Projektionsverfahren zur Approximation von Matrixfunktionen mit Anwendungen auf die Implementierung exponentieller Integratoren

I n a u g u r a l - D i s s e r t a t i o n

zur

Erlangung des Doktorgrades der

Mathematisch-Naturwissenschaftlichen Fakultät

der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Jörg Niehoff

aus Bocholt

Dezember 2006

Gedruckt mit der Genehmigung der

Mathematisch-Naturwissenschaftlichen Fakultät der

Heinrich-Heine-Universität Düsseldorf

Referentin: Prof. Dr. M. Hochbruck

Koreferent: Prof. Dr. A. Frommer

Tag der mündlichen Prüfung: 30.01.2007

Projektionsverfahren zur Approximation von Matrixfunktionen mit Anwendungen auf die Implementierung exponentieller Integratoren

Jörg Niehoff

Zusammenfassung

In dieser Arbeit werden Approximationen an Produkte von Matrixfunktionen mit Vektoren,

$$z = f(A)b$$

mit Krylovverfahren betrachtet, wobei $A \in \mathbb{C}^{N,N}$ eine große, dünn besetzte Matrix ist, $b \in \mathbb{C}^N$ ein Vektor und f eine Funktion. Motiviert wird diese Betrachtung dadurch, dass die effiziente Berechnung dieser Approximationen wesentlich für die Implementierung exponentieller Integratoren ist, wie sie zur Zeit intensiv in der wissenschaftlichen Literatur diskutiert werden.

Als Erstes wird ein Restartverfahren hergeleitet und dessen superlineare Konvergenz gezeigt. Scharfe Fehlerschranken und eine Deflated Variante werden anschließend angegeben.

Ebenfalls motiviert durch exponentielle Integratoren, insbesondere durch exponentielle Runge-Kutta-Verfahren, ist die Entwicklung von Verfahren zur Approximation von Produkten

$$z_i := f(hA)b_i, \qquad b_i = g(t + c_i h), \qquad i = 1, \ldots, n,$$

wobei g eine glatte Funktion ist, h die Zeitschrittweite und c_i die Knoten einer Quadraturformel sind. Wir stellen zwei neue Verfahren dar, die ausgehend von einem Krylovraum bzgl. A und b_1 die Approximationen $z_2, \ldots, z_n$ mit deutlich geringerem Aufwand berechnen, als bei einer naiven Implementierung, die zuvor berechnete Krylovräume nicht ausnutzt. Das erste Verfahren ist ein Lanczos-Galerkin Projektionsverfahren. Für dieses Verfahren geben wir einen Konvergenzbeweis und scharfe Fehlerschranken an und entwickeln Varianten. Das zweite neue Verfahren, ist ein Orthogonalprojektionsverfahren. Dieses verwendet die QR-Zerlegung der Vektoren $b_1, \ldots, b_n$. Für die Erstellung eines Abbruckkriteriums benötigen wir eine genaue theoretische Analyse der Faktoren der QR-Zerlegung. Dies gilt insbesondere für Varianten dieses Verfahrens, die wir durch das Verwenden von Restarts, Up-/Downdating Methoden und/oder der Hinzunahme von Ritzvektoren erstellen.

Bei allen neu vorgestellten Verfahren ist die größte Schwierigkeit, dass für Matrixfunktionen kein Residuum zur Verfügung steht und wir deshalb viele in der Literatur vorgeschlagene Techniken für Verfahren zur Lösung linearer Gleichungssysteme nicht direkt verwenden werden können.

Schließlich werden umfangreiche numerische Beispiele präsentiert. Es stellt sich heraus, dass mit den neu vorgeschlagenen Techniken die Ersparnis beim Rechenaufwand gegenüber einer naiven Implementierung über 90 % liegen kann. Damit ist erstmals gezeigt, dass exponentielle Integratoren auch für solch große Beispiele konkurrenzfähig sind.

Inhaltsverzeichnis

1 Einleitung **7**

2 Grundlagen **11**
2.1 Matrixfunktionen 11
2.1.1 Krylovverfahren für Matrixfunktionen 11
2.1.2 Shift-and-invert Lanczosverfahren 14
2.2 Exponentielle Integratoren 15
2.2.1 Exponentielle Runge-Kutta Verfahren 15
2.2.2 Exponentielle Rosenbrock Verfahren 17
2.3 Notation 18

3 Restarts **19**
3.1 Restarts für verschobene lineare Gleichungssysteme 19
3.2 Restarts für Matrixfunktionen 22
3.3 Konvergenz und Fehlerschranken 27
3.4 Deflated Restarts 34
3.5 Numerische Beispiele 36
3.5.1 Dreidimensionale Wärmeleitungsgleichung 36
3.5.2 Zweidimensionale Wärmeleitungsgleichung 37

4 Lanczos-Galerkin Projektionsverfahren **41**
4.1 Gleichungssysteme mit mehreren rechten Seiten 41
4.2 Lanczos-Galerkin Projektion für Matrixfunktionen 43
4.3 Konvergenz und Fehlerschranken 46
4.4 Lanczos-Galerkin mit mehreren Vektoren 53

5 Orthogonalprojektionsverfahren **59**
5.1 Analyse der Orthogonalisierung 60
5.2 Orthogonalprojektionsverfahren 64
5.2.1 Grundidee 64
5.2.2 Restart 68
5.2.3 Zirkulierende Orthogonalprojektion 71
5.2.4 Hinzunahme von Ritzvektoren 75
5.2.5 Reorthogonalisierung der Gram-Schmidt QR-Zerlegung 81

6 Numerische Beispiele **83**
6.1 Testprobleme 83
6.2 Orthogonalprojektion mit Ritzvektoren 85
6.3 Vergleich von Orthogonal- und Lanczos-Galerkin Projektionsverfahren . 88
6.3.1 Zeitaufwand 89
6.3.2 Speicheraufwand 92
6.4 Vergleich verschiedener Integratoren 97

7 Bemerkungen und Danksagung **105**

Kapitel 1

Einleitung

In dieser Arbeit werden Approximationen an Produkte der Form

$$z = f(A)b \tag{1.1}$$

mit Krylovverfahren betrachtet, wobei $A \in \mathbb{C}^{N,N}$ eine große, dünn besetzte Matrix ist, $b \in \mathbb{C}^N$ ein Vektor und f eine Funktion. Eine wesentliche Rolle in dieser Arbeit spielt die Exponentialfunktion.

Krylovverfahren zur Lösung linearer Gleichungssysteme sind schon seit mehr als 50 Jahren bekannt. Zunächst wurden sie als direkte Verfahren interpretiert, erst 25 Jahre später erkannte man ihr Potential als iterative Verfahren. Prominentester Vertreter ist das Verfahren der konjugierten Gradienten, welches eine Galerkinapproximation für lineare Gleichungssysteme mit symmetrischer und positiv definiter Matrix liefert. Seit etwa 1985 wurden zahlreiche Varianten für nicht symmetrische Probleme vorgestellt und ausgetestet, hier sind besonders das FOM und das GMRES-Verfahren zu nennen.

Krylovapproximationen an Produkte von Matrixfunktionen mit Vektoren, insbesondere an $\exp(A)b$, wurden zuerst in der chemischen Physik durch Nauts und Wyatt [58] und Park und Light [64] vorgestellt. In der mathematischen Fachliteratur wurden sie inzwischen ausführlich diskutiert, wobei hier besonders die Arbeiten von Druskin und Knizhnerman [17, 18], Gallopoulos und Saad [28], Knizhnerman [43, 44], und Hochbruck und Lubich [36] zu nennen sind. In der letzten Arbeit wird neben scharfen Fehlerschranken auch eine neue Klasse von Integrationsverfahren, sogenannte exponentielle Integratoren vorgeschlagen. Für diese wird schließlich in Hochbruck, Lubich und Selhofer [35] eine effiziente Implementierung mit Hilfe von Krylovverfahren vorgestellt, die als Startpunkt für weitere Aktivitäten auf diesem Gebiet angesehen werden kann. Zur Zeit beschäftigen sich weltweit diverse Arbeitsgruppen mit der Konstruktion, Analyse und Implementierung exponentieller Integratoren und/oder der Approximation von $f(A)b$ für verschiedene Anwendungen. Die Aktualität des Themas wird auch durch Vorträge und Minisymposium bei wichtigen Tagungen dokumentiert.

Die Konvergenzresultate für die Approximation der Matrixexponentialfunktion zeigen, dass für symmetrische Matrizen ein superlineares Konvergenzverhalten etwa nach $\|A\|$ Iterationsschritten beginnt. Ist diese Norm groß – etwa bei feinen Ortsdiskretisie-

rungen partieller Differentialgleichungen – so ist die naive Anwendung eines Krylovverfahren in der Regel nicht mehr möglich, da Rechen- und Speicheraufwand zu hoch sind. Die Lösung linearer Gleichungssysteme mit Krylovverfahren kann man durch Vorkonditionierung (z.B. Mehrgitterverfahren oder unvollständige Zerlegungen) beschleunigen, den Speicheraufwand kann man durch Restarts begrenzen. Beide Techniken basieren wesentlich darauf, dass das Residuum der aktuellen Näherung an die Lösung des Gleichungssystems nicht nur theoretisch sondern auch im praktischen Algorithmus verfügbar ist. Dies ist leider nicht mehr der Fall, wenn wir eine beliebige Funktion f betrachten, zum Beispiel die Exponentialfunktion. Ohne zum Residuum überzugehen, ist es auch unklar, wie man gute Startwerte für die Iteration auszunutzen kann.

Die Frage der Vorkonditionierung des Exponentialoperators ist inzwischen für symmetrische Matrizen gelöst. Van den Eshof und Hochbruck [81] und von Moret und Novati [54] haben unabhängig voneinander sogenannte Shift-and-invert Verfahren vorgeschlagen, die einen Krylovraum bzgl. $(I + \gamma A)^{-1}$ statt bzgl. A verwenden. Die verschobenen linearen Gleichungssysteme können entweder mit schnellen direkten Lösern [54] oder mit Hilfe des vorkonditionierten Verfahrens der konjugierten Gradienten [81] iterativ gelöst werden. Für diese Verfahren ist gezeigt worden, dass die Konvergenz zwar sublinear aber unabhängig von der Norm der Matrix ist. In der Regel benötigt man nur sehr wenige Iterationsschritte bis zum Erreichen der gewünschten Genauigkeit. Weitere aktuelle Arbeiten, die rationale Approximationen verwenden, gibt es von Lopez und Simoncini [50], Moret und Novati [55], Bergamaschi und Vianello [5], Bergamaschi, Caliari und Vianello [4], Bergamaschi, Caliari, Martínez und Vianello [3], vgl. auch den Übersichtsartikel von Frommer und Simoncini [27] und weitere darin aufgeführte Referenzen. Ebenfalls als rationale Approximation kann man Approximation mit Hilfe von Talbotquadraturformeln interpretieren, wie sie Trefethen, Weidemann und Schmelzer [80] vorgeschlagen haben.

Die Fragestellung hinter der vorliegenden Arbeit ist durch die Implementierung exponentieller Integratoren motiviert, insbesondere exponentieller Runge-Kutta-Verfahren für semilineare parabolische Differentialgleichungen, die zur Zeit intensiv untersucht werden [6, 7, 10, 11, 13, 37, 38, 45, 52, 62, 79]. Bei diesen Integratoren bleibt die Matrix während der gesamten Integrationsdauer konstant, lediglich die Vektoren, mit denen die (verschiedenen) Matrixfunktionen multipliziert werden, ändern sich und zwar bei genügend glatten Lösungen langsam mit der Zeit. Im Gegensatz zu dem kürzlich von Berland, Skaflestad und Wright [8] entwickelten Softwarepaket *EXPINT* beschäftigen wir uns mit dem Fall, dass eine explizite Berechnung von $f(hA)$ nicht möglich ist.

Schwerpunkt meiner Arbeit bilden daher Verfahren zur Approximation von Produkten

$$z_i := f(hA)b_i, \qquad b_i = g(t + c_i h), \qquad i = 1, \ldots, n, \tag{1.2}$$

wobei g eine glatte Funktion, h die Zeitschrittweite und c_i die Knoten einer Quadraturformel sind. Im Mittelpunkt steht dabei die Frage, wie schon berechnete Informationen aus vorherigen Approximationen weiter verwendet werden können obwohl Residuen nicht explizit verfügbar sind. Wir stellen zwei neue Verfahren dar, die ausgehend von einem Krylovraum bzgl. A und b_1 die Approximationen $z_2, \ldots, z_n$ mit deutlich geringerem Aufwand berechnen, als bei einer naiven Implementierung, die zuvor berechnete

Krylovräume nicht ausnutzt.

Die Konstruktion unserer Verfahren basiert wesentlich auf Herleitung der Krylovapproximation über die Cauchyintegralformel und einem verallgemeinerten Residuum für Matrixfunktionen wie sie auch in der Arbeit von Hochbruck und Lubich [36] verwendet wird. Es ist nämlich bekannt, dass die Residuen der Galerkinapproximationen an die verschobenen Gleichungssysteme für jede Wahl des Shifts in einem eindimensionalen Raum liegen. Dies liefert zunächst eine Möglichkeit, Restarts für Matrixfunktionen zu definieren und gleichzeitig die Konvergenzresultate aus [36] auf diesen Fall zu verallgemeinern. Unabhängig hiervon haben Eiermann und Ernst [20] ebenfalls erkannt, dass man solche Restarts auch durchführen kann, in dem man ausnutzt, dass die Krylovapproximation auch als Interpolation der Funktion in den Ritzwerten aufgefasst werden können [70]. Die beiden Algorithmen sind mathematisch äquivalent, die Herleitung über die Cauchyintegralformel hat aber den Vorteil, dass sie aussagekräftige Fehlerschranken liefert. Sie erlaubt zudem auch Deflated Restarts auf Matrixfunktionen zu übertragen.

Das erste Verfahren zur Approximation von (1.2) beruht darauf, die Grundidee der Lanczos-Galerkin Projektionsverfahren zur Approximation linearer Gleichungssysteme auch für Matrixfunktionen zu nutzen. Um einen guten Startvektor für die Lösung $Ax = b_2$ im Krylovraum bzgl. A und b_1 zu erhalten, projizieren Parlett [65] und Saad [69] b_2 auf diesen Krylovraum und berechnen einen neuen Krylovraum bzgl. des Residuums dieser Galerkinapproximation. Betrachtet man verschobene lineare Gleichungssysteme, so spannen die Residuumvektoren zu allen Shifts einen zweidimensionalen Raum auf. Das Band Krylov Verfahrens von Ruhe [67], angewandt auf die Basisvektoren dieses zweidimensionalen Raums, erstellt uns einen Unterraum, in dem wir mit Hilfe einer Galerkinbedingung Approximationen an die verschobenen Gleichungssysteme erstellen. Über die Cauchyintegralformel erhalten wir damit eine Lanczos-Galerkin Approximation an $f(A)b_2$ unter Verwendung eines guten Startvektors. Durch diese Herleitung lassen sich dann Konvergenzresultate und Fehlerschranken beweisen. Das Verfahren verallgemeinern wir dann für n Approximationen.

Ein anderer Ansatz ist, eine Orthonormalbasis $q_1, \ldots, q_n$ der Vektoren $b_1, \ldots, b_n$ zu berechnen und Krylovräume bzgl. A und q_i anstatt solcher bzgl. b_i zu verwenden. Wir zeigen, dass der obere Dreiecksfaktor der QR-Zerlegung in der i-ten Zeile Einträge in der Größenordnung $O(h^{i-1})$ hat und wie dies ausgenutzt werden kann, um die Gesamtzahl der Iterationsschritte zu reduzieren. Ähnliche Ansätze machen auch Fischer [23] und Lötstedt und Nilsson [51] zur Lösung linearer Gleichungssystemen mit verschiedenen rechten Seiten. Die Hauptschwierigkeit bei unserem Verfahren ist es, verlässliche Abbruchkriterien für die Iteration zu finden, da für Matrixfunktionen – wie schon mehrfach erwähnt – das Residuum nicht zugänglich ist. Wir benötigen daher eine genauere theoretische Analyse als in [23, 51]. Dies gilt insbesondere für die Varianten dieses Verfahrens, die durch das Einfügen von Restarts, Up-/Downdating Methoden und/oder Ritzvektoren erstellt werden.

Die vorliegende Arbeit ist folgendermaßen gegliedert: Im Kapitel 2 über die Grundlagen der Arbeit stellen wir zunächst das Standard Lanczosverfahren und das Shift-and-invert Lanczosverfahren zur approximativen Lösung von (1.1) vor, auf denen wir unsere neuen Verfahren aufbauen. Anschließend erläutern wir kurz exponentielle Integratoren

und geben einige Beispiele explizit an.

Im Kapitel 3 leiten wir die Restartmethode zur Approximation von Matrixfunktionen her und zeigen die Äquivalenz zu der Restartmethode von Eiermann und Ernst [20]. Anschließend zeigen wir die superlineare Konvergenz und Fehlerschranken. Nachdem die Deflated Variante des Restartverfahren beschrieben wurde, werden am Ende des Kapitels numerische Beispiele angegeben. Wir zeigen, dass man bei gewissen Problemstellungen mit dem Restartverfahren nicht nur Speicherplatz spart, sondern auch eine Beschleunigung der Berechnung der Approximation erreichen kann.

Im Kapitel 4 stellen wir das Lanczos-Galerkin Projektionsverfahren zur Approximation der Produkte (1.2) vor. Wir motivieren das Verfahren zunächst zur Approximation von Produkten einer Matrixfunktion mit zwei Vektoren und verallgemeinern es später. Die Konvergenz des Verfahrens wird bewiesen und Fehlerschranken hergeleitet. Am Ende des Kapitels werden unterschiedliche Varianten zur Approximation von Produkten einer Matrixfunktion mit mehreren Vektoren angegeben.

Im Kapitel 5 leiten wir das Verfahren, das die QR-Zerlegung der Vektoren $b_1, \ldots, b_n$ aus (1.2) nutzt, her und bezeichnen es als das *Orthogonalprojektionsverfahren*. Wir entwickeln Abbruchkriterien sowohl für das Grundverfahren als auch für dessen Varianten, die Restarts, zirkulierende QR Räume und/oder Ritzvektoren nutzen.

Numerische Beispiele zu den Verfahren aus den Kapiteln 4 und 5 findet man in Kapitel 6. Wir testen dort zunächst, in wie weit die Hinzunahme von Ritzvektoren zu dem Orthogonalprojektionsverfahren sinnvoll ist. Dann vergleichen wir die verschiedenen Projektionsverfahren untereinander und mit einem Krylovverfahren, dass keine schon vorhandenen Informationen nutzt. Anschließend untersuchen wir, ob man mit diesen neu vorgestellten Krylovverfahren exponentielle Integratoren effizienter implementieren kann. Als Integratoren verwenden wir das exponentielle Eulerverfahren und das Verfahren von Krogstad [45]. Wir werden zeigen, dass man erheblich Zeitaufwand sparen kann, in manchen Beispielen sogar insgesamt weit über 90 %, wenn man diese Verfahren benutzt. Das mit unseren Verfahren optimierte Krogstadverfahren vergleichen wir dann mit anderen in der Literatur bekannten Integratoren wie Radau-Kollokationsverfahren, Mehrschrittverfahren und Runge-Kutta-Chebyshev-Verfahren. Es zeigt sich, dass mit den hier vorgestellten Techniken exponentielle Integratoren zumindest in gewissen Anwendungen effizienter sind als die Vergleichsverfahren.

Zum Schluss dieser Arbeit ziehen wir noch ein kurzes Resumee.

Kapitel 2

Grundlagen

2.1 Matrixfunktionen

Die Berechnung von Matrixfunktionen $f(A)$ hat schon eine lange Geschichte. Die prominenteste und meist beschriebenste Funktion ist die Exponentialfunktion e^A. Es gibt viele verschiedene Möglichkeiten diese zu berechnen. Es werden unter anderem Padé-Approximationen, Interpolationsverfahren, Matrixfaktorisierungs- und Splittingverfahren angewandt. Moler und van Loan haben einige dieser direkten Verfahren zur Berechnung der Exponentialfunktion in [53] zusammengefasst. Wir betrachten Matrixfunktionen mit großen, dünn besetzten Matrizen. Für diese Matrixfunktionen sind keine direkten Methoden anwendbar, da sie zu langsam sind oder zuviel Speicherplatz brauchen. Deshalb benutzen wir iterative Verfahren, insbesondere Krylovverfahren.

2.1.1 Krylovverfahren für Matrixfunktionen

Bei diesen Verfahren berechnet man eine Approximation an $z = f(A)b$ mit Hilfe des Krylovunterraumes $\mathcal{K}_m(A, b)$ mit

$$\mathcal{K}_m(A, b) = \text{span}\{b, Ab, \ldots, A^{m-1}b\}.$$

Entwickelt wurden solche Verfahren zuerst in der chemischen Physik von Nauts und Wyatt [58] und Park und Light [64]. Mathematisch analysiert und weiterentwickelt wurden sie zuerst für die Exponentialfunktion unter anderen von Knizhnerman [43, 44], Druskin und Knizhnerman [17], Druskin, Greenbaum und Knizhnerman [16], Gallopoulos und Saad [28], Hochbruck und Lubich [36], Nour-Omid [60], Saad [70] und Stewart und Leyk [77]. Aber auch andere Matrixfunktionen können auf diese Weise approximiert werden. Arbeiten der letzten Jahre, die sich mit Matrixfunktionen beschäftigen, sind [3, 4, 5, 20, 27, 50, 54, 55, 80, 81]. Geeignete Krylovverfahren sind sowohl der Arnoldi- [1] als auch der Lanczosprozess [47]. Da wir uns in dieser Arbeit auf symmetrische Probleme konzentrieren, verwenden wir den symmetrischen Lanczosprozess, aber fast alle theoretische Herleitungen sind auch für das Arnoldiverfahren gültig. Es gibt drei verschiedene Ansätze, um diese Approximationen mit Hilfe von Krylovräumen herzuleiten

und zu analysieren. Wir benutzen die Darstellung von (1.1) in der Cauchyintegralformel. Man kann aber auch den Krylovunterraum als einen Suchraum, in den projiziert wird, interpretieren oder man nimmt an, dass man die Funktion in einigen Punkten interpoliert, wie es Saad [70] vorgeschlagen hat. Diese verschiedenen Ansätze werden unter anderem in einer Arbeit von Hochbruck und Hochstenbach [34] genauer beschrieben.

Im Folgenden leiten wir die Approximation an (1.1) her, die dieser Arbeit als Grundlage dient. Wir können mit der Cauchyintegralformel das Produkt (1.1) durch

$$z = f(A)b = \frac{1}{2\pi i}\int_\Gamma f(\lambda)(\lambda I - A)^{-1}b \, \mathrm{d}\lambda \tag{2.1}$$

darstellen, wobei Γ eine Randkurve eines stückweisen glatten, begrenzten Gebietes Ω ist, welches das Spektrum von A enthält. Weiter muss f analytisch in Ω und stetig auf dem Abschluß von Ω sein. I bezeichnet die Einheitsmatrix. Nun muss man das lineare Gleichungssystem

$$x(\lambda) := (\lambda I - A)^{-1}b \tag{2.2}$$

lösen. Zur approximativen Lösung eines linearen Gleichungssystems gibt es viele iterative Verfahren. Wir verwenden im Folgenden Krylovverfahren. Ist $V_m = [v_1, \ldots, v_m]$ die Basis des m-ten Krylovraumes $\mathcal{K}_m(A, b)$ aus dem Krylovprozess mit $v_1 = b$ und $V_m^H V_m = I$, dann gilt die Krylov Relation

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T.$$

Hierbei ist e_i der i-te Einheitsvektor und H_m eine obere Hessenbergmatrix. Beim Lanczosverfahren ist, ist H_m eine Tridiagonalmatrix. Es gilt folgende Relation

$$(\lambda I - A)V_m = V_m(\lambda I - H_m) - h_{m+1,m} v_{m+1} e_m^T.$$

Die Galerkin Approximation an (2.2) ist dann

$$x_m(\lambda) = V_m(\lambda I - H_m)^{-1}\beta e_1 \tag{2.3}$$

mit $\beta = \|b\|$. Diese erhält man, indem man verlangt, dass die Galerkin Bedingung

$$0 = V_m^H r_m(\lambda) = V_m^H(b - (\lambda I - A)x_m(\lambda))$$

bezüglich des Residuums

$$r_m(\lambda) = b - (\lambda I - A)x_m(\lambda)$$

erfüllt ist. Damit ist die Approximation z_m an (2.1) durch

$$\begin{aligned} z_m &= \frac{1}{2\pi i}\int_\Gamma f(\lambda)V_m(\lambda I - H_m)^{-1}\beta e_1 \mathrm{d}\lambda \\ &= V_m f(H_m)\beta e_1 \end{aligned} \tag{2.4}$$

gegeben. Da nun die Dimension m von H_m klein ist, kann man $f(H_m)$ mit Hilfe eines der direkten Verfahren, die am Anfang dieses Abschnittes kurz erwähnt wurden, berechnen.

Wir verwenden meist Verfahren, die Padé-Approximationen oder eine Diagonalisierung verwenden. Gallopoulos und Saad [28] zeigen, dass der Fehler der Approximation (2.4) proportional zu $\|A\|^m/m!$ ist, das heißt, dass das Verfahren für $m \gg \|A\|$ superlinear konvergiert. Dagegen zeigen Hochbruck und Lubich [36], dass die superlineare Konvergenz in einigen Fällen schon früher beginnt. Zum Beispiel für symmetrische, negativ definite Matrizen beginnt sie für $m \geq \sqrt{\|A\|}$, aber für schiefhermitsche Matrizen fängt die Superlinearität erst bei $m \approx \|A\|$ an. Das Lanczosverfahren, welches auf oben beschriebener Weise die Approximation an $f(A)b$ berechnet, nennen wir *Standard Lanczosverfahren*. Wir betrachten nun den wesentlichen Speicher- und Rechenaufwand dieses Verfahrens.

Speicheraufwand: Bei großen Problemen ist der Hauptspeicheraufwand durch die Anzahl der Vektoren der Dimension N, die gespeichert werden müssen, gegeben. Zur Berechnung der Approximation z_m an $f(A)b$ müssen m Vektoren aus dem Krylovverfahren und die Approximation selbst gespeichert werden. Das sind $m+1$ Vektoren der Dimension N.

Rechenaufwand: Der wesentliche Rechenaufwand des symmetrischen Lanczosverfahrens besteht im k-ten Iterationsschritt aus einer Matrixvektormultiplikation mit A und einer Funktionsauswertung von H_m, da die noch vorkommenden zwei Saxpy's und zwei Skalarprodukte dazu vernachlässigbar sind. Ein Saxpy bezeichnet eine Operation $\alpha x + y$, wobei x und y Vektoren sind und α ein Skalar ist. Wendet man das Arnoldiverfahren an, muss man neben der Matrixvektormultiplikation oder der Funktionsauswertung im k-ten Schritt k Saxpy's und $k+1$ Skalarprodukte berechnen. In dieser Arbeit werden wir aber bei allen Untersuchungen des Rechenaufwandes nur den des Lanczosverfahrens betrachten.

Bemerkung: Um die Approximation (2.3) zu erstellen, haben wir eine Galerkin Bedingung genutzt, wie man sie auch für das FOM Verfahren von Saad [68] zur Lösung linearer Gleichungssysteme verwendet. Eine andere Klasse von Krylovverfahren verwenden Minimierungseigenschaften, um eine Approximation zu erstellen. Ein Vertreter dieser Klasse ist das GMRES Verfahren von Saad und Schultz [72]. Wie wir in (2.1) - (2.4) gesehen haben, ist unsere Vorgehensweise so, dass wir die Matrixfunktion mit einer Cauchyintegralformel darstellen. Dann betrachten wir die verschobenen, linearen Gleichungssysteme, bestehend aus den Resolventen $(\lambda I - A)^{-1}$ und dem Vektor b. Wendet man auf diese Gleichungsysteme Approximationsverfahren an, so hängen die Approximationen von λ ab. Wir suchen Verfahren, die Approximationen erstellen, deren λ-Abhängigkeit auch durch eine Resolvente gegeben ist. Somit können wir diese Approximationen mit der Cauchyintegralformel als Approximation einer Matrixfunktion darstellen. Krylov-Galerkin-Approximationen, angewandt auf eine Resolvente multipliziert mit einem Vektor, geben wieder eine Resolvente als Approximationen an, Minimierungsmethoden nicht. Deshalb können wir in dieser Arbeit keine Minimierungsmethoden verwenden und betrachten nur Galerkinverfahren.

2.1.2 Shift-and-invert Lanczosverfahren

Ein anderes, für diese Arbeit wichtiges, Krylovverfahren zur Lösung von (1.1) mit symmetrischer Matrix A ist ein vorkonditioniertes Lanczosverfahren, welches von van den Eshof und Hochbruck [81] und Moret und Novati [54] beschrieben wird. Im Folgenden nennen wir dieses Verfahren *Shift-and-invert Lanczosverfahren*. Man nutzt hier aus, dass bei Matrixfunktionen, die sich ähnlich verhalten wie die Exponentialfunktion, die größten Eigenwerte und die zugehörigen Eigenvektoren von A stark die Lösung von $f(A)b$ bestimmen. Deshalb versucht man die Matrix A so zu transformieren, dass das Lanczosverfahren diese Vektoren schnell findet. Hierfür wird der Lanczosprozess auf $(I+\gamma A)^{-1}$ mit $\gamma > 0$ angewandt und man erhält nach m Schritten die folgende Relation

$$(I+\gamma A)^{-1}V_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{2.5}$$

mit $V_m e_1 = b$ und $V_m^T V_m = I$. Dann ist die Approximation an (1.1)

$$z_m = V_m f(\widetilde{H}_m)\beta e_1, \qquad \widetilde{H}_m = \frac{1}{\gamma}(H_m^{-1} - I).$$

Der optimale Wert für γ wird in [81] tabellarisch in Abhängigkeit der gewünschten Genauigkeit angegeben. Ebenfalls wird gezeigt, dass die Konvergenzgeschwindigkeit dieses Verfahrens unabhängig von dem Diskretisierungsgitter ist, mit dem die Matrix A aus einer Differentialgleichungen erstellt wurde. Als Fehlerschätzer für den relativen Fehler der Approximation wird

$$\|z - z_m\| \lesssim \frac{\delta_m}{1-\delta_m}\|z_m\| \tag{2.6}$$

mit

$$\delta_m = \frac{\|z_m - z_{m-i}\|}{\|z_m\|}$$

benutzt. Hierbei ist i sicherheitshalber flexibel gehalten, um eventuelle ungewollte Abbrüche in den ersten Schritten zu vermeiden. In den meisten Experimenten reicht es $i = 1$ zu setzen.

Bei der Erstellung der Lanczosrelation (2.5) muss im i-ten Iterationsschritt ein Gleichungssystem der Art $(I+\gamma A)^{-1}v_i$ gelöst werden. Man kann diese Gleichungssysteme sowohl direkt als auch iterativ lösen. Als direkte Löser kann man zum Beispiel die LU-Zerlegung von A oder den Backslashbefehl von Matlab nutzen, als iterativer Löser wird in [81] das mit einer unvollständigen LU-Zerlegung vorkonditionierte cg-Verfahren vorgeschlagen. Zusätzlich wird angegeben, zu welcher Genauigkeit das cg-Verfahren das Gleichungssystem lösen muss, dies hängt von dem Fehler der Approximation des Lanczosverfahrens im aktuellen Schritt ab. Mit wachsender Iterationszahl wird die geforderte Genauigkeit immer größer. Der wesentliche Speicheraufwand dieses Verfahrens, ohne Berücksichtigung des Speicheraufwands für die Berechnung des linearen Gleichungssystems, ist derselbe wie im letzten Abschnitt des Standardverfahrens. Der Rechenaufwand ändert sich aber dahingehend, dass nun pro Iterationsschritt anstatt der Berechnung einer Matrixvektormultiplikation ein lineares Gleichungssystem gelöst werden muss.

2.2 Exponentielle Integratoren

Exponentielle Integratoren zur Lösung von Differentialgleichungen existieren schon lange, aber erst in den letzten Jahren wurde ihre praktische Bedeutung erkannt. Mit exponentiellen Integratoren möchte man ganz allgemein steife oder oszillatotische Differentialgleichungen lösen. Die ersten exponentiellen Einschrittverfahren wurden vorgestellt in [19, 25, 48, 66, 78, 79], wobei in den meisten Arbeiten die Variation-der-Konstanten Formel zur Lösung von (2.7) benutzt wurde. Hierbei musste man in jedem Schritt die Exponentialfunktion angewandt auf A lösen. Exponentielle Mehrschrittverfahren wurden in [38, 46, 59, 84] betrachtet. Dass die Forschung auf dem Gebiet der exponentiellen Integratoren in den letzten Jahren intensiv betrieben wird, zeigt eine Auswahl an Arbeiten [6, 7, 10, 11, 13, 37, 38, 45, 52, 62, 79] aus diesem Zeitraum.

2.2.1 Exponentielle Runge-Kutta Verfahren

In dieser Arbeit konzentrieren wir uns auf die Lösung von semilinearen, parabolischen Differentialgleichungen

$$u'(t) = Au(t) + g(u,t), \qquad u(t_0) = u_0. \tag{2.7}$$

Hierbei ist A eine symmetrische und negativ definite Matrix. Wir nutzen exponentielle Runge-Kutta Verfahren zur Lösung von (2.7). Darunter fällt das exponentielle Eulerverfahren und Verfahren höherer Ordnung, wie das von Krogstad [45], Cox und Matthews [13] und von Hochbruck und Ostermann [37]. Ein allgemeines s stufiges exponentielles Runge-Kutta Verfahren berechnet ausgehend von $u_n \approx u(t_n)$ die Näherung an die exakte Lösung $u(t_{n+1})$ durch

$$\begin{aligned} u_{n+1} &= u_n + h\sum_{i=1}^{s} b_i(hA)\big(G_{n,i} + Au_n\big), \\ U_{n,i} &= u_n + h\sum_{j=1}^{s} a_{ij}(hA)\big(G_{n,j} + Au_n\big), \\ G_{n,i} &= g(t_n + c_i h, U_{n,i}). \end{aligned}$$

Diese Verfahren wurden für semilineare parabolische Probleme von Hochbruck und Ostermann [37] untersucht. Hierbei sind die Koeffizienten $b_i(hA)$ und $a_{ij}(hA)$ Linearkombinationen von

$$\varphi_{k,i}(hA) := \varphi_k(c_i hA) \qquad i = 1,\dots,s, \qquad k = 1,\dots,s \tag{2.8}$$

mit

$$\varphi_0(z) = e^z, \quad \varphi_{k+1}(z) = \frac{\varphi_k(z) - 1/k!}{z}.$$

Für die Berechnung der Näherungslösung braucht man also Approximationen an

$$\varphi_{k,i}(hA)\big(G_{n,j} + Au_n\big), \quad k,i = 1,\dots,s. \tag{2.9}$$

In jedem Schritt des exponentiellen Runge-Kutta Verfahrens müssen wir Produkte der Funktionen, ausgewertet an $c_k hA$ mit s verschiedenen Vektoren, berechnen. Da wir die Approximationen an (2.9) mit Krylovverfahren erstellen, können wir alle Approximation an $\varphi_{k,i}(hA)$, angewandt auf denselben Vektor, in demselben Krylovunterraum berechnen. Deshalb müssen wir in jedem Schritt der exponentiellen Runge-Kutta Methode insgesamt nur s neue Krylovunterräume erstellen.

Ein exponentielles Runge-Kutta Verfahren zur Lösung eines semilinearen Problems hat steife Ordnung p, falls für den Fehler der Näherung mit $t_n = t_0 + nh$

$$\|u_n - u(t_n)\| \leq Ch^{p+1}$$

gilt, wobei die Konstante C von T mit $0 \leq nh \leq T$ und der Lösung u abhängt, aber unabhängig von n, h und $\|A\|$ ist.

Nun geben wir zwei exponentielle Runge-Kutta Verfahren explizit an, da wir diese im Kapitel 6 (über numerische Experimente) anwenden. Das erste Verfahren ist das exponentielle Eulerverfahren. Es ist ein Einschrittverfahren und durch

$$u_{n+1} = u_n + h\varphi_1(hA)(g(t_n, u_n) + Au_n)$$

charakterisiert. Man muss also pro Zeitschritt eine Matrixfunktion angewandt auf einen Vektor berechnen.

Das zweite Verfahren ist das Verfahren von Krogstad [45]. Es ist ein vierstufiges exponentielles Runge-Kutta Verfahren, dessen klassische Ordnung 4 ist. Hochbruck und Ostermann zeigen aber in [37], dass es für steife Problem im schlimmsten Fall Ordnung 3 hat. Die Näherung an die Lösung $u(t_{n+1})$ zum Zeitschritt t_{n+1} ist durch

$$\begin{aligned}
U_{n,1} =& u_n, \\
U_{n,2} =& u_n + \frac{h}{2}\varphi_{1,2}(hA)\tilde{g}_1, \\
U_{n,3} =& u_n + \left(\frac{h}{2}\varphi_{1,3}(hA) - h\varphi_{2,3}(hA)\right)\tilde{g}_1 + h\varphi_{2,3}(hA)\tilde{g}_2, \\
U_{n,4} =& u_n + (h\varphi_{1,4}(hA) - 2h\varphi_{2,4}(hA))\tilde{g}_1 + 2h\varphi_{2,4}(hA)\tilde{g}_3, \\
u_{n+1} =& hb_1\tilde{g}_1 + hb_2\tilde{g}_2 + hb_3\tilde{g}_3 + hb_4\tilde{g}_4
\end{aligned}$$

mit

$$\tilde{g}_i = g(t_n + c_i h, U_{n,i}) + Au_n, \qquad i = 1, \ldots, 4$$

gegeben. Die Knoten sind

$$c_1 = 0, \qquad c_2 = c_3 = \frac{1}{2}, \qquad c_4 = 1$$

und die Gewichte

$$\begin{aligned}
b_1 &= \varphi_1 - 3\varphi_2(hA) + 4\varphi_3(hA), \\
b_2 = b_3 &= 2\varphi_2(hA) - 4\varphi_3(hA), \\
b_4 &= 4\varphi_3(hA) - \varphi_2(hA).
\end{aligned}$$

Pro Zeitschritt muss man beim Krogstad Verfahren also 4 Krylovräume erstellen, um die jeweiligen Produkte von Matrixfunktionen und Vektoren zu berechnen.

2.2.2 Exponentielle Rosenbrock Verfahren

Eine andere Klasse von exponentiellen Integratoren sind exponentielle Rosenbrockverfahren, vorgestellt von Hochbruck, Ostermann und Schweitzer [39]. Mit diesen löst man Differentialgleichungen der Form

$$u'(t) = A_n u(t) + g_n(u,t), \qquad u(t_0) = u_0, \tag{2.10}$$

wobei nun A_n und g_n von den Approximationen zum Zeitschritt t_n abhängen. Im Gegensatz zu (2.7) ändert sich hier die Matrix in jedem Zeitschritt. Das Schema eines s-stufigen exponentiellen Rosenbrockverfahren ist

$$\begin{aligned} U_{n,i} &= e^{c_i h A_n} u_n + h \sum_{j=1}^{i-1} a_{ij}(hA_n) g_n(U_{n,j}), \\ u_{n+1} &= e^{hA_n} u_n + h \sum_{i=1}^{s} b_i(hA_n) g_n(U_{n,i}). \end{aligned}$$

Wie in (2.8) sind auch bei diesem Verfahren die Koeffizienten $a_{ij}(hA_n)$ und $b_i(hA_n)$ Linearkombinationen der verschiedenen Funktionen $\varphi_{k,i}(hA_n)$. Der Fehler des Verfahrens im $(n+1)$-ten Zeitschritt wird durch $\|u_{n+1} - \widehat{u}_{n+1}\|$ mit

$$\widehat{u}_{n+1} = e^{hA_n} u_n + h \sum_{i=1}^{s} \widehat{b}_i(hA_n) g_n(U_{n,i})$$

geschätzt.

Das Verfahren *Rosenbrock43*, das wir in dem Kapitel über numerische Beispiele verwenden werden, ist ein dreistufiges exponentielles Rosenbrockverfahren mit steifer Ordnung vier mit einem Fehlerschätzer der dritten Ordnung. Dieses Verfahren ist durch die Koeffizienten

$$c_1 = 0, \qquad c_2 = \frac{1}{2}, \qquad c_3 = 1,$$

$$\begin{aligned} b_1 &= \varphi_1 - 14\varphi_3 + 36\varphi_4, \\ b_2 &= 16\varphi_3 - 48\varphi_4, \\ b_3 &= -2\varphi_3 + 12\varphi_4 \end{aligned}$$

und

$$a_{21} = 1/2\varphi_{1,2}, \qquad a_{32} = \varphi_1$$

gegeben, wobei die nicht aufgeführten Koeffizienten gleich Null sind. Die Koeffizienten für $\widehat{u}_{n+1}$ sind

$$\widehat{b}_1 = \varphi_1 - 14\varphi_3, \qquad \widehat{b}_2 = 16\varphi_3, \qquad \widehat{b}_3 = -2\varphi_3.$$

2.3 Notation

Zum Schluß des Grundlagenkapitels wollen wir noch einige Notationen einführen. Die Norm $\|.\|$ ohne Indizes steht für die Euklidnorm sowohl bei Vektoren als auch bei Matrizen. $\mathbb{P}_m$ bezeichnet den Raum der Polynome, deren Grad nicht größer als m sein darf. Sind $x_1, \ldots, x_n$ Vektoren der Dimension N, dann bezeichnen wir den, von diesen Vektoren aufgespannten, Raum mit $\text{span}\{x_1, \ldots, x_n\}$. Sei f eine n-mal stetig differenzierbare Funktion und damit die Punkte $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$ gegeben, dann bezeichnet $\delta^m f[x_0, x_1, \ldots, x_m]$ mit $m \leq n$ die m-ten dividierten Differenzen von f bezüglich $x_0, \ldots, x_m$. Diese sind rekursiv definiert durch

$$\delta^m f[x_0, x_1, \ldots, x_m] = \frac{\delta^{m-1} f[x_1, \ldots, x_m] - \delta^{m-1} f[x_0, \ldots, x_{m-1}]}{x_m - x_0}$$

mit $\delta^0 f[x_i] = f(x_i)$.

