzverhalten der unterschiedlichen Methoden abstrahiert. Für die restlichen Ausführungen dieses numerischen Experimentes werden jeweils 100 Iterationen eines Verfahrens betrachtet. Da alle Verfahren aus den gleichen Operationen, nämlich Matrix-Vektor-Produkten, Skalarprodukten und Linearkombinationen aufgebaut sind und deren jeweilige Anzahl in allen betrachteten Methoden vergleichbar

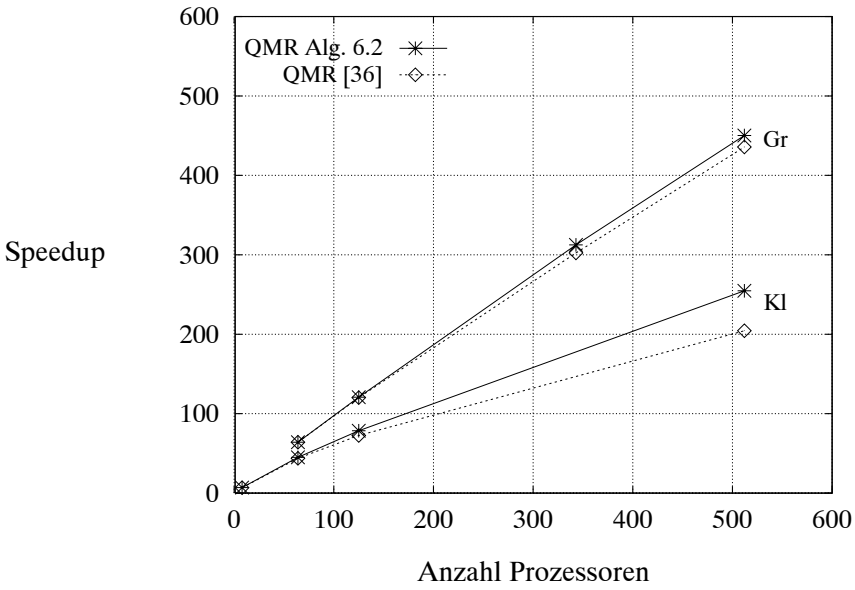


Abbildung 7.7: Speedup der ursprünglichen QMR–Version und der parallelen QMR–Variante für zwei unterschiedlich feine Diskretisierungsgitter (**Gr** und **Kl**) aus Exp. 7.3

ist, zeigen die Verfahren bezogen auf 100 Iterationen ein ähnliches Skalierungsverhalten auf Parallelrechnern. Aus diesem Grunde steht in den folgenden Ausführungen das Verfahren QMR im Mittelpunkt. Dabei wird der Unterschied zwischen der ursprünglichen QMR–Version [36, Alg.7.1] und den parallelen QMR–Varainten aus Alg. 6.2 und [9] betrachtet. In Abb. 7.7 sind die parallelen Ergebnisse der Implementierungen auf CRAY T3E für zwei unterschiedlich feine Diskretisierungsgitter dargestellt. Dabei wird weiterhin die Differentialgleichung (7.1) mit (7.4) und (7.5) betrachtet. Ein $280 \times 280 \times 280$ Gitter mit einer Schrittweite von $1/281$ in jeder Dimension führt zu einem Gleichungssystem mit $21\,952\,000$ Unbekannten und $153\,193\,600$ Nichtnull-Elementen. Da die Speicheranforderungen dieses Gleichungssystems eine Prozessoranzahl von mindestens 64 bedingen, wird der Speedup dieses **Gr**oßen Gleichungssystems in der Abbildung wie folgt berechnet. Der Speedup für 64 Prozessoren wird als ideal angenommen, so daß die Werte für die restlichen Prozessoranzahlen vermöge dieser Setzung berechnet werden. Die dargestellten Kurven sind daher als "optimistische" Annäherungen an die tatsächlichen Kurven zu bewerten. Die Abbildung zeigt die hohe Skalierbarkeit sowohl der ursprünglichen QMR–Version als auch der parallelen Variante aus Alg. 6.2. Beide Algorithmen weisen eine nahezu konstante Effizienz bis zur höchsten Prozessoranzahl von 512 auf.

In einem zweiten Beispiel ist eine gröbere Diskretisierung so gewählt, daß der Speicherplatz eines einzelnen Prozessors zur Berechnung des Gesamtproblems ausreicht. Dabei wird ein $80 \times 80 \times 80$ Gitter mit einer Schrittweite von $1/81$ in jeder Dimension verwendet. Die Ordnung des resultierenden Systems beträgt $512\,000$. Die Koeffizientenmatrix verfügt über $3\,545\,600$ Nichtnull-Elemente. Die Abbildung zeigt für dieses **Kl**einere System niedrigere Werte für den Speedup. Da der Berechnungsaufwand für die Matrix-Vektor-Produkte den Gesamtrechenaufwand einer Iteration nicht mehr so stark dominiert wie im Fall des **Gr**ößeren Gleichungssystems, besitzen die Kommunikationszeiten für die Berechnung der Skalarprodukte einen größeren Anteil an der Ausführungszeit einer Iteration. Daher ist der Unterschied der parallelen Variante, bei der zur Berechnung der Skalarprodukte globale Synchronisation eingespart wird, zu der ursprünglichen Version hier deutlicher als bei dem **Gr**ößeren Gleichungssystem. Dieses Effekt nimmt mit anwachsender Prozessoranzahl zu und zeigt die Überlegenheit der parallelen Variante vorausgesetzt,

daß beide QMR–Algorithmen gleich schnell konvergieren.

