Eine Methode zur vollständigen Bestimmung der Eigenzustände reeller symmetrischer Profilmatrizen

vorgelegt von
Diplom-Ingenieur
Martin Ruess

an der Fakultät VI
- Bauingenieurwesen und Angewandte Geowissenschaften -
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
Dr.-Ing.

genehmigte Dissertation

Promotionsausschuß :

Vorsitzender	:	Prof. Dr.-Ing. Bernd Hillemeier
Gutachter	:	Prof. Dr. Dr.h.c.mult. Peter Jan Pahl
Gutachter	:	PD Dr.-Ing. habil. Frank Molkenthin

Tag der wissenschaftlichen Aussprache : 27. Juni 2005

Berlin 2005
D 83 (Diss. TU Berlin)

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fachgebiet *Theoretische Methoden der Bau- und Verkehrstechnik* der Technischen Universität Berlin.

Es war die *"täuschend einfache Formulierung der Eigenwertaufgabe" (Wilkinson)*, die mich zu einer Arbeit auf diesem Gebiet verleitet, aber auch inspiriert hat und die stets Ansporn bei der Bearbeitung und Lösung der Aufgabe war. Meine Tätigkeit in der Lehre, der enge Kontakt und die Diskussion mit den Studierenden haben diese Arbeit wertvoll begleitet und unterstützt.

Die Unterstützung des Fachgebietes und insbesondere die fachliche Anleitung und Unterstützung durch Herrn Prof. Dr. Dr.h.c.mult. Peter Jan Pahl haben zum Abschluß dieser Arbeit maßgeblich beigetragen. Als mein Mentor und Lehrer hat er mir eine Vielzahl von wertvollen Gelegenheiten zu selbständigem wissenschaftlichem Arbeiten ermöglicht. Er hat meine Freude und meine Neugierde an der Wissenschaft erkannt, gefördert und nachhaltig geprägt. Für das Vertrauen in meine Arbeit und in meine Person bedanke ich mich sehr herzlich.

Herrn PD. Dr.-Ing.habil. Frank Molkenthin möchte ich sehr herzlich für die spontane Bereitschaft zur Übernahme eines Gutachtens danken. Die kritische Beurteilung und die intensive Diskussion der Arbeit hat sich als außerordentlich wertvoll erwiesen.

Herrn Prof. Dr.-Ing. Bernd Hillemeier danke ich für die Übernahme des Vorsitz im Promotionsausschuß. Seine engagierte und exzellente Einführung und Leitung hat der wissenschaftlichen Aussprache einen besonderen Rahmen verliehen.

Meine Eltern haben mir mein Studium ermöglicht und mich stets in meinem Tun und Handeln bestärkt. Sie haben mir damit eine wichtige Sicherheit und Unterstützung gegeben. Mein Bruder Andreas gab mir den Antrieb zu meiner Ausbildung. Er gibt mir die fachliche und persönliche Unterstützung, die für diesen Weg so wichtig und wertvoll ist. Vielen Dank für Eure Unterstützung und Euer Vertrauen.

Bedanken möchte ich mich auch bei den Menschen, die mir sehr nahe stehen und mich auf einem wichtigen Stück meines Weges mit viel Geduld und Verständnis begleitet haben.

Berlin, im Juni 2005

Martin Ruess

Für Jakob

Kurzfassung

Die Bearbeitung von Ingenieuraufgaben erfordert häufig die Bestimmung einer größeren Anzahl von Eigenwerten und Eigenvektoren großer symmetrischer Profilmatrizen. Schwingungs- und Stabilitätsuntersuchungen an Bauwerken und Tragkonstruktionen gehören zu den klassischen Aufgabenbereichen im Bauwesen, die als Teil der Aufgabenstellung die Lösung einer allgemeinen oder speziellen Eigenwertaufgabe beinhalten.

Die Inverse Matrixiteration ist ein neues, profilerhaltendes Verfahren zur Lösung der speziellen Eigenwertaufgabe großer symmetrischer Profilmatrizen. Das Verfahren basiert auf dem QR-Algorithmus zur Bestimmung der Eigenzustände vollbesetzter Matrizen. Durch eine Reihe von Erweiterungen wurde eine Methode entwickelt, die sich durch ein hohes Maß an Flexibilität und Zuverlässigkeit hinsichtlich des Berechnungsablaufs und hinsichtlich der Eigenschaften der Lösung auszeichnet. Die Methode bestimmt eine beliebige Anzahl Eigenwerte und Eigenvektoren unabhängig voneinander. Die Eigenwerte konvergieren schrittweise und in sortierter Reihenfolge, beginnend mit dem betragskleinsten Eigenwert. Dies ermöglicht den effektiven Einsatz einer Spektralverschiebung zur Bestimmung von Eigenwerten in beliebigen Bereichen des Eigenwertspektrums der Aufgabe. Durch eine kontinuierliche Deflation der Matrix wird die Dimension der Aufgabe stetig reduziert. Der numerische Aufwand des Verfahrens nimmt mit zunehmender Anzahl bestimmter Eigenwerte kontinuierlich ab. Das Verfahren bestimmt betragsgleiche Eigenwerte mit gleichen und unterschiedlichen Vorzeichen und Cluster schlecht getrennter Eigenwerte mit hoher Genauigkeit und Zuverlässigkeit.