Kapitel 3

Restarts

Restartverfahren verwendet man, wenn bei großen Problemen nach einigen Iterationen kein Speicherplatz mehr zur Verfügung steht. Ein geschickter Restart behält die relevanten Informationen bei gleichzeitiger Entfernung der Komponenten, die am meisten Speicherplatz benötigen. Manchmal kann ein Restart auch helfen, Rechenzeit zu sparen.

In diesem Kapitel beschäftigen wir uns mit Restarts für Krylovverfahren zur Approximation von $f(A)b$ mit $A \in \mathbb{C}^{N,N}$ und $b \in \mathbb{C}^N$. So ein Verfahren haben auch schon Eiermann und Ernst [20] entwickelt. Wir gelangen zu demselben Verfahren aber mit einer anderen Herleitung. Während Eiermann und Ernst über die polynomiale Darstellung der Krylovverfahren gehen, motivieren wir das Verfahren über Restarts von verschobenen linearen Gleichungssystemen [73]. Diese Herleitung ermöglicht es, zusätzlich Fehlerschranken und einen anderen Konvergenzbeweis anzugeben. Am Ende des Kapitels beschreiben wir eine „Deflated Restart“ Variante und präsentieren numerische Beispiele.

Im Grundlagenkapitel 2.1 über Matrixfunktionen haben wir gesehen, dass man eine Matrixfunktion mit Hilfe der Cauchyintegralformel

$$f(A)b = \frac{1}{2\pi i} \int_\Gamma f(\lambda)(\lambda I - A)^{-1} b \, \mathrm{d}\lambda \tag{3.1}$$

darstellen kann. Unser Restartverfahren für Matrixfunktionen basiert auf Restartverfahren zur Lösung des verschobenen linearen Gleichungssystems

$$(\lambda I - A)x(\lambda) = b. \tag{3.2}$$

Im Folgenden setzen wir ohne Einschränkung $\|b\| = 1$.

3.1 Restarts für verschobene lineare Gleichungssysteme

In der Geschichte von Krylovverfahren wurden häufig bei der Vorstellung neuer Verfahren Restartvarianten mit eingeführt. Zum Beispiel für FOM und GMRES wurden direkt Restartversionen durch Saad [68] und Saad und Schultz [72] vorgestellt, diese sind auch in [71] nochmal aufgeführt. Weiter untersucht wurden diese unter anderem von Beattie,

Embree und Rossi [2] und Eiermann, Ernst und Schneider [21]. Speziell für verschobene lineare Gleichungssysteme haben Frommer und Glässener [26] ein neu gestartetes GMRES Verfahren betrachtet und Simoncini [73] hat ein neu gestartetes FOM Verfahren beschrieben.

Wir haben in Kapitel 2.1.1 gesehen, dass man zur Approximation von Matrixfunktionen keine Methoden benutzen kann, die auf GMRES zurückgehen. Deshalb ist die Grundlage dieses Abschnittes die Arbeit von Simoncini [73], in der sie ein Restartverfahren basierend auf FOM für einige verschobene lineare Gleichungssysteme

$$(\lambda_i I - A)x_i = b$$

mit $i = 1, \ldots, s$ beschreibt. Wir verallgemeinern ihre Herleitung für (3.2), indem wir unendlich viele $\lambda \in \Gamma$ betrachten. Hierbei ist Γ die Kurve, über die in der Cauchyintegralformel (3.1) integriert wird.

Die wesentliche Erkenntnis für dieses Verfahren ist, dass alle Residuenvektoren Vielfache von v_{m+1} sind, vergleiche dazu auch Hochbruck, Lubich, Selhofer [35, Abschnitt 6.3] (verallgemeinertes Residuum für Matrixfunktionen) und Saad [70].

Sei der m-te Krylovraum $\mathcal{K}_m(A, b)$ berechnet, dann gilt die Relation

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \qquad V_m^H V_m = I \tag{3.3}$$

und für verschobene Matrizen

$$(\lambda I - A)V_m = V_m(\lambda I - H_m) - h_{m+1,m} v_{m+1} e_m^T \tag{3.4}$$

mit $v_1 = b$. Dann kann man folgendes Lemma zeigen (vgl. [73] und [35]).

Lemma 3.1. *Sei $x_m(\lambda) = V_m y_m(\lambda)$ die Galerkinapproximation an die Lösung von $(\lambda I - A)x(\lambda) = b$ in $\mathcal{K}_m(\lambda I - A, b)$, wobei V_m die Relation (3.4) erfüllt. Dann existiert ein Skalar $\beta_m(\lambda)$, so dass*

$$r_m(\lambda) = \beta_m(\lambda) v_{m+1} \tag{3.5}$$

gilt.

Beweis. Ist $y_m(\lambda)$ die Lösung von

$$(\lambda I - H_m)y_m(\lambda) = e_1,$$

dann gilt für das Residuum

$$\begin{aligned} r_m(\lambda) &= b - (\lambda I - A)x_m(\lambda) \\ &= b - (\lambda I - A)V_m y_m(\lambda) \\ &= V_m e_1 - V_m(\lambda I - H_m)y_m(\lambda) + h_{m+1,m} v_{m+1} e_m^T y_m(\lambda) \\ &= h_{m+1,m} e_m^T y_m(\lambda) v_{m+1}. \end{aligned}$$

Setzen wir nun

$$\beta_m(\lambda) = h_{m+1,m} e_m^T y_m(\lambda),$$

dann gilt die Aussage. □

Im Folgenden betrachten wir das Verfahren nur für einen Restart, eine Verallgemeinerung für mehrere Restarts folgt später. Wir fassen nochmal zusammen: Falls noch kein Restart erfolgt und der m-te Krylovraum $\mathcal{K}_m(A,b)$ berechnet ist, dann gelten die Relationen (3.3) und (3.4). Die Galerkinapproximation an die Lösung von (3.2) ist $x_m(\lambda) = V_m y_m(\lambda)$ mit

$$y_m(\lambda) = (\lambda I - H_m)^{-1} e_1. \tag{3.6}$$

Das Residuum ist $r_m(\lambda) = \beta_m(\lambda) v_{m+1}$ mit

$$\beta_m(\lambda) = h_{m+1,m} e_m^T y_m(\lambda) = h_{m+1,m} e_m^T (\lambda I - H_m)^{-1} e_1. \tag{3.7}$$

Nun wollen wir einen Restart durchführen. Als neuen Startvektor wählen wir $r_m(\lambda)$. Dann gilt $\|r_m(\lambda)\| = |\beta_m(\lambda)|$ und somit für den ersten neuen Krylovvektor

$$\widehat{v}_1 = \frac{r_m(\lambda)}{\|r_m(\lambda)\|} = \pm v_{m+1}.$$

Wir wählen $\widehat{v}_1 = v_{m+1}$ und berechnen den $\widehat{m}$-ten Krylovraum $\mathcal{K}_{\widehat{m}}(A, v_{m+1})$. Dann gelten die Relationen

$$A\widehat{V}_{\widehat{m}} = \widehat{V}_{\widehat{m}} \widehat{H}_{\widehat{m}} + \widehat{h}_{\widehat{m}+1,\widehat{m}} \widehat{v}_{\widehat{m}+1} e_{\widehat{m}}^T, \qquad \widehat{V}_{\widehat{m}}^H \widehat{V}_{\widehat{m}} = I$$

und

$$(\lambda I - A)\widehat{V}_{\widehat{m}} = \widehat{V}_{\widehat{m}}(\lambda I - \widehat{H}_{\widehat{m}}) - \widehat{h}_{\widehat{m}+1,\widehat{m}} \widehat{v}_{\widehat{m}+1} e_{\widehat{m}}^T.$$

Die Approximation an (3.2) ist nun

$$\widehat{x}_{\widehat{m}}(\lambda) = x_m(\lambda) + \widehat{V}_{\widehat{m}} \widehat{y}_{\widehat{m}}(\lambda), \tag{3.8}$$

wobei man $\widehat{y}_{\widehat{m}}(\lambda)$ durch die Galerkin Bedingung bezüglich des Residuums

$$\widehat{r}_{\widehat{m}} = b - (\lambda I - A)\widehat{x}_{\widehat{m}}(\lambda)$$

erhält. Diese ist

$$\begin{aligned} 0 &= (\widehat{V}_{\widehat{m}})^H \widehat{r}_{\widehat{m}}(\lambda) \\ &= (\widehat{V}_{\widehat{m}})^H (r_m(\lambda) - (\lambda I - A)\widehat{V}_{\widehat{m}} \widehat{y}_{\widehat{m}}(\lambda)) \\ &= \beta_m(\lambda) e_1 - (\lambda I - \widehat{H}_{\widehat{m}}) \widehat{y}_{\widehat{m}}(\lambda). \end{aligned}$$

Also gilt

$$\widehat{y}_{\widehat{m}}(\lambda) = (\lambda I - \widehat{H}_{\widehat{m}})^{-1} \beta_m(\lambda) e_1 \tag{3.9}$$

und für das Residuum

$$\begin{aligned} \widehat{r}_{\widehat{m}}(\lambda) &= b - (\lambda I - A)(x_m(\lambda) + \widehat{V}_{\widehat{m}} \widehat{y}_{\widehat{m}}(\lambda)) \\ &= \beta_m(\lambda)\widehat{v}_1 - (\lambda I - A)\widehat{V}_{\widehat{m}} \widehat{y}_{\widehat{m}}(\lambda) \\ &= \beta_m(\lambda) \left(\widehat{v}_1 - (\lambda I - A)\widehat{V}_{\widehat{m}} (\lambda I - \widehat{H}_{\widehat{m}})^{-1} e_1 \right) \\ &= \beta_m(\lambda) \left(\widehat{h}_{\widehat{m}+1,\widehat{m}} \widehat{v}_{\widehat{m}+1} e_{\widehat{m}}^T (\lambda I - \widehat{H}_{\widehat{m}})^{-1} e_1 \right) \\ &= \beta_m(\lambda) \widehat{\beta}_{\widehat{m}}(\lambda) \widehat{v}_{\widehat{m}+1} \end{aligned}$$

mit

$$\widehat{\beta}_{\widehat{m}}(\lambda) = \widehat{h}_{\widehat{m}+1,\widehat{m}} e_{\widehat{m}}^T (\lambda I - \widehat{H}_{\widehat{m}})^{-1} e_1.$$

3.2 Restarts für Matrixfunktionen

Die Approximation an $z = f(A)b$ unter Verwendung der Approximation $x_m(\lambda)$ an $(\lambda I - A)x(\lambda) = b$ mit Hilfe der Cauchyintegralformel ist

$$\begin{aligned} z_m &= \frac{1}{2\pi i}\int_\Gamma f(\lambda)x_m(\lambda)\mathrm{d}\lambda \\ &= V_m\frac{1}{2\pi i}\int_\Gamma f(\lambda)(\lambda I - H_m)^{-1}e_1\mathrm{d}\lambda \\ &= V_m f(H_m)e_1. \end{aligned}$$

Eine Approximation $\widehat{z}_{\widehat{m}}$ nach einem Restart an $z = f(A)b$ bekommen wir durch Einsetzen der Approximation $\widehat{x}_{\widehat{m}}(\lambda)$ (siehe (3.8)) nach einem Restart an das verschobene lineare Gleichungssystem in die Cauchyintegraldarstellung. Es ist also

$$\begin{aligned} \widehat{z}_{\widehat{m}} &= \frac{1}{2\pi i}\int_\Gamma f(\lambda)\big[x_m(\lambda) + \widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda)\big]\mathrm{d}\lambda \\ &= z_m + \frac{1}{2\pi i}\int_\Gamma f(\lambda)\widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda)\mathrm{d}\lambda. \end{aligned} \tag{3.10}$$

Satz 3.2. *Die Darstellung* (3.10) *ist äquivalent zu*

$$\widehat{z}_{\widehat{m}} = \widetilde{V} f(\widetilde{H})e_1$$

mit

$$\widetilde{H} = \left[\begin{array}{c|c} H_m & 0 \\ \hline h_{m+1,m}e_1e_m^T & \widehat{H}_{\widehat{m}} \end{array}\right].$$

und $\widetilde{V} := [V_m \quad \widehat{V}_{\widehat{m}}]$.

Beweis. (3.10) ist äquivalent zu

$$\widehat{z}_{\widehat{m}} = \frac{1}{2\pi i}\int_\Gamma f(\lambda)[V_m y_m(\lambda) + \widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda)]\mathrm{d}\lambda, \tag{3.11}$$

wobei

$$(\lambda I - H_m)y_m(\lambda) = e_1 \tag{3.12}$$

gilt und nach (3.9)

$$(\lambda I - \widehat{H}_{\widehat{m}})\widehat{y}_{\widehat{m}}(\lambda) = \beta_m(\lambda)e_1 = h_{m+1,m}e_m^T y_m(\lambda)e_1. \tag{3.13}$$

Die Gleichungen (3.12) und (3.13) sind äquivalent zu

$$\left[\begin{array}{c|c} \lambda I - H_m & 0 \\ \hline -h_{m+1,m}e_1e_m^T & \lambda I - \widehat{H}_{\widehat{m}} \end{array}\right]\left[\begin{array}{c} y_m(\lambda) \\ \hline \widehat{y}_{\widehat{m}}(\lambda) \end{array}\right] = e_1$$

oder zu

$$\widetilde{y}(\lambda) = (\lambda I - \widetilde{H})^{-1} e_1$$

mit

$$\widetilde{y}(\lambda) = \begin{bmatrix} y_m(\lambda) \\ \widehat{y}_{\widehat{m}}(\lambda) \end{bmatrix} \quad \text{und} \quad \widetilde{H} = \left[\begin{array}{c|c} H_m & 0 \\ \hline h_{m+1,m} e_1 e_m^T & \widehat{H}_{\widehat{m}} \end{array}\right].$$

Mit $\widetilde{V} := [V_m \quad \widehat{V}_{\widehat{m}}]$ folgt aus (3.11)

$$\begin{aligned} \widehat{z}_{\widehat{m}} &= \frac{1}{2\pi i} \int_\Gamma f(\lambda) \widetilde{V} \widetilde{y}(\lambda) \mathrm{d}\lambda \\ &= \frac{1}{2\pi i} \widetilde{V} \int_\Gamma f(\lambda) (\lambda I - \widetilde{H})^{-1} e_1 \mathrm{d}\lambda \\ &= \widetilde{V} f(\widetilde{H}) e_1. \end{aligned}$$

Damit ist die Aussage gezeigt. □

Die Matrix $\widetilde{H}$ ist eine obere Hessenbergmatrix oder im Lanczosfall eine Tridiagonalmatrix. Ferner gilt die Relation

$$\begin{aligned} A\widetilde{V} &= [AV_m \quad A\widehat{V}_{\widehat{m}}] \\ &= [V_m H_m + h_{m+1,m} v_{m+1} e_m^T \quad \widehat{V}_{\widehat{m}} \widehat{H}_{\widehat{m}} + \widehat{h}_{\widehat{m}+1,\widehat{m}} \widehat{v}_{\widehat{m}+1} e_{\widehat{m}}^T] \\ &= [V_m \quad \widehat{V}_{\widehat{m}}] \left[\begin{array}{c|c} H_m & 0 \\ \hline h_{m+1,m} e_1 e_m^T & \widehat{H}_{\widehat{m}} \end{array}\right] + [0 \quad \widehat{h}_{\widehat{m}+1,\widehat{m}} \widehat{v}_{\widehat{m}+1} e_{\widehat{m}}^T] \\ &= \widetilde{V} \widetilde{H} + \widehat{h}_{\widehat{m}+1,\widehat{m}} \widehat{v}_{\widehat{m}+1} e_{m+\widehat{m}}^T. \end{aligned}$$

Eiermann und Ernst haben in [20] die Darstellung

$$\widehat{z}_{\widehat{m}} = z_m + \gamma_m \widehat{V}_{\widehat{m}} \Delta_0(\widehat{H}_{\widehat{m}}) e_1 \tag{3.14}$$

mit

$$\Delta_0(\lambda) := \frac{f(\lambda) - q_0(\lambda)}{\omega_m(\lambda)}$$

und $\gamma_m := \prod_{k=1}^{m} h_{k+1,k}$ für (3.10) angegeben. Im Folgenden zeigen wir, dass beide Darstellungen äquivalent sind.

Lemma 3.3. *Es gilt*

$$\beta_m(\lambda) = \frac{1}{\omega_m(\lambda)} \prod_{k=1}^{m} h_{k+1,k},$$

wobei ω_m das charakteristische Polynom von H_m ist.

Beweis. Wir betrachten nun

$$\beta_m(\lambda) = h_{m+1,m} e_m^T y_m(\lambda) = h_{m+1,m} e_m^T (\lambda I - H_m)^{-1} e_1.$$

Setze $U_\lambda := \lambda I - H_m$ mit $U_\lambda = [u_1, \ldots, u_m]$, damit ist $y_m(\lambda) = U_\lambda^{-1} e_1$ und nach der Cramer'schen Regel gilt

$$e_m^T y_m(\lambda) = \frac{\det([u_1, \ldots, u_{m-1}, e_1])}{\det(U_\lambda)}.$$

Das Nennerpolynom ist das charakteristische Polynom ω_m von H_m, denn

$$\det(U_\lambda) = \det(\lambda I - H_m) = \omega_m(\lambda).$$

Setze $[u_1, \ldots, u_{m-1}, e_1] =: [\bar{u}_1, \ldots, \bar{u}_m] = \bar{U}_\lambda$, dann ist

$$\bar{U}_\lambda = \left[\begin{array}{ccccc|c} & & & & & 1 \\ & & & & & 0 \\ & & \lambda I - H_{m-1} & & & \vdots \\ & & & & & 0 \\ \hline 0 & \ldots & 0 & -h_{m,m-1} & & 0 \end{array}\right].$$

Laut dem Entwicklungssatz von Laplace gilt bei der Entwicklung nach der m-ten Spalte von $\bar{U}_\lambda$

$$\det(\bar{U}_\lambda) = \sum_{i=1}^{m} (-1)^{i+m} \bar{u}_{i,m} \det\left[\left(\bar{U}'_\lambda\right)_{i,m}\right],$$

wobei $\left(\bar{U}'_\lambda\right)_{i,m} \in \mathbb{C}^{m-1,m-1}$ durch Streichung der i-ten Zeile und m-ten Spalte von $\bar{U}_\lambda$ entsteht und $\bar{u}_{i,j}$ der (i,j)-te Eintrag von $\bar{U}_\lambda$ ist. Da aber $\bar{u}_{i,m} = 0$ für $j = 2, \ldots, m$ ist, gilt

$$\det(\bar{U}_\lambda) = (-1)^{1+m} \bar{u}_{1,m} \det((\bar{U}'_\lambda)_{1,m}) = (-1)^{m+1} \det((\bar{U}'_\lambda)_{1,m})$$

mit

$$\left(\bar{U}'_\lambda\right)_{1,m} = \lambda \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ \vdots & & & \ddots & 1 \\ 0 & \ldots & \ldots & \ldots & 0 \end{bmatrix} - \begin{bmatrix} h_{2,1} & \ldots & \ldots & \ldots & h_{2,m} \\ 0 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \ldots & \ldots & 0 & h_{m,m-1} \end{bmatrix}.$$

Mit

$$\det((\bar{U}'_\lambda)_{1,m}) = \prod_{k=1}^{m-1} (-h_{k+1,k}) = (-1)^{m-1} \prod_{k=1}^{m-1} h_{k+1,k}$$

gilt nun

$$\det(\bar{U}_\lambda) = (-1)^{m+1} (-1)^{m-1} \prod_{k=1}^{m-1} h_{k+1,k} = \prod_{k=1}^{m-1} h_{k+1,k}.$$

Somit ist

$$\begin{aligned}\beta_m(\lambda) &= h_{m+1,m} e_m^T y_m(\lambda) \\ &= h_{m+1,m} \frac{\det(\bar{U}_\lambda)}{\det(U_\lambda)} \\ &= \frac{1}{\omega_m(\lambda)} \prod_{k=1}^{m} h_{k+1,k}.\end{aligned}$$

□

Im Beweis des nächsten Satzes verwenden wir Resultate aus Davis [15, Kapitel 3.6].

Satz 3.4. *Sei q ein Polynom vom Grad $\leq m$, dann gilt*

$$\frac{1}{2\pi i} \int_\Gamma \frac{q(t)}{(t-z)(t-z_0)\dots(t-z_m)} \mathrm{d}t = 0,$$

wobei die Pole $z_0, \dots, z_m$ im Gebiet liegen, das von der Kurve Γ umschlossen wird.

Beweis. Seien $z_0, \dots, z_m$ paarweise verschieden,

$$I = \frac{1}{2\pi i} \int_\Gamma \frac{q(t)}{\omega(t)} \mathrm{d}t$$

und

$$\omega(t) = (t-z_0)(t-z_1)\dots(t-z_m).$$

Da die Nullstellen von ω einfach sind und q analytisch in Γ ist, gilt nach dem Residuensatz

$$I = \sum_{k=0}^{m} \frac{q(z_k)}{\omega'(z_k)}.$$

Dies ist gerade die Definition der dividierten Differenzen, also gilt

$$I = \delta^m q[z_0, \dots, z_m].$$

Ebenso gilt dann

$$\delta^{m+1} q[z, z_0, \dots, z_m] = \frac{1}{2\pi i} \int_\Gamma \frac{q(t)}{(t-z)(t-z_0)\dots(t-z_m)} \mathrm{d}t.$$

Da

$$\delta^{m+1} q[z, z_0, \dots, z_m] = \frac{q^{(m+1)}(\zeta)}{(m+1)!}$$

für ein $\zeta \in (\min(z, z_i), \max(z, z_i))$ gilt, folgt wegen $q^{(m+1)} \equiv 0$ die Behauptung. Sind die Pole $z_0, \dots, z_m$ nicht alle paarweise verschieden, gilt die Aussage weiterhin, vergleiche hierfür [15, Korollar 3.6.3]. □

Nun gilt für die Approximation (3.10) mit Lemma 3.3 auch die folgende Darstellung:

$$\begin{aligned}\widehat{z}_{\widehat{m}} &= z_m + \frac{1}{2\pi i}\int_\Gamma f(\lambda)\widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda)\mathrm{d}\lambda \\ &= z_m + \frac{1}{2\pi i}\int_\Gamma f(\lambda)\widehat{V}_{\widehat{m}}(\lambda I - \widehat{H}_{\widehat{m}})^{-1}\beta_m(\lambda)e_1\mathrm{d}\lambda \\ &= z_m + \widehat{V}_{\widehat{m}}\prod_{k=1}^{m} h_{k+1,k}\frac{1}{2\pi i}\int_\Gamma \frac{f(\lambda)}{\omega_m(\lambda)}(\lambda I - \widehat{H}_{\widehat{m}})^{-1}e_1\mathrm{d}\lambda.\end{aligned}$$

Mit $\gamma_m = \prod_{k=1}^{m} h_{k+1,k}$ und Satz 3.4 gilt dann weiter

$$\widehat{z}_{\widehat{m}} = z_m + \gamma_m\widehat{V}_{\widehat{m}}\frac{1}{2\pi i}\int_\Gamma \frac{f(\lambda) - q_0(\lambda)}{\omega_m(\lambda)}(\lambda I - \widehat{H}_{\widehat{m}})^{-1}e_1\mathrm{d}\lambda, \tag{3.15}$$

wobei $q_0 \in \mathbb{P}_{m-1}$ das Interpolationspolynom von f an den Nullstellen von $\omega_m \in \mathbb{P}_m$ ist. Da die Nullstellen $\lambda_1, \ldots, \lambda_m$ von ω_m hebbare Polstellen von

$$\Delta_0(\lambda) = \frac{f(\lambda) - q_0(\lambda)}{\omega_m(\lambda)}$$

sind, kann man die Cauchy Integralformel auf (3.15) anwenden. Es gilt also die Formel (3.14), die Eiermann und Ernst [20] hergeleitet haben.

Um mehrere Restarts zu betrachten, führen wir nun den Index $^{(i)}$ ein, der den i-ten Restart kennzeichnen soll. Falls noch nicht neu gestartet wurde, ist $i = 0$. Allgemein gilt der folgende Satz, dessen Beweis des Satzes auch von Eiermann und Ernst in [20] in ähnlicher Form angeführt wurde.

Satz 3.5. *Die Approximation an $z = f(A)b$ ist nach m_k Krylovschritten des k-ten Restarts*

$$z_{m_k}^{(k)} = \widetilde{V}^{(k)} f(\widetilde{H}^{(k)})e_1 \tag{3.16}$$

$$= z_{m_{k-1}}^{(k-1)} + V_{m_k}^{(k)}\big(f(\widetilde{H}^{(k)})e_1\big)_{\nu_{k-1}+1:\nu_k}. \tag{3.17}$$

Dabei ist

$$\widetilde{H}^{(k)} = \left[\begin{array}{c|c} \widetilde{H}^{(k-1)} & 0 \\ \hline h_{m_{k-1}+1,m_{k-1}}^{(k-1)} e_1 e_{\nu_{k-1}}^T & H_{m_k}^{(k)} \end{array}\right],$$

$\widetilde{V}^{(k)} = \begin{bmatrix}\widetilde{V}^{(k-1)} & V_{m_k}^{(k)}\end{bmatrix}$ *und* $\nu_k = \sum_{j=0}^{k} m_j$. *Für* $k = 0$ *sei* $\widetilde{H}^{(0)} = H_m$, $\widetilde{V}^{(0)} = V_m$ *und* $\nu_0 = m$. *Hierbei ist* $\widetilde{H}^{(k)}$ *eine obere Hessenbergmatrix bzw. im Lanczosfall eine Tridiagonalmatrix.*

Beweis. Gleichung (3.16) folgt direkt aus Satz 3.2. Es gilt

$$f(\widetilde{H}^{(k)}) = \left[\begin{array}{c|c} f(\widetilde{H}^{(k-1)}) & 0 \\ \hline \widetilde{M} & f(H_{m_k}^{(k)}) \end{array}\right]$$

mit einer bestimmten Matrix $\widetilde{M}$, die wir aber für den Beweis nicht kennen müssen. Dann gilt

$$\begin{aligned} \widetilde{V}^{(k)} f(\widetilde{H}^{(k)}) e_1 &= \begin{bmatrix} \widetilde{V}^{(k-1)} & V_{m_k}^{(k)} \end{bmatrix} \left[\begin{array}{c} f(\widetilde{H}^{(k-1)}) e_1 \\ \hline \big(f(\widetilde{H}^{(k)}) e_1\big)_{\nu_{k-1}+1:\nu_k} \end{array}\right] \\ &= z_{m_{k-1}}^{(k-1)} + V_{m_k}^{(k)} \big(f(\widetilde{H}^{(k)}) e_1\big)_{\nu_{k-1}+1:\nu_k}. \end{aligned}$$

Somit ist auch (3.17) gezeigt. □

Satz 3.5 gibt den für die Praxis entscheidenden Hinweis, dass man nur die aktuelle Basis $V_{m_k}^{(k)}$ und die letzte Näherung $z_{m_{k-1}}^{(k-1)}$ speichern muss, aber nicht die Basen aller früheren Zykeln. Die Matrix $\widetilde{H}^{(k-1)} \in \mathbb{C}^{\nu_{k-1},\nu_{k-1}}$ muss allerdings komplett gespeichert werden. Der Speicheraufwand hierfür ist aber im Vergleich zu dem Speicherplatz von $\widetilde{V}^{(k-1)}$ sehr gering, da in allen relevanten Anwendungen $\nu_{k-1} \ll N$ gilt. Zur Berechnung von $z_{m_k}^{(k)}$ ist der wesentliche Speicheraufwand durch die Vektoren $v_1^{(k)}, \ldots, v_{m_k+1}^{(k)}$ und $z_{m_{k-1}}^{(k-1)}$ gegeben. Falls man alle m_i gleich setzt, also $m := m_1 = m_2 = \ldots$, dann muss man im Restartverfahren maximal $m+2$ Vektoren der Dimension N speichern.

3.3 Konvergenz und Fehlerschranken

Das Ziel dieses Abschnittes ist es, die Konvergenz des Restartverfahrens zu zeigen und dessen Fehler

$$\epsilon_{m_k}^{(k)} := \|f(A)b - z_{m_k}^{(k)}\| = \left\| \frac{1}{2\pi i} \int_\Gamma f(\lambda) \big((\lambda I - A)^{-1} b - \sum_{i=0}^{k} V_{m_i}^{(i)} y_{m_i}^{(i)}(\lambda) \big) \, \mathrm{d}\lambda \right\|$$

abzuschätzen. Dazu sei E eine konvexe, beschränkte Menge in der komplexen Ebene und sei ϕ die konforme Abbildung, die das Äußere von E auf das Äußere des Einheitskreises $|w| > 1$ überträgt, wobei $\phi(z) = z/\rho + O(1)$ ist, falls $z \to \infty$ für ein $\rho > 0$. Außerdem sei Γ der Rand des stückweise glatten und beschränkten Gebietes G, welches E enthält. Des Weiteren sei f analytisch in G und konstant auf dem Abschluss von G.

Satz 3.6. *[36, Hochbruck, Lubich] Sei $\mathbb{P}_m$ der Raum aller Polynome mit maximalen Grad m. Dann gilt unter den obigen Voraussetzungen und für eine Matrix $A \in \mathbb{C}^{N,N}$, deren Wertebereich in E liegt,*

$$\min_{\substack{p \in \mathbb{P}_m \\ p(\lambda)=1}} \|p(A)\| \leq M |\phi(\lambda)|^{-m}$$

mit $M = \frac{3l(\partial E)}{2\pi d(\partial E)}$. *Hierbei ist* $l(\partial E)$ *die Länge des Randes von* ∂E *und* $d(S)$ *der minimale Abstand zwischen* $\mathcal{F}(A)$ *und einer Teilmenge* S *der komplexen Ebene. Falls* E *ein Intervall ist, gilt* $M = 3$.

Nun kann man folgendes Lemma zeigen.

Lemma 3.7. *Mit den Voraussetzungen und Notationen aus Satz 3.6 gilt*

$$|\beta_{m_i}^{(i)}(\lambda)| \leq M\|P_{m_i}^{(i)}(\lambda)\||\phi(\lambda)|^{-m_i}.$$

mit dem Projektor $P_{m_i}^{(i)}(\lambda) = I - V_{m_i+1}^{(i)}(\lambda I - \bar{H}_{m_i}^{(i)})(\lambda I - H_{m_i}^{(i)})^{-1}{V_{m_i}^{(i)}}^H$.

Beweis. Zur Erinnerung:

$$(\lambda I - A)V_{m_i}^{(i)} = V_{m_i}^{(i)}(\lambda I - H_{m_i}^{(i)}) - h_{m_i+1,m_i}^{(i)} v_{m_i+1}^{(i)} e_{m_i}^T.$$

Sei

$$\widetilde{r}_{m_i}^{(i)}(\lambda) := v_1^{(i)} - (\lambda I - A)V_{m_i}^{(i)}\widetilde{y}_{m_i}^{(i)}(\lambda)$$

mit

$$\widetilde{y}_{m_i}^{(i)}(\lambda) = (\lambda I - H_{m_i}^{(i)})^{-1}e_1.$$

Dann gilt mit Lemma 3.1

$$\|\widetilde{r}_{m_i}^{(i)}(\lambda)\| = |\beta_{m_i}^{(i)}(\lambda)|.$$

Ebenfalls gilt

$$\begin{aligned}\|\widetilde{r}_{m_i}^{(i)}(\lambda)\| &= \|v_1^{(i)} - (\lambda I - A)V_{m_i}^{(i)}\widetilde{y}_{m_i}^{(i)}\| \\ &= \|(I - V_{m_i+1}^{(i)}(\lambda I - \bar{H}_{m_i}^{(i)})(\lambda I - H_{m_i}^{(i)})^{-1}{V_{m_i}^{(i)}}^H)v_1^{(i)}\| \\ &= \|P_{m_i}^{(i)}(\lambda)v_1^{(i)}\|\end{aligned}$$

mit

$$\bar{H}_{m_i}^{(i)} = \begin{bmatrix} H_{m_i}^{(i)} \\ h_{m_i+1,m_i}^{(i)} e_{m_i}^T \end{bmatrix}.$$

Wegen

$$P_{m_i}^{(i)}(\lambda)(\lambda I - A)V_{m_i}^{(i)} = 0$$

gilt für beliebiges $y \in \mathbb{C}^{m_i}$

$$\begin{aligned}\|\widetilde{r}_{m_i}^{(i)}(\lambda)\| &= \|P_{m_i}^{(i)}(\lambda)v_1^{(i)} - (\lambda I - A)V_{m_i}^{(i)}y\| \\ &= \|P_{m_i}^{(i)}(\lambda)p_{m_i}^{(i)}(A)v_1^{(i)}\| \\ &\leq \|P_{m_i}^{(i)}(\lambda)\|\|p_{m_i}^{(i)}(A)\|,\end{aligned}$$

wobei $p_{m_i}^{(i)}$ ein beliebiges Polynom vom Grad $\leq m_i$ mit $p_{m_i}^{(i)}(\lambda) = 1$ ist. Aus Satz 3.6 folgt

$$|\beta_{m_i}^{(i)}(\lambda)| \leq \|P_{m_i}^{(i)}(\lambda)\|\|p_{m_i}^{(i)}(A)\| \leq M\|P_{m_i}^{(i)}(\lambda)\||\phi|^{-m_i}.$$

Damit ist das Lemma bewiesen. □

Jetzt kann man die Konvergenz des Restartverfahrens zeigen.

Satz 3.8. *Mit den Voraussetzungen und Notationen aus Satz 3.6 gilt für den Fehler nach m_k Schritten des k-ten Restarts der Krylovapproximation an $f(A)b$, wenn für den i-ten Restart m_i Schritte, $i = 0, \ldots, k-1$, benötigt wurden,*

$$\|f(A)b - z_{m_k}^{(k)}\| \leq \frac{M^k}{d(\Gamma)\pi} \int_\Gamma |f(\lambda) - q_{\nu_k - 1}(\lambda)| \cdot \left(\prod_{i=0}^{k-1} \|P_{m_i}^{(i)}(\lambda)\| \right) \cdot |\phi(\lambda)|^{-\nu_k} \ |\mathrm{d}\lambda|$$

mit $\nu_k = \sum_{i=1}^{k} m_k$.

Beweis. Sei

$$\begin{aligned} d_{\nu_k}(\lambda) :&= (\lambda I - A)^{-1} b - V_{m_0}^{(0)} y_{m_0}^{(0)}(\lambda) - \sum_{i=1}^{k} V_{m_i}^{(i)} y_{m_i}^{(i)}(\lambda) \\ &= (\lambda I - A)^{-1} V_{m_0}^{(0)} e_1 - V_{m_0}^{(0)} (\lambda I - H_{m_0}^{(0)})^{-1} e_1 - \sum_{i=1}^{k} V_{m_i}^{(i)} y_{m_i}^{(i)}(\lambda). \end{aligned}$$

Wegen

$$(\lambda I - A) V_{m_0}^{(0)} = V_{m_0}^{(0)} (\lambda I - H_{m_0}^{(0)}) - h_{m_0+1, m_0}^{(0)} v_{m_0+1}^{(0)} e_{m_0}^T$$

gilt

$$(\lambda I - A)^{-1} V_{m_0}^{(0)} = V_{m_0}^{(0)} (\lambda I - H_{m_0}^{(0)})^{-1} + h_{m_0+1, m_0}^{(0)} (\lambda I - A)^{-1} v_{m_0+1}^{(0)} e_{m_0}^T (\lambda I - H_{m_0}^{(0)})^{-1}$$

und damit

$$\begin{aligned} d_{\nu_k}(\lambda) &= h_{m_0+1, m_0}^{(0)} e_{m_0}^T (\lambda I - H_{m_0}^{(0)})^{-1} e_1 (\lambda I - A)^{-1} v_{m_0+1}^{(0)} - \sum_{i=1}^{k} V_{m_i}^{(i)} y_{m_i}^{(i)}(\lambda) \\ &= \beta_{m_0}^{(0)}(\lambda) (\lambda I - A)^{-1} v_{m_0+1}^{(0)} - V_{m_1}^{(1)} (\lambda I - H_{m_1}^{(1)})^{-1} \beta_{m_0}^{(0)}(\lambda) e_1 - \sum_{i=2}^{k} V_{m_i}^{(i)} y_{m_i}^{(i)}(\lambda) \\ &= \beta_{m_0}^{(0)}(\lambda) \big((\lambda I - A)^{-1} V_{m_1}^{(1)} e_1 - V_{m_1}^{(1)} (\lambda I - H_{m_1}^{(1)})^{-1} e_1 \big) - \sum_{i=2}^{k} V_{m_i}^{(i)} y_{m_i}^{(i)}(\lambda) \\ &= \beta_{m_0}^{(0)}(\lambda) \big(\beta_{m_1}^{(1)}(\lambda) (\lambda I - A)^{-1} V_{m_2}^{(2)} e_1 \big) - \sum_{i=2}^{k} V_{m_i}^{(i)} y_{m_i}^{(i)}(\lambda) \\ &= \kappa_{k-1}(\lambda) \big((\lambda I - A)^{-1} V_{m_k}^{(k)} e_1 - V_{m_k}^{(k)} (\lambda I - H_{m_k}^{(k)})^{-1} e_1 \big), \end{aligned}$$

wobei

$$\kappa_{k-1}(\lambda) := \prod_{i=0}^{k-1} \beta_{m_i}^{(i)}(\lambda)$$

ist. Setzt man $\widetilde{v} := v_{m_{k-1}+1}^{(k-1)} = V_{m_k}^{(k)} e_1$, dann ist mit

$$\Delta_{m_k}(\lambda) := (\lambda I - A)^{-1} - V_{m_k}^{(k)} (\lambda I - H_{m_k}^{(k)})^{-1} V_{m_k}^{(k)^T}$$

nun

$$d_{\nu_k}(\lambda) = \kappa_{k-1}(\lambda) \Delta_{m_k} \widetilde{v}.$$

Wegen $\Delta_{m_k}(\lambda)(\lambda I - A) V_{m_k}^{(k)} = 0$ gilt dann

$$\begin{aligned} d_{\nu_k}(\lambda) &= \kappa_{k-1}(\lambda) \Delta_{m_k}(\lambda) \big(\widetilde{v} - (\lambda I - A) V_{m_k}^{(k)} y\big), \qquad \forall y \in \mathbb{C}^{m_k} \\ &= \kappa_{k-1}(\lambda) \Delta_{m_k}(\lambda) p_{m_k}(A) \widetilde{v}, \end{aligned}$$

wobei $p_{m_k} \in \mathbb{P}_{m_k}$ ein beliebiges Polynom mit $p_{m_k}(\lambda) = 1$ ist. Da

$$\|(\lambda I - A)\| \leq \operatorname{dist}(\lambda, \mathcal{F}(A))^{-1}$$

und

$$\|(\lambda I - H_{m_k}^{(k)})\| \leq \operatorname{dist}(\lambda, \mathcal{F}(A))^{-1}$$

ist, gilt

$$\|\Delta_{m_k}(\lambda)\| \leq 2d(\Gamma)^{-1}. \tag{3.18}$$

Somit ist

$$\begin{aligned} \|d_{\nu_k}(\lambda)\| &\leq |\kappa_{k-1}(\lambda)| 2d(\Gamma)^{-1} \|p_{m_k}(A)\| \\ &\leq |\kappa_{k-1}(\lambda)| \frac{2M}{d(\Gamma)} |\phi(\lambda)|^{-m_k}, \end{aligned}$$

wobei man für die letzte Abschätzung Satz 3.6 verwendet. Mit Lemma 3.7 gilt

$$\begin{aligned} |\kappa_{k-1}(\lambda)| &\leq \prod_{i=0}^{k-1} |\beta_{m_i}^{(i)}(\lambda)| \\ &\leq M^k \prod_{i=0}^{k-1} \|P_{m_i}^{(i)}(\lambda)\| |\phi(\lambda)|^{-m_i} \\ &= M^k |\phi(\lambda)|^{-\nu_{k-1}} \prod_{i=0}^{k-1} \|P_{m_i}^{(i)}(\lambda)\|. \end{aligned}$$

Setzt man

$$C_p^{(k-1)}(\lambda) := \prod_{i=0}^{k-1} \|P_{m_i}^{(i)}(\lambda)\|,$$

dann ist

$$\|d_{\nu_k}(\lambda)\| \leq \frac{2}{d(\Gamma)} M^{k+1} |\phi(\lambda)|^{-\nu_k} C_p^{(k-1)}(\lambda)$$

und daraus folgt

$$\begin{aligned}\|f(A)b - z_{m_k}^{(k)}\| &\leq \frac{1}{2\pi}\int_\Gamma |f(\lambda)| \cdot \|d_{\nu_k}(\lambda)\| \ |\mathrm{d}\lambda| \\ &\leq \frac{M^{k+1}}{d(\Gamma)\pi}\int_\Gamma |f(\lambda)| \cdot C_p^{(k-1)}(\lambda) \cdot |\phi(\lambda)|^{-\nu_k} \ |\mathrm{d}\lambda|. \end{aligned} \tag{3.19}$$

Es gilt aber auch

$$\begin{aligned} z_{m_k}^{(k)} &= \widetilde{V}^{(k)} f(\widetilde{H}^{(k)}) e_1 \\ &= \widetilde{V}^{(k)} q_{\nu_k - 1}(\widetilde{H}^{(k)}) e_1 \\ &= q_{\nu_k - 1}(A) b \end{aligned}$$

für ein $q_{\nu_k-1} \in \mathbb{P}_{\nu_k-1}$, so dass

$$\begin{aligned} f(A)b - z_{m_k}^{(k)} &= f(A)b - \widetilde{V}^{(k)} f(\widetilde{H}^{(k)}) e_1 \\ &= f(A)b - \widetilde{V}^{(k)} f(\widetilde{H}^{(k)}) e_1 + \widetilde{V}^{(k)} q_{\nu_k-1}(\widetilde{H}^{(k)}) e_1 - \widetilde{V}^{(k)} q_{\nu_k-1}(\widetilde{H}^{(k)}) e_1 \\ &= f(A)b - \widetilde{V}^{(k)} f(\widetilde{H}^{(k)}) e_1 + \widetilde{V}^{(k)} q_{\nu_k-1}(\widetilde{H}^{(k)}) e_1 - q_{\nu_k-1}(A) b \\ &= (f - q_{\nu_k-1})(A)b - \widetilde{V}^{(k)} (f - q_{\nu_k-1})(\widetilde{H}^{(k)}) e_1. \end{aligned}$$

Also kann man statt der Abschätzung (3.19) auch

$$\|f(A)b - z_{m_k}^{(k)}\| \leq \frac{M^{k+1}}{d(\Gamma)\pi}\int_\Gamma |f(\lambda) - q_{\nu_k-1}(\lambda)| \cdot C_p^{(k-1)}(\lambda) \cdot |\phi(\lambda)|^{-\nu_k} \ |\mathrm{d}\lambda| \tag{3.20}$$

verwenden. □

Die Norm des Projektors $P_{m_i}^{(i)}(\lambda)$ lässt sich durch

$$\|P_{m_i}^{(i)}(\lambda)\| \leq 1 + \frac{|h_{m_i+1,m_i}^{(i)}|}{d(\Gamma)}$$

abschätzen, damit gilt das folgende Korollar:

Korollar 3.9. *Mit den Voraussetzungen und Notationen aus Satz 3.6 gilt für den Fehler nach m_k Schritten des k-ten Restarts der Krylovapproximation an $f(A)b$, wenn für den i-ten Restart m_i Schritte mit $i = 0, \ldots, k-1$ benötigt wurden,*

$$\|f(A)b - z_{m_k}^{(k)}\| \leq \frac{M^k c_P^{(k-1)}}{d(\Gamma)\pi}\int_\Gamma |f(\lambda) - q_{\nu_k-1}(\lambda)| \cdot |\phi(\lambda)|^{-\nu_k} \ |\mathrm{d}\lambda| \tag{3.21}$$

mit $\nu_k = \sum_{i=1}^{k} m_k$ *und*

$$c_P^{(k-1)} = \prod_{i=0}^{k-1}\left(1 + \frac{|h_{m_i+1,m_i}^{(i)}|}{d(\Gamma)}\right).$$

Die Abschätzung (3.21) kann man umschreiben als

$$\begin{aligned}\|f(A)b - z_{m_k}^{(k)}\| &\leq M^k c_p^{(k-1)} \frac{3l(\partial E)}{2\pi^2 d(\partial E) d(\Gamma)} \int_\Gamma |f(\lambda) - q_{\nu_k - 1}(\lambda)| \cdot |\phi(\lambda)|^{-\nu_k} \, |\mathrm{d}\lambda| \\ &\leq M^k c_p^{(k-1)} \Psi(\nu_k)\end{aligned}$$

mit

$$\Psi(\nu) = \frac{l(\partial E)}{2\pi d(\partial E) d(\Gamma)} \int_\Gamma |f(\lambda) - q_{\nu-1}(\lambda)| \cdot |\phi(\lambda)|^{-\nu} \, |\mathrm{d}\lambda|.$$

Dieses $\Psi(\nu)$ entspricht gerade der Abschätzung des Fehlers $\|f(A)b - V_\nu f(H_\nu)e_1\|$ für den Krylovprozess ohne Restarts, die in Lemma 1 der Arbeit von Hochbruck und Lubich [36] gezeigt wurde. Nun können wir alle Approximationsresultate dieser Arbeit nehmen, sie mit M^k multiplizieren und dann für die Restartverfahren anwenden. Die folgenden Resultate sind genau auf diese Weise aus der Arbeit [36] übernommen worden und werden deshalb nicht noch einmal explizit bewiesen. In beiden Fällen ist E ein Intervall und somit ist $M = 3$.