Weiterhin sind die Unterschiede zwischen der ursprünglichen Version und den parallelen Varianten um so stärker, je dünnbesetzter die Koeffizientenmatrizen der zu lösenden Gleichungssysteme sind. Dazu wird an Stelle der dreidimensionalen Differentialgleichung (7.1) mit (7.4) und (7.5) nun das aus [9] stammenden zweidimensionale Analogon

$$-\Delta u - 20\left(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y}\right) = f \qquad \text{in} \quad \Omega = (0,1)\times(0,1)$$

betrachtet, wobei die rechte Seite f so gewählt wird, daß für die Lösung der Differentialgleichung

$$u(x,y) = \frac{1}{2}\sin(4\pi x)\sin(6\pi y)$$

gilt. Zur Diskretisierung werden zentrierte Differenzen zweiter Ordnung auf einem 440×440 Gitter mit einer Schrittweite von $1/441$ in jeder Dimension verwendet. Das resultierende lineare Gleichungssystem besitzt die Ordnung $193\,600$ mit $966\,240$ Nichtnull-Elementen. Der Konvergenzverlauf der ursprünglichen QMR–Version [36, Alg.7.1] und der parallelen Variante [9] ist in diesem Beispiel nahezu identisch und ist in [9, Fig. 1] abgebildet. Die parallelen Ergebnisse der Implementierungen auf Intel Paragon XP/S 10 sind in Abb. 7.8 dargestellt. Bei einer vergleichsweise geringen Prozessoranzahl von 121 treten hier bereits markante Unterschiede im Verhalten des Speedup auf, die in der linken Hälfte der Abbildung wiedergegeben sind. Die rechte Seite der Abbildung zeigt den Zeitgewinn der parallelen Variante im Vergleich zur ursprünglichen Version. Genauer stellt diese Abbildung den Wert von $1 - T_{[9]}(p)/T_{[36]}(p)$ dar, wobei $T_{[9]}(p)$ und $T_{[36]}(p)$ die Laufzeiten der Versionen aus [9] und [36, Alg.7.1] auf p Prozessoren bezeichnen. Dabei sei auf die unterschiedlichen Topologien der verwendeten Rechner hingewiesen. Da der Durchmesser einer zweidimensionalen größer als der einer dreidimensionalen Gitterstruktur ist, fallen die Unterschiede der verglichenen QMR–Varianten auf Intel Paragon XP/S 10 deutlicher als auf CRAY T3E aus.

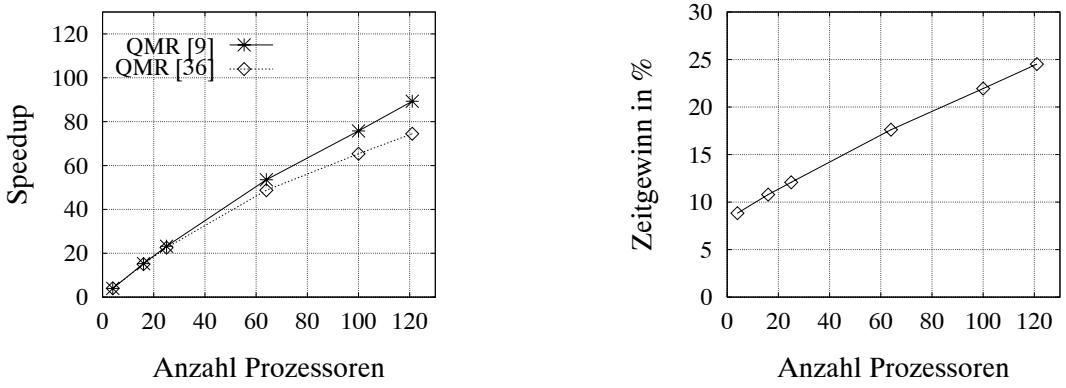


Abbildung 7.8: Speedup und relativer Zeitgewinn der Variante [9] gegenüber der Variante [36] für das zweidimensionale Beispiel aus Exp. 7.3

Experiment 7.4. Dieses Beispiel dient der Veranschaulichung eines grundsätzlichen Unterschiedes zwischen den Verfahren, die das Quasi-Residuum in der euklidischen und in der 1-Norm minimieren. Die in der euklidischen Norm minimierenden Verfahren verfügen nach (6.12) über eine obere Schranke der Form

$$\|\mathbf{r}_n\|_2 \le \sqrt{n+1}\ \min_{\mathbf{z}\in\mathbb{C}^n}\|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n\mathbf{z}\|_2 =: \sqrt{n+1}\,\tau_n,$$

wobei der Wert von τ_n innerhalb dieser Verfahren durch eine einfache Rekursion zur Verfügung steht. Die obere Schranke ist in praktischen Anwendungen ein mächtiges Hilfsmittel, das die aufwendige Operation eines zusätzlichen Matrix-Vektor-Produktes zur Berechnung des tatsächlichen Residuums $\mathbf{b} - \mathbf{A}\mathbf{x}_n$ ersetzt. Während die obere Schranke hier jedoch lediglich eine approximative Beziehung zwischen $\|\mathbf{r}_n\|_2$ und der euklidischen Norm τ_n des Quasi-Residuums liefert, gilt in den 1-Norm-Varianten eine Gleichheit der Form

$$\|\mathbf{r}_n\|_2 = \min_{\mathbf{z}\in\mathbb{C}^n} \|\mathbf{f}_{n+1} - \widehat{\mathbf{H}}_n \mathbf{z}\|_1 =: \widehat{\tau}_n .$$

Dabei wird der Wert $\widehat{\tau}_n$ der 1-Norm des Quasi-Residuums durch den in den Verfahren durchgeführten Minimierungsprozeß erzeugt und ist daher ohne zusätzliche Kosten verfügbar. In allen numerischen Beispielen der vorliegenden Arbeit wird in endlicher Arithmetik kein Unterschied zwischen der tatsächlichen Residuumsnorm $\|\mathbf{b} - \mathbf{A}\mathbf{x}_n\|_2$ und dem Wert $\widehat{\tau}_n$ festgestellt. In praktischen Anwendungen, in denen das tatsächliche Residuum normalerweise nicht in jedem Iterationsschritt berechnet wird, ist daher bei einem Vergleich der in der euklidischen und 1-Norm minimierenden Verfahren der Unterschied zwischen der oberen Schranke $\sqrt{n+1}\ \tau_n$ und dem Wert $\widehat{\tau}_n$ von Bedeutung.