Eine wesentliche Verbesserung des Konvergenzverlaufs der Iteration wurde durch eine wiederholte Prekonditionierung der Matrix und die Jacobi-Randkorrektur erreicht. Beide Methoden sind profilerhaltend und werden lokal zur Vermeidung einer Stagnation der Konvergenz eingesetzt.

Das entwickelte Verfahren wurde in einer objektorientierten Entwicklungsumgebung implementiert. Für die Umsetzung und Untersuchung des neu entwickelten Verfahrens und der Vergleichsverfahren wurde ein Softwarepaket entwickelt und implementiert, das gestützt auf die Finite-Element-Methode als Untersuchungs- und Entwicklungsumgebung zur Bearbeitung numerischer Problemstellungen dient.

Zur Beurteilung und Einordnung der Inversen Matrixiteration wurden zwei Vergleichsverfahren implementiert, die den Entwicklungsstand der Methoden zur Lösung der reellen, symmetrischen Eigenwertaufgabe widerspiegeln. Untersucht wurde der numerische Aufwand der Verfahren, der Speicherbedarf, das Konvergenzverhalten, sowie die Zuverlässigkeit und Genauigkeit der Berechnungsergebnisse. Mit dem Vergleich der Verfahren wird die Inverse Matrixiteration bewertet und eingeordnet. Die Genauigkeit und Zuverlässigkeit der Ergebnisse bei der Eigenwert- und Eigenvektorbestimmung rechtfertigen einen teilweise erhöhten numerischen Aufwand.

Für das neu entwickelte Verfahren wurde mit der Untersuchung des Schwingungsverhaltens von Bauwerken ein Anwendungsfall beispielhaft dokumentiert.

Abstract

Eigenvalue problems are common in engineering tasks. In particular the prediction of structural stability and dynamic behavior are important aspects in engineering that lead to eigenvalue problems for which a set of successive eigenvalues and eigenvectors must be determined.

The Inverse Matrixiteration is a new method for the solution of the standard eigenvalue problem with large symmetric profile matrices. The method is based on the well-known QR-method for dense matrices. With several extensions a new, flexible and reliable method was developed that is highly suited for the independent computation of any set of eigenvalues and eigenvectors of profile matrices. The method retains the profile structure of the matrix. The eigenvalues converge stepwise in sorted order, starting with the eigenvalue of smallest modulus. An effective spectral shift strategy in combination with deflation of computed eigenstates allows a selective computation of eigenstates in any part of the complete eigenvalue spectrum. The dimension of the eigenvalue problem decreases stepwise, thus reducing the numerical effort of the computation. The rate of convergence increases continuously. Eigenvalues of equal sign or clustered eigenvalues are often determined in the same iteration cycle.

A repeated preconditioning process in combination with Jacobi rotations in the parts of strongest convergence of the matrix lead to a significant improvement of the global convergence behavior. Both developments retain the profile structure of the matrix.

A suitable algorithm is developed and implemented in an object-oriented Finite-Element-Application for the analysis of structural problems in engineering tasks.

Several examples demonstrate the flexibility and reliability of the new method. Two state-of-the-art solution methods are documented and implemented to compare the efficiency of the Inverse Matrixiteration. Different criteria including the numerical effort and the storage requirements as well as the reliability and the accuracy of the computation are introduced for a classification of the new method. The strength of the Inverse Matrixiteration lies in the computation of any subset of eigenstates of very large matrices with profile structure. The numerically stable and very reliable computation of selected eigenvectors justify a slightly higher numerical effort.

The advantageous use of the new method is illustrated with a detailed example for the dynamic analysis of structures in Civil Engineering.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

ALLGEMEIN

$Mflop/s$	Millions of floating point operations per second
$\mathcal{K}^m$	Krylov-Raum der Dimension m