Satz 3.10. *Sei A eine hermitsche, negative semidefinite Matrix mit Eigenwerten in dem Intervall $[-4\rho, 0]$. Dann kann der Fehler $\epsilon_{m_k}^{(k)} := \|e^{hA}b - z_{m_k}^{(k)}\|$ nach m_k Schritten des k-ten Restarts der Krylovapproximation von $e^{hA}b$, wenn für den i-ten Restart m_i Schritte mit $i = 0, \ldots, k-1$ benötigt wurden, auf folgende Weise abgeschätzt werden:*

$$\epsilon_{m_k}^{(k)} \;\leq\; 3^k \gamma_1^{(k)} 10 e^{-\nu_k^2/(5\rho h)}, \qquad \sqrt{4\rho h} \leq \nu_k \leq 2\rho h \tag{3.22}$$

$$\epsilon_{m_k}^{(k)} \;\leq\; 3^k \gamma_2^{(k)} 10 (\rho h)^{-1} e^{-\rho h} \left(\frac{e\rho h}{\nu_k}\right)^{\nu_k}, \qquad \nu_k \geq 2\rho h \tag{3.23}$$

mit

$$\gamma_1^{(k)} = \prod_{i=0}^{k-1} \left(1 + \frac{4|h_{m_i+1,m_i}^{(i)}|\rho h^2}{\nu_k^2}\right), \qquad \gamma_2^{(k)} = \prod_{i=0}^{k-1} \left(1 + \frac{|h_{m_i+1,m_i}^{(i)}|}{\rho(\frac{\nu_k}{\rho h} + \frac{\rho h}{\nu_k} - 2)}\right)$$

und $\nu_k = \sum_{i=1}^{k} m_i$.

Bemerkung: Eine schärfere Fehlerschranke für (3.22) (mit $\beta > 0.92$) ist

$$\epsilon_{m_k}^{(k)} \leq 3^k \gamma_1^{(k)} \left(12\frac{\rho h}{\nu_k^2} + 8\frac{\sqrt{\rho h}}{\nu_k}\right) e^{-\beta\nu_k^2/(4\rho h)} \tag{3.24}$$

und für (3.23)

$$\epsilon_{m_k}^{(k)} \leq 3^k \gamma_2^{(k)} \left(5(\rho h)^{-1} + 3\sqrt{\pi}(\rho h)^{-1/2}\right) e^{(\rho h)^2/\nu_k} e^{-2\rho h} \left(\frac{e\rho h}{\nu_k}\right)^{\nu_k}. \tag{3.25}$$

Bemerkung: In diesem Fall ist eine grobe Abschätzung für $|h_{m_i+1,m_i}^{(i)}|$ für alle i durch

$$|h_{m_i+1,m_i}^{(i)}| \leq \|A\| \leq 4\rho$$

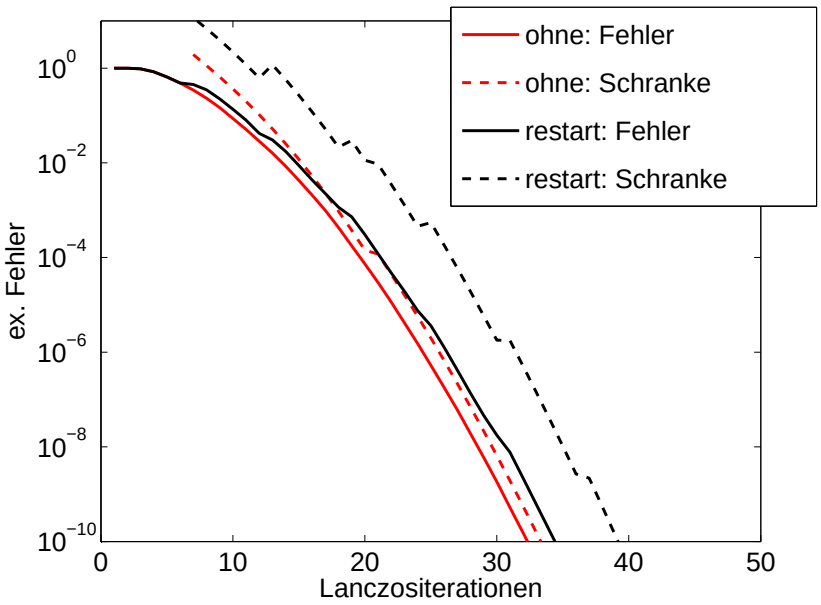

Abbildung 3.1: Fehler und Fehlerschranken für das symmetrische Beispiel 3.1: ohne Restart und Restart nach 6 Lanczositerationen

gegeben.

Beispiel 3.1. Um ein Beispiel dieser Fehlerschranke anzuführen, nehmen wir eine Diagonalmatrix $A \in \mathbb{C}^{N,N}$ mit $N = 1000$, deren äquidistante Eigenwerte auf dem Intervall $[-40, 0]$ liegen, einen Zufallsvektor $b \in \mathbb{R}^N$ und berechnen die Lanczosapproximation an $e^A b$. Die Abbildung 3.1 zeigt den exakten Fehler und die zugehörige Fehlerschranke pro Lanczositeration für den Lanczosprozess ohne Restarts (rot) und für den Lanczosprozess, der nach jeweils 6 Krylovschritten neu gestartet wird (schwarz). Die Fehlerschranken (3.24) und (3.25) werden durch entsprechende gestrichelte Linien dargestellt. Hierbei verwenden wir nicht $\gamma_1^{(k)}$ und $\gamma_2^{(k)}$, sondern berechnen $\max\limits_{\lambda \in \Gamma} \|P_{m_i}^{(i)}(\lambda)\|$ exakt. $\diamond$

Man sieht, dass für dieses Beispiel die Fehlerschranken für das Restartverfahren nicht ganz so gut sind wie für das Lanczosverfahren ohne Restarts, aber dennoch hinreichend scharf.

Als Anmerkung wird folgender Satz angegeben, er kann mit dem Beweis von Lemma 1 in [36] und obigen Resultaten gezeigt werden.

Satz 3.11. *Sei A eine schiefhermitsche Matrix mit Eigenwerten im Intervall $i[-2\rho,\ 2\rho]$. Dann kann man den Fehler nach m_k Schritten des k-ten Restarts der Krylovapproximation von $e^{hA}b$, wenn für den i-ten Restart m_i Schritte mit $i = 0, \ldots, k-1$ benötigt wurden, durch*

$$\epsilon_{m_k}^{(k)} \leq 3^k \gamma^{(k)} 12 e^{-(\rho h)^2/\nu_k} \left(\frac{e\rho h}{\nu_k}\right)^{\nu_k}, \quad \nu_k \geq 2\rho h \tag{3.26}$$

mit

$$\gamma^{(k)} = \prod_{i=0}^{k-1} \left(1 + \frac{|h_{m_i+1,m_i}|}{\rho(4 + (\frac{\nu_k}{\rho h} - \frac{\rho h}{\nu_k})^2)} \right)$$

und $\nu_k = \sum_{i=1}^{k} m_i$ *beschränken.*

Bemerkung: Eine schärfere Fehlerschranke für (3.26) ist gegeben durch

$$\epsilon_{m_k}^{(k)} \leq 3^{k-1}\gamma^{(k)} \left(4(\rho h)^{-1} + 11(\rho h)^{-1/2} \right) e^{-(\rho h)^2/\nu_k} \left(\frac{e\rho h}{\nu_k} \right)^{\nu_k}, \quad \nu_k \geq 2\rho h. \tag{3.27}$$

Bemerkung: In diesem Fall ist eine grobe Abschätzung für $|h_{m_i+1,m_i}^{(i)}|$ für alle i durch

$$|h_{m_i+1,m_i}^{(i)}| \leq \|A\| \leq 2\rho$$

gegeben.

3.4 Deflated Restarts

In diesem Abschnitt wird ein Verfahren vorgestellt, dass bei den jeweiligen Restarts im Krylovverfahren noch Ritzvektoren der aktuellen Approximation verwendet. Restartverfahren zur Lösung linearer Gleichungssysteme, die Ritzvektoren zu den Krylovräumen hinzufügen, nennt man *Deflated* oder *Thick Restart* Verfahren. Deshalb nennen wir unser Verfahren Krylovverfahren mit Deflated Restarts. Wir verwenden dazu ein Deflated FOM Verfahren (FOM-DR) von R. Morgan [56] für lineare Gleichungssysteme. Dieses FOM-DR Verfahren verwendet die *Thick-Restart-Technik* von Wu und Simon [86]. Wir verallgemeinern die FOM-DR Methode für verschobene lineare Gleichungssysteme ähnlich wie im Abschnitt 3.1 und nutzen sie für die Matrixfunktion.

Wieder betrachten wir zuerst wegen der besseren Darstellung nur einen Restart und verallgemeinern das Verfahren später für mehrere Restarts. Für die verschobenen linearen Gleichungssystemen

$$(\lambda I - A)x(\lambda) = b \tag{3.28}$$

erhalten wir wieder die Gleichungen (3.4), (3.5), (3.6) und (3.7). Nun berechnet man zu k Eigenwerten von H_m, falls diese diagonalisierbar ist, die Eigenvektoren $\widetilde{u}_1, \ldots, \widetilde{u}_k$. Zum Beispiel betrachtet man bei einem parabolischen Problem die größten Eigenwerte, da diese die Lösung stark bestimmen. Man setzt $\widetilde{U}_k := [\widetilde{u}_1, \ldots, \widetilde{u}_k] \in \mathbb{C}^{m,k}$ und D_k als Diagonalmatrix mit den zugehörigen Eigenwerten auf der Diagonalen, so dass $H_m\widetilde{U}_k = \widetilde{U}_k D_k$ ist. Nun orthonormiert man $\widetilde{U}_k$, das heißt, man berechnet die QR-Zerlegung $\widetilde{U}_k = U_k R_k$ mit einer orthonormalen Matrix U_k und einer oberen Dreiecksmatrix R_k. Damit definiert man

$$\widehat{V}_k = V_m U_k$$

und

$$\widehat{H}_k = U_k^H H_m U_k.$$

Es gilt die Relation

$$\begin{aligned} A\widehat{V}_k &= AV_mU_k \\ &= V_mH_mU_k + h_{m+1,m}v_{m+1}e_m^TU_k \\ &= \widehat{V}_kU_k^HH_mU_k + h_{m+1,m}v_{m+1}e_m^TU_k \\ &= \widehat{V}_k\widehat{H}_k + h_{m+1,m}v_{m+1}e_m^TU_k. \end{aligned}$$

Man setzt jetzt $\widehat{v}_{k+1} = v_{m+1}$ und wendet wieder Kryloviterationen an, um $\widehat{V}$ und $\widehat{H}$ zu vergrößern. Dann erhält man die Relation

$$(\lambda I - A)\widehat{V}_{\widehat{m}} = \widehat{V}_{\widehat{m}}(\lambda I - \widehat{H}_{\widehat{m}}) - \widehat{h}_{\widehat{m}+1,\widehat{m}}\widehat{v}_{\widehat{m}+1}e_{\widehat{m}}^T. \tag{3.29}$$

Die Matrix $\widehat{H}_{\widehat{m}}$ hat folgende Struktur: Die Zeilen 1 bis $k+1$ bilden eine vollbesetzte Matrix der Dimension $(k+1) \times \widehat{m}$ und die Zeilen $k+2$ bis $\widehat{m}$ bilden eine obere Hessenbergmatrix, beziehungsweise im Lanczosfall eine Tridiagonalmatrix. Die Approximation an (3.28) ist nun $\widehat{x}_{\widehat{m}}(\lambda) = x_m(\lambda) + \widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda)$ mit

$$\widehat{y}_{\widehat{m}}(\lambda) = (\lambda I - \widehat{H}_{\widehat{m}})^{-1}\beta_m(\lambda)e_{k+1}.$$

Das Residuum ist $\widehat{r}_{\widehat{m}}(\lambda) = \widehat{\beta}_{\widehat{m}}(\lambda)\beta_m(\lambda)\widehat{v}_{\widehat{m}+1}$ mit

$$\widehat{\beta}_{\widehat{m}}(\lambda) = \widehat{h}_{\widehat{m}+1,\widehat{m}}e_{\widehat{m}}^T\widehat{y}_{\widehat{m}}(\lambda).$$

Man kann hier wieder einen Restart einfügen, das heißt, man berechnet wieder k Ritzvektoren von $\widehat{H}_{\widehat{m}}$ und geht dann genau so vor wie oben gezeigt.

Wendet man dies auf die Matrixfunktion an, so bekommt man eine Approximation $\widehat{z}_{\widehat{m}}$ nach einem Deflated Restart an $z = f(A)b$ auf gleiche Weise, wie wir (3.10) erstellt haben. Diese Approximation ist

$$\widehat{z}_{\widehat{m}} = \frac{1}{2\pi i}\int_\Gamma f(\lambda)\big[x_m(\lambda) + \widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda)\big]\,\mathrm{d}\lambda$$

und es gilt der folgende Satz.

Satz 3.12. *Falls man einen Deflated Restart mit k Ritzvektoren auf oben beschriebene Weise durchführt, dann ist*

$$\widehat{z}_{\widehat{m}} = \widetilde{V}f(\widetilde{H})e_1$$

mit

$$\widetilde{H} = \left[\begin{array}{c|c} H_m & 0 \\ \hline h_{m+1,m}^{(0)}e_{k+1}e_m^T & \widehat{H}_{\widehat{m}} \end{array}\right]$$

und $\widetilde{V} := [V_m \quad \widehat{V}_{\widehat{m}}]$.

Bemerkung: $\widetilde{H}$ ist eine obere Hessenbergmatrix beziehungsweise Tridiagonalmatrix mit Ausnahme eines $(k+1) \times \widehat{m}$ Blockes.

Beweis. Der Beweis geht genau so wie der Beweis von Satz 3.2. Die einzige Änderung ist, dass man

$$(\lambda I - \widehat{H}_{\widehat{m}})\widehat{y}_{\widehat{m}}(\lambda) = \beta_m(\lambda)e_{k+1}$$

verwendet. □

Wie man die Approximationen an $z = f(A)b$ nach mehreren Deflated Restarts am Besten berechnet, zeigt eine Abwandlung von Satz 3.5:

Satz 3.13. *Die Approximation an $z = f(A)b$ ist nach m_l Krylovschritten des l-ten Deflated Restarts mit jeweiliger Verwendung von k Ritzvektoren*

$$z_{m_l}^{(l)} = \widetilde{V}^{(l)} f(\widetilde{H}^{(l)})e_1 \tag{3.30}$$

$$= z_{m_{l-1}}^{(l-1)} + V_{m_l}^{(l)} \big(f(\widetilde{H}^{(l)})e_1\big)_{\nu_{l-1}+1:\nu_l}. \tag{3.31}$$

Dabei ist

$$\widetilde{H}^{(l)} = \left[\begin{array}{c|c} \widetilde{H}^{(l-1)} & 0 \\ \hline h_{m_{l-1}+1,m_{l-1}}^{(l-1)} e_{k+1} e_{\nu_{l-1}}^T & H_{m_l}^{(l)} \end{array}\right], \tag{3.32}$$

$\widetilde{V}^{(l)} = \begin{bmatrix}\widetilde{V}^{(l-1)} & V_{m_l}^{(l)}\end{bmatrix}$ *und* $\nu_l = \sum_{j=0}^{l} m_j$. *Für* $l = 0$ *sei* $\widetilde{H}^{(0)} = H_m$, $\widetilde{V}^{(0)} = V_m$ *und* $\widetilde{m} = m$.

Bemerkung: Hierbei wird $H_{m_i}^{(i)}$ immer wie $\widehat{H}_{\widehat{m}}$ in (3.29) erstellt und hat obere Hessenberg beziehungsweise Tridiagonal Struktur bis auf einen vollbesetzten Anteil.

Der wesentliche Speicheraufwand des Deflated Restart Verfahrens ist gleich dem des Restart Verfahrens ohne Verwendung von Ritzvektoren.

3.5 Numerische Beispiele

3.5.1 Dreidimensionale Wärmeleitungsgleichung

Die dreidimensionale Wärmeleitungsgleichung entsteht aus dem Anfangs-Randwertproblem:

$$\begin{aligned} u' - \Delta u &= 0 && \text{auf } (0,1)^3 \times (0,T), \\ u(x,t) &= 0 && \text{auf } \partial(0,1)^3, \quad \forall t \in [0,T], \\ u(x,0) &= u_0(x), && x \in (0,1)^3. \end{aligned}$$

Falls man den Laplace-Operator auf einem gleichmäßigen Gitter mit jeweils N inneren Punkten in jeder Raumdimension diskretisiert, reduziert sich das Problem zu dem Anfangswertproblem

$$\begin{aligned} u'(t) &= Au(t), \quad t \in (0,T), \\ u(0) &= u_0, \end{aligned}$$

	k	Iterat.	cpu-Zeit	Ersparnis in %
ohne Restart	-	190	82.320	–
mit Restart	0	237	41.250	49.89
	1	216	35.400	57.00
	2	209	33.760	58.99
	3	203	33.040	59.86
	4	200	32.820	60.13
	5	198	33.360	59.48
	6	196	33.420	59.40
	7	196	34.680	57.87
	8	196	35.650	56.69
Lanczos{40}	-	190	53.470	35.05

Tabelle 3.1: Vergleich der versch. Lanczosverfahren für die dreidim. Wärmeleitungsgleichung

mit einer $n \times n$ Matrix A ($n = N^3$) und einem Anfangsvektor u_0. Die Lösung ist gegeben durch

$$u(t) = e^{tA} u_0. \tag{3.33}$$

Wir setzen nun $N = 50$ und $t = 0.1$ und vergleichen das Lanczosverfahren ohne Restart mit Lanczosverfahren inklusive Restarts nach $m = 40$ Vektoren mit k Ritzvektoren. Außerdem wird zum Vergleich ein Lanczosverfahren hinzugenommen, dass m Lanczosvektoren speichert und die restlichen Vektoren nur zur Berechnung der Ergebnisse wieder konstruiert. Dieses Verfahren bezeichnen wir mit *Lanczos*$\{m\}$. Die Lanczosverfahren laufen so lange bis der exakte Fehler kleiner als $1.2 \cdot 10^{-6}$ ist. Tabelle 3.1 gibt die jeweilige Anzahl der Iterationen, die jeweilige Zeit der Berechnung in Sekunden und im Vergleich zum Lanczosverfahren ohne Restart die Zeitersparnis in Prozent an. In Abbildung 3.2 werden die jeweiligen exakten Fehler des gleichen Beispiels pro Kryloviteration für die verschiedenen Verfahren verglichen. Die Verfahren, die verglichen werden, sind das Lanczosverfahren ohne Restart (o.r.) und die neu gestarteten Lanczosverfahren mit $k = 0, 1, \ldots, 8$ Ritzvektoren. Man sieht nun, dass in diesem Beispiel durch Restarts neben der Speicherplatzersparnis auch eine Beschleunigung der Rechenzeit erreicht wird. Das dies nicht immer der Fall sein muss, zeigt das nächste Beispiel.

3.5.2 Zweidimensionale Wärmeleitungsgleichung

Wie in der Einleitung gesagt, wollen wir die Berechnung von Differentialgleichungen mit exponentiellen Integratoren verbessern. Die wichtigsten Differentialgleichungen für diese Arbeit sind semilineare, parabolische Differentialgleichungen:

$$u'(t) = Au(t) + g(u,t), \qquad u(t_0) = u_0.$$

In vielen Beispielen ist $Au(t)$ die Diskretisierung des Laplace Operators Δu. In exponentiellen Integratoren werden dann Produkte von der Art $\varphi_{k,i}(hA)v$ berechnet (vergl.

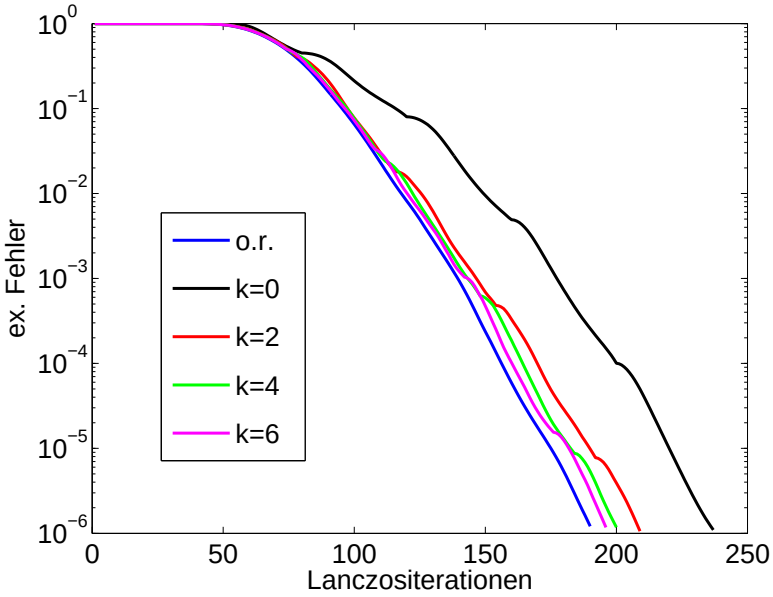

Abbildung 3.2: Exakter Fehler der versch. Verfahren für die dreidim. Wärmeleitungsgleichung

Kapitel 2.2). Deshalb betrachten wir die Approximation von

$$\varphi_1(A)b$$

unter Verwendung unserer Restart Methoden. A ist die Diskretisierungsmatrix der Diskretisierung auf [0 1] mit $N = 100$ inneren Gitterpunkten in jeder Raumdimension des zweidimensionalen Laplace Operators. Mit $n = N^2$ ist$A \in \mathbb{R}^{n,n}$ und $b \in \mathbb{R}^n$ ein Zufallsvektor. Wir vergleichen wieder die gleichen Verfahren wie im letzten Beispiel. Wieder ist die maximale Anzahl der gespeicherten Vektoren $m = 40$ und wir berechnen die Approximationen bis zu einer Genauigkeit von $1.35 \cdot 10^{-6}$. Daraus ergeben sich Tabelle 3.2 und Abbildung 3.3. Hier sieht man, dass durch einen Restart nicht immer Rechenzeit gespart werden kann.

	k	Iterat.	cpu-Zeit	Ersparnis in %
ohne Restart	-	85	1.550	–
mit Restart	0	235	40.690	−2525.16
	1	135	3.590	−131.61
	2	109	1.980	−27.74
	3	104	1.780	−14.84
	4	92	1.400	9.68
	5	92	1.450	6.45
	6	91	1.440	7.10
Lanczos{40}	-	84	1.610	−3.87

Tabelle 3.2: Vergleich der versch. Lanczosverfahren für die zweidimensionale Wärmeleitungsgleichung

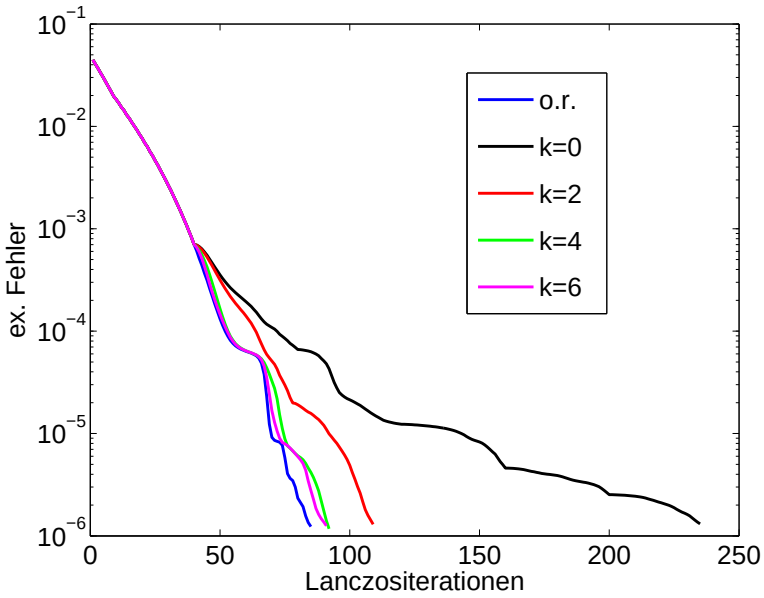

Abbildung 3.3: Exakter Fehler der versch. Verfahren für die zweidimensionale Wärmeleitungsgleichung

Kapitel 4

Lanczos-Galerkin Projektionsverfahren

In den nächsten beiden Kapiteln zeigen wir, wie man Approximationen von Matrixfunktionen, angewandt auf mehrere Vektoren,

$$f(A)b_i, \qquad b_i = g(t + c_i h) \tag{4.1}$$

auf effiziente Weise berechnet. Hierbei sind die Vektoren b_i Funktionsauswertungen einer hinreichend glatten Funktion $g : \mathbb{R} \to \mathbb{C}^N$ und $A \in \mathbb{C}^{N,N}$.

Wir betrachten dieses Problem, da wir in Kapitel 2.2 gesehen haben, dass zur Lösung einer parabolischen Differentialgleichung mit Hilfe von exponentiellen Integratoren Matrixfunktionen, angewandt auf verschiedene Vektoren (siehe (2.9)), approximiert werden müssen. Die Vektoren hängen voneinander durch Funktionsauswertungen ab und sind deshalb nicht gleichzeitig verfügbar.

4.1 Gleichungssysteme mit mehreren rechten Seiten

Wenn wir die Matrixfunktionen durch die Cauchyintegralformel darstellen, ist das Hauptproblem die Lösung verschobener linearer Gleichungssysteme mit verschiedenen rechten Seiten

$$(\lambda I - A)x_i(\lambda) = b_i, \qquad i = 1, 2, \ldots. \tag{4.2}$$

Man möchte die Approximationen an die $x_i(\lambda)$ nicht einzeln, ohne Berücksichtigung der anderen Approximationen, berechnen, sondern versuchen, möglichst viele Informationen der schon berechneten Approximationen zu nutzen. Dabei wird erwartet, dass man insgesamt Berechnungsaufwand und Berechnungszeit einspart. In der Literatur gibt es zwei verschiedene Verfahrensklassen zur Approximation von linearen Gleichungssystemen mit unterschiedlichen rechten Seiten

$$Ax_i = b_i, \qquad i = 1, \ldots, n$$

mit Hilfe von Krylovverfahren, nämlich *Blockmethoden* und *Projektionsmethoden*.

Ein Blockverfahren, das gleichzeitig verschiedene rechte Seiten berechnet, gibt es fast zu jedem iterativen Verfahren zur Lösung linearer Gleichungssysteme. Anstatt auf einen einzelnen Vektor wird hier ein Krylovverfahren auf eine Matrix $B = [b, \ldots, b_n]$ angewendet. Das erste Blockverfahren ist das Blockverfahren der konjugierten Gradienten (BCG-Verfahren) von D. O'Leary [61]. Des Weiteren gibt es das "Block Generalized Minimum Residual" (Bl-GMRES) Verfahren von B. Vital [85], das "Block Quasi Minimum Residual" (Bl-QMR) Verfahren von R. W. Freund und M. Malhotra [24], das Block FOM (BFOM) Verfahren, eingeführt von Y. Saad [71], das "Block Minimal Residual Smoothing" (BMRS) Verfahren von K. Jbilou [40], das Block BICGSTAB (Bl-BICGSTAB) von El Guennouni, Jbilou und Sadok [31], das "block modified Chebyshev" Verfahren von D. Calvetti und L. Reichel [9] und das Block-EN Verfahren von G. Gu und H. Wu [30]. Diese Verfahren wurden dann weiter entwickelt, zum Beispiel BCG von Feng, Owen und Perić [22].

Außerdem gibt es sogenannte globale Block Verfahren. Hier wird, im Gegensatz zu den oben aufgeführten Verfahren, in der Frobeniusnorm auf Krylovräume projiziert. Es gibt das "global full orthogonalization" (Gl-FOM) und das "global generalized residual" (Gl-GMRES) Verfahren von K. Jbilou, A. Messaoudi und H. Sadok [41], das globale Hessenberg und globale CMRH Verfahren von M. Heyouni [33].

Vorreiter bei der Verfahrensklasse der Projektionsmethoden waren B. N. Parlett [65] und Y. Saad [69]. Sie berechnen die Lösung des ersten linearen Gleichungssystems und projizieren das Residuum des zweiten Systems mit einer Galerkinprojektion auf den Krylovraum des ersten Systems. Nun starten sie ihr Verfahren mit dem projizierten Residuum neu oder führen das alte Verfahren weiter und hoffen dadurch, weniger Schritte machen zu müssen, um eine geeignete Approximation an die Lösung zu bekommen. Viele andere gehen ähnlich vor. Papadrakakis und Smerou [63] benutzen für die Projektion die Lanczosvektoren, van der Vorst [83] verwendet die Residuenvektoren aus dem CG-Verfahren und Smith, Peterson, Mittra [75] benutzen ein vereinfachtes CG-Verfahren, welches auf den Richtungsvektoren beruht. Solche Verfahren werden auch Seed (dt. Samen, Keim) Verfahren genannt. Diese werden von P. Joly [42] und T. Chan und W. Wan [12] untersucht. T. Chan und W. Wan [12] geben ein Seed-Block-Verfahren an. V. Simoncini und E. Gallopoulos [74] kombinieren die Seed-Idee mit der Hybrid-Technik. C. Musschoot [57] hat ein Verfahren entwickelt, das auf der BICGSTAB-Methode aufbaut.

Vorteil der Projektionstechniken ist, dass bei diesen Verfahren die verschiedenen Vektoren b_i noch nicht alle bekannt sein müssen und deshalb voneinander abhängen können. Dies ist bei den Blockmethoden nicht der Fall, hier müssen alle Vektoren von Anfang an bekannt sein und sind somit nicht für unser Problem geeignet.

Verfahren, die speziell die Matrixfunktion, angewandt auf verschiedene Vektoren, betrachten, sind in der Literatur noch nicht bekannt. In diesem Kapitel betrachten wir ein Projektionsverfahren zur Approximation von (4.1) und im nächsten Kapitel verwenden wir für ein anderes Projektionsverfahren die QR Zerlegung der verschiedenen Vektoren. Numerische Beispiele für beide Verfahren werden im letzten Kapitel angegeben.

4.2 Lanczos-Galerkin Projektion für Matrixfunktionen

Um die Darstellung übersichtlich zu gestalten, betrachten wir hier zunächst den Fall, dass wir eine Matrixfunktion, angewandt auf zwei verschiedene Vektoren,

$$z = f(A)b \qquad \text{und} \qquad \widehat{z} = f(A)\widehat{b}$$

approximieren wollen. Wegen der Cauchyintegraldarstellung der Matrixfunktion betrachten wir die verschobenen linearen Gleichungssysteme

$$(\lambda I - A)x(\lambda) = b \qquad \text{und} \qquad (\lambda I - A)\widehat{x}(\lambda) = \widehat{b}.$$

Im Folgenden nutzen wir die Idee, die neben Parlett [65] auch Saad [69] für sein Lanczos-Galerkin Projektionsverfahren hatte. Man projiziert den Vektor $\widehat{b}$ auf den schon berechneten Krylovraum des ersten linearen Gleichungssystems und berechnet damit eine Galerkinapproximation im gleichen Raum. Diese Galerkinapproximation wird dann als erste Approximation an die Lösung des zweiten linearen Gleichungssystems benutzt. Um die Näherung zu verfeinern, verwendet man ein Verfahren, das auf dem Residuum des projizierten Vektors im ersten Raum beruht. Hierbei ist wieder, wie im letzten Kapitel 3.1 wesentlich, dass man dieses Residuum durch schon bekannte Vektoren darstellen kann.

Zur Berechnung von $z = f(A)b$ nutzen wir die Krylovrelation

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \qquad v_1 = \frac{b}{\|b\|}, \qquad V_m^H V_m = I.$$

Dann ist die Standard Krylovapproximation an z

$$z_m = V_m f(H_m)\|b\| e_1,$$

dies haben wir schon in Kapitel 2.1 gesehen. Nun wollen wir die berechneten Krylovkomponenten V_m und H_m nutzen, um eine Näherungslösung an $\widehat{z} = f(A)\widehat{b}$ zu erhalten. Anders ausgedrückt suchen wir eine gute Startapproximation $\widehat{x}_0(\lambda)$ an $\widehat{x}(\lambda)$ in dem schon berechneten Krylovunterraum $\mathcal{K}_m(A, b)$. Daher setzen wir $\widehat{x}_0(\lambda) = V_m \widehat{y}_0(\lambda)$ und das Residuum ist

$$\begin{aligned} \widehat{r}_0(\lambda) :&= \widehat{b} - (\lambda I - A)\widehat{x}_0(\lambda) \\ &= \widehat{b} - (\lambda I - A)V_m \widehat{y}_0(\lambda) \\ &= \widehat{b} - V_m(\lambda I - H_m)\widehat{y}_0(\lambda) + h_{m+1,m} v_{m+1} e_m^T \widehat{y}_0(\lambda). \end{aligned} \tag{4.3}$$

Verlangen wir, dass das Residuum $\widehat{r}_0$ senkrecht auf dem Raum V_m stehen soll, so muss

$$0 = {V_m}^H \widehat{b} - (\lambda I - H_m)\widehat{y}_0(\lambda)$$

gelten. Also ist die Startnäherung $\widehat{x}_0(\lambda) = V_m \widehat{y}_0(\lambda)$ an $\widehat{x}(\lambda)$ durch das lineare Gleichungssystem

$$(\lambda I - H_m)\widehat{y}_0 = {V_m}^H \widehat{b} \tag{4.4}$$

gegeben. Setzt man dies in (4.3) ein, so ist das Residuum

$$\widehat{r}_0(\lambda) = (I - V_m {V_m}^H)\widehat{b} + \beta_m(\lambda) v_{m+1} \tag{4.5}$$

mit $\beta_m(\lambda) = h_{m+1,m} e_m^T \widehat{y}_0(\lambda)$. Approximiert man die Lösungen der linearen Gleichungssysteme in der Cauchyintegralformel mit $\widehat{x}_0(\lambda)$, dann bekommt man die erste Näherung an $\widehat{z}$ in dem schon berechneten Krylovraum $\mathcal{K}(A, b)$ und diese ist

$$\widehat{z}_0 = V_m f(H_m) {V_m}^H \widehat{b}. \tag{4.6}$$

Falls die Genauigkeit dieser Näherung noch nicht gut genug ist, wollen wir einen neuen Unterraum, zum Beispiel einen Krylovunterraum, mit Hilfe des Residuums (4.5) erzeugen, um in diesem Raum die Approximation zu verfeinern. Im Kapitel 3.1 über Restarts von verschobenen linearen Gleichungssystemen gab es eine ähnliche Situation, dort war aber das Residuum ein Vielfaches von v_{m+1} und somit war das normierte Residuum unabhängig von λ. Dies ist hier nicht der Fall. Man muss sich hier eine etwas andere Strategie überlegen.