Exemplarisch wird der Unterschied zwischen den in der euklidischen und in der 1-Norm minimierenden Verfahren durch den folgenden Vergleich von TFQMR und TFQMR$_1$ beschrieben. Dazu wird eine Diskretisierung von (7.2) mit Lösung (7.3) mit unterschiedlich feinen Gittern betrachtet. In Abb. 7.9 sind die Ergebnisse für ein resultierendes System der Ordnung $1\,953\,125$ mit $13\,578\,125$ Nichtnull-Elementen dargestellt. Da für TFQMR$_1$ die Kurven von $\widehat{\tau}_n$ und der tatsächlichen Residuumsnorm in der logarithmischen Darstellung zusammenfallen, ist in der Abbildung lediglich die tatsächliche TFQMR$_1$–Residuumsnorm aufgenommen. Um die Rechenzeit zu be-

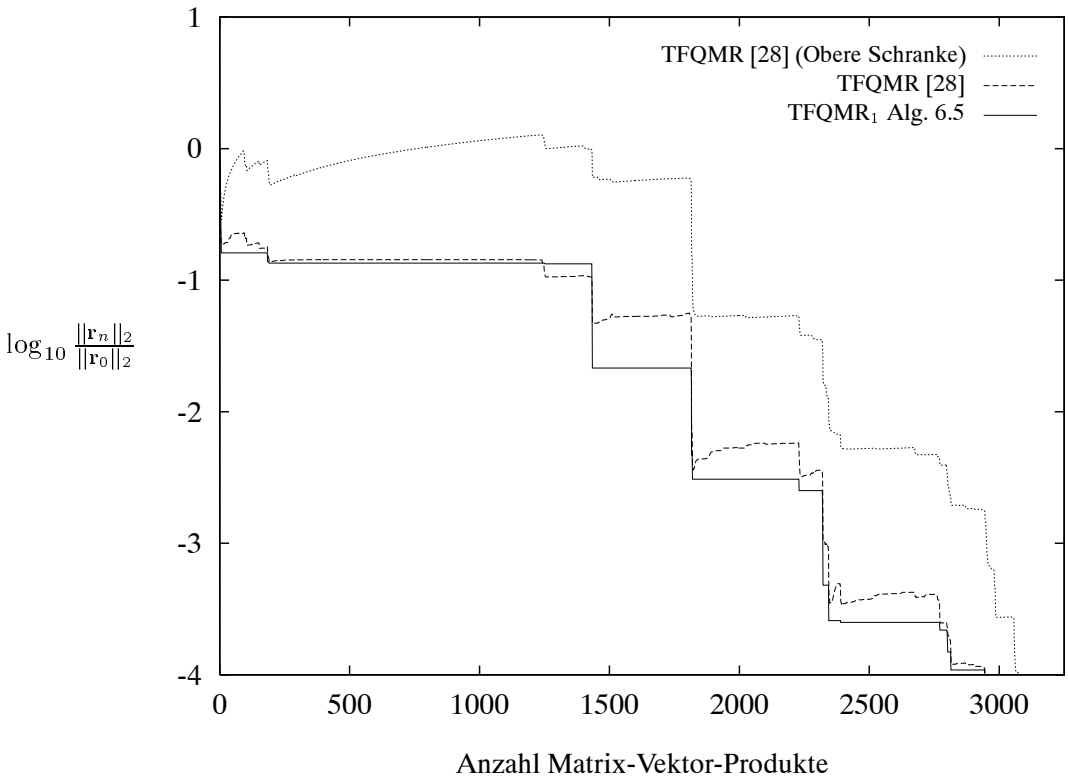


Abbildung 7.9: Vergleich von oberer Schranke und tatsächlichem Konvergenzverlauf aus Exp. 7.4

grenzen, wird als Abbruchkriterium

$$\frac{\|\mathbf{r}_n\|_2}{\|\mathbf{r}_0\|_2} \leq \epsilon = 10^{-4}$$

verwendet. Die Konvergenzverläufe in der tatsächlichen Residuumsnorm sind für beide Methoden ähnlich. Jedoch deutet der Unterschied zur in TFQMR verfügbaren Schranke an, daß TFQMR$_1$ wie nachfolgend beschrieben zu bevorzugen ist.

Um eine praktische Anwendung zu simulieren, wird das Abbruchkriterium unter Beibehaltung von $\epsilon = 10^{-4}$ wie folgt verändert. Für TFQMR wird das tatsächliche Residuum erst dann berechnet, wenn die obere Schranke kleiner oder gleich $\epsilon \cdot \|\mathbf{r}_0\|_2$ ist. In den restlichen Iterationsschritten wird dann ausschließlich die tatsächliche Residuumsnorm für das Abbruchkriterium verwendet. Für TFQMR$_1$ erfolgt der Wechsel zur tatsächlichen Residuumsnorm, sobald $\widehat{\tau}_n \leq \epsilon \cdot \|\mathbf{r}_0\|_2$. Da $\widehat{\tau}$ mit der tatsächlichen Residuumsnorm zusammenfällt, ist in allen Experimenten nur eine einzige Berechnung des tatsächlichen Residuums notwendig. Die Ergebnisse dieser Untersuchung sind in Tab. 7.2 zusammengefaßt. Jede Spalte der Tabelle entspricht einer Diskretisierung einer speziellen Schrittweite, die zu einem resultierenden Gleichungssystem führt, dessen Ordnung und Anzahl von Nichtnull-Elementen angegeben sind. Die letzte Zeile zeigt, daß TFQMR$_1$ eine signifikante Anzahl an Matrix-Vektor-Produkten gegenüber TFQMR einspart.

Tabelle 7.2: Vergleich zwischen TFQMR und TFQMR$_1$ für vier unterschiedliche Diskretisierungsgitter aus Exp. 7.4. Die letzte Zeile zeigt die Anzahl der von TFQMR$_1$ im Vergleich zu TFQMR eingesparten Matrix-Vektor-Produkte.

Gleichungssystem	*A*	*B*	*C*	*D*
Ordnung des Systems	421 857	1 000 000	1 953 125	3 375 000
Anzahl Nichtnull-Elemente	2 919 375	6 940 000	13 578 125	23 490 000
Matrix-Vektor-Produkte TFQMR	1 180	3 020	3 086	2 274
Matrix-Vektor-Produkte TFQMR$_1$	1 058	2 344	2 946	2 210
Einsparung Matrix-Vektor-Produkte	122	676	140	64

Spalte *C* dieser Tabelle entspricht dem in Abb. 7.9 abgebildeten Experiment. Die in der Tabelle aufgeführten 140 eingesparten Matrix-Vektor-Produkte, die TFQMR$_1$ weniger als TFQMR benötigt, sind in Abb. 7.9 bei $\epsilon = 10^{-4}$ wiederzufinden. Dieser Abbildung entnimmt man außerdem, daß der Unterschied zwischen den in diesen beiden Verfahren verfügbaren Schranken bei $\epsilon = 10^{-4}$ besonders klein ist. Bei anderen Genauigkeiten werden wesentlich größere Einsparungen erzielt. Um beispielsweise eine Genauigleit von $\epsilon = 4 \cdot 10^{-3}$ zu erreichen, benötigt TFQMR 2772 Matrix-Vektor-Produkte während TFQMR$_1$ dafür 953 weniger, nämlich lediglich 1819 benötigt.