Da $\widehat{r}_0(\lambda)$ für jedes λ in einem zweidimensionalen Unterraum liegt, kann man die Band Lanczos Methode von Ruhe [67] für zwei Vektoren nutzen. Das Band Lanczos Verfahren kann man auch zu einem Band Arnoldi Verfahren umschreiben, das zum Beispiel von Saad in [71, Kapitel 6.12] beschrieben wird. Hierfür setzen wir $\widehat{v}_1 = v_{m+1}$ und bestimmen $\widehat{v}_2$ so, dass $[\widehat{v}_1,\ \widehat{v}_2]$ eine Orthonormalbasis des Raumes ist, der von v_{m+1} und $(I - V_m {V_m}^H)\widehat{b}$ aufgespannt wird. Dann ist also

$$\widehat{v}_2 = \frac{1}{\alpha_{22}} \left[(I - V_m {V_m}^H)\widehat{b} - \alpha_{12}\widehat{v}_1 \right]$$

mit

$$\begin{aligned} \alpha_{12} &= \widehat{v}_1^H (I - V_m {V_m}^H)\widehat{b}, \\ \alpha_{22} &= \| (I - V_m {V_m}^H)\widehat{b} - \alpha_{12}\widehat{v}_1 \| \end{aligned}$$

und es gilt

$$(I - V_m {V_m}^H)\widehat{b} = \alpha_{12}\widehat{v}_1 + \alpha_{22}\widehat{v}_2$$

und

$$\widehat{r}_0(\lambda) = (\alpha_{12} + \beta_m(\lambda))\widehat{v}_1 + \alpha_{22}\widehat{v}_2.$$

Das Residuum $\widehat{r}_0(\lambda)$ liegt im $\operatorname{span}\{\widehat{v}_1,\ \widehat{v}_2\}$. Wendet man auf $\widehat{v}_1$ und $\widehat{v}_2$ das Block Krylovverfahren von Ruhe an, erhält man die Relation

$$A\widehat{V}_{\widehat{m}} = \widehat{V}_{\widehat{m}}\widehat{H}_{\widehat{m}} + \widehat{v}_{\widehat{m}+1}(\widehat{h}_{\widehat{m}+1,\widehat{m}} e_{\widehat{m}}^T + \widehat{h}_{\widehat{m}+1,\widehat{m}-1} e_{\widehat{m}-1}^T) + \widehat{h}_{\widehat{m}+2,\widehat{m}}\widehat{v}_{\widehat{m}+2} e_{\widehat{m}}^T$$

mit $\widehat{V}_{\widehat{m}+2}^H \widehat{V}_{\widehat{m}+2} = I$. Die Matrix $\widehat{H}_{\widehat{m}}$ ist eine obere Hessenbergmatrix mit einer zusätzlichen unteren Nebendiagonalen und im Lanczosfall ist sie eine Bandmatrix mit Bandweite $p = 2$. $\widehat{V}_{\widehat{m}}$ ist aber keine Orthonormalbasis eines Krylovraumes mehr, sondern des Unterraumes

$$\begin{aligned} \widehat{\mathcal{K}}_{\widehat{m}}(A, \widehat{v}_1, \widehat{v}_2) &:= \operatorname{span}\{\widehat{v}_1, \widehat{v}_2, \ldots, \widehat{v}_m\} \\ &= \operatorname{span}\{\widehat{v}_1, \widehat{v}_2, A\widehat{v}_1, A\widehat{v}_2, \ldots, A^{\lceil \frac{\widehat{m}+1}{2} \rceil} v_1, A^{\lceil \frac{\widehat{m}}{2} \rceil}\widehat{v}_2\}. \end{aligned}$$

Hier bezeichnet $\lceil\cdot\rceil$ die nächst kleinere ganze Zahl. Ist ein $z \in \widehat{\mathcal{K}}_{\widehat{m}}(A, \widehat{v}_1, \widehat{v}_2)$, dann kann man

$$z = p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}(A)\widehat{v}_1 + p_{2,\lceil\frac{\widehat{m}}{2}\rceil}(A)\widehat{v}_2 \tag{4.7}$$

durch Polynome $p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil} \in \mathbb{P}_{\lceil\frac{\widehat{m}+1}{2}\rceil}$ und $p_{2,\lceil\frac{\widehat{m}}{2}\rceil} \in \mathbb{P}_{\lceil\frac{\widehat{m}}{2}\rceil}$ darstellen.

Wir suchen nun ein $\widehat{x}_{\widehat{m}}(\lambda)$, so dass $\widehat{x}_0(\lambda) + \widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda)$ eine gute Näherung an $\widehat{x}(\lambda)$ ist. Dafür betrachten wir wieder das Residuum

$$\begin{aligned}
\widehat{r}_{\widehat{m}}(\lambda) :&= \widehat{b} - (\lambda I - A)(\widehat{x}_0(\lambda) + \widehat{x}_{\widehat{m}}(\lambda)) \\
&= \widehat{r}_0(\lambda) - (\lambda I - A)\widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda) \\
&= (I - V_m {V_m}^H)\widehat{b} + \beta_m(\lambda)v_{m+1} - \widehat{V}_{\widehat{m}}(\lambda I - \widehat{H}_{\widehat{m}})\widehat{y}_{\widehat{m}}(\lambda) \\
&\quad + \widehat{v}_{\widehat{m}+1}(\widehat{h}_{\widehat{m}+1,\widehat{m}}e_{\widehat{m}}^T + \widehat{h}_{\widehat{m}+1,\widehat{m}-1}e_{\widehat{m}-1}^T)\widehat{y}_{\widehat{m}}(\lambda) + \widehat{h}_{\widehat{m}+2,\widehat{m}}\widehat{v}_{\widehat{m}+2}e_{\widehat{m}}^T\widehat{y}_{\widehat{m}}(\lambda) \\
&= \alpha_{12}\widehat{v}_1 + \alpha_{22}\widehat{v}_2 + h_{m+1,m}\widehat{v}_1 e_m^T\widehat{y}_0(\lambda) - \widehat{V}_{\widehat{m}}(\lambda I - \widehat{H}_{\widehat{m}})\widehat{y}_{\widehat{m}}(\lambda) \\
&\quad + \widehat{v}_{\widehat{m}+1}(\widehat{h}_{\widehat{m}+1,\widehat{m}}e_{\widehat{m}}^T + \widehat{h}_{\widehat{m}+1,\widehat{m}-1}e_{\widehat{m}-1}^T)\widehat{y}_{\widehat{m}}(\lambda) + \widehat{h}_{\widehat{m}+2,\widehat{m}}\widehat{v}_{\widehat{m}+2}e_{\widehat{m}}^T\widehat{y}_{\widehat{m}}(\lambda).
\end{aligned}$$

Um die Galerkin Bedingung zu erfüllen, soll dieses Residuum senkrecht auf $\widehat{V}_{\widehat{m}}$ stehen, also muss

$$0 = \alpha_{12}e_1 + \alpha_{22}e_2 + h_{m+1,m}e_1e_m^T\widehat{y}_0(\lambda) - (\lambda I - \widehat{H}_{\widehat{m}})\widehat{y}_{\widehat{m}}(\lambda)$$

gelten. Das lineare Gleichungssystem zur Berechnung von $\widehat{y}_{\widehat{m}}(\lambda)$ ist also

$$(\lambda I - \widehat{H}_{\widehat{m}})\widehat{y}_{\widehat{m}}(\lambda) = \alpha_{12}e_1 + \alpha_{22}e_2 + h_{m+1,m}e_1e_m^T\widehat{y}_0(\lambda). \tag{4.8}$$

Für das Residuum gilt dann

$$\widehat{r}_{\widehat{m}}(\lambda) = \widehat{v}_{\widehat{m}+1}(\widehat{h}_{\widehat{m}+1,\widehat{m}}e_{\widehat{m}}^T + \widehat{h}_{\widehat{m}+1,\widehat{m}-1}e_{\widehat{m}-1}^T)\widehat{y}_{\widehat{m}}(\lambda) + \widehat{h}_{\widehat{m}+2,\widehat{m}}\widehat{v}_{\widehat{m}+2}e_{\widehat{m}}^T\widehat{y}_{\widehat{m}}(\lambda).$$

Die beiden Gleichungssysteme (4.4) und (4.8) sind äquivalent zu dem großen Gleichungssystem

$$\left[\begin{array}{c|c} \lambda I - H_m & 0 \\ \hline -h_{m+1,m}e_1e_m^T & \lambda I - \widehat{H}_{\widehat{m}} \end{array}\right] \left[\begin{array}{c} \widehat{y}_0(\lambda) \\ \widehat{y}_{\widehat{m}}(\lambda) \end{array}\right] = \left[\begin{array}{c} {V_m}^H\widehat{b} \\ \alpha_{12}e_1 + \alpha_{22}e_2 \end{array}\right].$$

Damit gilt

$$\widehat{x}_{\widehat{m}}(\lambda) = \widehat{x}_0(\lambda) + \widehat{V}_{\widehat{m}}\widehat{y}_{\widehat{m}}(\lambda) = \widetilde{V}\left[\begin{array}{c} \widehat{y}_0(\lambda) \\ \widehat{y}_{\widehat{m}}(\lambda) \end{array}\right] = \widetilde{V}(\lambda I - \widetilde{H})^{-1}\widetilde{b}$$

mit $\widetilde{V} = [V_m,\ \widehat{V}_{\widehat{m}}]$,

$$\widetilde{H} = \left[\begin{array}{c|c} H_m & 0 \\ \hline h_{m+1,m}e_1e_m^T & \widehat{H}_{\widehat{m}} \end{array}\right] \qquad \text{und} \qquad \widetilde{b} = \left[\begin{array}{c} {V_m}^H\widehat{b} \\ \alpha_{12}e_1 + \alpha_{22}e_2 \end{array}\right].$$

Die Matrix $\widetilde{H}$ ist eine obere Hessenbergmatrix mit einer zusätzlichen unteren Nebendiagonalen und im Lanczosfall eine Bandmatrix mit Bandweite $p = 2$. Die Näherung an $\widehat{z} = f(A)\widehat{b}$ ist daher durch

$$\widehat{z}_{\widehat{m}} = \widetilde{V}f(\widetilde{H})\widetilde{b} = \widehat{z}_0 + \widehat{V}_{\widehat{m}}(f(\widetilde{H})\widetilde{b})_{(m+1):(m+\widehat{m})}$$

gegeben. Die letzte Gleichung zeigt man genauso wie Satz 3.5. Außerdem gilt noch folgende Relation

$$A\widetilde{V} = \widetilde{V}\widetilde{H} + \widehat{v}_{\widehat{m}+1}(\widehat{h}_{\widehat{m}+1,\widehat{m}}e_{\widehat{m}}^T + \widehat{h}_{\widehat{m}+1,\widehat{m}-1}e_{\widehat{m}-1}^T) + \widehat{h}_{\widehat{m}+2,\widehat{m}}\widehat{v}_{\widehat{m}+2}e_{\widehat{m}}^T. \tag{4.9}$$

4.3 Konvergenz und Fehlerschranken

In diesem Abschnitt leiten wir einen Beweis zur Konvergenz dieses Verfahrens und eine Abschätzung des Fehlers

$$\widehat{\epsilon}_{\widehat{m}} := \|f(A)\widehat{b} - \widehat{z}_{\widehat{m}}\| = \left\| \frac{1}{2\pi i} \int_\Gamma f(\lambda)\widehat{d}_{\widehat{m}}(\lambda)\mathrm{d}\lambda \right\|$$

her. Hierfür werden unter anderem wieder Techniken benutzt, die auch im Kapitel 3.3 und in [36] verwendet wurden. Es gelten die gleichen Voraussetzungen wie am Anfang des Kapitels 3.3.

Es ist

$$\begin{aligned}
\widehat{d}_{\widehat{m}}(\lambda) &= (\lambda I - A)^{-1}\widehat{b} - \widehat{x}_{\widehat{m}}(\lambda) \\
&= (\lambda I - A)^{-1}\widehat{b} - \widetilde{V}(\lambda I - \widetilde{H})^{-1}\widetilde{b} \\
&= (\lambda I - A)^{-1}\widehat{b} - V_m(\lambda I - H_m)^{-1}{V_m}^H\widehat{b} \\
&\quad - \widehat{V}_{\widehat{m}}(\lambda I - \widehat{H}_{\widehat{m}})^{-1}(\alpha_{12}e_1 + \alpha_{22}e_2 + h_{m+1,m}e_1e_m^T\widehat{y}_0(\lambda)) \\
&= (\lambda I - A)^{-1}\widehat{b} - (\lambda I - A)^{-1}V_m{V_m}^H\widehat{b} \\
&\quad + h_{m+1,m}(\lambda I - A)^{-1}v_{m+1}e_m^T(\lambda I - H_m)^{-1}{V_m}^H\widehat{b} \\
&\quad - \widehat{V}_{\widehat{m}}(\lambda I - \widehat{H}_{\widehat{m}})^{-1}\widehat{V}_{\widehat{m}}^H(\alpha_{12}\widehat{v}_1 + \alpha_{22}\widehat{v}_2 + h_{m+1,m}\widehat{v}_1e_m^T\widehat{y}_0(\lambda)) \\
&= \Delta_{\widehat{m}}\left[(\alpha_{12} + \beta_m(\lambda))\widehat{v}_1 + \alpha_{22}\widehat{v}_2\right]
\end{aligned} \tag{4.10}$$

mit $\beta_m(\lambda) = h_{m+1,m}e_m^T\widehat{y}_0(\lambda)$ und $\Delta_{\widehat{m}} = (\lambda I - A)^{-1} - \widehat{V}_{\widehat{m}}(\lambda I - \widehat{H}_{\widehat{m}})^{-1}\widehat{V}_{\widehat{m}}^H$. Wegen $\widehat{V}_{\widehat{m}}^H(\lambda I - A)\widehat{V}_{\widehat{m}} = (\lambda I - \widehat{H}_{\widehat{m}})$ gilt

$$\Delta_{\widehat{m}}(\lambda I - A)\widehat{V}_{\widehat{m}} = 0. \tag{4.11}$$

Außerdem gibt es für ein beliebiges $y \in \mathbb{C}^{\widehat{m}}$ wegen (4.7) Polynome $p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}$ und $p_{2,\lceil\frac{\widehat{m}}{2}\rceil}$, so dass

$$\begin{aligned}
&(\alpha_{12} + \beta_m(\lambda))\widehat{v}_1 + \alpha_{22}\widehat{v}_2 - (\lambda I - A)\widehat{V}_{\widehat{m}}y \\
&= p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}(A)(\alpha_{12} + \beta_m(\lambda))\widehat{v}_1 + p_{2,\lceil\frac{\widehat{m}}{2}\rceil}(A)\alpha_{22}\widehat{v}_2
\end{aligned} \tag{4.12}$$

gilt, da $\widehat{V}_{\widehat{m}}y \in \widehat{\mathcal{K}}_{\widehat{m}}(A, \widehat{v}_1, \widehat{v}_2)$ ist. Die Polynome $p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}$ und $p_{2,\lceil\frac{\widehat{m}}{2}\rceil}$ sind Polynome vom Grad $\leq \lceil\frac{\widehat{m}+1}{2}\rceil$ beziehungsweise $\leq \lceil\frac{\widehat{m}}{2}\rceil$ mit $p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}(\lambda) = p_{2,\lceil\frac{\widehat{m}}{2}\rceil}(\lambda) = 1$. Mit (4.10),

(4.11) und (4.12) gilt für ein beliebiges $y \in \mathbb{C}^{\widehat{m}}$

$$\begin{aligned}\|\widehat{d}_{\widehat{m}}(\lambda)\| &= \|\Delta_{\widehat{m}} \left[(\alpha_{12} + \beta_m(\lambda))\widehat{v}_1 + \alpha_{22}\widehat{v}_2 - (\lambda I - A)\widehat{V}_{\widehat{m}} y\right] \| \\ &= \|\Delta_{\widehat{m}} \left[p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}(A)(\alpha_{12} + \beta_m(\lambda))\widehat{v}_1 + p_{2,\lceil\frac{\widehat{m}}{2}\rceil}(A)\alpha_{22}\widehat{v}_2\right] \| \\ &\leq \|\Delta_{\widehat{m}}\| \left[\|p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}(A)\|(|\alpha_{12}| + |\beta_m(\lambda)|) + \|p_{2,\lceil\frac{\widehat{m}}{2}\rceil}(A)\||\alpha_{22}|\right]. \end{aligned} \tag{4.13}$$

Wir betrachten die einzelnen Komponenten. Für die Norm von $\Delta_{\widehat{m}}$ gilt wie für (3.18)

$$\|\Delta_{\widehat{m}}\| \leq 2d(\Gamma)^{-1} \tag{4.14}$$

und für die Norm der Polynome wegen Satz 3.6

$$\|p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}(A)\| \leq M|\phi(\lambda)|^{-\lceil\frac{\widehat{m}+1}{2}\rceil} \leq M|\phi(\lambda)|^{-\lceil\frac{\widehat{m}}{2}\rceil}$$

und

$$\|p_{2,\lceil\frac{\widehat{m}}{2}\rceil}(A)\| \leq M|\phi(\lambda)|^{-\lceil\frac{\widehat{m}}{2}\rceil}.$$

Des Weiteren gilt

$$|\alpha_{12}| = |{v_{m+1}}^H(I - V_m {V_m}^H)\widehat{b}| \leq \|(I - V_m {V_m}^H)\widehat{b}\|, \tag{4.15}$$

$$|\alpha_{22}| = \|(I - V_m {V_m}^H)\widehat{b} - \alpha_{12} v_{m+1}\| \leq 2\|(I - V_m {V_m}^H)\widehat{b}\|. \tag{4.16}$$

Nun schätzen wir noch $|\beta_m(\lambda)|$ ab. Es gilt

$$\begin{aligned}\widehat{r}_0(\lambda) &= \widehat{b} - (\lambda I - A)V_m \widehat{y}_m \\ &= \widehat{b} - V_{m+1}(\lambda I - \bar{H}_m)(\lambda I - H_m)^{-1}{V_m}^H \widehat{b} \\ &= (I - V_{m+1}(\lambda I - \bar{H}_m)(\lambda I - H_m)^{-1}{V_m}^H)\widehat{b} \\ &=: P_m^G(\lambda)\widehat{b}\end{aligned}$$

mit

$$\bar{H}_m = \begin{bmatrix} H_m \\ h_{m+1,m} e_m^T \end{bmatrix}.$$

Zusammen mit (4.5),

$$P_m^G(\lambda)(\lambda I - A)V_m = 0$$

und

$$P_m^G(\lambda)(I - V_m {V_m}^H) = I - V_m {V_m}^H$$

gilt dann

$$\begin{aligned}
\beta_m(\lambda)v_{m+1} &= \widehat{r}_0(\lambda) - (I - V_m {V_m}^H)\widehat{b} \\
&= P_m^G(\lambda)\widehat{b} - P_m^G(\lambda)(I - V_m {V_m}^H)\widehat{b} \\
&= P_m^G(\lambda) V_m {V_m}^H \widehat{b} \\
&= P_m^G(\lambda) \sum_{l=1}^{m} v_l {v_l}^H \widehat{b} \\
&= P_m^G(\lambda) \left(\sum_{l=1}^{m} (v_l - (\lambda I - A) V_m y_l) {v_l}^H \widehat{b} \right), \qquad \forall y_l \in \mathbb{C}^m, l = 1, \ldots, m
\end{aligned}$$

so dass

$$|\beta_m(\lambda)| \leq \|P_m^G(\lambda)\| \sum_{l=1}^{m} |{v_l}^H \widehat{b}| \min_{y \in \mathbb{C}^m} \|v_l - (\lambda I - A) V_m y\|.$$

Wir schreiben $v_l = q_{l-1}(A)v_1$ für ein $q_{l-1} \in \mathbb{P}_{l-1}$ und erhalten

$$\begin{aligned}
\min_{y \in \mathbb{C}^m} \|v_l - (\lambda I - A) V_m y\| &= \min_{z \in \mathcal{K}_m} \|v_l - (\lambda I - A) z\| \\
&= \min_{\substack{p \in \mathbb{P}_m \\ p(\lambda) = 0}} \|q_{l-1}(A)v_1 - p(A)v_1\| \\
&\leq \min_{\substack{p \in \mathbb{P}_{m-l+1} \\ p(\lambda) = 0}} \|(I - p(A)) q_{l-1}(A) v_1\| \\
&= \min_{\substack{p \in \mathbb{P}_{m-l+1} \\ p(\lambda) = 1}} \|p(A) q_{l-1}(A) v_1\| \\
&\leq \min_{\substack{p \in \mathbb{P}_{m-l+1} \\ p(\lambda) = 1}} \|p(A)\| \|q_{l-1}(A) v_1\| \\
&= \min_{\substack{p \in \mathbb{P}_{m-l+1} \\ p(\lambda) = 1}} \|p(A)\| \|v_l\| \\
&\leq M |\phi(\lambda)|^{-(m-l+1)}.
\end{aligned}$$

In der letzten Ungleichung wurde wieder Satz 3.6 angewendet. Damit ist

$$|\beta_m(\lambda)| \leq M \|P_m^G(\lambda)\| \sum_{l=1}^{m} |{v_l}^H \widehat{b}| |\phi(\lambda)|^{-(m-l+1)}. \tag{4.17}$$

Für die Norm des Projektors $P_m^G(\lambda)$ gilt

$$\|P_m^G(\lambda)\| \leq 1 + |h_{m+1,m}| d(\Gamma)^{-1}. \tag{4.18}$$

Insgesamt folgt aus (4.13) - (4.18)

$$\begin{aligned}\|\widehat{d}_{\widehat{m}}(\lambda)\| &\leq \|\Delta_{\widehat{m}}\| \left[\|p_{1,\lceil\frac{\widehat{m}+1}{2}\rceil}(A)\|(|\alpha_{12}|+|\beta_m(\lambda)|)+\|p_{2,\lceil\frac{\widehat{m}}{2}\rceil}(A)\||\alpha_{22}|\right] \\ &\leq 6d(\Gamma)^{-1}M\|(I-V_m{V_m}^H)\widehat{b}\||\phi(\lambda)|^{-\lceil\frac{\widehat{m}}{2}\rceil} \\ &\quad + 2d(\Gamma)^{-1}M^2\left(1+\frac{|h_{m+1,m}|}{d(\Gamma)}\right)\sum_{l=1}^{m}\left(|{v_l}^H\widehat{b}||\phi(\lambda)|^{-(m-l+1+\lceil\frac{\widehat{m}}{2}\rceil)}\right).\end{aligned}$$

Wir betrachten den Fehler der Approximation an die Matrixfunktion

$$\begin{aligned}\widehat{\epsilon}_{\widehat{m}} &= \|f(A)\widehat{b}-\widetilde{V}f(\widetilde{H})\widetilde{b}\| \\ &= \|\frac{1}{2\pi i}\int_\Gamma f(\lambda)\widehat{d}_{\widehat{m}}(\lambda)\mathrm{d}\lambda\| \\ &\leq \frac{1}{2\pi}\int_\Gamma |f(\lambda)|\|\widehat{d}_{\widehat{m}}(\lambda)\||\mathrm{d}\lambda| \\ &\leq \frac{3M}{\pi d(\Gamma)}\|(I-V_m{V_m}^H)\widehat{b}\|\int_\Gamma |f(\lambda)||\phi(\lambda)|^{-\lceil\frac{\widehat{m}}{2}\rceil}|\mathrm{d}\lambda| \\ &\quad + \frac{M^2}{\pi d(\Gamma)}\left(1+\frac{|h_{m+1,m}|}{d(\Gamma)}\right)\sum_{l=1}^{m}\left(|{v_l}^H\widehat{b}|\int_\Gamma |f(\lambda)||\phi(\lambda)|^{-(m-l+1+\lceil\frac{\widehat{m}}{2}\rceil)}|\mathrm{d}\lambda|\right).\end{aligned}$$

Damit ist der folgende Satz bewiesen:

Satz 4.1. *Mit den Voraussetzungen und M aus Satz 3.6 gilt*

$$\begin{aligned}\|f(A)\widehat{b}-\widehat{z}_{\widehat{m}}\| &\leq \frac{3M}{\pi d(\Gamma)}\|(I-V_m{V_m}^H)\widehat{b}\|\int_\Gamma |f(\lambda)||\phi(\lambda)|^{-\lceil\frac{\widehat{m}}{2}\rceil}|\mathrm{d}\lambda| \\ &\quad + \frac{M^2}{\pi d(\Gamma)}\left(1+\frac{|h_{m+1,m}|}{d(\Gamma)}\right)\sum_{l=1}^{m}\left(|{v_l}^H\widehat{b}|\int_\Gamma |f(\lambda)||\phi(\lambda)|^{-(m-l+1+\lceil\frac{\widehat{m}}{2}\rceil)}|\mathrm{d}\lambda|\right).\end{aligned}$$

Bemerkung: Sei $\delta > 0$. Falls $\|(I-V_l{V_l}^H)\widehat{b}\| \leq \delta$ ist, dann gilt

$$\|(I-V_m V_m^H)\widehat{b}\| \leq \|(I-V_l V_l^H)\widehat{b}\| \leq \delta$$

und für $l < k \leq m$

$$|v_k^H\widehat{b}| = |v_k^H(I-V_lV_l^H)\widehat{b}| \leq \|(I-V_lV_l^H)\widehat{b}\| \leq \delta.$$

Deshalb gilt, je kleiner das l mit $l \leq m$ ist, für das $\|(I-V_lV_l^H)\widehat{b}\| \leq \delta$ mit $0 < \delta \ll 1$ gilt, um so schneller konvergiert das Lanczos-Galerkin Projektionsverfahren. Ist dagegen die Norm $\|(I-V_m{V_m}^H)\widehat{b}\|$ groß, dann konvergiert das Verfahren langsam. Wir betrachten nun zwei Spezialfälle. Sei erstens $\widehat{b} = v_l$ für ein $1 < l \leq m$, dann gilt

$$\|f(A)\widehat{b}-\widehat{z}_{\widehat{m}}\| \leq \frac{M^2}{\pi d(\Gamma)}\left(1+\frac{|h_{m+1,m}|}{d(\Gamma)}\right)\int_\Gamma |f(\lambda)||\phi(\lambda)|^{-(m-l+1+\lceil\frac{\widehat{m}}{2}\rceil)}|\mathrm{d}\lambda|.$$

Je kleiner l ist, um so größer ist der Exponent $(m-l+1+\lceil\frac{\widehat{m}}{2}\rceil)$ von $|\phi(\lambda)|^{-1}$ und um so schneller konvergiert das Verfahren. Gilt nun im zweiten Spezialfall, dass $\widehat{b}$ senkrecht auf V_m steht, dann ist

$$\|f(A)\widehat{b}-\widehat{z}_{\widehat{m}}\| \leq \frac{3M}{\pi d(\Gamma)}\|\widehat{b}\|\int_\Gamma |f(\lambda)||\phi(\lambda)|^{-\lceil\frac{\widehat{m}}{2}\rceil}|\mathrm{d}\lambda|$$

und das Verfahren konvergiert ungefähr halb so schnell, als wenn wir die naive Implementierung des Krylovverfahrens ohne Projektion benutzen, vergleiche hierzu Satz 3.8.

Beispiel 4.1. In Abbildung 4.1 haben wir diese Spezialfälle veranschaulicht. Dabei berechnen wir zuerst die Standardapproximation an e^Ab, wobei $A \in \mathbb{R}^{N,N}$ mit $N = 1000$ eine Diagonalmatrix mit äquidistanten Eigenwerten auf dem Intervall $[-40, 0]$ und $b \in \mathbb{R}^N$ ein normierter Zufallsvektor ist, und tragen dessen exakten Fehler (schwarz) pro Kryloviteration auf. Dann haben wir $\widehat{b}$ als fünf verschiedene Vektoren gesetzt. Einmal steht $\widehat{b}$ senkrecht auf V_m und die anderen Male gilt $\widehat{b} = v_l$ mit $l = 5, 10, 15, 20$. Für diese Vektoren wurden jeweils die Approximationen an $e^A\widehat{b}$ mit Hilfe des Lanczos-Galerkin Projektionsverfahrens berechnet. Deren exakte Fehler werden nun im Anschluss an den exakten Fehler von e^Ab aufgetragen, damit man sehen kann, wie viele Kryloviterationen das Lanczos-Galerkin Projektionsverfahren insgesamt für die Vektoren b und $\widehat{b}$ in den verschiedenen Fällen benötigt. $\diamond$

Man sieht, dass wie vorhergesagt, je größer das l ist, um so mehr Kryloviterationen gebraucht werden. Wenn $\widehat{b}$ senkrecht auf V_m steht, benötigt man am meisten Iterationen.

Setzt man

$$\Psi(\nu) = \frac{M}{3d(\Gamma)}\int_\Gamma |f(\lambda)-q_{\nu-1}(\lambda)|\cdot|\phi(\lambda)|^{-\nu}\ |\mathrm{d}\lambda|,$$

da dies gerade der Abschätzung des Fehlers $\|f(A)b - V_\nu f(H_\nu)e_1\|$ für den Krylovprozess ohne Restarts entspricht, der in [36] untersucht wird, dann ist der Fehler

$$\begin{aligned}\widehat{\epsilon}_{\widehat{m}} \leq\ & 3\|(I-V_m{V_m}^H)\widehat{b}\|\Psi(\lceil\frac{\widehat{m}}{2}\rceil)\\ & + M\left(1+\frac{|h_{m+1,m}|}{d(\Gamma)}\right)\sum_{l=1}^{m}\left(|{v_l}^H\widehat{b}|\Psi(m-l+1+\lceil\frac{\widehat{m}}{2}\rceil)\right).\end{aligned}$$

Nun kann man auf gleiche Weise wie in [36] die Fehlerschranken für spezielle Matrizen A angeben.

Wir wollen nur die Korollare angeben, die den Fehler des Lanczos-Galerkin Projektionsverfahrens angewandt auf $e^A\widehat{b}$ für eine hermitsche oder eine schiefhermitsche Matrix in den oben genannten Spezialfällen abschätzt. Für $V_m^H\widehat{b} = 0$ gilt

$$\widehat{\epsilon}_{\widehat{m}} \leq 3\Psi(\lceil\frac{\widehat{m}}{2}\rceil),$$

wenn ohne Einschränkung $\|\widehat{b}\| = 1$ ist, und für $\widehat{b} = v_l$ mit $1 < l \leq m$

$$\widehat{\epsilon}_{\widehat{m}} \leq M\left(1+\frac{|h_{m+1,m}|}{d(\Gamma)}\right)\Psi(m-l+1+\lceil\frac{\widehat{m}}{2}\rceil).$$

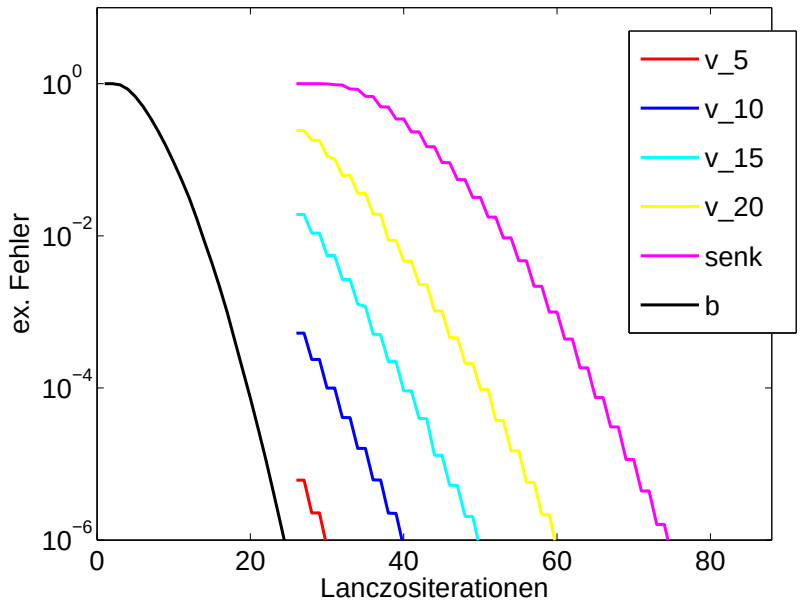

Abbildung 4.1: Exakter Fehler von $e^A b$ und $e^A \widehat{b}$ für versch. $\widehat{b}$ des Beispiels 4.1

Damit gilt nun das Korollar:

Korollar 4.2. *Sei A eine hermitsche, negative semidefinite Matrix mit Eigenwerten in dem Intervall $[-4\rho, 0]$. Dann kann der Fehler der Approximation des Lanczos-Galerkin Projektionsverfahrens an $e^{hA}\widehat{b}$ unter Verwendung des Krylovraumes $\mathcal{K}_m(A, b)$ für die Spezialfälle folgendermaßen abgeschätzt werden. Steht $\widehat{b}$ senkrecht auf V_m, dann ist (mit $\beta > 0.92$)*

$$\widehat{\epsilon}_{\widehat{m}} \leq 3\left(12\frac{\rho h}{\nu_1^2} + 8\frac{\sqrt{\rho h}}{\nu_1}\right) e^{-\beta\nu_1^2/(4\rho h)}, \qquad \sqrt{4\rho h} \leq \nu_1 \leq 2\rho h$$

$$\widehat{\epsilon}_{\widehat{m}} \leq 3\left(5(\rho h)^{-1} + 3\sqrt{\pi}(\rho h)^{-1/2}\right) e^{(\rho h)^2/\nu_1} e^{-2\rho h}\left(\frac{e\rho h}{\nu_1}\right)^{\nu_1}, \qquad \nu_1 \geq 2\rho h.$$

mit $\nu_1 = \lceil \frac{\widehat{m}}{2} \rceil$. Und für $\widehat{b} = v_l$ mit $1 < l \leq m$ gilt

$$\widehat{\epsilon}_{\widehat{m}} \leq 3\gamma_1\left(12\frac{\rho h}{\nu_2^2} + 8\frac{\sqrt{\rho h}}{\nu_2}\right) e^{-\beta\nu_2^2/(4\rho h)}, \qquad \sqrt{4\rho h} \leq \nu_2 \leq 2\rho h$$

$$\widehat{\epsilon}_{\widehat{m}} \leq 3\gamma_2\left(5(\rho h)^{-1} + 3\sqrt{\pi}(\rho h)^{-1/2}\right) e^{(\rho h)^2/\nu_2} e^{-2\rho h}\left(\frac{e\rho h}{\nu_2}\right)^{\nu_2}, \qquad \nu_2 \geq 2\rho h.$$

mit $\nu_2 = m - l + 1 + \lceil \frac{\widehat{m}}{2} \rceil$ und

$$\gamma_1 = 1 + \frac{4|h_{m+1,m}|\rho h^2}{\nu_2^2} \quad \text{und} \quad \gamma_2 = 1 + \frac{|h_{m+1,m}|}{\rho(\frac{\nu_2}{\rho h} + \frac{\rho h}{\nu_2} - 2)}.$$

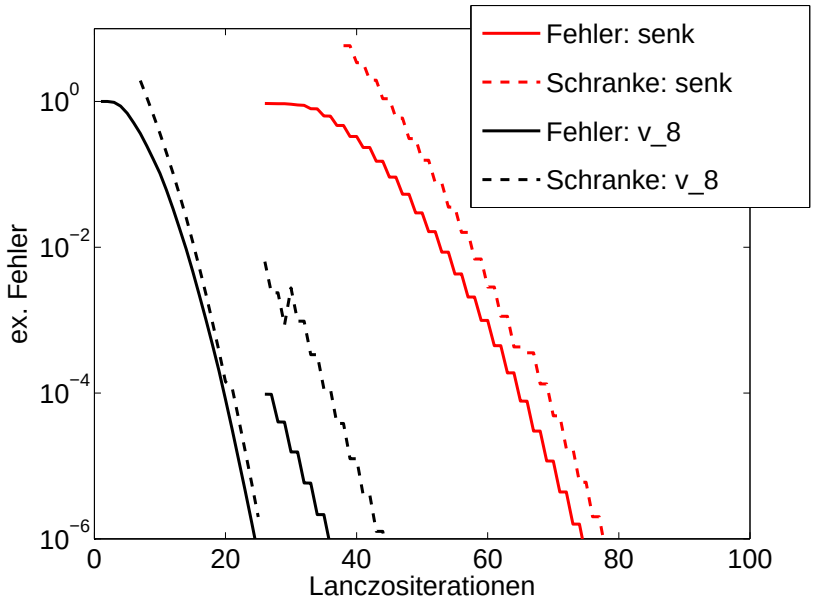

Abbildung 4.2: Fehler und Fehlerschranken für das symmetrische Beispiel 4.2 mit $\widehat{b}$ senkrecht auf V_m und $\widehat{b} = v_8$

Beispiel 4.2. Um ein Beispiel dieser Fehlerschranken anzuführen, nehmen wir wieder die Diagonalmatrix $A \in \mathbb{C}^{N,N}$ mit $N = 1000$, deren äquidistante Eigenwerte auf dem Intervall $[-40, 0]$ liegen, einen Zufallsvektor $b \in \mathbb{R}^N$ und berechnen die Approximation an $e^A b$ und $e^A \widehat{b}$ mit $V_m^H \widehat{b} = 0$ und $\widehat{b} = v_8$. Die Abbildung 4.2 zeigt den exakten Fehler und die zugehörige Fehlerschranke pro Lanczositeration für den Lanczosprozess mit Lanczos-Galerkin Projektion für beide Vektoren. Die Fehlerschranken aus Korollar 4.2 werden hierbei genutzt und durch entsprechende gestrichelte Linien dargestellt. $\diamond$

Man sieht, dass für dieses Beispiel die Fehlerschranke für $V_m^H \widehat{b} = 0$ scharf ist, aber für $\widehat{b} = v_8$ etwas zu pessimistisch.

Korollar 4.3. *Sei A eine schiefhermitsche Matrix mit Eigenwerten im Intervall $i[-2\rho,\ 2\rho]$. Dann kann der Fehler der Approximation des Lanczos-Galerkin Projektionsverfahrens an $e^{hA}\widehat{b}$ unter Verwendung des Krylovraumes $\mathcal{K}_m(A, b)$ für die Spezialfälle folgendermaßen abgeschätzt werden. Steht $\widehat{b}$ senkrecht auf V_m, dann ist*

$$\widehat{\epsilon}_{\widehat{m}} \leq \left(4(\rho h)^{-1} + 11(\rho h)^{-1/2}\right) e^{-(\rho h)^2/\nu_1} \left(\frac{e\rho h}{\nu_1}\right)^{\nu_1}, \quad \nu_1 \geq 2\rho h \tag{4.19}$$

mit $\nu_1 = \lceil \frac{\widehat{m}}{2} \rceil$. Und für $\widehat{b} = v_l$ mit $1 < l \leq m$ gilt

$$\widehat{\epsilon}_{\widehat{m}} \leq \gamma \left(4(\rho h)^{-1} + 11(\rho h)^{-1/2}\right) e^{-(\rho h)^2/\nu_2} \left(\frac{e\rho h}{\nu_2}\right)^{\nu_2}, \quad \nu_2 \geq 2\rho h \tag{4.20}$$

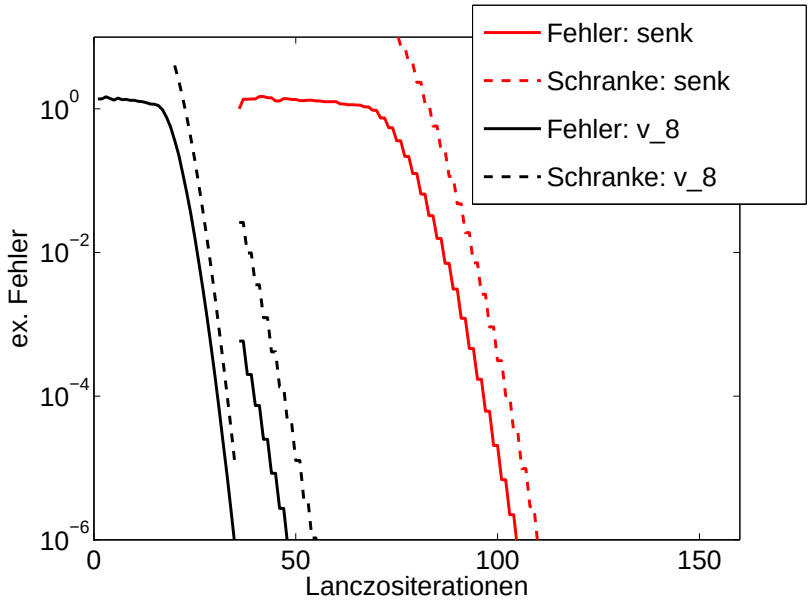

Abbildung 4.3: Fehler und Fehlerschranken für das schiefsymmetrische Beispiel 4.3 mit $\widehat{b}$ senkrecht auf V_m und $\widehat{b} = v_8$

mit $\nu_2 = m - l + 1 + \lceil \frac{\widehat{m}}{2} \rceil$ *und*

$$\gamma = 1 + \frac{|h_{m+1,m}|}{\rho(4 + (\frac{\nu_2}{\rho h} - \frac{\rho h}{\nu_2})^2)}.$$

Beispiel 4.3. Als Beispiel nehmen wir hier die Diagonalmatrix $A \in \mathbb{C}^{N,N}$ mit $N = 1000$ und äquidistanten Eigenwerten in dem Intervall $[-20i, 20i]$, einen Zufallsvektor $b \in \mathbb{R}^N$ und berechnen die Approximation an $e^A b$ und $e^A \widehat{b}$ mit $V_m^H \widehat{b} = 0$ und $\widehat{b} = v_8$. Die Abbildung 4.3 zeigt den exakten Fehler und die zugehörige Fehlerschranke pro Lanczositeration für den Lanczosprozess mit Lanczos-Galerkin Projektion für beide Vektoren. Hierfür werden die Fehlerschranken (4.19) und (4.20) genutzt, die sich als hinreichend scharf herausstellen. $\diamond$

4.4 Lanczos-Galerkin mit mehreren Vektoren

Wendet man die Matrixfunktion $f(A)$ auf mehrere Vektoren $b_1, \ldots, b_n$ an, so hat man verschiedene Möglichkeiten, das Lanczos-Galerkin Projektionsverfahren zu nutzen.

Eine Möglichkeit ist, bei jeder neuen Berechnung $f(A)b_i$, den Vektor b_i auf die Vereinigung aller bisher berechneten Unterräume zu projizieren. Im Folgenden zeigt der Index $^{(i)}$ die Zugehörigkeit zum Vektor b_i an. Wir nehmen an, die Approximationen an $f(A)b_1$ und $f(A)b_2$ sind berechnet, wie in Abschnitt 4.2 beschrieben. Dann gilt Relation

(4.9) in der Form

$$A\widetilde{V}_{\nu_2} = \widetilde{V}_{\nu_2}\widetilde{H}_{\nu_2} + v^{(2)}_{m_2+1}(h^{(2)}_{m_2+1,m_2}e^T_{\nu_2} + h^{(2)}_{m_2+1,m_2-1}e^T_{\nu_2-1}) + h^{(2)}_{m_2+2,m_2}v^{(2)}_{m_2+2}e^T_{\nu_2}$$

mit $\nu_i = \sum_{j=1}^{i} m_j$,

$$\widetilde{V}_{\nu_i} = [V^{(1)}_{m_1}, \ldots, V^{(i)}_{m_i}] \in \mathbb{C}^{N,\nu_i}$$

und

$$\widetilde{H}_{\nu_i} = \left[\begin{array}{c|c} \widetilde{H}_{\nu_{i-1}} & 0 \\ \hline \sum_{j=1}^{i-1} e_j\mathcal{H}^{(i)}_j & H^{(i)}_{m_i} \end{array}\right] \in \mathbb{C}^{\nu_i,\nu_i},$$

wobei

$$\mathcal{H}^{(i)}_j = \sum_{k=0}^{i-j} h^{(i)}_{m_i+j,m_i-k}e^T_{\nu_{i-1}}$$

ist. $\widetilde{H}_{\nu_i}$ ist eine obere Hessenbergmatrix mit $(i-1)$ zusätzlichen unteren Nebendiagonalen oder im Lanczosfall eine Bandmatrix mit Bandbreite $p = i$. Nun projiziert man b_3 auf $\widetilde{V}_{\nu_2}$ und berechnet an $z_3 = f(A)b_3$ die erste Approximation

$$z^{(3)}_0 = \widetilde{V}_{\nu_2} f(\widetilde{H}_{\nu_2})\widetilde{V}^H_{\nu_2} b_3$$

genauso wie in (4.6). Ist diese Approximation noch nicht gut genug, muss man sie verfeinern. Dazu betrachtet man mit

$$y^{(3)}_0(\lambda) = (\lambda I - \widetilde{H}_{\nu_2})^{-1}\widetilde{V}^H_{\nu_2} b_3$$

das Residuum der zugehörigen linearen Gleichungssysteme

$$\begin{aligned} r^{(3)}_0(\lambda) &= b_3 - (\lambda I - A)\widetilde{V}_{\nu_2} y^{(3)}_0(\lambda) \\ &= (I - \widetilde{V}_{\nu_2}\widetilde{V}^H_{\nu_2})b_3 + \beta^{(3)}_{1,m_2}(\lambda)v_{m_2+1} + \beta^{(3)}_{2,m_2}(\lambda)v_{m_2+2}. \end{aligned}$$

mit

$$\beta^{(3)}_{1,m_2}(\lambda) = (h^{(2)}_{m_2+1,m_2}e^T_{\nu_2} + h^{(2)}_{m_2+1,m_2-1}e^T_{\nu_2-1})y^{(3)}_0(\lambda) = \mathcal{H}^{(2)}_1 y^{(3)}_0(\lambda)$$

und

$$\beta^{(3)}_{2,m_2}(\lambda) = h^{(2)}_{m_2+2,m_2}e^T_{\nu_2}y^{(3)}_0(\lambda) = \mathcal{H}^{(2)}_2 y^{(3)}_0(\lambda).$$

Jetzt berechnet man eine Orthonormalbasis $[v^{(3)}_1,\ v^{(3)}_2,\ v^{(3)}_3]$ des Raumes, der durch die Vektoren, die das Residuum $r^{(3)}_0(\lambda)$ bilden, aufgespannt wird, wobei man $v^{(3)}_1 = v^{(2)}_{m_2+1}$ und $v^{(3)}_2 = v^{(2)}_{m_2+2}$ setzt. Dann ist

$$(I - \widetilde{V}_{\nu_2}\widetilde{V}^H_{\nu_2})b_3 = \sum_{k=1}^{3} \alpha_{k3}v^{(3)}_k.$$

Mit den drei Orthonormalbasisvektoren startet man das Band Krylovverfahren von Ruhe. Man bekommt nach m_3 Schritten die Relation

$$A\widetilde{V}_{\nu_3} = \widetilde{V}_{\nu_3}\widetilde{H}_{\nu_3} + v^{(3)}_{m_3+1}\mathcal{H}^{(3)}_1 + v^{(3)}_{m_3+2}\mathcal{H}^{(3)}_2 + v^{(3)}_{m_3+3}\mathcal{H}^{(3)}_2.$$

Die Approximation an z_3 ist dann

$$z^{(3)}_{m_3} = \widetilde{V}_{\nu_3} f(\widetilde{H}_{\nu_3})\widetilde{b}_3 = z^{(3)}_0 + V^{(3)}_{m_3}\left(f(\widetilde{H}_{\nu_3})\widetilde{b}_3\right)_{(\nu_2+1):\nu_3}$$

mit

$$\widetilde{b}_3 = \begin{bmatrix} \widetilde{V}^H_{\nu_2} b_3 \\ \sum_{k=1}^{3} \alpha_{k3} e_k \end{bmatrix}.$$

Für die nächsten Approximationen an $f(A)b_i$ mit $i > 3$ geht man wieder entsprechend vor.

Allgemein gilt, dass die Approximation an $z_i = f(A)b_i$ durch

$$z^{(i)}_{m_i} = \widetilde{V}_{\nu_i} f(\widetilde{H}_{\nu_i})\widetilde{b}_i = z^{(i)}_0 + V^{(i)}_{m_i}\left(f(\widetilde{H}_{\nu_i})\widetilde{b}_i\right)_{(\nu_{i-1}+1):\nu_i}$$

mit

$$\widetilde{b}_i = \begin{bmatrix} \widetilde{V}^H_{\nu_{i-1}} b_i \\ \sum_{k=1}^{i} \alpha_{ki} e_k \end{bmatrix}$$

gegeben ist, wobei die erste Approximation

$$z^{(i)}_0 = \widetilde{V}_{\nu_{i-1}} f(\widetilde{H}_{\nu_{i-1}})\widetilde{V}^H_{\nu_{i-1}} b_i$$

ist. Hierbei bekommt man die Faktoren α_{ki} durch die Zerlegung von $(I - \widetilde{V}_{\nu_{i-1}}\widetilde{V}^H_{\nu_{i-1}})b_i$ in die Orthonormalbasisvektoren, also

$$(I - \widetilde{V}_{\nu_{i-1}}\widetilde{V}^H_{\nu_{i-1}})b_i = \sum_{k=1}^{i} \alpha_{ki} v^{(i)}_k.$$

Der Vorteil dieses Verfahrens ist, dass die Wahrscheinlichkeit, dass man sehr wenige bis gar keine neuen Kryloviterationen für eine gute Approximation an $f(A)b_i$ braucht, mit jedem neuen Vektor b_i steigt, da schon ein großer Unterraum zur Verfügung steht. Ein Nachteil ist, dass man das ganze $\widetilde{V}_{\nu_i}$ speichern muss, da diese Matrix die Dimension $N \times \nu_i$ hat, kann der Speicherbedarf sehr groß werden. Ein weiterer Nachteil ist, dass Tests in der Praxis gezeigt haben, dass dieses Verfahren mit wachsendem i instabil wird und deshalb das Verfahren neu gestartet werden muss. Wir fassen den wesentlichen Speicher- und Rechenaufwand zur Berechnung von $f(A)b_i$ mit diesem Verfahren ohne Restart zusammen.

Speicheraufwand: Wir zählen nur wie viele Vektoren der Dimension N gespeichert werden müssen, da diese den wesentlichen Speicheraufwand ausmachen. Von den schon berechneten Krylovräumen befindet sich die Matrix $\widetilde{V}_{\nu_{i-1}}$ und

$$v^{(i-1)}_{m_{i-1}+1}, \ldots, v^{(i-1)}_{m_{i-1}+i-1}$$

noch im Speicher. Dann wird

$$v_1^{(i)} = v_{m_{i-1}+1}^{(i-1)}, \ldots, v_{i-1}^{(i)} = v_{m_{i-1}+i-1}^{(i-1)}$$

gesetzt und $v_i^{(i)}$ berechnet. Durch die folgende Band Lanczos Methode werden m_i neue Vektoren berechnet und als $v_l^{(i)}$ mit $l = i+1, \ldots, m_i + i$ gespeichert. Die dann berechnete Approximation an $f(A)b_i$ muss auch gespeichert werden, kann aber nach Verwendung wieder gelöscht werden. Insgesamt sind also dann

$$\nu_{i-1} + (i-1) + 1 + m_i + 1 = \nu_i + i + 1$$

Vektoren der Dimension N gespeichert. Der Speicheraufwand für $\widetilde{H}_{\nu_i} \in \mathbb{C}^{\nu_i,\nu_i}$ und $\widetilde{b}_i \in \mathbb{C}^{\nu_i}$ ist dagegen zu vernachlässigen.

Falls aber mit einem Verfahren Approximationen an Produkte von unterschiedlichen Funktionen einer unterschiedlich skalierten Matrix A mit demselben Vektor b_i, also an $f_{j_1}(h_{j_2}A)b_i$ mit $j_1 = 1, \ldots, \vartheta_1$, $j_2 = 1, \ldots, \vartheta_2$ und $\vartheta = \vartheta_1\vartheta_2$, berechnet werden sollen, wie das zum Beispiel bei einem mehrstufigem exponentiellen Runge-Kutta Verfahren der Fall ist, so müssen insgesamt ϑ Approximationen gespeichert werden, die aber nach Verwendung wieder gelöscht werden können. In diesem Fall sind dann zusammen $\nu_i + i + \vartheta$ Vektoren der Dimension N zu speichern.

Rechenaufwand: Ist das Krylovverfahren ein Lanczosverfahren, so muss man zusätzlich zu den m_i Matrixvektormultiplikationen mit A oder im shift-and-invert Fall Lösungen linearer Gleichungssysteme des Band Lanczos Algorithmus und den Funktionsauswertungen von $\widetilde{H}_{\nu_i}$ noch $\widetilde{V}_{\nu_{i-1}}\widetilde{V}_{\nu_{i-1}}^H b_i$ und die Orthogonalisierung von $b_i - \widetilde{V}_{\nu_{i-1}}\widetilde{V}_{\nu_{i-1}}^H b_i$ gegen die $v_1^{(i)}, \ldots, v_{i-1}^{(i)}$ berechnen. Dies ist der wesentliche Rechenaufwand des Verfahrens.

Da wir erwarten, dass für die Approximation von $f(A)b_i$ mit $i \geq 2$ nur noch sehr wenige Iterationen zusätzlich benötigt werden, kann man annehmen, dass ν_i nicht viel größer ist als m_1.

Eine andere Möglichkeit, die Matrixfunktion $f(A)$ auf mehrere Vektoren $b_1, \ldots, b_n$ anzuwenden, ist, dass man b_i nur auf den Krylovraum des ersten Vektors, der von $V_{m_1}^{(1)}$ aufgespannt wird, projiziert. Man interpretiert also jeden neuen Vektor b_i als „zweiten“ Vektor und geht dann wie im Abschnitt 4.2 vor, indem man $\widehat{b} = b_i$ setzt. Vorteil dieses Verfahrens ist, dass man nicht das ganze $\widetilde{V}_{\nu_i}$ speichern muss, sondern nur $V_{m_1}^{(1)}$ und die Vektoren, die für die Verfeinerung der neuen Approximation benötigt werden. Diese können aber, wenn die Approximation berechnet wurde, gelöscht werden. Der Nachteil ist, dass in Anwendungen mit wachsendem i der Vektor b_i sich nicht mehr hauptsächlich durch Vektoren aus $V_{m_1}^{(1)}$ darstellen läßt und daher die Norm $\|(I - V_{m_1}^{(1)}{V_{m_1}^{(1)}}^H)b_i\|$ groß wird. Dann konvergiert das Verfahren, wie wir im Satz 4.1 gesehen haben, recht langsam. Daher sollte man nun einen Restart durchführen, indem man b_i als den ersten Vektor interpretiert, an $f(A)b_i$ die Standard Krylovapproximation berechnet und dann die nächsten Vektoren wieder wie oben als „zweiten“ Vektor anwendet. Dies wiederholt

man solange, bis alle Approximationen erstellt sind. Auch für dieses Verfahren betrachten wir den wesentlichen Speicher- und Rechenaufwand zur Approximation von $f(A)b_i$ für den Fall, dass kein Restart benutzt wurde.

Speicheraufwand: Wieder interessieren wir uns nur für die Anzahl der gespeicherten Vektoren der Dimension N. Von dem ersten berechneten Krylovraum sind $v_1^{(1)}, \ldots, v_{m_1+1}^{(1)}$ im Speicher vorhanden. Dann setzt man $v_1^{(i)} = v_{m_1+1}^{(1)}$ und berechnet $v_2^{(i)}$. Nun folgen m_i Iterationen des Band Lanczos Verfahrens, wobei $v_3^{(i)}, \ldots, v_{m_i+2}^{(i)}$ Vektoren erstellt werden. Mit diesen Vektoren wird die Approximation an $f(A)b_i$ erstellt, die auch gespeichert werden muss. Also müssen $m_1 + m_i + 3$ Vektoren gespeichert werden. Nach der Berechnung und Verwendung der Approximation an $f(A)b_i$ können aber die $m_i + 1$ Vektoren $v_2^{(i)}, \ldots, v_{m_i+2}^{(i)}$ und die Approximation wieder gelöscht werden und der Speicheraufwand ist dann wieder so groß wie vorher.

Falls aber mit einem Verfahren Approximationen an Produkte von unterschiedlichen Funktionen einer unterschiedlich skalierten Matrix A mit demselben Vektor b_i, also an $f_{j_1}(h_{j_2}A)b_i$ mit $j_1 = 1, \ldots, \vartheta_1$, $j_2 = 1, \ldots, \vartheta_2$ und $\vartheta = \vartheta_1\vartheta_2$, berechnet werden sollen, befinden sich im Speicher insgesamt $m_1 + m_i + \vartheta + 2$ Vektoren der Dimension N, von denen man $m_i + \vartheta + 1$ später wieder entfernen kann.

Rechenaufwand: Zusätzlich zu den m_i Matrixvektormultiplikationen mit A oder im shift-and-invert Fall Lösungen linearer Gleichungssysteme des Band Lanczos Algorithmus und den Funktionsauswertungen von $\widehat{H}$, müssen $V_{m_1}^{(1)} {V_{m_1}^{(1)}}^H b_i$ und die Orthogonalisierung von $b_i - V_{m_1}^{(1)} {V_{m_1}^{(1)}}^H b_i$ gegen $v_{m_1+1}^{(1)}$ berechnet werden. Dies ist der wesentliche Rechenaufwand des Verfahrens.

Auch hier erwarten wir, dass für die Approximation von $f(A)b_i$ mit $i \geq 2$ nur noch sehr wenige Iterationen des Krylovverfahrens zusätzlich benötigt werden, das heißt, dass m_i klein ist. Dass dies auch der Fall ist, sehen wir im Kapitel über die numerischen Beispiele. Falls m_i zu groß wird, wird ein Restart eingefügt.

Vorstellbar ist auch ein Verfahren, das beide beschriebenen Vorgehensweisen kombiniert, indem man b_i immer nur auf die ersten l Unterräume, die durch die Berechnung der Approximationen an $f(A)b_k$ mit $k = 1, \ldots, l$ erstellt wurden, projiziert.

Kapitel 5

Orthogonalprojektionsverfahren

In diesem Kapitel stellen wir eine andere Methode zur Berechnung von Approximationen von Matrixfunktionen angewandt auf mehrere Vektoren

$$f(A)b_i, \qquad b_i = g(t + c_i h) \tag{5.1}$$

vor, wobei dieselben Bedingungen wie für (4.1) gelten. Im Gegensatz zum letzten Kapitel, in dem wir ein Lanczos-Galerkin Projektionsverfahren genutzt haben, werden wir im Folgenden ein Orthogonalprojektionsverfahren vorstellen, das die QR-Zerlegung der Vektoren b_i verwendet. Zur Berechnung von linearen Gleichungssystemen mit mehreren rechten Seiten haben auch schon Fischer [23] und Lötstedt und M. Nilsson [51] die QR-Zerlegung genutzt. Diese Arbeiten wollen wir zunächst kurz vorstellen.

P. Fischer projiziert zur Lösung von $Ax_i = b_i$ die aktuelle rechte Seite auf die vorherigen rechten Seiten. Dann löst er $A\widehat{x}_i = \widehat{b}_i$, wobei der Vektor $\widehat{b}_i$ den projizierten Vektor b_i darstellt. Die gesuchte Lösung x_i des linearen Gleichungssystems ist eine Linearkombination der Vektoren $\widehat{x}_1, \ldots, \widehat{x}_i$. Er projiziert die verschiedenen Vektoren b_i auf zwei unterschiedlichen Weisen mit dem Gram-Schmidt-Verfahrens, einmal verwendet er die Euklid Norm, das andere Mal die A-Norm $\|x\|_A = \sqrt{x^T Ax}$, wobei A symmetrisch, positiv definit ist. Um Speicherplatz zu sparen, führt er nach einigen rechten Seiten einen Restart durch.

Auch P. Lötstedt und M. Nilsson [51] benutzen die Gram-Schmidt-Orthogonalisierung. Stellt der Vektor $\tilde{x}_i$ die approximierte Lösung von $Ax_i = b_i$ dar, dann berechnen sie die QR-Zerlegung der Matrix $S = [s_1, \ldots, s_m]$ mit $s_i := A\tilde{x}_i = b_i - r_i$, wobei $r_i = b_i - A\tilde{x}_i$ das Residuum ist. Diese Approximation kann man mit einem beliebigen Verfahren zur approximativen Lösung von linearen Gleichungssystemen, von denen einige im Abschnitt 4.1 zitiert sind, berechnen. Mit der Matrix S kann dann $\tilde{x}_{m+1}$ entweder direkt ermittelt oder mit wenigen Schritten eines iterativen Verfahrens berechnt werden, wenn die Vektoren b_i glatt von einander abhängen. Im nächsten Schritt wird der QR-Zerlegung von S mit Hilfe von Up- und Downdating Methoden eine Spalte entnommen und eine neue hinzugefügt. Man kann auch Block-Verfahren benutzen, wenn einem im i-ten Schritt k neue rechte Seiten bekannt sind.

Wir leiten in diesem Kapitel ein Verfahren zur effizienten Berechnung von (5.1) her.

Da für Matrixfunktionen Residuen nicht direkt verfügbar sind, ist eine umfassende Analyse der QR-Zerlegung einer Matrix, der Spalten Funktionsauswertungen entsprechen, unabdingbar. Ohne diese Analyse ist es nicht möglich, geeignete Stopkriterien zu konstruieren und damit einen für die Praxis relevanten Algorithmus zu entwickeln. Aus unseren theoretischen Ergebnissen leiten wir ein neues Stopkriterium für Orthogonalprojektionsverfahren her. Anschließend erläutern wir Variationen des Verfahrens, die Restarts, Up- und Downdating oder Ritzvektoren verwenden. Auch für diese Verfahren ist die Herleitung des Stopkriteriums der aufwendigste Schritt.

5.1 Analyse der Orthogonalisierung

Wir nehmen an, wir kennen n Vektoren $b_1, \dots, b_n \in \mathbb{R}^N$, wobei die $b_i = g(t + c_i h)$ Auswertungen einer glatten Funktion $g : \mathbb{R} \to \mathbb{R}^N$ sind. Es sei

$$[b_1, \dots, b_n] = QR, \qquad Q = [q_1, \dots, q_n] \in \mathbb{R}^{N,n}, \quad R = (r_{i,j})_{i,j=1}^{n} \in \mathbb{R}^{n,n},$$

die QR-Zerlegung dieser Vektoren, wobei Q eine orthonormale Matrix und R eine obere Dreiecksmatrix ist. Zunächst zeigen wir einige Eigenschaften von R, wobei das Ergebnis des folgenden Lemmas wahrscheinlich bekannt ist, wir es aber in der von uns benötigten Form nicht in der Literatur gefunden haben.

Lemma 5.1. *Sei $g : \mathbb{R} \to \mathbb{R}^N$ n-mal stetig differenzierbar und $p_{n-1} \in \mathbb{P}_{n-1}$ das Lagrange Interpolationspolynom an g definiert in den Knoten $a = x_1, \dots, x_n < b$. Dann gilt für $k = 0, \dots, n-1$ mit $x_n < x < b$*

$$\|p_{n-1}^{(k)}(x) - g^{(k)}(x)\| \leq C_{n,k} M_n(x)(x - x_1)^{n-k}. \tag{5.2}$$

Dabei ist $M_i(x) = \max\limits_{\xi \in [x_1, x]} \|g^{(i)}(\xi)\|$ und die Konstante

$$C_{n,k} \leq \frac{1}{(n-k)!} + k(n-1)\left(\frac{\Delta}{\delta}\right)^{n-2}$$

mit $\delta := \min\limits_{\substack{j,k=1 \\ j \neq k}}^{n} |x_j - x_k|$ und $\Delta := |x_n - x_1|$ unabhängig von g.

Beweis. Nach Davis [15, Theorem 3.7.1] ist der Fehler einer Approximation durch Lagrange Interpolation in den Knoten $a \leq x_1 < \ldots < x_n < b$ mit

$$E_{n-1}(x) := g(x) - p_{n-1}(x) = \frac{1}{(n-1)!} \int_a^b g^{(n)}(\tau) \kappa_n(x, \tau) \mathrm{d}\tau$$

gegeben, wobei

$$\kappa_n(x, \tau) = (x - \tau)_+^{n-1} - \sum_{j=1}^{n} (x_j - \tau)_+^{n-1} \ell_j(x)$$

der zugehörige Peanokern,

$$(x-\tau)_+ = \begin{cases} x-\tau, & \text{für } x \geq \tau, \\ 0, & \text{für } x < \tau, \end{cases}$$

und $\ell_j(x) \in \mathbb{P}_{n-1}$ das j-te Lagrange Polynom ist. Sei $x_n < x < b$. Mit

$$\begin{aligned}
\int_a^b \frac{\partial^k}{\partial x^k}\kappa_n(x,\tau)\mathrm{d}\tau &= \int_a^b \frac{\partial^k}{\partial x^k}\left((x-\tau)_+^{n-1} - \sum_{j=1}^n (x_j-\tau)_+^{n-1}\ell_j(x)\right)\mathrm{d}\tau \\
&= \int_a^x \frac{\partial^k}{\partial x^k}(x-\tau)^{n-1}\mathrm{d}\tau - \sum_{j=1}^n \ell_j^{(k)}(x)\int_a^{x_j}(x_j-\tau)^{n-1}\mathrm{d}\tau \\
&= \int_a^x \frac{(n-1)!}{(n-k-1)!}(x-\tau)^{n-1-k}\mathrm{d}\tau + \sum_{j=1}^n \ell_j^{(k)}(x)\frac{1}{n}(x_j-a)^n \\
&= -\frac{(n-1)!}{(n-k)!}(x-a)^{n-k} + \sum_{j=1}^n \ell_j^{(k)}(x)\frac{1}{n}(x_j-a)^n
\end{aligned}$$

gilt für die k-te Ableitung des Interpolationsfehlers mit $k = 0, \ldots, n-1$

$$\begin{aligned}
\|E_{n-1}^{(k)}(x)\| &= \left\| \frac{1}{(n-1)!}\int_a^b g^{(n)}(\tau)\frac{\partial^k}{\partial x^k}\kappa_n(x,\tau)\mathrm{d}\tau \right\| \qquad (5.3) \\
&\leq \frac{1}{(n-1)!}\max_{\xi\in[a,x]}\|g^{(n)}(\xi)\|\int_a^x \left|\frac{\partial^k}{\partial x^k}\kappa_n(x,\tau)\right|\mathrm{d}\tau \\
&\leq M_n(x)\left(\frac{1}{(n-k)!}(x-a)^{n-k} + \sum_{j=1}^n |\ell_j^{(k)}(x)|\frac{1}{n!}(x_j-a)^n\right). \qquad (5.4)
\end{aligned}$$

Der Betrag der k-ten Ableitung des Lagrange Polynoms für $k = 1, \ldots, n-1$ ist

$$\begin{aligned}
|\ell_j^{(k)}(x)| &= \left| \left(\prod_{\substack{l=1 \\ l\neq j}}^n (x_j - x_l)\right)^{-1} \frac{\partial^k}{\partial x^k}\prod_{\substack{l=1 \\ l\neq j}}^n (x-x_l) \right| \\
&= \left| \left(\prod_{\substack{l=1 \\ l\neq j}}^n (x_j - x_l)\right)^{-1} \left(\sum_{j_1=1}^n \cdots \sum_{j_k=j_{k-1}+1}^n k! \prod_{\substack{l=1 \\ l\neq j, l\neq j_1,\ldots,l\neq j_k}}^n (x-x_l)\right)\right|.
\end{aligned}$$

Mit der Abschätung $x - x_l \leq x - x_1$ gilt

$$|\ell_j^{(k)}(x)| \leq k!(x-x_1)^{n-k-1}\left(\sum_{j_1=1}^n \cdots \sum_{j_k=j_{k-1}+1}^n 1\right)\left(\prod_{\substack{l=1 \\ l\neq j}}^n |x_j - x_l|\right)^{-1}.$$

Wegen

$$\left(\sum_{j_1=1}^{n}\cdots\sum_{j_k=j_{k-1}+1}^{n}1\right)=\frac{n!}{(k-1)!}$$

ist damit

$$|\ell_j^{(k)}(x)|\le n!k(x-x_1)^{n-k-1}\left(\prod_{\substack{l=1\\l\neq j}}^{n}|x_j-x_l|\right)^{-1}. \tag{5.5}$$

Da nach Voraussetzung $a=x_1$ gilt, folgt mit (5.4) und (5.5)

$$\|E_{n-1}^{(k)}(x)\|\le M(x)(x-x_1)^{n-k-1}\left(\frac{(x-x_1)}{(n-k)!}+\sum_{j=1}^{n}k\left(\prod_{\substack{l=1\\l\neq j}}^{n}|x_j-x_l|\right)^{-1}(x_j-x_1)^n\right).$$

Mit $(x_j-x_1)=0$ für $j=1$ und der Abschätzung $(x_j-x_1)\le(x-x_1)$ ist

$$\|E_{n-1}^{(k)}(x)\|\le M(x)(x-x_1)^{n-k}\left(\frac{1}{(n-k)!}+\sum_{j=2}^{n}k\left(\prod_{\substack{l=1\\l\neq j}}^{n}|x_j-x_l|\right)^{-1}(x_j-x_1)^{n-1}\right).$$

Da in der Summe alle $j>1$ sind, kommt in jedem Produkt der Faktor $|x_j-x_1|$ vor, den man dann rauskürzen kann. Somit ist

$$\|E_{n-1}^{(k)}(x)\|\le M(x)(x-x_1)^{n-k}\left(\frac{1}{(n-k)!}+\sum_{j=2}^{n}k\left(\prod_{\substack{l=2\\l\neq j}}^{n}|x_j-x_l|\right)^{-1}(x_j-x_1)^{n-2}\right).$$

Mit den jeweiligen Abschätzungen $|x_j-x_l|^{-1}\le\delta^{-1}$ für alle $l=2,\dots,n$ und $(x_j-x_1)\le\Delta$ ist

$$\|E_{n-1}^{(k)}(x)\|\le\left(\frac{1}{(n-k)!}+k(n-1)\left(\frac{\Delta}{\delta}\right)^{n-2}\right)M(x)(x-x_1)^{n-k}$$

und damit das Lemma bewiesen. □

Mit diesem Lemma zeigen wir den folgenden Satz:

Satz 5.2. *Sei $g:\mathbb{R}\to\mathbb{R}^N$ n-mal stetig differenzierbar, seien $0\le c_1<\ldots<c_n$ feste Knoten, $g_i:=g(t+c_ih)$ Funktionsauswertungen und sei die Schrittweite $h>0$. Die QR-Zerlegung*

$$[g_1,\dots,g_n]=QR$$

sei mit $Q = [q_1, \ldots, q_n]$ *und* $R = \{r_{ik}\}_{ik}$ *gegeben. Dann gilt für* $i, k = 1, \ldots, n$

$$|r_{ik}| \leq C_i M_{i-1} h^{i-1}, \qquad \text{für } i \leq k, \tag{5.6}$$

$$|r_{ik}| = 0, \qquad \text{für } i > k, \tag{5.7}$$

wobei

$$M_i = \max_{\xi \in [t+c_1 h, t+c_n h]} \|g^{(i)}(\xi)\|$$

ist und die Konstanten C_i *von den Knoten abhängen, aber unabhängig von* h *und* g *sind.*

Bemerkung: Die Matrix R aus dem obigen Satz hat dann folgende Struktur

$$R = \begin{bmatrix} O(1) & \cdots & \cdots & O(1) \\ & O(h) & \cdots & O(h) \\ & & \ddots & \vdots \\ & & & O(h^{n-1}) \end{bmatrix}$$

Falls die verschiedenen Knoten c_i nicht paarweise verschieden sind, ist der Rang von $[g_1, \ldots, g_n]$ kleiner als n und damit die Abschätzung (5.6) zu pessimistisch.

Beweis. Ohne Beschränkung der Allgemeinheit setzen wir $c_1 = 0$. Sei $Q = [q_1, \ldots, q_n]$. R ist eine obere Dreiecksmatrix und damit ist $r_{ik} = 0$ für $i > k$. Nach Konstruktion gilt für jedes $k \in \{1, \ldots, n\}$

$$g_k = r_{1k} q_1 + r_{2k} q_2 + \ldots + r_{kk} q_k.$$

Da Q orthogonal ist, gilt $r_{ik} = q_i^T g_k$ für $i \leq k$. Für $i = 1$ und $k = 1, \ldots, n$ gilt dann

$$|r_{1k}| = |q_1^T g_k| \leq \|g(t + c_k h)\| \leq M_0. \tag{5.8}$$

Damit ist die Aussage für $i = 1$ gezeigt. Von nun an betrachten wir $i > 1$. Sei p_{i-2} das Interpolationspolynom an g in den Knoten $t, t + c_2 h, \ldots, t + c_{i-1} h$, dann ist

$$p_{i-2}(\tau) = \sum_{j=1}^{i-1} \ell_j(\tau) g_j. \tag{5.9}$$

Der Fehler dieser Interpolation sei

$$E_{i-2}(\tau) := g(\tau) - p_{i-2}(\tau), \tag{5.10}$$

dann gilt für die Ableitungen dieser Fehlerfunktion

$$E_{i-2}^{(k)}(\tau) = g^{(k)}(\tau), \qquad \text{für } k \geq i - 1. \tag{5.11}$$

Es gilt für $i = 2, \ldots, n$ wegen (5.9)

$$q_i^T g^{(k)}(\tau) = q_i^T E_{i-2}^{(k)}(\tau) + \sum_{j=1}^{i-1} \ell_j^{(k)}(\tau) r_{ij} = q_i^T E_{i-2}^{(k)}(\tau), \qquad \text{für } k \geq 0. \tag{5.12}$$

Nun wollen wir $|r_{ik}|$ für $i = 2, \ldots, k$ und $k > i$ abschätzen. Für $k = i$ folgt direkt aus Lemma 5.1 und mit (5.12)

$$|r_{ii}| = |q_i^T g(t + c_i h)| \leq \|E_{i-2}((t + c_i h)\| \leq C_{i-1,0} M_{i-1}((c_i - c_1)h)^{i-1} \leq C_i M_{i-1} h^{i-1}.$$

Für $k > i$ erhält man aus der Taylorentwicklung mit einem $\theta \in (c_i, c_k)$

$$\begin{aligned}
|r_{ik}| &= |q_i^T g(t + c_k h)| \\
&\leq \|E_{i-2}(t + c_k h)\| \\
&\leq \sum_{j=0}^{i-2} \frac{(c_k - c_i)^j}{j!} h^j \|E_{i-2}^{(j)}(t + c_i h)\| + \frac{(c_k - c_i)^{i-1}}{(i-1)!} h^{i-1} \|E_{i-2}^{(i-1)}(t + \theta h)\|.
\end{aligned}$$

Mit (5.11) für $k = i - 1$ ist dann

$$\begin{aligned}
|r_{ik}| &\leq \sum_{j=0}^{i-2} \frac{(c_k - c_i)^j}{j!} h^j \|E_{i-2}^{(j)}(t + c_i h)\| + \frac{(c_k - c_i)^{i-1}}{(i-1)!} h^{i-1} \|g^{(i-1)}(t + \theta h)\| \\
&\leq \sum_{j=0}^{i-2} \frac{(c_k - c_i)^j}{j!} h^j \|E_{i-2}^{(j)}(t + c_i h)\| + \frac{(c_k - c_i)^{i-1}}{(i-1)!} h^{i-1} M_{i-1}
\end{aligned}$$

und mit Lemma 5.1 folgt

$$\begin{aligned}
|r_{ik}| &\leq \sum_{j=0}^{i-2} \frac{(c_k - c_i)^j}{j!} h^j C_{i-1,j} M_{i-1}((c_i - c_1)h)^{i-1-j} + \frac{(c_k - c_i)^{i-1}}{(i-1)!} h^{i-1} M_{i-1} \\
&\leq C_i M_{i-1} h^{i-1}.
\end{aligned}$$

Damit ist dieser Satz bewiesen. □

5.2 Orthogonalprojektionsverfahren

In diesem Abschnitt werden wir Orthogonalprojektionsverfahren zur Approximation von

$$z_i = f(A)b_i$$

vorstellen, die eine QR-Zerlegung der rechten Seiten benutzen. Hierfür verwenden wir die Ergebnisse aus dem vorherigen Abschnitt 5.1.

5.2.1 Grundidee

Es sei die QR-Zerlegung von $[b_1, \ldots, b_n] = QR$ mit einer orthonormalen Matrix Q und einer oberen Dreiecksmatrix R gegeben. Dann kann man jede Lösung

$$z_i = f(A)b_i = f(A)[q_1, \ldots, q_i] \begin{bmatrix} r_{1,i} \\ \vdots \\ r_{i,i} \end{bmatrix}, \qquad i = 1, \ldots, n \tag{5.13}$$

durch eine Linearkombination der Produkte $f(A)q_1, \ldots, f(A)q_i$ darstellen. Wir definieren

$$w_i := f(A)q_i \qquad \text{für} \qquad i = 1, \ldots, n.$$

Wir nehmen an, dass wir alle Approximationen $\widetilde{z}_i$ an z_i mit $i = 1, \ldots, n$ zu einer Genauigkeit *tol* berechnen wollen, das heißt, wir müssen

$$\|z_i - \widetilde{z}_i\| \leq tol, \qquad i = 1, \ldots, n \tag{5.14}$$

fordern. Die fundamentale Idee der Orthogonalprojektionsverfahren ist, dass man anstatt die Approximationen $\widetilde{z}_i$ direkt mit einem Krylovverfahren zu berechenen, diese durch eine Linearkombination der Vektoren $\widetilde{w}_l$ mit $l = 1, \ldots, i$ darzustellen, wobei die Vektoren $\widetilde{w}_i$ Näherungslösungen an die Produkte $w_i = f(A)q_i$ sind. Mit (5.13) ist diese Linearkombination durch

$$\widetilde{z}_i = \sum_{l=1}^{i} r_{l,i}\widetilde{w}_l \tag{5.15}$$

gegeben. Die Approximationen $\widetilde{w}_i$ mit $i = 1, \ldots, n$ berechnen wir mit Hilfe eines Krylovverfahrens zu der Genauigkeit tol_i, das heißt, dass

$$\|w_i - \widetilde{w}_i\| \leq tol_i \qquad \text{für} \qquad i = 1, \ldots, n \tag{5.16}$$

erfüllt sein muss. Wie groß diese Genauigkeit tol_i genau gewählt werden muss, um (5.14) zu garantieren, sehen wir später. Die Approximationen an $z_1, \ldots, z_n$ berechnen wir also durch Erstellung der Krylovräume

$$\mathcal{K}_{m_1}(A, q_1), \mathcal{K}_{m_2}(A, q_2), \ldots, \mathcal{K}_{m_n}(A, q_n),$$

anstatt der Krylovräume

$$\mathcal{K}_{\widehat{m}_1}(A, b_1), \mathcal{K}_{\widehat{m}_2}(A, b_2), \ldots, \mathcal{K}_{\widehat{m}_n}(A, b_n), \tag{5.17}$$

Wir werden zeigen, dass

$$tol_i = \frac{tol}{n} O(h^{1-i}) \tag{5.18}$$

gilt. Somit wird tol_i umso größer, je größer i wird und deshalb wird die benötigte Dimension m_i des jeweiligen Krylovraumes $\mathcal{K}_{m_i}(A, q_i)$ umso kleiner. Da *tol* gleichbleibt, wären die verschiedenen $\widehat{m}_i$ aus (5.17) alle ungefähr gleich groß, falls wir das Verfahren ohne Projektion verwenden würden. Also gilt, je mehr Produkte $f(A)q_i$ wir mit dem Orthogonalprojektionsverfahren berechnen, umso weniger Kryloviterationen braucht man insgesamt im Vergleich zu der Anzahl der Kryloviterationen des Standardverfahrens. Daraus folgt, dass man durch Verwendung des Orthogonalprojektionsverfahrens erheblich Rechenzeit sparen kann.

In fast allen Anwendungen, die wir betrachten, sind die b_i mit $i = 1, \ldots, n$ nicht gleichzeitig bekannt, sondern meistens hängt b_i von $z_{i-1} = f(A)b_{i-1}$ ab. Das heißt,

dass wir die QR-Zerlegung schrittweise mit einem Gram-Schmidt Verfahren aufbauen müssen: Wir nehmen an, dass die QR-Zerlegung

$$[b_1, \ldots, b_{i-1}] = Q_{i-1} R_{i-1}$$

mit $Q_{i-1} = [q_1, \ldots, q_{i-1}] \in \mathbb{R}^{N,i-1}$ und $R_{i-1} \in \mathbb{R}^{i-1,i-1}$, die Approximationen an $w_l = f(A)q_l$ mit $l = 1, \ldots, i-1$ zur Genauigkeit tol_l und damit die Approximationen an $z_l = f(A)b_l$ zur Genauigkeit tol bekannt sind. Nun wollen wir die Approximation an $z_i = f(A)b_i$ berechnen. Dafür müssen wir zuerst das b_i in die schon vorhandene QR-Zerlegung einfügen. Wir orthogonalisieren b_i gegen Q_{i-1}, indem wir

$$\widetilde{q}_i = (I - Q_{i-1} Q_{i-1}^T) b_i$$

berechnen. Dann ergänzt der normierte Vektor $q_i = \widetilde{q}_i / \|\widetilde{q}_i\|$ das Q_{i-1} zu

$$Q_i = [q_1, \ldots, q_{i-1}, q_i].$$

Das b_i lässt sich durch eine Linearkombination der q_l mit $l = 1, \ldots, i$ darstellen als

$$b_i = \sum_{l=1}^{i} r_{li} q_l.$$

Mit diesen neuen Koeffizienten erweitert man die Matrix R_{i-1} zu

$$R_i = \left[\begin{array}{c|c} & r_{1,i} \\ R_{i-1} & \vdots \\ & r_{i-1,i} \\ \hline 0 \cdots 0 & r_{i,i} \end{array}\right]. \tag{5.19}$$

Jetzt kann man die Approximation $\widetilde{w}_i$ an $w_i = f(A)q_i$ mit einem Krylovverfahren zur Genauigkeit tol_i berechnen und die gesuchte Approximation $\widetilde{z}_i$ an $z_i = f(A)b_i$ mit der Linearkombination (5.15) erstellen.

Im Folgenden zeigen wir, welchen Wert man für tol_i wählen muss, damit die Eigenschaft (5.18) erfüllt wird.

Toleranzen

Zunächst betrachten wir wieder den Fall, dass alle Vektoren $b_1, \ldots, b_n$ gleichzeitig bekannt sind, dann gilt mit Hilfe der QR Zerlegung der Vektoren

$$\|z_n - \widetilde{z}_n\| = \|\sum_{i=1}^{n} r_{i,n}(w_i - \widetilde{w}_i)\| \leq \sum_{i=1}^{n} |r_{i,n}| \|w_i - \widetilde{w}_i\|. \tag{5.20}$$

Um nun (5.14) für $i = n$ zu erfüllen, setzen wir für die Toleranz tol_i in (5.16)

$$tol_i = \frac{tol}{n|r_{i,n}|}. \tag{5.21}$$

Damit gilt

$$\|z_n - \widetilde{z}_n\| \leq \sum_{i=1}^{n} |r_{i,n}| tol_i = n\frac{tol}{n} = tol$$

und für $i < n$

$$\|z_i - \widetilde{z}_i\| \leq \sum_{j=1}^{i} |r_{j,i}| tol_j = \sum_{j=1}^{i} \frac{|r_{j,i}|}{n|r_{j,n}|} tol. \tag{5.22}$$

Wir haben angenommen, dass die $b_i = g(t+c_i h)$ mit $i = 1, \ldots, n$ Funktionsauswertungen einer glatten Funktion g sind, deshalb können wir Satz 5.2 anwenden und wissen dann, dass die Werte $r_{j,1}, \ldots, r_{j,j}$ für jedes $1 \leq j \leq n$ jeweils von derselben Größenordnung sind. Es gilt

$$r_{j,k} = O(h^{j-1}), \qquad k = 1, \ldots, j. \tag{5.23}$$

Nun können wir eine Abschätzung für (5.22) durch

$$\|z_i - \widetilde{z}_i\| \lesssim \sum_{j=1}^{i} \frac{|r_{j,n}|}{n|r_{j,n}|} tol = \frac{i}{n} tol \leq tol \tag{5.24}$$

angeben. Somit ist (5.14) durch die Wahl der Toleranz tol_i in (5.21) für alle $i = 1, \ldots, n$ erfüllt und mit (5.21) und (5.23) gilt

$$tol_i = \frac{tol}{n} O(h^{1-i}).$$

Sind nun die Vektoren $b_1, \ldots, b_n$ nicht gleichzeitig, sondern nur schrittweise bekannt, dann ist zu dem Zeitpunkt, an dem man die Approximation $\widetilde{w}_i$ an $w_i = f(A)q_i$ erstellen muss, das $r_{i,n}$ noch unbekannt. In (5.19) sieht man, dass zu diesem Zeitpunkt nur die Zahlen $r_{j,l}$ mit $1 \leq j, l \leq i$ bekannt sind. Deshalb wenden wir den Satz 5.2 an und sehen, dass $r_{i,n} \approx r_{i,i}$ ist. Da das $r_{i,i}$ bekannt ist, kann man anstatt (5.21) nun

$$tol_i \approx \frac{tol}{n|r_{i,i}|} \tag{5.25}$$

setzen und es gilt immer noch

$$tol_i = \frac{tol}{n} O(h^{1-i}). \tag{5.26}$$

Mit der Darstellung (5.25) von tol_i weiß man jetzt zu welcher Genauigkeit die Approximation $\widetilde{w}_i$ berechnet werden muss. Dass damit auch die Bedingung (5.14) erfüllt ist, zeigt

$$\|z_i - \widetilde{z}_i\| \leq \sum_{j=1}^{i} |r_{j,i}| tol_j \approx \sum_{j=1}^{i} \frac{|r_{j,i}|}{n|r_{j,j}|} tol \approx \sum_{j=1}^{i} \frac{|r_{j,j}|}{n|r_{j,j}|} tol = \frac{i}{n} tol \leq tol.$$

Wir haben gezeigt, wie man für dieses Verfahren die Toleranzen tol_i wählen muss.