Experiment 7.5. Abschließend werden Ergebnisse aus einem Beispiel aus der numerischen Simulation in der Halbleitertechnik beschrieben.* Die physikalische Problemstellung der Dotierung sowie die in dieser speziellen Simulation angewendeten numerischen Lösungverfahren sind in Kapitel 1 skizziert und werden ausführlich in [49, 57] beschrieben. Das Beispiel, dessen Ergebnisse hier beschrieben werden, simuliert die gleichzeitige Dotierung eines Siliziumsubstrats mit den drei Dotierstoffen Bor, Phosphor und Arsen. Die geometrische Struktur dieses Halbleitersubstrats ist in [49, Fig. 4] dargestellt. Bei der Diskretisierung innerhalb eines einzelnen Zeitschrittes

*Diese Untersuchungen wurden von Dr. M. Paffrath, Zentralabteilung Technik, SIEMENS AG, durchgeführt.

Tabelle 7.3: Gesamtrechenzeiten in Minuten aus Exp. 7.5 für unterschiedliche Prozessoranzahlen p

# FE-Knoten	$p = 1$	$p = 2$	$p = 4$
1 845	55	—	—
3 615	106	61	—
7 230	194	106	60

werden drei unterschiedlich feine Finite-Elemente-Gitter betrachtet. Dabei werden die Feinheiten von einem zum nächsten Gitter jeweils ungefähr verdoppelt. Die daraus resultierenden linearen Gleichungssysteme unterschiedlicher Größe werden in der Simulation ohne Vorkonditionierung mit der parallelen QMR–Variante [9] gelöst. Als Abbruchkrierium wird $\|\mathbf{r}_n\|_2/\|\mathbf{r}_0\|_2 < 10^{-4}$ verwendet. Die Simulation wird auf einem Netz aus vier SUN4–IPX Workstations durchgeführt, wobei die Kommunikation über Ethernet mit PVM durchgeführt wird.

In Tab. 7.3 sind die Rechenzeiten in Minuten der gesamten Simulation — also nicht nur der Lösung der linearen Gleichungssysteme — enthalten. Jede Spalte dieser Tabelle korrespondiert zu einem der drei unterschiedlich feinen Finite-Elemente-Gitter. Die Tabelle zeigt für ein festes Finite-Elemente-Gitter, dessen Anzahl von Knoten in der ersten Spalte angegeben ist, die Rechenzeiten mit zunehmender Porzessorzahl p. Dabei ist zu beachten, daß das kleinste Problem nur mit einem Prozessor und das mittlere Problem nur mit einem und zwei Prozessoren gerechnet wird. Dieses Beispiel dient nicht einem Vergleich zwischen der parallelen Variante und der ursprünglichen QMR–Version, sondern zeigt die Anwendbarkeit der entwickelten parallelen Variante in einem realen Beispiel der industriellen Fertigung.

Kapitel 8

Zusammenfassung und Ausblick

Die numerische Simulation umfaßt die mathematische Modellierung von Vorgängen aus Wissenschaften und deren technischen Anwendungen, den Entwurf von Algorithmen zur Lösung der darin enthaltenen Teilaufgaben und berücksichtigt dabei die zu verwendende Rechnerarchitektur. Werden komplexe Problemstellungen simuliert, erfordert dies den konzertierten Einsatz von ausgefeilten Methoden der Mathematik, Informatik und Informationstechnik, Natur- und Ingenieurwissenschaften. Eine der am häufigsten zu bewältigenden Aufgabenstellungen der numerischen Simulation, die zudem nicht selten den Gesamtrechenaufwand dominiert, ist die Lösung von linearen Gleichungssystemen. Oft sind die Koeffizientenmatrizen dieser Systeme sehr groß und außerdem dünnbesetzt, so daß direkte Verfahren wie beispielsweise die Gauß–Elimination aufgrund ihres hohen Speicher- und Rechenzeitbedarfs nicht eingesetzt werden können. Einen Ausweg bieten in diesen Fällen iterative Methoden, deren wohl mächtigste Vertreter in die Klasse der Krylov–Teilraumverfahren fallen. Wünschenswert sind Methoden sowohl mit kurzen Rekursionen als auch mit optimalem Konvergenzverhalten. Für Gleichungssysteme mit allgemeiner nicht-hermitescher Koeffizientenmatrix ist jedoch bekannt, daß man auf eine dieser beiden Eigenschaften verzichten muß. Unter praktischen Gesichtspunkten überwiegt die Forderung nach moderatem Speicher- und Rechenzeitbedarf pro Iterationsschritt, die sich in kurzen Rekursionen manifestiert. Da solche Methoden dann zwangsläufig über suboptimale Konvergenzeigenschaften verfügen, gibt es für jedes iterative Verfahren mit kurzen Rekursionen Anwendungsfälle, in denen die Konvergenzgeschwindigkeit nur gering ist. Aus dieser Tatsache resultiert die Notwendigkeit, eine Fülle von unterschiedlichen iterativen Verfahren zur Verfügung zu stellen. Je mehr iterative Verfahren mit kurzen Rekursionen bekannt sind, desto größer ist die Chance, für ein gegebenes Problem eine effiziente Methode mit gutem Konvergenzverhalten zu finden. Durch die Bereitstellung neuer iterativer Methoden mit kurzen Rekursionen leistet die vorliegende Arbeit dazu einen Beitrag.

Der Entwurf dieser neuen Iterationsverfahren erfolgt zielgerichtet auf ihre spätere Ausführung auf massiv-parallelen Rechnersystemen mit verteiltem Speicher hin. Die Konzentration beim Algorithmenentwurf auf Aspekte der Parallelverarbeitung ist im skizzierten Umfeld der numerischen Simulation von zentraler Bedeutung, da sich gerade dort die Anforderungen an Rechenzeit und Speicherplatz stets an der oberen Grenze der Rechnertechnologie orientieren und daher modernste Parallelrechner eine “natürliche” Rechnerarchitektur darstellen. Der Entwurf paralleler iterativer Verfahren macht Gebrauch von der klaren Trennung der zwei grundlegenden Ingredienzien, aus denen ein Krylov–Teilraumverfahren aufgebaut ist:

- Die Konstruktion einer Basis der zugrundeliegenden Krylov–Teilräume und
- die Festlegung eines darin enthaltenen Vektors als aktuelle Iterierte.