Nun betrachten wir den wesentlichen Speicher- und Rechenaufwand für die Berechnung der Approximation an $f(A)b_i$ mit diesem Orthogonalprojektionsverfahren.

Speicheraufwand: Der wesentliche Speicheraufwand ist durch die Anzahl der Vektoren der Dimension N gegeben. Zuerst fügt man b_i in die bestehende QR Zerlegung ein. Hierfür muss man die Vektoren $q_1, \ldots, q_i$ speichern. Der Vektor b_i kann dann gelöscht werden. Anschließend berechnet man die Krylovapproximation an $w_i = f(A)q_i$ mit m_i Iterationen. Zur Erstellung dieser Approximation muss man dann m_i Krylovvektoren der Dimension N speichern, diese können aber danach wieder gelöscht werden. Dann speichert man w_i zusätzlich zu den schon vorhandenen $w_1, \ldots, w_{i-1}$. Insgesamt sind also $2i + m_i$ Vektoren der Dimension N im Speicher, von denen man m_i wieder entfernen kann.

Falls aber mit einem Krylovraum Approximationen an Produkte von unterschiedlichen Funktionen einer unterschiedlich skalierten Matrix A mit demselben Vektor b_i, also an $f_{j_1}(h_{j_2}A)b_i$ mit $j_1 = 1, \ldots, \vartheta_1$, $j_2 = 1, \ldots, \vartheta_2$ und $\vartheta = \vartheta_1\vartheta_2$, berechnet werden sollen, wie das zum Beispiel bei einem mehrstufigen exponentiellen Runge-Kutta Verfahren der Fall ist, so müssen pro q_i insgesamt ϑ Approximationen an das Produkt einer Matrixfunktion mit q_i gespeichert werden. Vor der Approximation von Produkten von Matrixfunktionen und dem Vektor b_i sind dann also schon $(i-1)\vartheta$ Approximationen im Speicher und es werden noch ϑ neue Approximationen hinzugefügt. Dann befinden sich im Speicher insgesamt mit dem Anteil der QR Zerlegung $i(\vartheta+1)+m_i$ Vektoren der Dimension N, von denen man m_i wieder entfernen kann.

Rechenaufwand: Der Hauptrechenaufwand ist die Orthogonalisierung des Vektors b_i gegenüber den Vektoren $q_1, \ldots, q_{i-1}$, die Erstellung des Krylovraumes $\mathcal{K}_{m_i}(A, q_i)$ und damit die Berechnung der Matrixfunktionen an H_{m_i}. Bei der Erstellung des Krylovraumes mit dem Lanczosverfahren ist der wesentliche Rechenaufwand in jedem Iterationsschritt die Matrixvektormultiplikation mit A oder im shift-and-invert Fall das Lösen des Gleichungssystems.

Wie wir gesehen haben, kann man erwarten, dass m_i immer kleiner wird, je größer das i wird. Somit wird das m_i mit größer werdendem i deutlich kleiner als die Iterationsanzahl des Krylovverfahrens, mit dem man die Approximation an $f(A)b_i$ ohne Informationen über vorherige Vektoren berechnet.

5.2.2 Restart

Die oben beschriebene Methode kann nicht angewendet werden, falls man nicht weiß, für wie viele Vektoren b_i man $y_i = f(A)b_i$ berechnen muss, denn dann ist das n, das man zur Berechnung der tol_i in (5.21) oder (5.25) benötigt, unbekannt. Außerdem gibt es Schwierigkeiten, falls die Anzahl der Vektoren n zu groß wird. Je größer das i wird, für das man $z_i = f(A)b_i$ berechnen will, um so mehr Vektoren $q_1, \ldots, q_{i-1}$ gibt es, gegen die man $b_i \in \mathbb{R}^N$ orthogonalisieren muss. Wenn die Dimension des Problems sehr groß ist, wie es in unseren Anwendungen der Fall ist, wird die Rechenzeit für die Erstellung der QR-Zerlegung so teuer, dass man insgesamt keine Einsparung der Rechenzeit mehr erkennen kann. Ein weiteres Problem ist, dass der Speicherbedarf, wie wir gesehen haben, mit jedem neuen Vektor weiter anwächst. Für großes N kann der

vorhandene Speicherplatz schnell verbraucht sein. Also ist die erste einfache Idee, um das zu vermeiden, die Einführung von Restarts nach s Schritten. Das heißt, falls die QR-Zerlegung aus s Vektoren besteht, löschen wir im nächsten Schritt die gesamte QR-Zerlegung und fangen an, eine neue QR-Zerlegung zu erstellen und berechnen mit dem Standard Krylovverfahren eine Approximation an das Produkt der Matrixfunktion mit dem neuen Vektor. Das Q der QR-Zerlegung besteht aus maximal s Vektoren und somit bleibt der Aufwand der Orthogonalisierung eines Vektors b_i gegen das Q begrenzt. Die geforderte Genauigkeit der Approximation $\widetilde{w}_i$ an $w_i = f(A)q_i$ ist

$$tol_i = \frac{tol}{s|r_{i,i}|},$$

da i immer nur Element von $\{1, \ldots, s\}$ ist. Ein weiterer Grund, einen Restart einzuführen, ist, dass das Verfahren instabil wird, sobald das $|r_{i,i}|$ klein wird (Größenordnung: $\sim 10^{-7}$). Dafür geben wir ein Beispiel an.

Beispiel 5.1. Sei $A \in \mathbb{R}^{N,N}$ mit $N = 10^5$ eine Diagonalmatrix mit äquidistanten Eigenwerten auf dem Intervall $[-400, 0]$, sei $b_1 = Au$ mit einem normierten Zufallsvektor $u \in \mathbb{R}^N$ und

$$b_i = A(b_{i-1} + hz_{i-1}), \qquad i = 2, \ldots, n$$

mit $z_i = e^A b_i$, $h = 1/n$ und $n = 20$. Dieses Beispiel ist so gewählt, da es in dieser Form dem exponentiellen Eulerverfahren ähnelt. Wir berechnen die Approximationen an $e^A b_i$ mit $i = 1, \ldots, n$ zur Genauigkeit 10^{-8} mit dem naiven Lanczosverfahren, das keine Projektion benutzt, mit dem Lanczosverfahren, das das Orthogonalprojektionsverfahren aus dem letzten Abschnitt 5.2.1 ohne Restart (o.r.) nutzt, und mit dem Orthogonalprojektionsverfahren inklusive Restart nach jeweils $s = 8$ Vektoren. In Abbildung 5.1 sind die exakten Fehler pro Kryloviteration aller 20 Produkte nacheinander für die verschiedenen Verfahren dargestellt. Die gepunktete Linie gibt die geforderte Genauigkeit an. $\diamond$

Man sieht, dass das Orthogonalprojektionsverfahren ohne Restart nach ungefähr der Hälfte der Approximationen instabil wird, während das Verfahren mit Restart stabil bleibt.

Der Nachteil dieser Vorgehensweise ist, dass durch die komplette Löschung der QR-Zerlegung alle Informationen über vorherige Vektoren and Produkte verloren gehen.

Ist

$$\widehat{i} := \begin{cases} s & , \text{ falls } mod(i, s) = 0, \\ mod(i, s) & , \text{ sonst,} \end{cases} \tag{5.27}$$

definiert, wobei $\mathrm{mod}(\cdot, s)$ die Modulofunktion bezüglich s ist, kann man ähnlich wie im letzten Abschnitt sagen, dass Folgendes für den Speicher- und Rechenaufwand zur Approximation von $f(A)b_i$ gilt.

Speicheraufwand: Zur Berechnung der Approximation an $f(A)b_i$ ist der Speicher mit $2\widehat{i} + m_i$ Vektoren der Dimension N gefüllt, von denen man m_i wieder entfernen kann. Maximal befinden sich im Speicher also $2s + m_i$ Vektoren.

Falls aber mit einem Krylovraum Approximationen an Produkte von unterschiedlichen Funktionen einer unterschiedlich skalierten Matrix A mit demselben Vektor

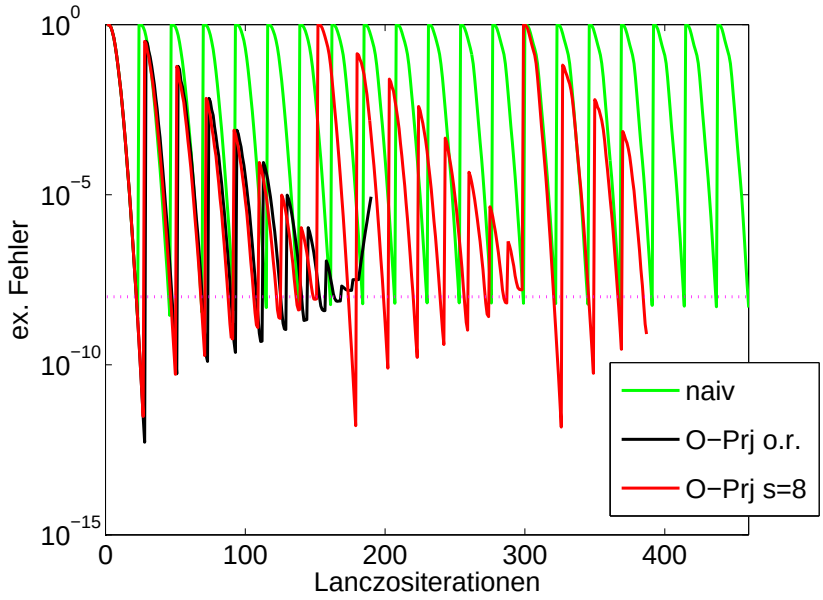

Abbildung 5.1: Fehler für die Lanczosapproximationen an $e^A b_i$ mit $i = 1, \ldots, 20$ aus dem Beispiel 5.1

b_i, also an $f_{j_1}(h_{j_2}A)b_i$ mit $j_1 = 1, \ldots, \vartheta_1$, $j_2 = 1, \ldots, \vartheta_2$ und $\vartheta = \vartheta_1\vartheta_2$, berechnet werden sollen, befinden sich im Speicher insgesamt $\widehat{i}(\vartheta + 1) + m_i$ Vektoren der Dimension N, von denen man m_i wieder entfernen kann. Maximal sind das also $s(\vartheta + 1) + m_i$ Vektoren.

Rechenaufwand: Der Hauptrechenaufwand ist die Orthogonalisierung des Vektors b_i gegenüber den Vektoren $q_1, \ldots, q_{\widehat{i}-1}$, die Erstellung des Krylovraumes $\mathcal{K}_{m_{\widehat{i}}}(A, q_{\widehat{i}})$ und damit die Berechnung der Matrixfunktionen an $H_{m_{\widehat{i}}}$. Bei der Erstellung des Krylovraumes ist der wesentliche Rechenaufwand in jedem Iterationsschritt die Matrix-Vektormultiplikation mit A oder im shift-and-invert Fall das Lösen des Gleichungssystems, das sind $m_{\widehat{i}}$ Matrix-Vektormultiplikationen oder Gleichungssysteme.

Das Orthogonalprojektionsverfahren mit Restarts ist im Algorithmus 1 nochmals kompakt dargestellt. Hierbei orthonormiert die Funktion $[q, r]$ =gram-schmidt(b, Q) den Vektor b gegen die Matrix Q mit einem Gram-Schmidt Verfahren. Der resultierende orthonormale Vektor ist q und r ist so gewählt, dass $b = [Q \quad q]r$ gilt. Die Funktion funAb$(A, q, fun, toli)$ berechnet die Approximation an das Produkt $fun(A)q$ zur Genauigkeit tol mit einem beliebigen Verfahren.

Algorithmus 1 Orthogonalprojektion mit Restarts

$z_i = f(A)b_i$, $i = 1, \ldots, n$ mit $\|z_i - \widetilde{z}_i\| \leq tol$

Eingabe: A, $b_1, \ldots, b_n$, fun, tol, s (z.B. $s = 12$)
Ausgabe: $\widetilde{z}_1, \ldots, \widetilde{z}_n$
Setze $j = 0$, $W = [\,]$
for $i = 1, \ldots, n$ **do**
  if $j = 0$ oder $j = s + 1$ **then**
    $r = \|b_i\|$
    $Q = b_i/r$, $R = r$
    $j = 1$
  else
    $[q, r]$ =gram-schmidt(b_i, Q)
    $Q = [Q \quad q]$, $R = \left[\begin{array}{c|c} R & \\ \hline 0 & r \end{array}\right]$
  end if
  $toli = tol/(s\ r(j))$
  w =funAb$(A, q, fun, toli)$
  $W = [W \quad w]$
  $z_i = Wr$
  $j = j + 1$
end for

5.2.3 Zirkulierende Orthogonalprojektion

In diesem Abschnitt werden wir eine Methode präsentieren, die immer eine QR-Zerlegung von maximal k Vektoren nutzt, aber weniger Informationen vorheriger Vektoren verliert als bei strikten Restarts aus dem letzten Anschnitt. Hierfür nutzen wir eine Up- und Downdating-Methode, die im Buch von Golub und van Loan [29, Seite 608] vorgestellt wird. Die Idee wird im Folgenden skizziert. Falls die QR-Zerlegung von $[b_1, \ldots, b_k]$ schon erstellt wurde und der nächste Vektor eingefügt werden soll, löschen wir den ersten Vektor b_1 aus dieser QR-Zerlegung und fügen dann erst den Vektor b_{k+1} der QR-Zerlegung hinzu. Dann erhalten wir die QR-Zerlegung von $[b_2, \ldots, b_{k+1}]$, bestehend aus k Vektoren. Diese Vorgehensweise wiederholen wir bis zum letzten Vektor. Nun beschreiben wir diese Methode detaillierter.

Wir setzen voraus, dass die QR-Zerlegung $[b_1, \ldots, b_k] = Q_k R_k$ und die Approximationen $\widetilde{w}_i$ an die Produkte $w_i = f(A)q_i$ für $i = 1, \ldots, k$ mit einer Genauigkeit

$$tol_i = \frac{tol}{k|r_{i,i}|} \quad \left(= \frac{tol}{k} O(h^{1-i})\right) \tag{5.28}$$

schon berechnet wurden. Also kennen wir die Approximationen $\widetilde{z}_i$ an $z_i = f(A)b_i$ und es gilt $\|z_i - \widetilde{z}_i\| \leq tol$. Nun wollen wir die Approximation an z_{k+1} berechnen. Dazu löschen

wir b_1 aus unserer QR-Zerlegung. Hierfür setzen wir

$$H := R_k \begin{bmatrix} 0 \\ I_{k-1} \end{bmatrix} = \begin{bmatrix} r_{1,2} & \dots & r_{1,k} \\ r_{2,2} & & \vdots \\ & \ddots & \vdots \\ 0 & & r_{k,k} \end{bmatrix}.$$

Dann berechnen wir die QR-Zerlegung $H = U\widehat{R}_{k-1}$ mit $U \in \mathbb{R}_{k-1}^{k,k-1}$ und $\widehat{R} \in \mathbb{R}^{k-1,k-1}$. Da H eine obere Hessenbergmatrix und $\widehat{R}_{k-1}$ eine obere Dreiecksmatrix ist, ist U ebenfalls eine obere Hessenbergmatrix. Setzen wir $\widehat{Q}_{k-1} := Q_k U$ mit $\widehat{Q}_{k-1} = [\widehat{q}_1, \dots, \widehat{q}_{k-1}]$, dann gilt

$$\widehat{Q}_{k-1}\widehat{R}_{k-1} = Q_k U \widehat{R}_{k-1} = Q_k H = [b_2, \dots, b_k].$$

Wir kennen also die QR-Zerlegung von $[b_2, \dots, b_k]$. Als nächstes orthogonalisieren wir b_{k+1} gegen $\widehat{Q}_{k-1}$ mit einem Gram-Schmidt Verfahren und erhalten $\widehat{q}_k$. Die Darstellung von b_{k+1} ist durch eine Linearkombination der $\widehat{q}_j$ mit $j = 1, \dots, k$ gegeben:

$$b_{k+1} = \sum_{j=1}^{k} \widehat{r}_{j,k}\widehat{q}_j.$$

Mit $\widehat{q}_k$ und den $\widehat{r}_{j,k}$ vervollständigen wir $\widehat{Q}_{k-1}$ und $\widehat{R}_{k-1}$ und erhalten so die QR-Zerlegung

$$[b_2, \dots, b_{k+1}] = \widehat{Q}_k \widehat{R}_k$$

mit $\widehat{Q}_k = [\widehat{Q}_{k-1}\ \widehat{q}_k]$ und $\widehat{R} = (\widehat{r}_{i,j})_{i,j=1}^{k}$. Nun wollen wir die Approximation $\widetilde{z}_{k+1}$ an

$$z_{k+1} = \sum_{j=1}^{k} f(A)\widehat{q}_j\widehat{r}_{j,k} = \sum_{j=1}^{k} v_j\widehat{r}_{j,k} \tag{5.29}$$

mit $v_j = f(A)\widehat{q}_j$ so berechnen, dass sie die Fehlerschranke

$$\|z_{k+1} - \widetilde{z}_{k+1}\| \leq tol \tag{5.30}$$

erfüllt. Sei $\widetilde{v}_j$ die Approximation an v_j. Für $j = 1, \dots, k-1$ muss man die Approximation $\widetilde{v}_j$ nicht neu berechnen, denn setzt man $W_k = [w_1, \dots, w_k]$ und $V_{k-1} = [v_1, \dots, v_{k-1}]$, dann ist

$$V_{k-1} = f(A)\widehat{Q}_{k-1} = f(A)Q_k U = W_k U.$$

Man berechnet eine neue Approximation $\widetilde{v}_k$ an $v_k = f(A)\widehat{q}_k$ mit der Eigenschaft

$$\|v_k - \widetilde{v}_k\| \leq \widehat{tol}_k.$$

Mit dieser erstellt man die Approximation an z_{k+1} durch eine Linearkombination wie in (5.29). Für die Approximationen an die folgenden Produkte $f(A)b_i$ mit $i = k+2, \dots$

geht man genauso vor, indem man wieder einen Vektor aus der QR-Zerlegung entfernt und einen neuen einfügt.

Die offenen Fragen sind, wie groß die Toleranz $\widehat{tol}_k$ gewählt werden muss, damit die Approximation an z_{k+1} die Fehlerschranke (5.30) erfüllt und ob dann auch für $\|v_i - \widetilde{v}_i\|$ mit $i = 1, \ldots, k$ eine zu (5.28) äquivalente Bedingung gilt. Wenn dies der Fall ist, kann man im nächsten Schritt genauso vorgehen. Die Antwort auf die Fragen geben wir im Folgenden.

Toleranzen

Der Fehler der Approximation $\widetilde{z}_{k+1}$ ist

$$\|z_{k+1} - \widetilde{z}_{k+1}\| = \|\sum_{j=1}^{k} \widehat{r}_{j,k}(v_j - \widetilde{v}_j)\|. \tag{5.31}$$

Es gilt

$$v_j = W_k u_j = \sum_{i=1}^{j+1} w_i u_{i,j} \tag{5.32}$$

für $j = 1, \ldots, k-1$, wobei u_j die j-te Spalte von U ist und $u_{i,j}$ der (i,j)-te Eintrag von U. Die Gleichung von (5.32) gilt wegen der oberen Hessenbergform von U, da dann $u_{i,j} = 0$ für $i = j+2, \ldots, k$ ist. Daher gilt für $j = 1, \ldots, k-1$

$$\|v_j - \widetilde{v}_j\| \leq \sum_{i=1}^{j+1} \|w_i - \widetilde{w}_i\||u_{i,j}| \leq \sum_{i=1}^{j+1} tol_i|u_{i,j}| = \sum_{i=1}^{j} tol_i|u_{i,j}| + tol_{j+1}|u_{j+1,j}|. \tag{5.33}$$

Wegen $\|u_j\| = 1$ gilt $|u_{j,i}| \leq 1$. Da U obere Hessenbergform und $\widehat{R}_k$ obere Dreiecksform hat und somit $u_{i+1,j} = 0$ für $j = 1, \ldots, i-1$ und $\widehat{r}_{j,i} = 0$ für $j = i+1, \ldots, k-1$ gilt, folgt aus $H = U\widehat{R}_{k-1}$

$$r_{i+1,i+1} = h_{i+1,i} = \sum_{j=1}^{k-1} u_{i+1,j}\widehat{r}_{j,i} = u_{i+1,i}\widehat{r}_{i,i}$$

für $i = 1, \ldots, k-1$. Wir erhalten mit (5.28) und (5.33)

$$\begin{aligned}
\|v_j - \widetilde{v}_j\| &\leq \sum_{i=1}^{j} tol_i + tol_{j+1}|u_{j+1,j}| \\
&\approx \sum_{i=1}^{j} \frac{tol}{k} O(h^{1-i}) + \frac{|r_{j+1,j+1}|tol}{k|r_{j+1,j+1}||\widehat{r}_{j,j}|} \\
&\lesssim \frac{tol}{k} O(h^{1-j}) + \frac{tol}{k|\widehat{r}_{j,j}|}.
\end{aligned} \tag{5.34}$$

Da $\widehat{R}_{k-1}$ zu der QR-Zerlegung von $[b_2, \ldots, b_k]$ gehört, können wir den Satz 5.2 anwenden und erhalten $\widehat{r}_{j,j} = O(h^{j-1})$ für $j = 1, \ldots, k-1$. Damit ist für $j = 1, \ldots, k-1$

$$\|v_j - \widetilde{v}_j\| \leq \frac{tol}{k} O(h^{1-j}). \tag{5.35}$$

Aus (5.31) kann man mit Hilfe von Satz 5.2

$$\begin{aligned}
\|z_{k+1} - \widetilde{z}_{k+1}\| &\leq \sum_{j=1}^{k-1} |\widehat{r}_{j,k}| O(h^{1-j}) + |\widehat{r}_{k,k}| \widehat{tol}_k \\
&\approx \sum_{j=1}^{k-1} \frac{tol}{k} + |\widehat{r}_{k,k}| \widehat{tol}_k \\
&= (k-1)tol + |\widehat{r}_{k,k}| \widehat{tol}_k
\end{aligned}$$

folgern. Um (5.30) zu erfüllen, setzen wir

$$\widehat{tol}_k = \frac{tol}{k|\widehat{r}_{k,k}|}.$$

Hieraus und aus (5.35) folgt

$$\|v_j - \widetilde{v}_j\| \leq \widehat{tol}_j, \qquad \text{für } j = 1, \ldots, k$$

mit

$$\widehat{tol}_j = \frac{tol}{k} O(h^{1-j}) = \frac{tol}{k|\widehat{r}_{j,j}|}.$$

Für die Berechnung der Approximation an das Produkt der Matrixfunktion mit dem nächsten Vektor ist damit also auch die Voraussetzung (5.28) gegeben.

Da dieses Verfahren nach einigen Schritten an Genauigkeit verliert, startet man es nach jeweils s Vektoren neu. Dabei ist es sinnvoll, das s so zu wählen, dass der Restart nach einem vollständigen Zyklen erfolgt, also s ein Vielfaches von k ist. Wir verdeutlichen die Instabilität mit einem Beispiel:

Beispiel 5.2. Die Matrix A und die Vektoren b_i mit $i = 1, \ldots, n$ seien die aus Beispiel 5.1. Hier berechnen wir die Approximationen an $e^A b_i$ mit $i = 1, \ldots, n$ zur Genauigkeit 10^{-8} mit dem Lanczosverfahren, das keine Informationen über vorherige Vektoren nutzt, mit dem zirkulierenden Orthogonalprojektionsverfahren mit $k = 3$ ohne Restart (o.r.) und schließlich mit demselben Verfahren nur diesmal mit Restart nach jeweils $s = 9$ Vektoren. In Abbildung 5.2 sind die exakten Fehler pro Kryloviteration aller 20 Produkte nacheinander für die verschiedenen Verfahren dargestellt. Die gepunktete Linie gibt die geforderte Genauigkeit an. $\diamond$

Nun betrachten wir den Speicher- und Rechenaufwand für die Approximation an $f(A)b_i$ mit Hilfe des k-zirkulierenden QR Raums mit Restart nach s Vektoren.

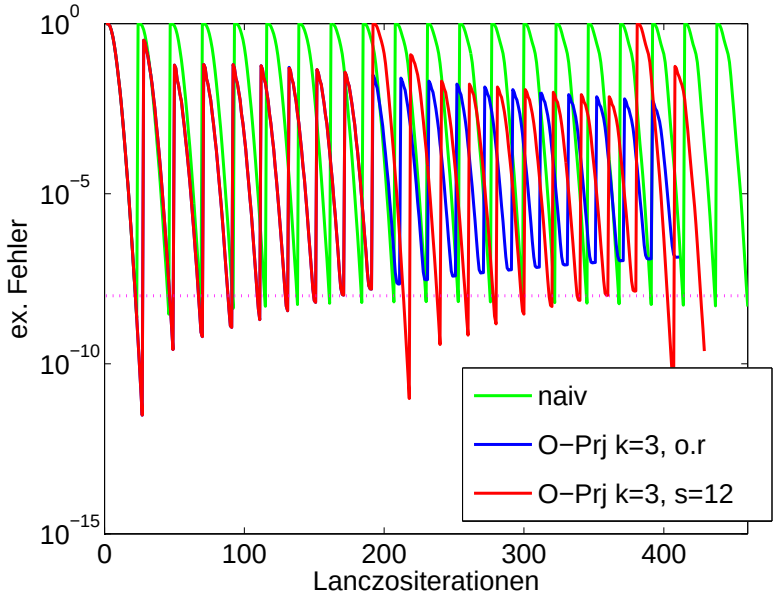

Abbildung 5.2: Fehler für die Lanczosapproximationen an $e^A b_i$ mit $i = 1, \ldots, 20$ aus dem Beispiel 5.2

Speicheraufwand: Da in diesem Verfahren maximal k Vektoren in der QR-Zerlegung sind, können wir mit denselben Argumenten wie bei der Betrachtung des Speicheraufwands im letzten Abschnitt 5.2.2 begründen, dass sich maximal $2k + m_i$ Vektoren im Speicher befinden.

Betrachtet man wieder die Approximationen an $f_{j_1}(h_{j_2} A)b_i$ mit $j_1 = 1, \ldots, \vartheta_1$, $j_2 = 1, \ldots, \vartheta_2$ und $\vartheta = \vartheta_1 \vartheta_2$, so muss man maximal $k(\vartheta + 1) + m_i$ Vektoren speichern.

Rechenaufwand: Falls $\widehat{i} \leq k$ ist, ist der Rechenaufwand derselbe wie im letzten Abschnitt. Ist $\widehat{i} > k$, besteht neben der Erstellung des Krylovraumes $\mathcal{K}_{m_{\widehat{i}}}(A, q_{\widehat{i}})$ und der Berechnung der Matrixfunktionen an $H_{m_{\widehat{i}}}$, der Hauptrechenaufwand in der Entfernung eines Vektors aus der QR-Zerlegung und dem anschließenden Einfügen von b_i in die QR-Zerlegung.

Einen kompakten Überblick des zirkulierenden Orthogonalprojektionsverfahrens mit Restarts gibt der Algorithmus 2 an. Neben den schon bekannten Funktionen *gramschmidt* und *funAb* erstellt die Funktion $[Q, R] = \text{qr}(X, 0)$ die reduzierte QR-Zerlegung von $X \in \mathbb{R}^{n,m}$ mit $Q \in \mathbb{R}^{n,m}$ und $R \in \mathbb{R}^{m,m}$.

5.2.4 Hinzunahme von Ritzvektoren

Eine weitere Variation dieses Verfahrens ist die zusätzliche Verwendung von Ritzvektoren der Matrix A in der QR Zerlegung. Dies ist motiviert dadurch, dass in parabolischen

Algorithmus 2 Zirkulierende Orthogonalprojektion mit Restarts

$z_i = f(A)b_i$, $i = 1, \ldots, n$ mit $\|z_i - \widetilde{z}_i\| \leq tol$

Eingabe: A, $b_1, \ldots, b_n$, fun, tol, k (z.B. $k = 4$), s (Vielfaches von k, z.B. $s = 3k$)
Ausgabe: $\widetilde{z}_1, \ldots, \widetilde{z}_n$
Setze $j = 0$, $W = [\,]$
for $i = 1, \ldots, n$ **do**
  if $j = 0$ oder $j = s + 1$ **then**
    $r = \|b_i\|$
    $Q = b_i/r$, $R = r$
    $j = 1$
  else
    if $j > k$ **then**
      $R = R \begin{bmatrix} 0 \\ I_{k-1} \end{bmatrix}$
      $[U, R] =$qr$(R, 0)$
      $Q = QU$
      $W = WU$
    end if
    $[q, r] =$gram-schmidt(b_i, Q)
    $Q = [Q \quad q]$, $R = \left[\begin{array}{c|c} R & \\ \hline 0 & r \end{array}\right]$
  end if
  $toli = tol/(k\ r(end))$
  $w =$funAb$(A, q, fun, toli)$
  $W = [W \quad w]$
  $z_i = Wr$
  $j = j + 1$
end for

Problemen die Lösung von $f(A)b$ von den größten Eigenwerten von A dominiert wird. Daher approximieren wir die, zu diesen Eigenwerten gehörenden, Eigenvektoren durch Ritzvektoren. Diese Ritzvektoren berechnen wir aus dem Krylovraum bezüglich A und b_1, den wir zur Berechnung der Approximation an $f(A)b_1$ erstellen, und fügen sie in die QR-Zerlegung ein. Die Vorgehensweise dieser Variante des Orthogonalprojektionsverfahrens wird im Folgenden beschrieben.

Wir nehmen an, dass man für $b_1, \ldots, b_n$ die Produkte $z_i = f(A)b_i,\ i = 1, \ldots, n$ mit der Genauigkeit *tol* approximieren möchte. Wir legen am Anfang fest, dass wir für dieses Verfahren l Ritzvektoren verwenden wollen. Dann berechnen wir die Approximation $\widetilde{z}_1$ an z_1 mit Hilfe eines Krylovverfahrens zur Genauigkeit

$$\|z_1 - \widetilde{z}_1\| = \frac{tol}{n+l}.$$

Wir teilen hier durch $(n + l)$ anstatt durch n, da wir durch die Ritzvektoren l neue Vektoren bekommen, die wir später in die QR-Zerlegung einfügen. Dann ist

$$\widetilde{z}_1 = V_m f(H_m)\|b_1\|e_1$$

mit der zugehörigen Krylovrelation

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \qquad v_1 = b_1/\|b_1\|, \qquad V_m^T V_m = I$$

gegeben. Wir berechnen l Eigenvektoren $\widehat{u}_1, \ldots, \widehat{u}_l$ von H_m. Für unsere Probleme betrachten wir immer die Eigenvektoren zu den l größten Eigenwerten. Setzen wir $\widehat{U}_l = [\widehat{u}_1, \ldots, \widehat{u}_l]$, dann bekommen wir mit

$$[u_1, \ldots, u_l] = U_l := V_m \widehat{U}_l$$

die Ritzvektoren u_i mit $i = 1, \ldots, l$ von A bezüglich H_m. Nun berechnen wir die Approximationen $\widetilde{x}_i$ an

$$x_i = f(A)u_i, \qquad i = 1, \ldots, l$$

zur Genauigkeit

$$\|x_i - \widetilde{x}_i\| \leq \frac{tol}{(l+1)(n+l)}$$

mit dem Krylovverfahren. Da die Ritzvektoren nicht exakt A invariant sind, kann man $x_i = f(A)u_i$ nicht direkt mit den Ritzwerten berechnen, aber wir können davon ausgehen, dass zur Erstellung der Approximationen $\widetilde{x}_i$ nur wenige Iterationen des Krylovverfahrens notwendig sind.

Jetzt erstellen wir die QR-Zerlegung

$$[u_1, \ldots, u_l, b_1] = Q_{l+1} R_{l+1}$$

mit $Q_{l+1} = [q_1, \ldots, q_{l+1}]$. Diese nutzen wir, um damit die Approximationen $\widetilde{z}_i$ an $z_i = f(A)b_i$ mit $i = 2, \ldots, n$ mit Hilfe des Orthogonalprojektionsverfahrens aus Abschnitt

5.2.1 zu berechnen. Hierfür fügen wir die Vektoren b_i nacheinander in die QR-Zerlegung mit Hilfe eines Gram-Schmidt Verfahrens ein, berechnen die Approximation $\widetilde{w}_{i+l}$ an

$$w_{i+l} = f(A)q_{i+l}, \qquad i = 2, \ldots, n$$

zur Genauigkeit

$$tol_{i+l} = \|w_{i+l} - \widetilde{w}_{i+l}\| \leq \frac{tol}{(l+n)|r_{i+l,i+l}|}$$

mit einem Krylovverfahren und erhalten dann die Approximation $\widetilde{z}_i$ als Linearkombination der Approximationen $\widetilde{w}_j$ an $w_j = f(A)q_j$ mit $j = 1, \ldots, i+l$. Für die Einträge $r_{k,j}$, $k, j = 1, \ldots, l+i$ von R_{l+i} aus der so berechneten QR-Zerlegung

$$[u_1, \ldots, u_l, b_1, \ldots, b_i] = Q_{l+i}R_{l+i},$$

die ungleich Null sind, gilt $|r_{k,j}| = O(1)$ mit $k = 1, \ldots, l$ und $j = 1, \ldots, l+i$, da die Ritzvektoren nicht als Funktionsauswertungen einer glatten Funktion angesehen werden können. Nach Satz 5.2 ist $|r_{k,j}| = O(h^{k-l-1})$ mit $k, j = l+1, \ldots, l+i$. Damit die gewünschte Bedingung

$$\|z_i - \widetilde{z}_i\| \leq tol, \qquad i = 2, \ldots, n$$

erfüllt ist, muss

$$tol_j = \|w_j - \widetilde{w}_j\| \leq \frac{tol}{n+l} \qquad \text{für} \quad j = 1, \ldots, l+1 \tag{5.36}$$

gelten. Denn mit Satz 5.2 ist dann

$$\begin{aligned}
\|z_i - \widetilde{z}_i\| &\leq \sum_{j=1}^{i+l} \|w_j - \widetilde{w}_j\||r_{j,i+l}| \\
&= \sum_{j=1}^{l+1} tol_j|r_{j,i+l}| + \sum_{j=l+2}^{l+i} tol_j|r_{j,i+l}| \\
&\lesssim (l+1)\frac{tol}{(n+l)} + \sum_{j=l+2}^{l+i} \frac{tol}{(n+l)|r_{j,j}|}|r_{j,i+l}| \\
&\lesssim (l+1)\frac{tol}{(n+l)} + (i-1)\frac{tol}{(n+l)} \\
&\leq tol.
\end{aligned}$$

Wir zeigen, dass (5.36) gilt: Für $j = 1, \ldots, l$ mit $R_j = \{r_{k,i}\}_{i,k=1}^{j}$ gilt

$$\begin{aligned}
\|w_j - \widetilde{w}_j\| &= \|[v_1, \ldots, v_j]\{R_j^{-1}\}_{:,j} - [\widetilde{v}_1, \ldots, \widetilde{v}_j]\{R_j^{-1}\}_{:,j}\| \\
&\leq \sum_{k=1}^{j} \|v_k - \widetilde{v}_k\||\{R_j^{-1}\}_{k,j}| \\
&\lesssim j\frac{tol}{(l+1)(n+l)}O(1) \\
&\lesssim \frac{tol}{n+l}
\end{aligned}$$

und für $j = l + 1$

$$\begin{aligned}
\|w_{l+1} - \widetilde{w}_{l+1}\| &\leq \|[v_1, \ldots, v_l, z_1]\{R_j^{-1}\}_{:,l+1} - [\widetilde{v}_1, \ldots, \widetilde{v}_j, \widetilde{z}_1]\{R_j^{-1}\}_{:,l+1}\| \\
&\leq \sum_{k=1}^{l} \|v_k - \widetilde{v}_k\| |\{R_j^{-1}\}_{k,l+1}| + \|z_1 - \widetilde{z}_1\| |\{R_j^{-1}\}_{l+1,l+1}| \\
&\lesssim l \frac{tol}{(l+1)(l+n)} O(1) + \frac{tol}{(l+n)|r_{l+1,l+1}|} \\
&\approx \frac{tol}{l+n}.
\end{aligned}$$

Damit haben wir ein Orthogonalprojektionsverfahren mit der Verwendung von Ritzvektoren zur Approximation von $f(A)b_1, \ldots, f(A)b_n$ vorgestellt.

Natürlich kann man diese Methode auch neustarten. Dafür löschen wir nicht wie in Abschnitt 5.2.2 die ganze QR-Zerlegung nach s Schritten, sondern nur den Teil der Zerlegung, der zu den Vektoren b_i gehört. Den ersten Teil Q_l und R_l der QR-Zerlegung, der zu den Ritzvektoren $u_1, \ldots, u_l$ gehört, behalten wir.

Ähnlich gehen wir in dem Fall des zirkulierenden QR Raumes vor. Wir entfernen anstatt des ersten Vektors q_1 den Vektor q_{l+1} aus der QR-Zerlegung. Ansonsten gehen wir genauso vor, wie im Abschnitt 5.2.3 über den zirkulierenden QR Raum beschrieben.

Die Hinzunahme von Ritzvektoren lohnt sich nicht, wenn das zu Grunde liegende Krylovverfahren das Shift-and-invert Lanczosverfahren ist, denn bei diesem Verfahren werden die Eigenvektoren ohnehin schon schnell approximiert.

Wir betrachten den wesentlichen Speicher- und Rechenaufwand für das zirkulierende Orthogonalprojektionsverfahren mit l Ritzvektoren und Restart.

Speicheraufwand: Zusätzlich zu dem wesentlichen Speicheraufwand für das Verfahren aus dem letzten Abschnitt 5.2.3, müssen hier noch die Vektoren gespeichert werden, die durch die Ritzvektoren hinzukommen. Das sind l Vektoren für die QR-Zerlegung und die Approximationen an die Produkte der Matrixfunktion mit den l Ritzvektoren. Also befinden sich im Speicher maximal $2(k+l)+m_i$ Vektoren und wenn man Produkte von ϑ verschiedenen Funktionen und b_i mit demselben Krylovraum berechnet, sind maximal $(k+l)(\vartheta+1)+m_i$ Vektoren im Speicher.

Rechenaufwand: Der wesentliche Rechenaufwand ist die Erstellung von $l+k$ Approximationen mit Krylovverfahren, die pro Approximation m_j mit $j = 1, \ldots, l+k$ Iterationen gebrauchen, und der Ausbau der QR-Zerlegung. Unsere Theorie zeigt, dass das m_j mit größer werdendem j immer kleiner wird und man somit Rechenzeit spart.

Der Algorithmus 3 gibt das Orthogonalprojektionsverfahren inklusive aller Varianten an. Die Funktion *funAb* gibt bei dem Aufruf $[w, H, V]$ =funAb$(A, q, fun, toli)$ neben der Approximation w an $fun(A)q$ im Krylovraum auch die Hessenbergmatrix H und die Krylovvektoren, gespeichert in V, des Krylovverfahrens an. Die Funktion eig(X, l) berechnet die l größten Eigenwerte der Matrix X.

Algorithmus 3 Zirkulierende Orthogonalprojektion mit Restarts und Ritzvektoren

$z_i = f(A)b_i$, $i = 1, \ldots, n$ mit $\|z_i - \widetilde{z}_i\| \leq tol$

Eingabe: A, $b_1, \ldots, b_n$, fun, tol, k, s, l (Anzahl der Ritzvektoren)
Ausgabe: $\widetilde{z}_1, \ldots, \widetilde{z}_n$
Setze $j = 0$, $W = [\,]$
for $i = 1, \ldots, n$ **do**
  if $j = 0$ oder $j = s + 1$ **then**
    $r = \|b_i\|$, $Q = b_i/r$, $R = r$
    $j = 1$
  else
    if $j > k$ **then**
      $R = R \begin{bmatrix} 0 \\ I_{k-1} \end{bmatrix}$
      $[U, R] = \mathrm{qr}(R, 0)$, $Q = QU$, $W = WU$
    end if
    $[q, r] = \text{gram-schmidt}(b_i, Q)$
    $Q = [Q \quad q]$, $R = \left[\begin{array}{c|c} R & \\ \hline 0 & r \end{array}\right]$
  end if
  $toli = tol/((k + l)\ r(end))$
  if $i = 1$ **then**
    $[w, H, V] = \mathrm{funAb}(A, q, fun, toli)$
    $W = [W\ w]$
    $z_1 = Wr$
    $U = V\ \mathrm{eig}(H, l)$
    $X = [\,]$
    $toli = tol/((l + 1)(k + l))$
    for $m = 1, \ldots, l$ **do**
      $X = [X \ \ \mathrm{funAb}(A, U(:, m), toli)]$
    end for
    $[Q, R] = \mathrm{qr}([U\ b_i], 0)$
    $W = [X\ z_1]R^{-1}$
    $X = [\,]$
  else
    $W = [W \ \ \mathrm{funAb}(A, q, fun, toli)]$
    $z_i = Wr$
  end if
  $j = j + 1$
end for

5.2.5 Reorthogonalisierung der Gram-Schmidt QR-Zerlegung

Für unsere Orthogonalprojektionsmethoden müssen wir eine QR-Zerlegung durch Orthogonalisierung einiger Vektoren $[v_1, \ldots, v_n] = Q_n R_n$ erstellen. Aber in der Praxis verliert man sehr schnell die Orthogonalität der Matrix Q_n, falls das klassische oder das modifizierte Gram-Schmidt Verfahren benutzt wird. Deshalb verwenden wir die Methode zur Reorthogonalisierung der Gram-Schmidt QR-Zerlegung von J.W. Daniel, W.B. Gragg, L. Kaufman und G.W. Stewart [14]. Der Algorithmus 4 stellt dieses Verfahren für die Vektoren $v_1, \ldots, v_n$ dar.

Algorithmus 4 Reorthogonalisierte Gram-Schmidt QR-Zerlegung

Wähle $0 \ll \eta < 1$ (z.B. $\eta = 1/\sqrt{2}$)
$R_1 = \|v_1\|$, $Q_1 = v_1/R_1$
for $i = 2, \ldots, n$ **do**
  $r_i^0 = 0, \quad v^0 = v_i,$
  for $k = 1, 2, 3, \ldots$ bis $\|v^k\| > \eta \|v^{k-1}\|$ **do**
    $s^k = Q_{i-1}^T v^{k-1},$
    $r_i^k = r_i^{k-1} + s^k,$
    $u^k = Q_{i-1} s^k,$
    $v^k = v^{k-1} - u^k,$
  end for
  $r_i = r_i^k, \;\; \rho_i = \|v^k\|, \;\; q_i = \frac{v^k}{\rho_i}$
  $Q_i = [q_1, \ldots, q_i]$
  $R_i = \begin{bmatrix} R_{i-1} & r_i \\ 0 & \rho_i \end{bmatrix}$
end for

Kapitel 6

Numerische Beispiele

In diesem Kapitel betrachten wir numerische Beispiele für die neu vorgestellten Projektionsverfahren zur effizienten Approximation einer Matrixfunktion, angewandt auf mehrere Vektoren. Dazu betrachten wir Differentialgleichungen, die wir mit exponentiellen Integratoren lösen. Innerhalb dieser exponentiellen Integratoren verwenden wir die Projektionsverfahren. Wir testen, wie sinnvoll die Hinzunahme von Ritzvektoren zu der Orthogonalprojektion ist, vergleichen die Projektionsverfahren mit einem naiven Verfahren und prüfen, ob die exponentiellen Integratoren mit Hilfe der Projektionsverfahren konkurrenzfähig zu anderen Integratoren sind.

Alle Beispiele sind mit der 7.0.1 Version von *MATLAB* auf einer *AMD Athlon(tm) 64 Processor 3200+* Maschine mit 2.2 GHz und 2 Gigabyte Hauptspeicher gerechnet worden.

6.1 Testprobleme

Im Folgenden geben wir die Differentialgleichungen an, die wir in den weiteren Abschnitten als Testprobleme nutzen.

Die Probleme, mit denen wir die meisten Testdurchläufe für die neu vorgestellten Verfahren durchführen, sind parabolische, semilineare Differentialgleichungen der Form

$$\frac{\partial u(x,t)}{\partial t} = \Delta u(x,t) + g(t,u(x,t)), \qquad u(x,t_0) = u_0, \qquad x \in [0,\ 1]^d \tag{6.1}$$

mit $t \in [0,1]$ und

$$g(t,u(x,t)) = \frac{1}{1+u(x,t)} + \Phi(x,t).$$

Die Differentialgleichung hat homogene Dirichlet Randbedingungen. Das Φ aus der Differentialgleichung ist hierbei so gewählt, dass die exakte Lösung der verschiedenen Probleme durch

$$u(x,t) = e^t \prod_{i=1}^{d} x_i(1-x_i)$$

gegeben ist, wobei das d die jeweilige Raumdimension des Problems ist.

Wir diskretisieren die Differentialgleichungen im Raum durch finite Differenzen auf einem Gitter mit N inneren Gitterpunkten in jeder Raumdimension. Der Laplace Operator Δu wird dann durch eine Matrixoperation Au ersetzt, wobei A die Poissonmatrix der Dimension $n = N^d$ ist. Diese Matrix ist symmetrisch, dünn besetzt und je nach Diskretisierung sehr groß.

Wir betrachten für dieses Problem die Beispiele:

Beispiel 6.1. Das eindimensionale, semilineare Beispiel erhalten wir durch die Diskretisierung mit finiten Differenzen der Differentialgleichung (6.1) mit $d = 1$. Die Dimension der Diskretisierungsmatrix ist gleich der Anzahl N der Gitterpunkte. Wir wählen N aus $[100,\ 10000]$. $\diamond$

Beispiel 6.2. Das zweidimensionale, semilineare Beispiel erhalten wir durch die Diskretisierung mit finiten Differenzen der Differentialgleichung (6.1) mit $d = 2$. Damit ist die Dimension der Diskretisierungsmatrix N^2. Wir wählen N aus $[100,\ 500]$. $\diamond$

Beispiel 6.3. Das dreidimensionale, semilineare Beispiel erhalten wir durch die Diskretisierung mit finiten Differenzen der Differentialgleichung (6.1) mit $d = 3$. Damit ist die Dimension der Diskretisierungsmatrix N^3. Wir wählen N aus $[10,\ 60]$. $\diamond$

Ein weiteres Testproblem beruht auf der Differentialgleichung

$$M\frac{\partial u(x,t)}{\partial t} = Au(x,t) + g(t,u(x,t)), \qquad u_0 = u(x,t_0) \tag{6.2}$$

mit $g(t,u) = 2e^t y$, wobei y ein Vektor gleicher Dimension wie u ist und dessen Einträge alle eins sind. Diese Differentialgleichung stellt ein mit N inneren Gitterpunkten diskretisiertes Wärmeleitungsproblem auf dem Einheitskreis dar. Die Diskretisierung mit finiten Elementen erzeugt die symmetrischen, dünnbesetzten Matrizen M und A der Dimension N.

Beispiel 6.4. Auf unterschiedliche Gittern finiter Elemente betrachten wir die Differentialgleichung 6.2. $\diamond$

In Abbildung 6.1 geben wir die Zerlegung des Einheitskreises mit finiten Elementen aus Beispiel 6.4 links mit $N = 41$ und rechts mit $N = 614$ an.

Um die Differentialgleichung (6.2) in eine, für uns anwendbaren, Form zu bringen, berechnet man die Cholesky Zerlegung von M und erhält $M = L^T L$. Dann ist (6.2) äquivalent zu

$$\frac{\partial z(x,t)}{\partial t} = L^{-T}AL^{-1}z(x,t) + L^{-T}g(t,L^{-1}z(x,t)), \qquad z_0 = Lu(x,t_0)$$

mit $z(x,t) = Lu(x,t)$.

Als letztes Beispiel betrachten wir ein nicht symmetrisches Problem. Die Gleichung

$$\frac{\partial u(x,t)}{\partial t} = \Lambda(u(x,t)) + g(t,u(x,t)), \qquad u(x,t_0) = u_0 \tag{6.3}$$

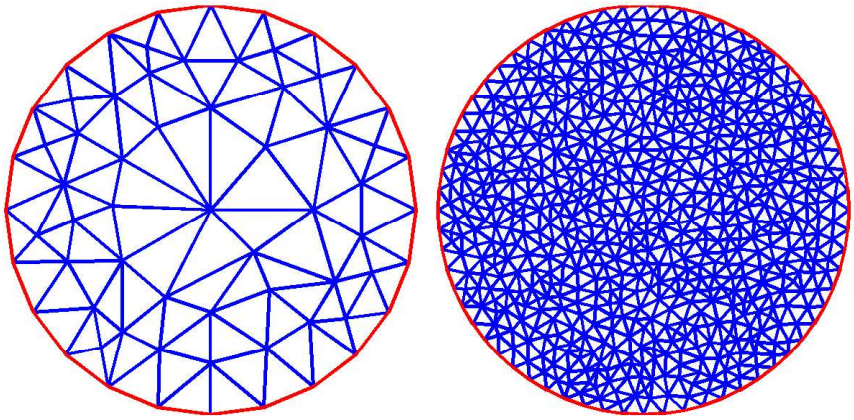

Abbildung 6.1: Zerlegung des Einheitskreises mit finiten Elementen aus Beispiel 6.4

mit $x \in [0,1]^2$, $t \in [0,1]$,

$$g(t, u(x,t)) = \gamma u(u - 1/2)(1 - u),$$

und

$$\Lambda(u(x,t)) = \epsilon \Delta u(x,t) - \alpha \ \mathrm{div}(u(x,t)).$$

beschreibt eine Differentialgleichung mit homogenen Neumann Randbedingungen. Wir erhalten die Matrixoperation $Au(x,t)$ durch eine Diskretisierung des Operators $\Lambda(u(x,t))$ durch finite Differenzen mit N Gitterpunkten inklusive der Randpunkte in jeder Raumdimension. Die Matrix A hat die Dimension $N^2 \times N^2$ und ist nicht symmetrisch.

Beispiel 6.5. Dieses Beispiel ist die Differentialgleichung (6.3) mit $\epsilon = 100$, $\alpha = -1$ und $\gamma = 100$ und erhalten so ein steifes Problem, da die für die Norm von A dann $\|A\| \approx 800N^2$ gilt. $\diamond$

6.2 Orthogonalprojektion mit Ritzvektoren

Wir wollen zunächst untersuchen, wie groß der Nutzen bei der in Abschnitt 5.2.4 beschriebenen Hinzunahme von Ritzvektoren zu dem Orthogonalprojektionsverfahren ist. Dazu lösen wir die Beispiele 6.2 und 6.3 mit dem exponentiellen Eulerverfahren. Hierbei approximieren wir die vorkommenden Produkte von Matrixfunktionen und Vektoren zur Genauigkeit 10^{-6} mit der mit $k = 4$ Vektoren zirkulierenden Orthogonalprojektionsmethode inklusive Restart nach $s = 12$ Vektoren und l Ritzvektoren. Als Krylovverfahren benutzen wir das Standard Lanczosverfahren. Wir betrachten dies nun für $l = 0, 1, 2, 3$, für verschiedene Zeitschrittweiten h des exponentiellen Integrators und für verschieden viele Diskretisierungsgitterpunkte N.

Die Abbildungen 6.2 und 6.3 stellen die Ergebnisse dieser Tests dar. Das linke Bild zeigt jeweils die Rechenzeiten, die das exponentielle Eulerverfahren mit dem Orthogonalprojektionsverfahrens inklusive l Ritzvektoren und einer Schrittweite h gebraucht.

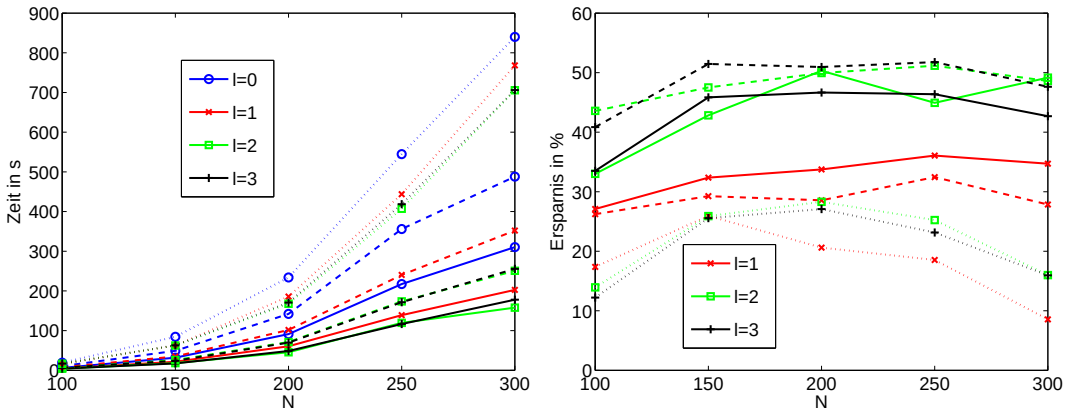

Abbildung 6.2: Rechenzeiten (links) und Verhältnis dieser (rechts) durch die Hinzunahme von Ritzvektoren bei der Orthogonalprojektionsmethode mit dem Lanczosverfahren für den exponentiellen Euler zur Lösung des zweidimensionalen Beispiels 6.2

Hierbei gibt die Farbe der Linien das l an und die Art der Linien das h, das heißt, eine durchgezogene Linie steht für $h = 0.1$, eine gestrichelte für $h = 0.05$ und eine gepunktete für $h = 0.01$. Die jeweilige rechte Graphik stellt das Verhältnis der Rechenzeit, die man mit Hinzunahme von Ritzvektoren gebraucht, im Vergleich zu der Rechenzeit dar, die man ohne Verwendung von Ritzvektoren benötigt. Die gerade, gelbe, durchgezogene Linie liegt bei Null Prozent, das heißt, wenn Punkte unter dieser Linien liegen, dann erhöht die Hinzunahme von Ritzvektoren die Rechenzeit.

Das Erste, was einem bei der Betrachtung der Beispiele auffällt, ist, dass sich bei beiden Fällen die Hinzunahme von Ritzvektoren lohnt. Für das Beispiel 6.2 spart man sowohl für $h = 0.1$ als auch für $h = 0.05$ bei der Verwendung von zwei oder drei Ritzvektoren 40 bis 50 Prozent Rechenzeit gegenüber keiner Verwendung von Ritzvektoren. Benutzt man einen Ritzvektor so spart man in diesen Fällen nur ungefähr 30 Prozent. Ist die Zeitschrittweite allerdings $h = 0.01$, so spart man mit jeder getesteten Anzahl von Ritzvektoren nur 20 bis 30 Prozent. Für das dreidimensionale Beispiel 6.3 sehen die Ergebnisse für $N \geq 30$ ähnlich aus, allerdings spart man in allen Fällen prozentual weniger Rechenzeit als für das zweidimensionale Beispiel 6.2. Aber auch in diesem Beispiel sieht man, dass die Ersparnis durch die Verwendung eines Ritzvektors geringer ist als wenn man zwei oder drei Ritzvektoren benutzt und dass sich bei $h = 0.01$ die Hinzunahme von Ritzvektoren weniger lohnt.

Verwendet man anstatt des exponentiellen Eulerverfahrens das Krogstadverfahren, dann erhält man analog zu den Abbildungen 6.2 - 6.3 die Abbildungen 6.4 - 6.5. Hierbei spart man bei der Verwendung von 3 Ritzvektoren jeweils am Meisten: für das zweidimensionale Beispiel 6.2 ungefähr 50 und für das dreidimensionale Beispiel 6.2 circa 30 Prozent Rechenzeit mit den Zeitschrittweiten $h = 0.1$ und $h = 0.05$. Mit zwei Ritzvektoren spart man noch im zweidimensionalen 30 bis 40 und im dreidimensionalen 10 bis 20 Prozent. Für beide Beispiele spart man mit der Hinzunahme von nur einem Ritzvektor

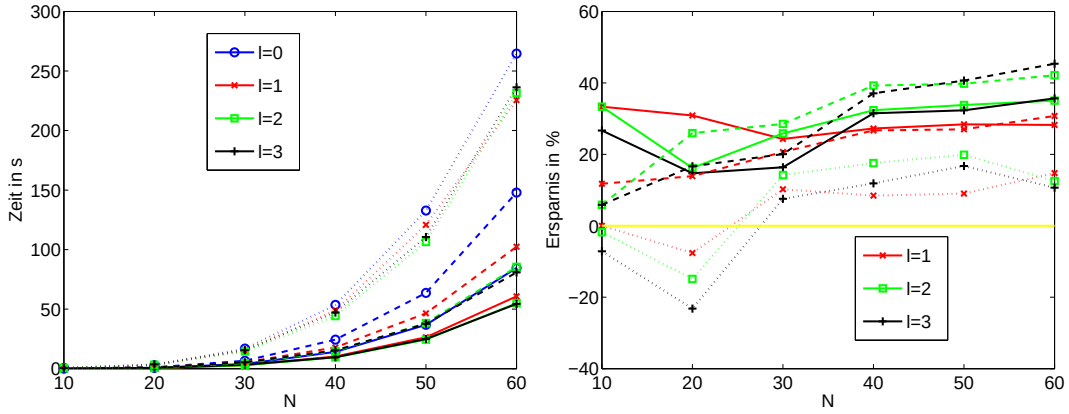

Abbildung 6.3: Rechenzeiten (links) und Verhältnis dieser (rechts) durch die Hinzunahme von Ritzvektoren bei der Orthogonalprojektionsmethode mit dem Lanczosverfahren für den exponentiellen Euler zur Lösung des dreidimensionalen Beispiels 6.3

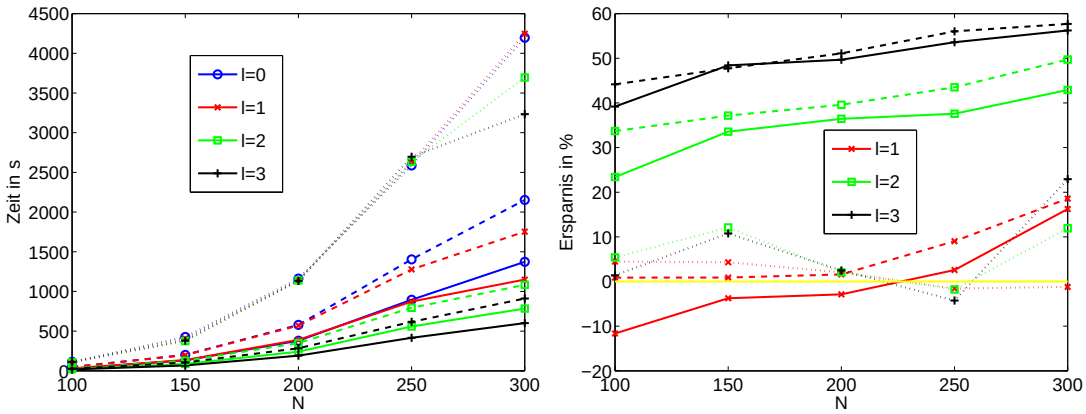

Abbildung 6.4: Rechenzeiten (links) und Verhältnis dieser (rechts) durch die Hinzunahme von Ritzvektoren bei der Orthogonalprojektionsmethode mit dem Lanczosverfahren für das Krogstadverfahren zur Lösung des zweidimensionalen Beispiels 6.2

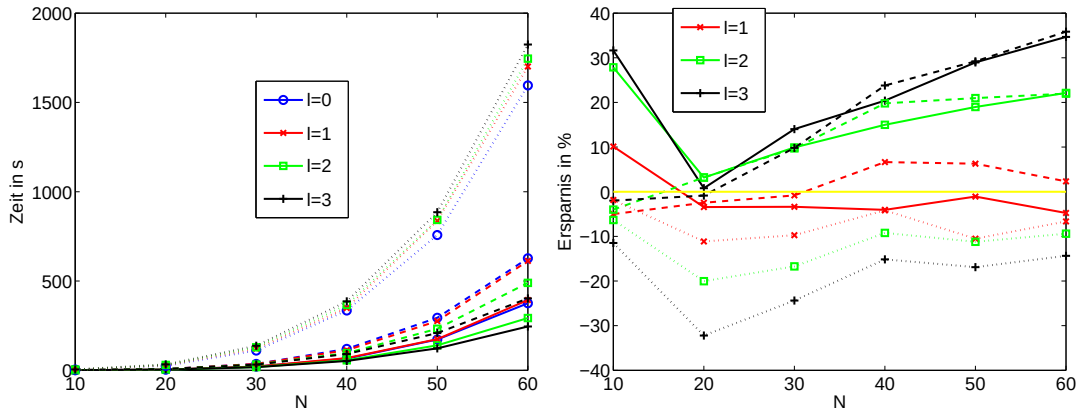

Abbildung 6.5: Rechenzeiten (links) und Verhältnis dieser (rechts) durch die Hinzunahme von Ritzvektoren bei der Orthogonalprojektionsmethode mit dem Lanczosverfahren für das Krogstadverfahren zur Lösung des dreidimensionalen Beispiels 6.3

keine Rechenzeit mehr. Auch bei der Zeitschrittweite $h = 0.01$ bewirkt die Hinzunahme von Ritzvektoren eher eine Verlangsamung des Verfahrens.

Wir haben also festgestellt, dass sich die Hinzunahme von Ritzvektoren für eine Zeitschrittweite $h = 0.01$ nicht lohnt. Dies lässt sich dadurch begründen, dass durch die kleine Zeitschrittweite nach Satz 5.2 die Einträge von R der QR-Zerlegung sehr klein sind. Daher braucht man schon nach wenigen Zeitschritten fast keine Iterationen mehr zur Berechnung der Produkte $f(A)b_i$. Die Hinzunahme von Ritzvektoren in diese QR-Zerlegung bringt dann keine Verbesserung sondern eher eine Störung.

Abschließend kann man zusammenfassen, dass es sinnvoll ist, Ritzvektoren in das Orthogonalprojektionsverfahren einzufügen, wenn die Vektoren, für die wir die Produkte mit der Matrixfunktion berechnen wollen, nicht zu sehr voneinander abhängen. Falls man Ritzvektoren verwendet, ist es besser, zwei bis drei zu verwenden als nur einen.

6.3 Vergleich von Orthogonal- und Lanczos-Galerkin Projektionsverfahren

In diesem Abschnitt lösen wir die Testprobleme mit exponentiellen Integratoren mit konstanter Zeitschrittweite h, um die Effektivität von Orthogonal- und Lanczos-Galerkin Projektionsverfahren zu untersuchen. Dabei unterscheiden wir folgende Verfahren:

Naiv: Die naive Implementierung des exponentiellen Integrators, welches ein Lanczosverfahren verwendet, das keine Informationen über vorherige Vektoren nutzt.

LG-Prj: Der exponentielle Integrator verwendet das Lanczos-Galerkin Projektionsverfahren aus Kapitel 4. Hierbei nutzen wir die Strategie, die am Ende des Abschnitts 4.4 vorgestellt wird und bei der man jeden neuen Vektor nur auf den

Krylovraum des ersten Vektors mit der Matrix A projiziert. Wir starten dieses Verfahren neu, wenn $\|(I - V_{m_1}^{(1)} V_{m_1}^{(1)^T}) b_i\| > 0.1$ oder wenn $m_i > m_1$ wird.

O-Prj: Der exponentielle Integrator verwendet das orthogonalprojizierte Lanczosverfahren aus Kapitel 5. Wir starten hier nach $s = 12$ Vektoren neu, verwenden den zirkulierenden QR Raum mit $k = 4$ und nutzen l Ritzvektoren.

Mit allen vorgestellten Lanczosverfahren berechnen wir Approximationen zu der Genauigkeit von 10^{-6} an die jeweiligen Matrixfunktionen, wobei wir für jedes Verfahren den Fehlerschätzer (2.6) nutzen.

Wir vergleichen die verschiedenen Verfahren *Naiv*, *LG-Prj* und *O-Prj* zur Lösung der unterschiedlichen Testprobleme. Hierbei betrachten wir jeweils die Gesamtanzahl der, für die Berechnung der Produkte von Matrixfunktionen und Vektoren benötigten, Lanczositerationen, die benötigte Rechenzeit und den Speicheraufwand des Lanczosverfahrens. Außerdem untersuchen wir das prozentuale Verhältnis der Rechenzeiten der beiden unterschiedlichen Projektionsverfahren im Vergleich zum naiven Verfahren.

6.3.1 Zeitaufwand

Zunächst verwenden wir die Verfahren *Naiv*, *LG-Prj* und *O-Prj* innerhalb des exponentiellen Euler Verfahrens zur Lösung der Beispiele 6.2 und 6.3 und vergleichen die Rechenzeiten. Wir benutzen jeweils für das orthogonalprojizierte Lanczosverfahren soviele Ritzvektoren l, wie das schnellste Verfahren aus dem letzten Abschnitt zu den gleichen Beispielen. In allen Verfahren ist das zu Grunde liegende Lanczosverfahren das Standard Lanczosverfahren. Für die zwei- und dreidimensionalen Beispiele 6.2 und 6.3 stellen wir in den Abbildungen 6.6 - 6.7 links die Rechenzeiten der jeweiligen Verfahren und rechts das prozentuale Verhältnis der Rechenzeiten gegenüber *Naiv* dar. Wir betrachten dies auf unterschiedlichen Diskretisierungsgittern mit N Gitterpunkten und mit unterschiedlichen Zeitschrittweiten h. Eine durchgezogene Linie entspricht $h = 0.1$, eine gestrichelte $h = 0.05$ und eine gepunktete $h = 0.01$. Man erkennt gut, dass man mit den Projektionsverfahren erheblich an Zeitaufwand sparen kann. Für das zweidimensionale Beispiel 6.2 spart man mit dem Lanczos-Galerkin Projektionsverfahren mit alle Zeitschrittweiten fast immer über 90 Prozent Rechenzeit gegenüber dem naiven Verfahren, mit dem Orthogonalprojektionsverfahren über 80 Prozent. Für das dreidimensionale Beispiel 6.3 spart man zwar nicht mehr ganz so viel, aber immer noch über 80 Prozent mit dem Lanczos-Galerkin Projektionsverfahren und 70 Prozent mit dem Orthogonalprojektionsverfahren für $N \geq 30$. Betrachtet man *Naiv*, *LG-Prj* und *O-Prj* mit dem Krogstadverfahren, so erhalten wir, wie man in Abbildung 6.8 und 6.9 sieht, fast die gleichen Ergebnisse wie bei Verwendung des exponentiellen Eulerverfahrens.

Verwendet man als Grundlage für die verschiedenen Verfahren das Shift-and-invert Lanczosverfahren anstatt des Standard Lanczosverfahrens, so spart man mit den Projektionsverfahren immer noch einiges an Rechenzeit gegenüber dem naiven Verfahren. Wir betrachten nur das zweidimensionale Beispiel 6.2, da sich für das dreidimensionale die Verwendung des Shift-and-invert Lanczosverfahrens nicht lohnt. Abbildung 6.10 zeigt die Ergebnisse bezüglich des exponentiellen Eulerverfahrens und Abbildung 6.11

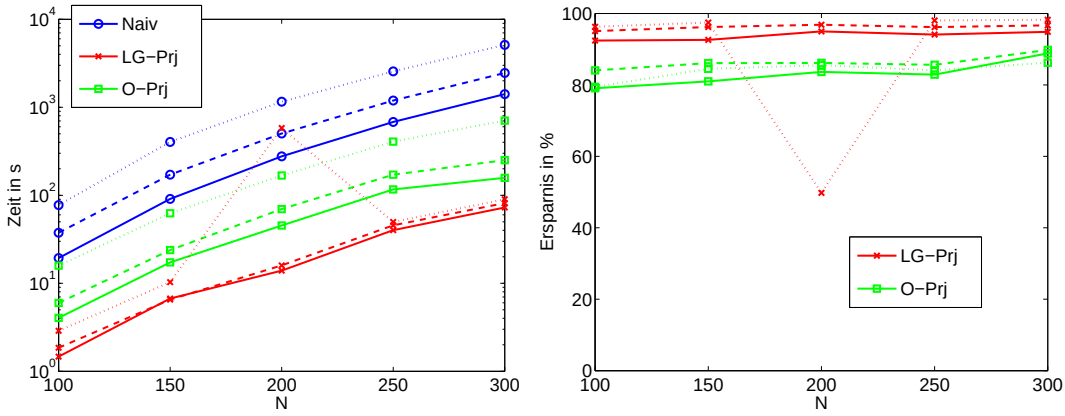

Abbildung 6.6: Rechenzeiten (links) und prozentuales Verhältnis (rechts) der verschiedenen Varianten des exponentiellen Euler Verfahrens zur Lösung des zweidimensionalen Beispiels 6.2

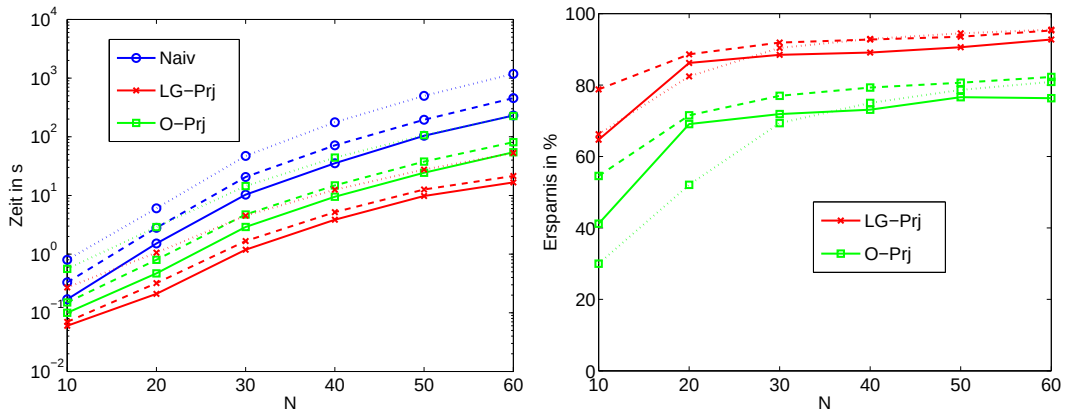

Abbildung 6.7: Rechenzeiten (links) und prozentuales Verhältnis (rechts) der verschiedenen Varianten des exponentiellen Euler Verfahrens zur Lösung des dreidimensionalen Beispiels 6.3

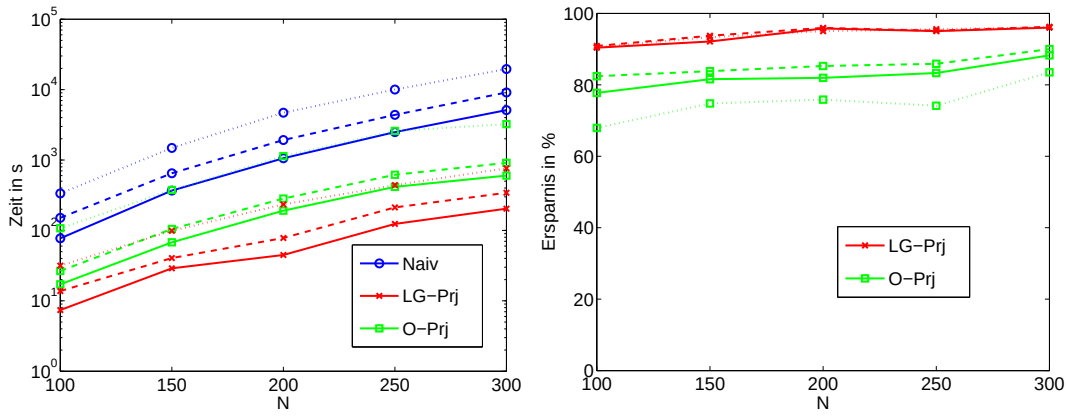

Abbildung 6.8: Rechenzeiten (links) und prozentuales Verhältnis (rechts) der verschiedenen Varianten des Krogstadverfahrens zur Lösung des zweidimensionalen Beispiels 6.2

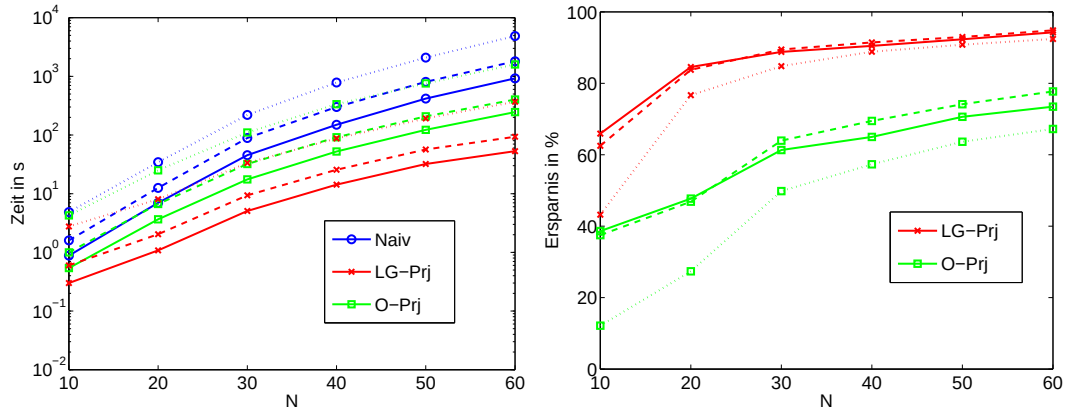

Abbildung 6.9: Rechenzeiten (links) und prozentuales Verhältnis (rechts) der verschiedenen Varianten des Krogstadverfahrens zur Lösung des dreidimensionalen Beispiels 6.3

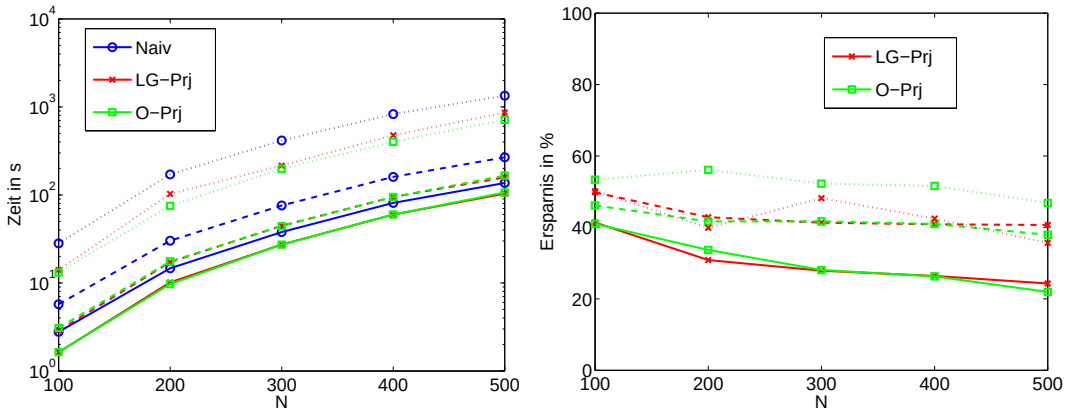

Abbildung 6.10: Rechenzeiten (links) und prozentuales Verhältnis (rechts) der verschiedenen Varianten des exponentiellen Euler Verfahrens mit Shift-and-invert Lanczosverf. zur Lösung des zweidimensionalen Beispiels 6.2

bezüglich des Krogstadverfahrens. Im Fall des exponentiellen Eulerverfahrens braucht man mit beiden Projektionsverfahren bei gleicher Zeitschrittweite ungefähr den gleichen Zeitaufwand und spart mit $h = 0.1$ um die 30 Prozent Rechenzeit, mit $h = 0.05$ mehr als 40 Prozent und mit $h = 0.01$ ungefähr 50 Prozent. Betrachtet man das Krogstadverfahren, so spart man mit dem Lanczos-Galerkin Verfahren immer mehr als mit dem Orthogonalprojektionsverfahren. Einzige Ausnahme ist der Fall mit Schrittweite $h = 0.01$ und großer Dimension. Mit dem Lanczos-Galerkin Verfahren spart man meist zwischen 50 und 70 Prozent und mit dem Orthogonalprojektionsverfahren 30 bis 50 Prozent an Rechenzeit gegenüber dem naiven Verfahren.

Insgesamt hat man also gesehen, dass man mit Hilfe der Projektionsverfahren einen exponentiellen Integrator erheblich beschleunigen kann und hoffen kann, dass diese exponentiellen Integratoren dadurch konkurrenzfähig zu anderen Integratoren werden. Dies untersuchen wir im Abschnitt 6.4.

6.3.2 Speicheraufwand

Neben dem Rechenaufwand ist auch der Speicheraufwand ein wichtiges Kriterium für ein gutes Verfahren. Ist der Speicheraufwand eines Verfahrens zu groß, so kann es sein, dass einige Probleme, die mit anderen Integratoren leicht berechnet werden können, mit diesem Verfahren auf Grund fehlenden Speicherplatzes nicht gelöst werden können. In den Kapiteln über die Projektionsverfahren haben wir unter der Rubrik Speicheraufwand gesehen, dass der Speicheraufwand von den Lanczositerationen abhängt, die zur Approximation der Produkte der Matrixfunktionen mit den Vektoren benötigt werden. Daher betrachten wir den Speicheraufwand und die benötigte Anzahl an Iterationen für die Verfahren *Naiv*, *LG-Prj* und *O-Prj*. Wir verwenden dafür nur das zweidimensionale Beispiel 6.2, da bei den anderen Beispielen die Ergebnisse ähnlich sind. Abbildung 6.12

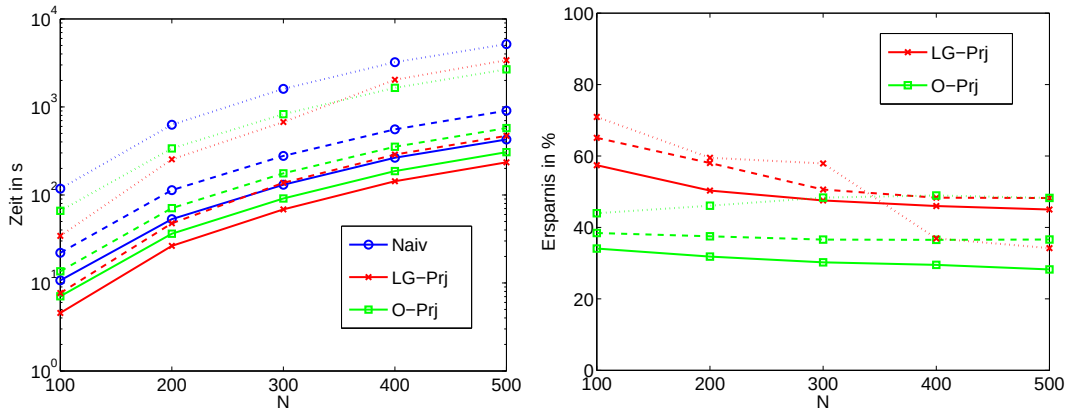

Abbildung 6.11: Rechenzeiten (links) und prozentuales Verhältnis (rechts) der verschiedenen Varianten des Krogstadverfahrens mit Shift-and-invert Lanczosverf. zur Lösung des zweidimensionalen Beispiels 6.2

zeigt für ein Diskretisierungsgitter mit $N = 200$ Gitterpunkten in jeder Raumdimension die Vergleiche bezüglich des exponentiellen Eulerverfahren mit Zeitschrittweite $h = 0.05$. Das zu Grunde liegende Krylovverfahren ist das Standard Lanczosverfahren. Das linke Bild gibt die Anzahl der benötigten Lanczositerationen pro Zeitschritt an. Im ersten Zeitschritt des Verfahrens, das die Orthogonalprojektion verwendet, ist die Anzahl der Iterationen angegeben, die insgesamt zur Berechnung der Produkte der Matrixfunktion mit dem ersten Vektor und den l Ritzvektoren benötigt werden. Im rechten Bild ist die Anzahl der maximal gespeicherten Vektoren in jedem Zeitschritt aufgetragen. In Abbildung 6.13 sind dieselben Ergebnisse für das Krogstadverfahren dargestellt, da sich die Verfahren beim Euler- und Krogstadverfahren gleich verhalten, diskutieren wir sie gleichzeitig. Das naive Verfahren braucht mit beiden Integratoren, wie zu erwarten war, in jedem Zeitschritt etwa konstant viele Lanczositerationen. Diese liegen außer im ersten Schritt bei circa 220 Lanczositerationen pro Zeitschritt. Da man pro Zeitschritt einen Vektor der Dimension $n = N^2$ speichern muss, ist die Anzahl der gespeicherten Vektoren gleich der Anzahl der Lanczositerationen. Das Lanczos-Galerkin Projektionsverfahren braucht im ersten Zeitschritt genauso viele Iterationen wie das naive Verfahren, da ja dort auch nur ein Standard Lanczositerationsschritt durchgeführt wird. Aber in den folgenden Zeitschritten wird nur ein Iterationsschritt durchgeführt, das heißt, dass die Startapproximation, die man durch die Projektion auf den Raum der Krylovvektoren aus dem ersten Zeitschritt bekommt, schon gut genug ist. Deshalb müssen auch nahezu nur die Krylovvektoren aus dem ersten Zeitschritt gespeichert werden, das sind ungefähr 150 Vektoren. Das Verfahren, das die Orthogonalprojektion nutzt, braucht in jedem Schritt mehr Lanczositerationen als das Lanczos-Galerkin Projektionsverfahren. Minimal benötigt es zwei Iterationen. Da nach jedem zwölften Produkt von Matrixfunktion und Vektor ein Restart gemacht wird, braucht man dann wieder mehr Iterationen. Die Iterationsanzahl schwankt also hier immer zwischen 2 und 200. Zusätzlich zu den,

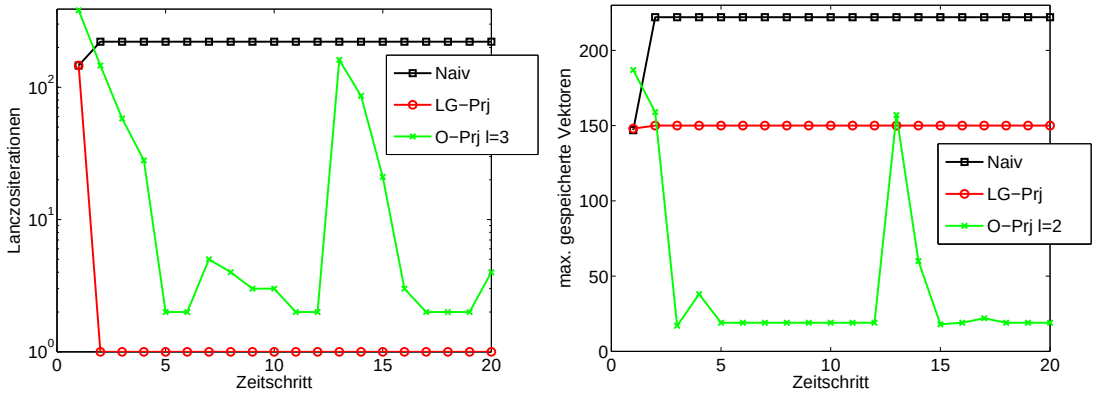

Abbildung 6.12: Lanczositerationen und Speicheraufwand der verschiedenen Varianten des exponentiellen Euler Verfahrens mit $h = 0.05$ zur Lösung des zweidimensionalen Beispiels 6.2 mit $N = 200$

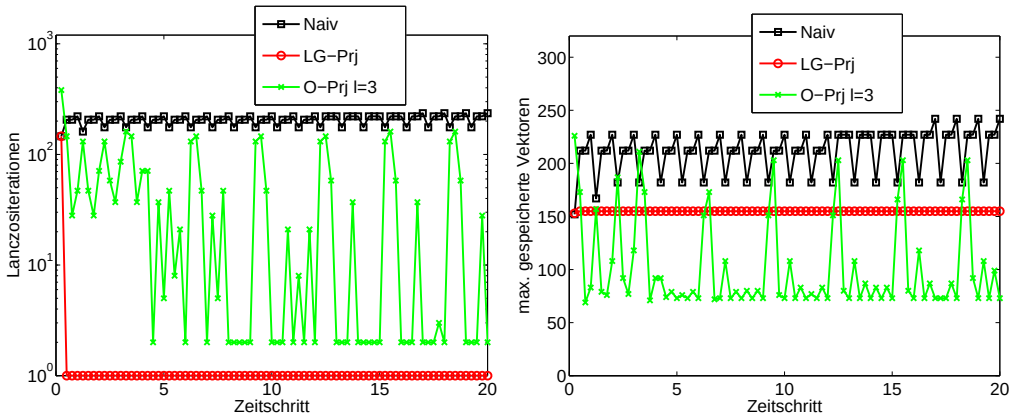

Abbildung 6.13: Lanczositerationen und Speicheraufwand der verschiedenen Varianten des Krogstadverfahrens mit $h = 0.05$ zur Lösung des zweidimensionalen Beispiels 6.2 mit $N = 200$

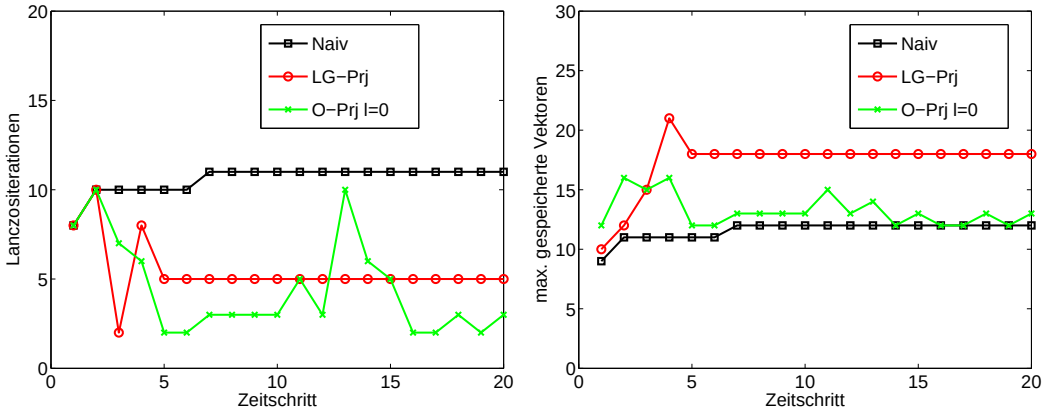

Abbildung 6.14: Shift-and-invert Lanczositerationen und Speicheraufwand der verschiedenen Varianten des exponentiellen Euler Verfahrens mit $h = 0.05$ zur Lösung des zweidimensionalen Beispiels 6.2 mit $N = 500$

durch die Iteration erstellten, Vektoren, müssen noch die Vektoren gespeichert werden, die durch die Verwendung der QR Zerlegung hinzukommen, wie wir das im Abschnitt 5.2.4 gesehen haben. Insgesamt ist der Speicheraufwand für das Verfahren mit der Orthogonalprojektion am kleinsten, dagegen ist die Iterationsanzahl pro Zeitschritt für das Lanczos-Galerkin Verfahren am kleinsten.