Da zu letztgenanntem Punkt in allen in dieser Arbeit betrachteten iterativen Methoden lediglich solche Operationsarten verwendet werden, die auf Parallelrechnern mit verteiltem Speicher oh-

ne Kommunikation der Prozessoren untereinander implementierbar sind, werden die parallelen Eigenschaften von Krylov–Teilraumverfahren ausschließlich durch den erstgenannten Punkt bestimmt. Die parallelen Gesichtspunkte, die beim Entwurf der neuen Krylov–Teilraumverfahren zu berücksichtigen sind, werden so auf die Verfahren zur Erzeugung einer Basis der zugrundeliegenden Krylov–Teilräume zurückgeführt. Diese Kapselung der parallelen Aspekte erlaubt eine Reduzierung des parallelen Algorithmenentwurfs auf das Aufspannen der Vektorräume. Durch die Betrachtung der zugrundeliegenden Vektorräume gelingt der Entwurf neuer paralleler Varianten des nicht-hermiteschen Lanczos–Algorithmus, mit dem eine Basis der Krylov–Teilräume erzeugt wird. In diesen parallelen Varianten des Lanczos–Algorithmus sind alle Operationen, die globale Synchronisation erfordern und damit auf massiv-parallelen Rechnersystemen den bei weitem wichtigsten limitierenden Faktor darstellen, innerhalb eines einzelnen Iterationsschrittes unabhängig voneinander. Aufbauend auf diesen Erkenntnissen werden neue parallele Versionen der Methode der quasi-minimalen Residuen (QMR) und der bikonjugierten Gradienten (BCG) zur iterativen Lösung von linearen Gleichungssystemen hergeleitet. In diesen beiden Verfahren wird der Lanczos–Algorithmus jeweils als zugrundeliegende Technik zur Erzeugung der Krylov–Teilräume verwendet.

Zur Festlegung der aktuellen Iterierten in einem Krylov–Teilraumverfahren wird in der vorliegenden Arbeit ein neuer Ansatz eingeführt, der wie folgt einsetzbar ist. Gegeben sei ein beliebiges Verfahren zur Erzeugung von Krylov–Teilräumen bezüglich der Koeffizientenmatrix $\mathbf{A}$ des zu lösenden Gleichungssystems, wobei die ersten n erzeugten Basisvektoren als Spaltenvektoren einer Matrix $\mathbf{V}_n$ aufgefaßt werden. Genügt dieses Verfahren einer Matrixgleichung der Form $\mathbf{A}\mathbf{V}_{n+1} = \mathbf{V}_n\mathbf{T}_n$ mit einer tridiagonalen Matrix $\mathbf{T}_n$, ist der 1-Norm-Ansatz der quasi-minimalen Residuen zur Herleitung iterativer Verfahren anwendbar. Daß diese Annahme an den zugrundeliegenden Prozeß zur Erzeugung der Krylov–Teilräume keinesfalls restriktiv ist, zeigt die vorliegende Arbeit durch die Verwendung sowohl des Lanczos–Algorithmus in zwei unterschiedlichen Formen, nämlich basierend auf Drei-Term-Rekursionen und gekoppelten Zwei-Term-Rekursionen, als auch von CGS und Bi-CGSTAB. Diese vier Algorithmen zur Erzeugung von Krylov–Teilräumen erfüllen obige Matrixgleichung. Der 1-Norm-Ansatz der quasi-minimalen Residuen definiert die aktuelle Iterierte $\mathbf{x}_n$ durch 1-Norm-Minimierung des zweiten Faktors in der Darstellung des Residuums $\mathbf{r}_n = \mathbf{V}_{n+1}\mathbf{s}_n$ und führt in den resultierenden iterativen Verfahren zu kurzen Rekursionen. Daß die resultierenden Methoden auf kurzen Rekursionen basieren, ist gleichzeitig notwendig und überraschend. Kurze Rekursionen sind einerseits notwendig, weil dieser Ansatz auf der Minimierung lediglich eines Faktors des Residuums basiert. Würde man dazu lange Rekursionen benötigen, wäre es sinnvoller, den dann hohen Aufwand an Speicher und Rechenzeit in die Minimierung des gesamten Residuums zu investieren und so Verfahren mit optimalem Konvergenzverhalten herzuleiten. Andererseits ist das Resultat der kurzen Rekursionen für diesen Ansatz auch überraschend, weil dazu eine Folge von 1-Norm-Minimierungsproblemen effizient gelöst werden muß und dabei nicht auf Standardtechniken wie im euklidischen Fall zurückgegriffen werden kann.

Damit entwickelt die vorliegende Arbeit neue Ideen sowohl bei der Erzeugung der Krylov–Teilräume, nämlich die Bereitstellung paralleler Varianten des Lanczos–Algorithmus, als auch bei der Festlegung der Iterierten, genauer den (seriellen) 1-Norm-Ansatz der quasi-minimalen Residuen. Zudem lassen sich diese beiden Punkte mit in der Literatur bekannten Techniken zur Herleitung von Krylov–Teilraumverfahren kombinieren. Dies erlaubt den Entwurf einer Fülle von Algorithmen zur iterativen Lösung linearer Gleichungssysteme. Neben den bereits erwähnten parallelen Varianten von QMR und BCG werden die folgenden 1-Norm-Verfahren eingeführt. QMR_1 wird in zwei Implementierungen vorgestellt, die auf dem Lanczos–Algorithmus mit Drei-Term-Rekursionen und mit gekoppelten Zwei-Term-Rekursionen beruhen. TFQMR_1 verwendet zum Aufspannen der zugrundeliegenden Krylov–Teilräume die Methode der quadrierten konjugierten