Betrachten wir auf gleiche Weise das Beispiel 6.2 mit dem Shift-and-invert Lanczosverfahren und $N = 500$, so bekommen wir Abbildung 6.14 und 6.15. Bei diesen Verfahren müssen wegen der Vorkonditionierung des Lanczosverfahrens wesentlich weniger Iterationsschritte gemacht werden. Die Anzahl der Lanczositerationen bei dem naiven Verfahren liegt bei ungefähr 10 pro approximiertes Produkt. Die benötigten Iterationen für die Projektionsverfahren liegen darunter. Hierbei ist das Projektionsverfahren im Gegensatz zur Verwendung des Standard Lanczosverfahrens nicht immer nach einem Iterationsschritt fertig, sondern braucht immer noch ungefähr 5 Schritte. Man sieht auch, dass ab und zu ein Restart gemacht wird, zum Beispiel im zweiten Zeitschritt des exponentiellen Euler Verfahrens. Dadurch, dass so wenige Iterationsschritte benötigt werden, liegt die Anzahl der zu speichernden Vektoren der Projektionsverfahren über der des naiven Verfahrens. Denn da bei den Projektionsverfahren im Gegensatz zu dem naiven Verfahren immer noch Informationen schon approximierter Produkte gespeichert werden, kann der Speicheraufwand der Projektionsverfahren größer als der des naiven Verfahrens werden, wenn der Unterschied der Anzahl der benötigten Lanczositerationen nicht sehr groß ist.

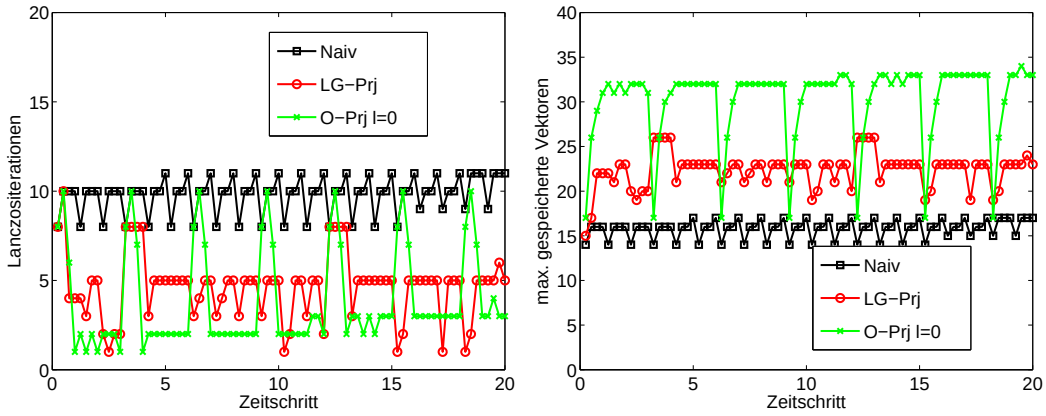

Abbildung 6.15: Shift-and-invert Lanczositerationen und Speicheraufwand der verschiedenen Varianten des Krogstadverfahrens mit $h = 0.05$ zur Lösung des zweidimensionalen Beispiels 6.2 mit $N = 500$

6.4 Vergleich verschiedener Integratoren

In diesem Abschnitt vergleichen wir unterschiedliche Integratoren zur Lösung der Testprobleme und untersuchen, ob die exponentiellen Integratoren mit den neu vorgestellten Projektionsverfahren konkurrenzfähig sind. Die verwendeten Integratoren sind:

ode15s: Variation des Matlab *ode15s* Integrators durch Einfügung der Matlab Permutation *colamd* für die LU Zerlegungen zur Beschleunigung der Rechenzeit.

Krogstad: Das Krogstadverfahren mit einem projizierten Lanczosverfahren und geeigneter Schrittweite. Es wird jeweils angegeben, welches Projektionsverfahren und welches Lanczosverfahren genutzt wird. Falls das Shift-and-invert Lanczosverfahren verwendet wird, werden die vorkommenden linearen Gleichungssysteme mit der LU Zerlegung gelöst, wobei diese durch die Permutation *colamd* beschleunigt wird.

Radau5: Das Radau5 Verfahren von Hairer und Wanner [32] unter Verwendung der Permutation *colamd* bei den LU Zerlegungen.

RKC: Der Runge-Kutta-Chebyshev Integrator von Sommeijer, Shampine und Verwer [76], der auf den expliziten Runge-Kutta-Chebyshev Methoden von van der Houwen und Sommeijer [82] beruht. Eine Matlab Implementierung dieses Verfahrens ist von Liljo [49] geschrieben worden.

Rosenbrock43: Das exponentielle Rosenbrockverfahren der Variation '43' [39] aus Kapitel 2.2.2 mit ebenfalls einem projizierten Lanczosverfahren unter eventueller Verwendung der LU Zerlegung mit der Permutation *colamd*.

Das Krogstadverfahren benutzt konstante Zeitschrittweite, falls nicht anders angegeben ist $h = 0.1$. Die anderen Integratoren dagegen verwenden eine Zeitschrittweitensteuerung. Wir berechnen die Näherungen an die Testprobleme mit unterschiedlichen Diskretisierungsgittern N. Dazu verwenden wir die angegebenen Verfahren zur vorgebenen absoluten Toleranz von 10^{-5}.

Die Tabelle 6.1 gibt den relativen Fehler und die jeweiligen Rechenzeiten zur Berechnung des eindimensionalen Beispiels 6.1 an. In Abbildung 6.16 werden die Rechenzeiten der Tabelle nochmals graphisch dargestellt. Für das Krogstadverfahren und das Rosenbrockverfahren wird das Lanczos-Galerkin Projektionsverfahren mit der Shift-and-invert Lanczosmethode verwendet.

Tabelle 6.2 und Abbildung 6.17 stellen den Vergleich zur Lösung des zweidimensionalen Beispiels 6.2 dar. Auch hier verwenden wir das Lanczos-Galerkin Projektionsverfahren mit dem Shift-and-invert Lanczosverfahren.

Die Resultate der Lösung des dreidimensionalen Beispiels 6.3 zeigen Tabelle 6.2 und Abbildung 6.17, dabei wird das Lanczos-Galerkin Projektionsverfahren mit dem Standard Lanczosverfahren benutzt.

Das Beispiel 6.4 auf dem Einheitskreis lösen wir mit den Integratoren *ode15s*, *radau5* und zwei Krogstadvarianten, wobei das eine die Orthogonalprojektion und das anderen

N		ode15s	Krogstad	Radau5	RKC	Rosenbrock43
100	Zeit	0.24	0.6	0.29	0.32	0.71
	Fehler	9.83e-07	6.10e-06	1.59e-05	3.29e-06	9.55e-07
500	Zeit	0.07	0.35	0.08	1.6	0.44
	Fehler	9.83e-07	6.10e-06	1.44e-05	3.33e-06	3.00e-06
1000	Zeit	0.08	0.44	0.13	3.93	0.48
	Fehler	9.83e-07	6.15e-06	1.43e-05	3.33e-06	3.97e-06
5000	Zeit	0.44	1.55	0.64	68.93	1.44
	Fehler	9.47e-07	6.10e-06	1.41e-05	3.26e-06	1.90e-05
10000	Zeit	0.96	3.01	1.31	277.04	1.98
	Fehler	3.35e-06	6.10e-06	1.41e-05	3.43e-06	1.39e-05

Tabelle 6.1: Verschiedene Integratoren zur Lösung des eindimensionalen Beispiels 6.1

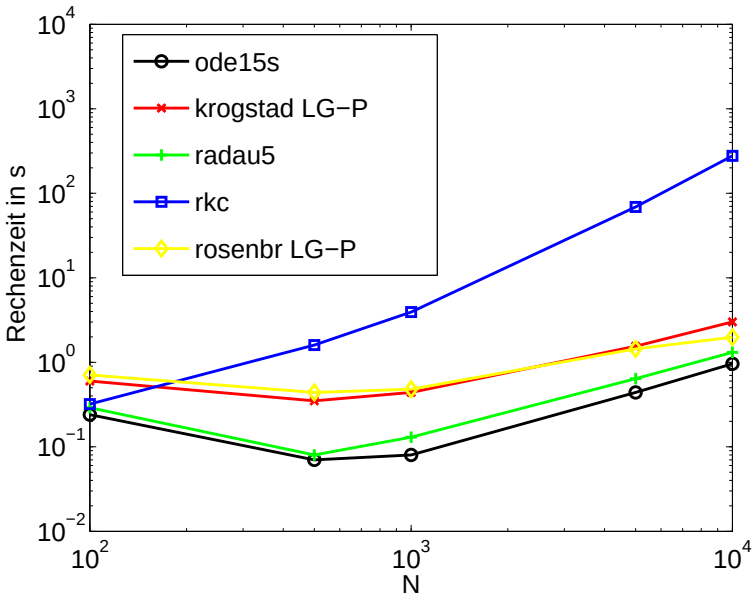

Abbildung 6.16: Verschiedene Integratoren zur Lösung des eindimensionalen Beispiels 6.1

N		ode15s	Krogstad	Radau5	RKC	Rosenbrock43
100	Zeit	3.52	3.81	7.12	7.99	5.16
	Fehler	8.23e-08	4.05e-06	3.64e-03	2.69e-05	4.05e-06
200	Zeit	21.35	22.38	47.52	77.45	23.32
	Fehler	8.22e-08	3.89e-06	3.63e-03	2.62e-05	2.08e-05
300	Zeit	63.85	57.43	163.1	261.63	66.99
	Fehler	8.22e-08	3.90e-06	3.63e-03	2.60e-05	7.26e-06
400	Zeit	153.45	116.8	360.67	645.86	153.23
	Fehler	8.22e-08	3.88e-06	3.63e-03	2.58e-05	1.22e-05
500	Zeit	284.55	204.92	681.72	1267.04	282.21
	Fehler	8.22e-08	3.92e-06	3.63e-03	2.58e-05	4.12e-05

Tabelle 6.2: Verschiedene Integratoren zur Lösung des zweidimensionalen Beispiels 6.2

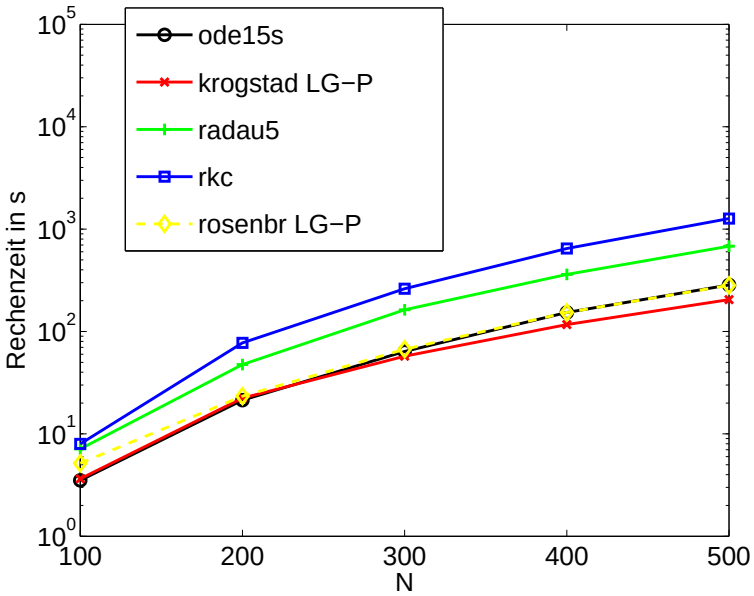

Abbildung 6.17: Verschiedene Integratoren zur Lösung des zweidimensionalen Beispiels 6.2

N		ode15s	Krogstad	Radau5	RKC	Rosenbrock43
10	Zeit	0.69	0.31	1.46	0.32	1.2
	Fehler	4.14e-06	3.70e-06	4.12e-04	3.80e-06	3.25e-05
20	Zeit	30.46	0.76	88.01	1.83	3.2
	Fehler	4.58e-06	3.71e-06	2.44e-04	3.58e-06	1.75e-04
30	Zeit	492.66	3.23	1576.35	9.89	11.9
	Fehler	4.67e-06	3.71e-06	1.87e-04	3.22e-06	1.18e-04
40	Zeit	-	8.65	-	31.11	29.87
	Fehler	-	3.71e-06	-	3.38e-06	3.09e-04
50	Zeit	-	18.77	-	74.36	75.29
	Fehler	-	3.71e-06	-	3.57e-06	1.49e-04

Tabelle 6.3: Verschiedene Integratoren zur Lösung des dreidimensionalen Beispiels 6.3

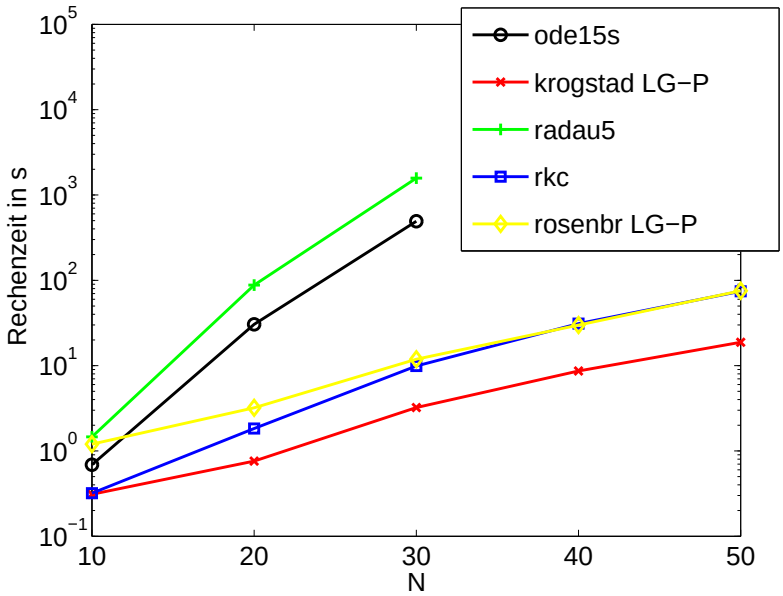

Abbildung 6.18: Verschiedene Integratoren zur Lösung des dreidimensionalen semilinearen Beispiels 6.3

N		ode15s	Krogstad O-Prj	Krogstad LG-Prj	Radau5
7378	Zeit	52.03	25.32	27.66	47.02
	Fehler	3.76e-06	8.66e-07	8.91e-07	1.44e-04
10498	Zeit	84.13	42.11	55.6	90.31
	Fehler	3.29e-06	6.60e-07	8.92e-07	7.99e-05
16667	Zeit	154.77	78.69	97.83	188.55
	Fehler	3.77e-06	8.68e-07	8.92e-07	7.42e-06
29799	Zeit	350.94	183.16	240.65	492.6
	Fehler	3.42e-06	8.58e-07	8.91e-07	3.67e-06
67316	Zeit	1057.09	606.93	815.66	1647.39
	Fehler	2.83e-06	8.93e-07	8.92e-07	2.52e-05
74108	Zeit	1230.2	694.81	1100.95	2100.47
	Fehler	3.21e-06	8.90e-07	8.91e-07	2.04e-05

Tabelle 6.4: Verschiedene Integratoren zur Lösung des Beispiels 6.4

die Lanczos-Galerkin Projektion nutzt. Beide Projektionsverfahren nutzen das Shift-and-invert Lanczosverfahren. Das Resultat sieht man in Tabelle 6.4 und Abbildung 6.19.

Als letztes lösen wir das Beispiel 6.5. Da es sich um ein nicht symmetrisches Problem handelt, verwenden wir für das Krogstad- und das Rosenbrockverfahren das Lanczos-Galerkin Projektionsverfahren basierend auf dem Arnoldiverfahren. Die Tabelle 6.5 und die Abbildung 6.20 zeigen die Ergebnisse der unterschiedlichen Verfahren, dabei betrachten wir nicht das RKC-Verfahren, da es nicht für unsymmetrische Probleme geeeignet ist. Hier ist die konstante Zeitschrittweite des Krogstadverfahrens $h = 0.04$.

In den meisten Beispielen wählt man das Lanczos-Galerkin Projektionsverfahren, da es schneller als das Orthogonalprojektionsverfahren ist. Es gibt aber auch Ausnahmen, wie bei dem Problem des Beispiels 6.4. Man kann in allen Anwendungen deutlich sehen, dass man mit unseren neu vorgestellten Strategien zur effektiven Implementierung von exponentiellen Integratoren diese wesentlich beschleunigen und damit wettbewerbsfähig machen kann. Dies war das Ziel dieser Arbeit.

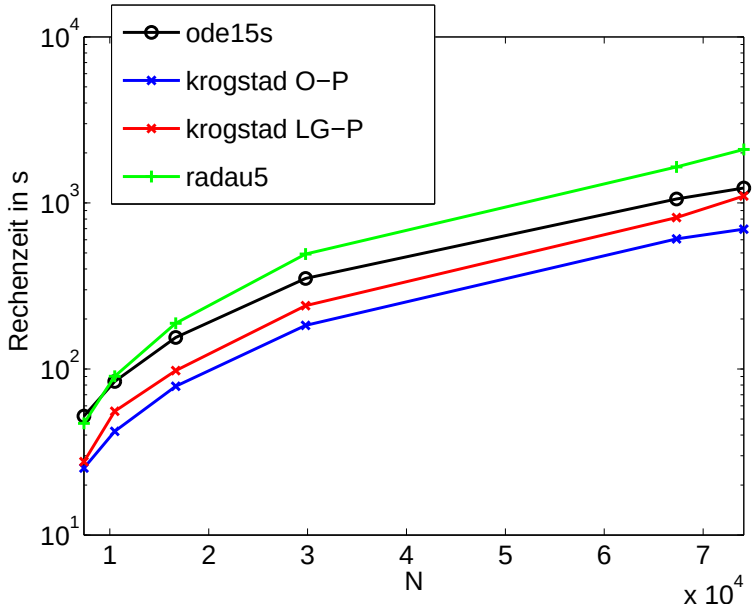

Abbildung 6.19: Verschiedene Integratoren zur Lösung des Beispiels 6.4

N		ode15s	Krogstad	Radau5	Rosenbrock43
100	Zeit	44.16	6.22	41.34	24.69
	Fehler	1.04e-08	7.80e-09	3.19e-13	2.55e-09
150	Zeit	113.95	16.55	129.12	72.05
	Fehler	4.01e-08	1.84e-09	2.14e-13	1.18e-09
200	Zeit	255.94	30.26	298.04	143.77
	Fehler	5.95e-08	1.37e-09	2.62e-12	2.05e-09
250	Zeit	397.94	45.36	586.5	254.93
	Fehler	1.26e-10	3.32e-11	4.14e-13	7.22e-09
300	Zeit	626.73	67.5	1093.02	423.71
	Fehler	6.79e-09	9.07e-11	5.04e-14	6.04e-06

Tabelle 6.5: Verschiedene Integratoren zur Lösung des Beispiels 6.5

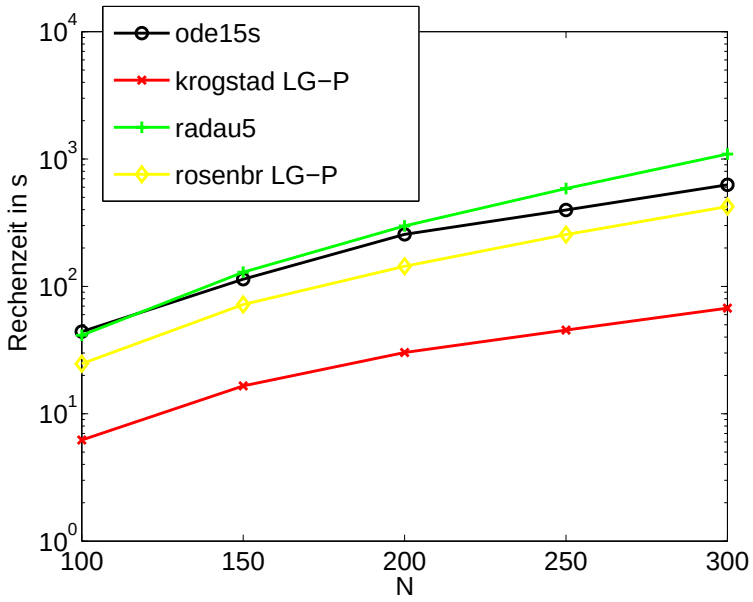

Abbildung 6.20: Verschiedene Integratoren zur Lösung des Beispiels 6.5

Kapitel 7

Bemerkungen und Danksagung

In dieser Arbeit sind Restartverfahren zur Approximation einer Matrixfunktion multipliziert mit einem Vektor vorgestellt, deren Konvergenz bewiesen, Fehlerschranken erstellt und es ist durch numerische Beispiele gezeigt worden, dass man mit diesem Verfahren Speicherplatz und manchmal auch Rechenzeit sparen kann. Des Weiteren sind zwei neue Verfahren für eine effiziente Berechnung von Produkten einer Matrixfunktion mit verschiedenen Vektoren vorgestellt worden. Das eine Verfahren ist ein Lanczos-Galerkin Projektionsverfahren, das andere ein Orthogonalprojektionsverfahren. Für beide Verfahren sind analytische Resultate und verschiedene Variationen dieser angegeben worden. Am Ende dieser Arbeit wurde mit numerischen Beispielen gezeigt, dass die neuen Verfahren die Effizienz exponentieller Integratoren stark verbessern und damit diese konkurrenzfähig zu anderen Integratoren machen können.

Zur weiteren Verbesserung von exponentiellen Integratoren gibt es noch andere Bereiche, in denen man weiter Forschung betreiben könnte. Ein Themengebiet zum Beispiel, das hier nicht behandelt wurde, aber sich nun anschließen würde, ist die Suche nach Verfahren, die die Approximation von Produkten sich leicht verändernder Matrizen mit einem Vektor beziehungsweise verschiedenen Vektoren auf effiziente Weise berechnet.

An dieser Stelle möchte ich mich recht herzlich bei all denen bedanken, die an der Entstehung dieser Arbeit beteiligt waren. Besonderer Dank gilt dabei meiner Referentin und Betreuerin Prof. Dr. Marlis Hochbruck für ihre ständige Gesprächsbereitschaft und ihren Anregungen, ohne die diese Arbeit nicht möglich gewesen wäre. Ebenso möchte ich Dr. Jasper van den Eshof danken, dass er mir gerade zu Beginn meines wissenschaftlichen Arbeitens immer hilfsbereit zur Seite stand. Michael Hönig danke ich füs Korrekturlesen und Jörg Sautter für seine Matrizen. Bei allen Mitarbeitern der Lehrstühle für Angewandte Mathematik und Mathematische Optimierung an der Heinrich-Heine Universität Düsseldorf bedanke ich mich für das gute Arbeitsklima und die tolle Atmosphäre untereinander. Abschließend möchte ich mich bei meiner ganzen Familie und besonders bei meiner Frau Christelle für ihre Liebe und Unterstützung bedanken.

Dieser Dissertation ist im Rahmen des Projektes *Iterationsverfahren für instationäre Differentialgleichungen* der Deutschen Forschungsgemeinschaft (DFG) enstanden.

Literaturverzeichnis

[1] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[2] C. Beattie, M. Embree, and J. Rossi. Convergence of restarted Krylov subspaces to invariant subspaces. *SIAM J. Matrix Anal. Appl.*, 25(4):1074–1109, 2004.

[3] L. Bergamaschi, M. Caliari, A. Martínez, and M. Vianello. Comparing Leja and Krylov approximations of large scale matrix exponentials. In V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, editors, *Computational Science — ICCS 2006*, volume 3994 of *Lecture Notes in Computer Science*, pages 685–692. Springer Berlin/Heidelberg, 2006.

[4] L. Bergamaschi, M. Caliari, and M. Vianello. The ReLPM exponential integrator for FE discretizations of advection-diffusion equations. In M. Bubak, G. D. v. Albada, P. M. A. Sloot, and J. Dongarra, editors, *Computational Science — ICCS 2004*, volume 3039 of *Lecture Notes in Computer Science*, pages 434–442. Springer Berlin/Heidelberg, 2004.

[5] L. Bergamschi and M. Vianello. Efficient conputation of the exponential operator for large, sparse, symmetric matrices. *Numer. Linear Algebra*, 7:27–45, 2000.

[6] H. Berland, B. Owren, and B. Skaflestad. B-series and order conditions for exponential integrators. *SIAM J. Numer. Anal.*, 43(4):1715–1727, 2005.

[7] H. Berland and B. Skaflestad. Solving the nonlinear Schrödinger equation using exponential integrators. Technical report, Norwegian University of Science and Technology, Trondheim, 2005.

[8] H. Berland, B. Skaflestad, and W. M. Wright. EXPINT - A matlab package for exponential integrators. Technical report, Norwegian University of Science and Technology, Trondheim, 2005.

[9] D. Calvetti and L. Reichel. Application of a block modified Chebyshev algorithm to the iterative solution of symmetric linear systems with multiple right hand sides. *Numer. Math.*, 68:3–16, 1994.

[10] M. P. Calvo and C. Palencia. A class of explicit multistep exponential integrators for semilinear problems. *Numer. Math.*, 102(3):367–381, 2005.

[11] E. Celledoni, A. Marthinsen, and B. Owren. Commutator-free lie group methods. *Future Generation Computer Systems*, 19(3):341–352, 2003.

[12] T. F. Chan and W. L. Wan. Analysis of projection methods for solving linear systems with multiple right-hand sides. *SIAM J. Sci. Comp.*, 18(6):1698–1721, November 1997.

[13] S. M. Cox and P. C. Matthews. Exponential time differencing for stiff systems. *J. Comput. Phys.*, 176(2):430–455, 2002.

[14] J. Daniel, W.B. Gragg, L. Kaufman, and G.W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comp.*, 30:772–795, 1976.

[15] P. J. Davis. *Interpolation and Approximation*. Dover, NY, 1975.

[16] V. Druskin, A. Greenbaum, and L. Knizhnerman. Using nonorthogonal Lanczos vectors in the computation of matrix functions. *SIAM J. Sci. Comp.*, 19(1):38–54, January 1998.

[17] V. Druskin and L. Knizhnerman. Krylov subspace approximation of eigenpairs and matrix functions in exact and computer arithmetic. *Num. Lin. Alg. Appl.*, 2(3):205–217, 1995.

[18] V. L. Druskin and L. A. Knizhnerman. Two polynomial methods of calculating functions of symmetric matrices. *U.S.S.R Comput. Maths. Math. Phys.*, 29:112–121, 1989.

[19] B. Ehle and J. Lawson. Generalized Runge–Kutta processes for stiff initial value problems. *J. Inst. Math. Appl.*, 16:11–21, 1975.

[20] M. Eiermann and O. Ernst. A restarted Krylov subspace method for the evaluation of matrix functions. Technical report, University of Freiberg, 2005.

[21] M. Eiermann, O. Ernst, and O. Schneider. Analysis of acceleration strategies for restarted minimal residual methods. *J. Comp. Appl. Math.*, 123(1–2):261–292, 2000.

[22] Y. T. Feng, D. Perić, and D. R. J. Owen. A block conjugate gradient method applied to linear systems with multiple right-hand sides. *Comp. Meth. Appl. Mech. Eng.*, 127:203–215, 1995.

[23] P.F. Fischer. Projection techniques for iterative solution of $A\underline{x} = \underline{b}$ with successive right-hand sides. *Comp. Meth. in Appl. Mech*, 163:193–204, 1998.

[24] R. W. Freund and M. Malhotra. A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides. *Linear Algebra and Its Applications*, 254:119–157, 1997.

[25] A. Friedli. Verallgemeinerte Runge–Kutta Verfahren zur Lösung steifer Differentialgleichungssysteme. In *Numerical Treatment of Differential Equations, Lecture Notes in Math.*, pages 35–50. Springer, Berlin, 1978.

[26] A. Frommer and U. Glässner. Restarted GMRES for shifted linear systems. *SIAM J. Sci. Comp.*, 19(1):15–26, January 1998.

[27] A. Frommer and V. Simoncini. Matrix functions. Technical report, Dipartimento di Matematica, Bologna, 2006.

[28] E. Gallopoulos and Y. Saad. Efficient solution of parabolic equations by Krylov approximation methods. *SIAM J. Sci. Statist. Comput.*, 13:1236–1264, 1992.

[29] G. H. Golub and C. F. Van Loan. *Matrix Computation*. John Hopkins University Press, 1996. Third Edition.

[30] G.-D. Gu and H-B. Wu. A Block EN Algorithm for Nonsymmetric Linear Systems with Multiple Right Hand Sides. *Lin. Alg. And its Appl.*, 299:1–20, 1999.

[31] A. El Guennouni, K. Jbilou, and H. Sadok. A block version of BICGSTAB for linear systems with multiple right-hand sides. *Electr. Trans. Num. Ana.*, 16:129–142, 2003.

[32] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer-Verlag, New York, 2d edition, 1996.

[33] M. Heyouni. The global Hessenberg and CMRH methods for linear systems with multiple right-hand sides. *Numer. Algorithms*, 26(4):317–332, April 2001.

[34] M. Hochbruck and M. E. Hochstenbach. Subspace extraction for matrix functions. *submitted to Elsevier Science*, 2005.

[35] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comp.*, 19(5):1552–1574, 1998.

[36] M. Hochbruck and Ch. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Num. Analy.*, 34(5):1911–1925, 1997.

[37] M. Hochbruck and A. Ostermann. Explicit exponential Runge–Kutta methods for semilinear parabolic problems. *SIAM JU. Num. Analy.*, 43(3):1069–1090, 2005.

[38] M. Hochbruck and A. Ostermann. Exponential Runge–Kutta methods for parabolic problems. *Appl. Num. Math.*, 53(2–4):323–339, May 2005.

[39] M. Hochbruck, A. Ostermann, and J. Schweitzer. Exponential integrators of Rosenbrock-type. volume 18/2006 of *Oberwolfach Reports*. Mathematisches Forschungsinstitut Oberwolfach, 2006.

[40] K. Jbilou. Smoothing iterative block methods for linear systems with multiple right-hand sides. *J. Comp. Appl. Math.*, 107:97–109, 1999.

[41] K. Jbilou, A. Messaoudi, and H. Sadok. Global FOM and GMRES algorithms for matrix equations. *Appl. Num. Math.*, 31(1):49–63, September 1999.

[42] P. Joly. Résolution de systèmes linéaires avec plusieurs seconds membres par la méthode du gradient conjugué. Technical Report R91012, Université Pierre et Marie Curie, 1991.

[43] L. A. Knizhnerman. Calculation of functions of unsymmetric matrices using Arnoldi's method. *Comput. Math. and Math. Phys.*, 31:1–9, 1992.

[44] L. A. Knizhnerman. Error bounds in Arnoldi's method: The case of a normal matrix. *Comput. Math. and Math. Phys.*, 32:1199–1211, 1992.

[45] S. Krogstad. Generalized integrating factor methods for stiff PDEs. *Journal of Computational Physics*, 203(1):72–88, 2005.

[46] J. D. Lambert and S. T. Sigurdsson. Multistep methods with variable matrix coefficients. *SIAM J. Numer. Anal.*, 9:715–733, 1972.

[47] 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*, 45:255–282, 1950.

[48] J. D. Lawson. Generalized Runge–Kutta processes for stable systems with large Lipschitz constants. *SIAM J. Numer. Anal.*, 4:372–380, 1967.

[49] J. Liljo. Numerische Verfahren zur Lösung parabolischer partieller Differentialgleichungen. *Master Arbeit, Mathematisches Institut, Heinrich-Heine Universität Düsseldorf*, 2006.

[50] L. Lopez and V. Simoncini. Analysis of projection methods for rational funciton approximation to the matrix exponential. *SIAM J. Numer. Anal.*, 44:613–635, 2006.

[51] P. Lötstedt and M. Nilsson. A minimal residual interpolation method for linear equations with multiple right hand sides. *SIAM J. Sci. Comput.*, 25(6):2126–2144, 2004.

[52] B. Minchev. Exponential integrators for semilinear problems, phd thesis. Technical report, University of Bergen, Norway, 2004.

[53] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, March 2003.

[54] I. Moret and P. Novati. RD rational approximations of the matrix exponential. *BIT*, 44:595–615, 2004.

[55] I. Moret and P. Novati. Interpolation functions of matrices on zeros of quasi-kernel polynomials. *Numer. Linear Algebra*, 11:337–353, 2005.

[56] R. B. Morgan. GMRES with deflated restarting. *SIAM J. Sci. Statist. Comput.*, 24(1):20–27, 2002.

[57] C. Musschoot. A Lanczos-type method for solving nonsymmetric linear systems with multiple right-hand sides - matrix and polynomial interpretation. *J. Comp. Appl. Math.*, 101:61–85, 1999.

[58] A. Nauts and R.E. Wyatt. New approach to many-state quantum dynamics: The recursive-residue-generation method. *Phys. Rev. Lett.*, 51:2238–2241, 1983.

[59] S. Nørsett. An A-stable modification of the Adams–Bashforth methods. In *Conference on the Numerical Solution of Differential Equations, J. Morris, ed., Lecture Notes in Math. 109*, pages 214–219. Springer, 1969.

[60] B. Nour-Omid. Applications of the Lanczos algorithm. *Comput. Phys. Comm.*, 53:157–168, 1989.

[61] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Lin. Alg. Appl.*, 29:293–322, 1980.

[62] A. Ostermann, M. Thalhammer, and W. M. Wright. A class of explicit exponential general linear methods. *BIT*, 46(2):409–432, 2006.

[63] M. Papadrakakis and S. Smerou. A new implementation of the Lanczos method in linear problems. *Int. Journal for Numer. Meth. in Eng.*, 29:141–159, 1990.

[64] T.J. Park and J.C. Light. Unitary quantum time evolution by iterative Lanczos reduction. *J. Chem. Phys.*, 85:5870–5876, 1986.

[65] B. N. Parlett. A new look at the Lanczos algorithm for solving symmetric systems of linear equations. *Lin. Alg. Appl.*, 29:323–346, 1980.

[66] D. Pope. An exponential method of numerical integration of ordinary differential equations. *Comm. ACM*, 6:491–493, 1963.

[67] A. Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Math. Comput.*, 33:680–687, 1979.

[68] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comp.*, 37:105–126, 1981.

[69] Y. Saad. On the Lanczos method for solving symmetric linear systems with several right-hand sides. *Math. Comp.*, 48:651–662, 1987.

[70] Y. Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):209–228, February 1992.

[71] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS, Boston, 1996.

[72] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 7(3):856–869, July 1986.

[73] V. Simoncini. Restarted full orthogonalization method for shifted linear systems. *BIT Numerical Math.*, 43:459–466, 2003.

[74] V. Simoncini and E. Gallopoulos. An iterative method for nonsymmetric systems with multiple right–hand sides. *SIAM J. Sci. Statist. Comput.*, 16:917–933, 1995.

[75] C.F Smith, A.F. Peterson, and R. Mittra. A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields. *IEEE Trans. Ant. Prop.*, 37:1490–1493, 1989.

[76] B.P. Sommeijer, L.F. Shampine, and J.G. Verwer. RKC: an explicit solver for parabolic PDEs. *J. Comp. Appl. Math.*, 88:315–326, 1991.

[77] D. E. Stewart and T. S. Leyk. Error estimates for Krylov subspace approximations of matrix exponentials. *J. Comput. Appl. Math.*, 72:359–406, 1996.

[78] K. Strehmel and R. Weiner. B-convergence results for linearly implicit one step methods. *BIT*, 27:264–281, 1987.

[79] K. Strehmel and R. Weiner. *Linear-implizite Runge–Kutta Methoden und ihre Anwendungen*. Teubner, Leipzig, 1992.

[80] L. N. Trefethen, J. A. C. Weidemann, and T. Schmelzer. Talbot quadratures and rational approximations. *BIT, Numerical Mathematics*, 46:653–670, 2006.

[81] J. van den Eshof and M. Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM J. Sci. Comp.*, 27(4):1438–1457, 2006.

[82] P.J. van der Houwen and B.P. Sommeijer. On the internal stability of explicit m-stage Runge-Kutta methods for large values of m. *ZAMM*, 60:479–485, 1980.

[83] H.A. van der Vorst. An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A. *J. Comput. Appl. Math.*, 18:249–263, 1987.

[84] J. Verwer. On generalized linear multistep methods with zero-parasitic roots and an adaptive principal root. *Numer. Math.*, 27:143–155, 1977.

[85] B. Vital. Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseurs. Technical report, Université de Rennes, 1990.

[86] K. Wu and H. Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM J. Matrix Ana. Appl.*, 22(2):602–616, 2000.

Die hier vorgelegte Dissertation habe ich eigenständig und ohne unerlaubte Hilfe angefertigt. Die Dissertation wurde in der vorgelegten oder in ähnlicher Form noch bei keiner anderen Institution eingereicht. Ich habe bisher keine erfolglosen Promotionsversuche unternommen.

Düsseldorf, den 05.12.2006

(Jörg Niehoff)