Gradienten (CGS) und ist daher eng verwandt mit der Methode der transpositionsfreien quasi-minimalen Residuen (TFQMR). In dem Verfahren QMRCGSTAB$_1$ werden die zugrundeliegenden Krylov–Teilräume durch die Methode der bikonjugierten stabilisierten Gradienten (Bi-CGSTAB) aufgespannt, wodurch eine enge Beziehung zu QMRCGSTAB entsteht. Die Iterierten dieser 1-Norm-Versionen lassen sich als jeweils beste Galerkin–ähnliche Approximation an die exakte Lösung interpretieren. Anhand von numerischen Experimenten wird verdeutlicht, daß die 1-Norm-Verfahren einen ähnlichen Konvergenzverlauf wie die entsprechenden, in der euklidischen Norm quasi-minimierenden Verfahren besitzen. In realen Anwendungen der Praxis wie beispielsweise bei der Diskretisierung von partiellen Differentialgleichungen liegt ihr Vorteil in der expliziten Verfügbarkeit der Norm des Residuums zu jedem Zeitpunkt des Iterationsverfahrens. Die durchgeführten numerischen Experimente belegen, daß diese Eigenschaft zu signifikanten Einsparungen an Matrix-Vektor-Produkten und damit der bei weitem rechenintensivsten Operation eines Krylov–Teilraumverfahrens führen kann. Beispielsweise treten Fälle auf, in denen die in der euklisichen Norm minimierenden Verfahren bis zu 50% mehr Matrix-Vektor-Produkte als die hier entwickelten 1-Norm-Verfahren benötigen, um eine vorgegebene Genauigkeit zu erreichen.

Im Unterschied zu dem weitgehend übereinstimmenden Konvergenzverhalten der 1-Norm-Verfahren und ihren euklidischen Entsprechungen zeigt ein Vergleich der parallelen Varianten mit den Algorithmen in ihrer ursprünglichen Version in den vorgestellten numerischen Beispielen markante Unterschiede, sowohl im positiven als auch im negativen Sinne. Die praktische Anwendbarkeit der mit dieser Arbeit bereitgestellten Techniken wird in einer speziellen industriellen Anwendung aus der Halbleitersimulation nachgewiesen. Wie weitere durchgeführte Experimente zeigen, ist in anderen Fällen jeweils eine Analyse des Konvergenzverhaltens erforderlich. Konvergieren jedoch die parallelen Varianten genauso schnell wie die ursprünglichen Versionen, so werden die ohnehin bereits guten Parallelisierungseigenschaften der ursprünglichen Versionen von denen der parallelen Varianten noch übertroffen. Dabei nimmt der Vorteil, den die parallelen Varianten besitzen, mit zunehmender Prozessoranzahl deutlich zu, so daß die entwickelten parallelen Varianten besonders für hohe Prozessoranzahlen zu bevorzugen sind.

Die durchgeführten Experimente weisen darauf hin, daß die Unterschiede im Konvergenzverhalten zwischen den parallelen QMR–Varianten und der ursprünglichen QMR–Version allein auf die verschiedenen Varianten des zugrundeliegenden Lanczos–Algorithmus zurückzuführen sind. Eine an diese Arbeit anknüpfende Untersuchung soll sich deshalb mit der Fragestellung befassen, ob diese Unterschiede im Konvergenzverhalten eliminiert werden können, indem Look-Ahead-Techniken in den zugrundeliegenden Lanczos–Algorithmus integriert werden. Wäre dies der Fall, so würden die parallelen Varianten für massiv-parallele Rechnersysteme mehr als nur eine Alternative zu den ursprünglichen Versionen darstellen. Einen weiteren zu untersuchenden Punkt bilden Vorkonditionierungstechniken.

Literaturverzeichnis

[1] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge, 1994.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation*. Prentice-Hall, Englewood Cliffs, 1989.

[3] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

[4] C. Brezinski and M. Redivo-Zaglia. Treatment of Near-Breakdown in the CGS Algorithm. *Numerical Algorithms*, 7:33–73, 1994.

[5] C. Brezinski and H. Sadok. Avoiding Breakdown in the CGS Algorithm. *Numerical Algorithms*, 1:199–206, 1991.

[6] C. Brezinski, M. R. Zaglia, and H. Sadok. A Breakdown-Free Lanczos Type Algorithm for Solving Linear Systems. *Numerische Mathematik*, 63(1):29–38, 1992.

[7] P. N. Brown. A Theoretical Comparison of the Arnoldi and GMRES Algorithms. *SIAM Journal on Scientific and Statistical Computing*, 12(1):58–78, 1991.

[8] H. M. Bücker. Isoefficiency Analysis of Parallel QMR–Like Iterative Methods and its Implications on Parallel Algorithm Design. In K. Ecker, J. Apsel, and T. Ripke, editors, *Fourteenth Workshop on Parallel Processing, Lessach, Austria, September 25–29, 1995*, Institutsberichte des Instituts für Informatik, Informatik-Bericht 96/1, pages 28–49. Technische Universität Clausthal, Clausthal-Zellerfeld, 1996.

[9] H. M. Bücker and M. Sauren. A Parallel Version of the Quasi-Minimal Residual Method Based on Coupled Two-Term Recurrences. In J. Waśniewski, J. Dongarra, K. Madsen, and D. Olesen, editors, *Applied Parallel Computing: Industrial Computation and Optimization, Proceedings of the Third International Workshop, PARA '96, Lyngby, Denmark, August 18–21, 1996*, volume 1184 of *Lecture Notes in Computer Science*, pages 157–165, Berlin, 1996. Springer.

[10] H. M. Bücker and M. Sauren. A Parallel Version of the Unsymmetric Lanczos Algorithm and its Application to QMR. Internal Report KFA–ZAM–IB–9605, Research Centre Jülich, Jülich, Germany, March 1996.

[11] H. M. Bücker and M. Sauren. A Variant of the Biconjugate Gradient Method Suitable for Massively Parallel Computing. In G. Bilardi, A. Ferreira, R. Lüling, and J. Rolim, editors, *Solving Irregularly Structured Problems in Parallel, Proceedings of the Fourth International Symposium, IRREGULAR'97, Paderborn, Germany, June 12–13, 1997*, volume 1253 of *Lecture Notes in Computer Science*, pages 72–79, Berlin, 1997. Springer. Also available in extended form as Internal Report KFA–ZAM–IB–9702, Research Centre Jülich, Jülich, Germany, April 1997.

[12] R. Bulirsch, A. Gilg, K. Merten, and K. Steger. Numerische Simulation in der Halbleiterindustrie. *Informatik Forschung und Entwicklung*, 5(1):42–56, 1990.

[13] T. F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, and C. H. Tong. A Quasi-Minimal Residual Variant of the Bi-CGSTAB Algorithm for Nonsymmetric Systems. *SIAM Journal on Scientific Computing*, 15(2):338–347, 1994.

[14] E. W. Cheney. *Introduction to Approximation Theory*. International Series in Pure and Applied Mathematics. McGraw-Hill, New York, 1966.

[15] T. F. Coleman and Y. Li. A Globally and Quadratically Convergent Affine Scaling Method for Linear L_1 Problems. *Mathematical Programming*, 56:189–222, 1992.

[16] J. Cullum and R. A. Willoughby. A Practical Procedure for Computing Eigenvalues of Large Sparse Nonsymmetric Matrices. In J. Cullum and R. A. Willoughby, editors, *Large Scale Eigenvalue Problems, Proceedings of the IBM Europe Institute Workshop on Large Scale Eigenvalue Problems, Oberlech, Austria, July 8–12, 1985*, number 127 in North-Holland Mathematics Studies, pages 193–240, Amsterdam, The Netherlands, 1986. North-Holland.

[17] J. K. Cullum and A. Greenbaum. Relations between Galerkin and Norm-Minimizing Iterative Methods for Solving Linear Systems. *SIAM Journal on Matrix Analysis and Applications*, 17(2):223–247, 1996.

[18] E. Dekel, D. Nassimi, and S. Sahni. Parallel Matrix and Graph Algorithms. *SIAM Journal on Computing*, 10(4):657–675, 1981.

[19] J. W. Demmel. Trading Off Parallelism and Numerical Stability. In M. S. Moonen, G. H. Golub, and B. L. R. De Moor, editors, *Linear Algebra for Large Scale and Real-Time Applications*, volume 232 of *NATO ASI Series E: Applied Sciences*, pages 49–68. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993. Proceedings of the NATO Advanced Study Institute on Linear Algebra for Large Scale and Real-Time Applications, Leuven, Belgium, August 1992.

[20] J. W. Demmel, M. T. Heath, and H. A. van der Vorst. Parallel Numerical Linear Algebra. In *Acta Numerica 1993*, pages 111–197. Cambridge University Press, Cambridge, 1993.

[21] V. Faber and T. Manteuffel. Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method. *SIAM Journal on Numerical Analysis*, 21(2):352–362, 1984.

[22] R. W. Farebrother. *Linear Least Squares Computations*. Marcel Dekker, New York, 1987.

[23] B. Fischer. *Polynomial Based Iteration Methods for Symmetric Linear Systems*. Advances in Numerical Mathematics. Wiley and Teubner, Chichester, 1996.

[24] R. Fletcher. Conjugate Gradient Methods for Indefinite Systems. In G. A. Watson, editor, *Numerical Analysis Dundee 1975*, volume 506 of *Lecture Notes in Mathematics*, pages 73–89, Berlin, 1976. Springer.

[25] R. W. Freund. Conjugate Gradient-Type Methods for Linear Systems with Complex Symmetric Coefficient Matrices. *SIAM Journal on Scientific and Statistical Computing*, 13(1):425–448, 1992.

[26] R. W. Freund. Quasi-Kernel Polynomials and Convergence Results for Quasi-Minimal Residual Iterations. In D. Braess and L. L. Shumaker, editors, *Numerical Methods of Approximation Theory*, volume 9, pages 77–95. Birkhäuser, Basel, 1992.

[27] R. W. Freund. Quasi-Kernel Polynomials and their Use in Non-Hermitian Matrix Iterations. *Journal of Computational and Applied Mathematics*, 43:135–158, 1992.

[28] R. W. Freund. A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems. *SIAM Journal on Scientific Computing*, 14(2):470–482, 1993.

[29] R. W. Freund. The Look-Ahead Lanczos Process for Large Nonsymmetric Matrices and Related Algorithms. In M. S. Moonen, G. H. Golub, and B. L. R. De Moor, editors, *Linear Algebra for Large Scale and Real-Time Applications*, volume 232 of *NATO ASI Series E: Applied Sciences*, pages 137–163. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993. Proceedings of the NATO Advanced Study Institute on Linear Algebra for Large Scale and Real-Time Applications, Leuven, Belgium, August 1992.

[30] R. W. Freund. The Look-Ahead Lanczos Process for Nonsymmetric Matrices and its Applications. In J. D. Brown, M. T. Chu, D. C. Ellison, and R. J. Plemmons, editors, *Proceedings of the Cornelius Lanczos International Centenary Conference, Raleigh, NC, December 12–17, 1993*, pages 33–47, Philadelphia, PA, 1994. SIAM.

[31] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative Solution of Linear Systems. In *Acta Numerica 1992*, pages 1–44. Cambridge University Press, Cambridge, 1992.

[32] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal. An Implementation of the Look-Ahead Lanczos Algorithm for Non-Hermitian Matrices, Part I. RIACS Technical Report 90.45, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, November 1990.

[33] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal. An Implementation of the Look-Ahead Lanczos Algorithm for Non-Hermitian Matrices. *SIAM Journal on Scientific Computing*, 14(1):137–158, 1993.

[34] R. W. Freund and N. M. Nachtigal. An Implementation of the Look-Ahead Lanczos Algorithm for Non-Hermitian Matrices, Part II. RIACS Technical Report 90.46, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, November 1990.

[35] R. W. Freund and N. M. Nachtigal. QMR: A Quasi-Minimal Residual Method for Non-Hermitian Linear Systems. *Numerische Mathematik*, 60(3):315–339, 1991.

[36] R. W. Freund and N. M. Nachtigal. An Implementation of the QMR Method Based on Coupled Two-Term Recurrences. *SIAM Journal on Scientific Computing*, 15(2):313–337, 1994.

[37] R. W. Freund and T. Szeto. A Transpose-Free Quasi-Minimal Residual Squared Algorithm for Non-Hermitian Linear Systems. In R. Vichnevetsky, D. Knight, and G. Richter, editors, *Advances in Computer Methods for Partial Differential Equations VII*, pages 258–264. IMACS, 1992.

[38] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, third edition, 1996.

[39] G. H. Golub and H. A. van der Vorst. Closer to the Solution: Iterative Linear Solvers. Technical Reports of the Scientific Computing/Computational Mathematics Program SCCM–96–17, Stanford University, Stanford, 1996. To appear in I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, Oxford University Press, 1997.

[40] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, PA, 1997.

[41] A. Gupta, V. Kumar, and A. Sameh. Performance and Scalability of Preconditioned Conjugate Gradient Methods on Parallel Computers. Technical Report TR 92–64, Department of Computer Science, University of Minnesota, Minneapolis, MN–55455, November 1992. Revised April 1994.

[42] M. H. Gutknecht. A Completed Theory of the Unsymmetric Lanczos Process and Related Algorithms, Part I. *SIAM Journal on Matrix Analysis and Applications*, 13(2):594–639, 1992.

[43] M. H. Gutknecht. A Completed Theory of the Unsymmetric Lanczos Process and Related Algorithms, Part II. *SIAM Journal on Matrix Analysis and Applications*, 15(1):15–58, 1994.

[44] M. H. Gutknecht. Solving Linear Systems with the Lanczos Process. In *Acta Numerica 1997*, pages 271–398. Cambridge University Press, Cambridge, 1997.

[45] M. T. Heath, E. Ng, and B. W. Peyton. Parallel Algorithms for Sparse Linear Systems. *SIAM Review*, 33(3):420–460, 1991.

[46] M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[47] F. Hoßfeld. Partielle Differentialgleichungen: Die permanente Herausforderung. In W. E. Nagel, editor, *Partielle Differentialgleichungen, Numerik und Anwendungen*, volume 18 of *Konferenzen des Forschungszentrums Jülich*, pages 1–9. Forschungszentrum Jülich, Jülich, 1996.

[48] T. Huckle. Low-Rank Modification of the Unsymmetric Lanczos Algorithm. *Mathematics of Computation*, 64(212):1577–1588, 1995.

[49] M. Kahlert, M. Paffrath, and U. Wever. Grid Partitioning versus Domain Decomposition: A Comparison of Some Industrial Problems on Workstation Clusters. *Surveys on Mathematics for Industry*, 6:133–166, 1996.

[50] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin/Cummings, Redwood City, 1994.

[51] C. Lanczos. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.

[52] C. Lanczos. Solutions of Systems of Linear Equations by Minimized Iterations. *Journal of Research of the National Bureau of Standards*, 49(1):33–53, 1952.

[53] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, 1974.

[54] Y. Li. A Globally Convergent Method for l_p Problems. *SIAM Journal on Optimization*, 3(3):609–629, 1993.

[55] G. Manzini. Sparse Matrix Vector Multiplication on Distributed Architectures: Lower Bounds and Average Complexity Results. *Information Processing Letters*, 50(5):231–238, 1994.

[56] O. A. McBryan and E. F. Van de Velde. Hypercube Algorithms and Implementations. *SIAM Journal on Scientific and Statistical Computing*, 8(2):s227–s287, 1987.

[57] M. Paffrath, W. Jacobs, W. Klein, E. Rank, K. Steger, U. Weinert, and U. Wever. Concepts and Algorithms in Process Simulation. *Surveys on Mathematics for Industry*, 3:149–183, 1993.

[58] C. C. Paige and M. A. Saunders. Solution of Sparse Indefinite Systems of Linear Equations. *SIAM Journal on Numerical Analysis*, 12:617–629, 1975.

[59] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, 1980.

[60] B. N. Parlett. Reduction to Tridiagonal Form and Minimal Realizations. *SIAM Journal on Matrix Analysis and Applications*, 13(2):567–593, 1992.

[61] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A Look-Ahead Lanczos Algorithm for Unsymmetric Matrices. *Mathematics of Computation*, 44(169):105–124, 1985.

[62] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, 1996.

[63] Y. Saad and M. H. Schulz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[64] M. Sauren. Kommunikationsminimale Algorithmen zur Lösung großer dünnbesetzter linearer Gleichungssysteme auf massiv-parallelen Systemen. Bericht Jül–3396, Forschungszentrum Jülich, Jülich, Juni 1997.

[65] M. Sauren and H. M. Bücker. On Deriving the Quasi-Minimal Residual Method. Internal Report KFA–ZAM–IB–9608, Research Centre Jülich, Jülich, Germany, April 1996. Accepted for publication in *SIAM Review*.

[66] P. Sonneveld. CGS, a Fast Lanczos-Type Solver for Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 10(1):36–52, 1989.

[67] H. A. van der Vorst. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[68] H. A. van der Vorst. Parallel Iterative Solution Methods for Linear Systems arising from Discretized PDE's. In *Special Course on Parallel Computing in CFD*, number R–807 in AGARD Reports, pages 3–1–3–39. AGARD, Neuilly-sur-Seine, France, 1995. Paper presented in an AGARD-FDP-VKI Special Course, held at the VKI, Rhode-Saint-Genese, Belgium, from 15–19 May and 16–20 October at NASA Ames, USA.

[69] G. A. Watson. *Approximation Theory and Numerical Methods*. John Wiley & Sons, Chichester, 1980.

[70] R. Weiss. *Parameter-Free Iterative Linear Solvers*, volume 97 of *Mathematical Research*. Akademie Verlag, Berlin, 1996.

[71] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965.

[72] Q. Ye. A Breakdown-Free Variation of the Nonsymmetric Lanczos Algorithms. *Mathematics of Computation*, 62(205):179–207, 1994.

[73] Y. Zhang. Primal-Dual Interior Point Approach for Computing the l_1-Solutions and l_∞-Solutions of Overdetermined Systems. *Journal of Optimization Theory and Applications*, 77(2):323–341, 1993.

Liste iterativer Methoden

BCG	Biconjugate Gradient [24, 52]
Bi-CGSTAB	Biconjugate Gradient Stabilized [67]
CG	Conjugate Gradient [46]
CGS	Conjugate Gradient Squared [66]
GMRES	Generalized Minimal Residual [63]
MINRES	Minimal Residual [58]
QMR (Drei-Term-Rekursionen)	Quasi-Minimal Residual [35]
QMR_1 (Drei-Term-Rekursionen)	1-Norm Quasi-Minimal Residual, Alg. 6.1
QMR (Zwei-Term-Rekursionen)	Quasi-Minimal Residual [36]
QMR_1 (Zwei-Term-Rekursionen)	1-Norm Quasi-Minimal Residual, Alg. 6.3
QMRCGSTAB	QMR variant of Bi-CGSTAB [13]
$QMRCGSTAB_1$	1-Norm QMR variant of Bi-CGSTAB, Alg. 6.7
TFQMR	Transpose-Free Quasi-Minimal Residual [28]
$TFQMR_1$	1-Norm Transpose-Free Quasi-Minimal Residual, Alg. 6.5

Algorithmenverzeichnis

Abbildungsverzeichnis