GRIECHISCHE BUCHSTABEN

α_1	Rotation um die x_1-Achse
α_2	Rotation um die x_2-Achse
$\bar{\delta}$	Statistischer Mittelwert
$\breve{\delta}$	Standardabweichung
δ_{im}	Kronecker-Symbol : $\delta_{im} = \left\{ \begin{array}{ccccc} 1 & : & i & = & m \\ 0 & : & i & \neq & m \end{array} \right\}$
ϵ	Darstellungsgenauigkeit. Kleinste Fließkommazahl im Rechner für die gilt : $(1. + \epsilon) > 1.$
λ, λ_i	Eigenwert, i-ter Eigenwert
$\sigma(\mathbf{A})$	Vollständiges Eigenwertspektrum von $\mathbf{A}$
$\psi(\mathbf{A})$	Teilmenge des Eigenwertspektrums von $\mathbf{A}$
ω, ω_i	Eigenkreisfrequenz, i-te Eigenkreisfrequenz
Ω	Erregerkreisfrequenz
$\boldsymbol{\epsilon}^T$	:= $[\epsilon_{11}, \epsilon_{22}, \epsilon_{33}, \epsilon_{23}, \epsilon_{31}, \epsilon_{12}]$ Ingenieurdehnungen
$\boldsymbol{\sigma}^T$	:= $[\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{23}, \sigma_{31}, \sigma_{12}]$ Ingenieurspannungen

SKALARE

a	Skalar, $a \in \mathbb{R}$
a_{ik}	Matrixkoeffizient (i=Zeilenindex, k=Spaltenindex)
f	Technische Frequenz [Hz] $\hat{=}\frac{\sqrt{\lambda}}{2\,\pi}$
u_3	transversale Verschiebung in Richtung der x_3-Achse
C	Referenzkonfiguration eines Kontinuums
$\hat{C}$	Momentankonfiguration eines Kontinuums
N	Matrixdimension

VEKTOREN

$\mathbf{a}$	Vektor, $\mathbf{a} \in \mathbb{R}^{(N\times 1)}$
$\mathbf{x}, \mathbf{x}_i$	Eigenvektor, i-ter Eigenvektor
$\dot{\mathbf{x}}$	1. Ableitung des Ortsvektors nach der Zeit $(= \frac{\partial \mathbf{x}}{\partial t})$
$\ddot{\mathbf{x}}$	2. Ableitung des Ortsvektors nach der Zeit $(= \frac{\partial^2 \mathbf{x}}{\partial t^2})$

MATRIZEN

$\mathbf{A}$	Matrix, $\mathbf{A} \in \mathbb{R}^{(N\times N)}$
$\tilde{\mathbf{A}}$	Submatrix = Diagonalblock aus $\mathbf{A}$
$\hat{\mathbf{A}}, \check{\mathbf{A}}$	Durch Transformation veränderte Matrix $\mathbf{A}$
$\mathbf{A}^H$	hermitesch Transponierte der Matrix $\mathbf{A} \in \mathbb{C}$, entspricht $\mathbf{A}^T$, falls $\mathbf{A} \in \mathbb{R}$
$\mathbf{A}_{sub}^{(p\times p)}$	Submatrix der Dimension $(p \times p)$
$\mathbf{C}_M$	Materialmatrix $\begin{bmatrix} c_1 & c_2 & c_2 & 0 & 0 & 0 \\ c_2 & c_1 & c_2 & 0 & 0 & 0 \\ c_2 & c_2 & c_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_3 \end{bmatrix}$ $\begin{aligned} c_1 &= \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \\ c_2 &= \frac{\nu}{1-\nu}\, c_1 \\ c_3 &= \frac{E}{2(1+\nu)} \end{aligned}$

$\mathbf{D}$	Diagonalmatrix $(diag[d_{ii}])$
$\mathbf{D}^T$	Differentialoperator $\begin{bmatrix} \frac{\partial}{\partial x_1} & 0 & 0 & 0 & \frac{\partial}{\partial x_3} & \frac{\partial}{\partial x_2} \\ 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_1} \\ 0 & 0 & \frac{\partial}{\partial x_3} & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} & 0 \end{bmatrix}$
$\mathbf{E}, \mathbf{F}$	Diagonalmatrix $(diag[e_{ii}], e_{ii} = \pm 1)$, Identitätsmatrix
$\mathbf{I}$	Identitätsmatrix
$\mathbf{L}, \mathbf{C}$	Linksdreiecksmatrix mit Diagonalelementen $l_{ii} = 1$
$\mathbf{P}_{ik}, \mathbf{R}_{ik}$	Rotationsmatrix
$\mathbf{R}, \mathbf{U}$	Rechtsdreiecksmatrix
$\mathbf{T}$	Permutationsmatrix
$\mathbf{X}, \mathbf{Q}, \mathbf{P}$	Orthonormalmatrix

NORMEN

$\parallel \mathbf{a} \parallel_2$	Euklidische Vektornorm : $\parallel \mathbf{a} \parallel_2 = (\sum_i a_i^2)^{0.5}$
$\parallel \mathbf{A} \parallel_2$	Spektralnorm : $\parallel \mathbf{A} \parallel_2 = max \mid \lambda_i \mid, \quad \lambda_i \in \sigma(\mathbf{A})$
$\parallel \mathbf{A} \parallel_F$	Frobeniusnorm : $\parallel \mathbf{A} \parallel_F = (\sum_i \sum_j a_{ij}^2)^{0.5}$

Kapitel 1

Einleitung

Die Bearbeitung von Aufgabenstellungen aus den Natur- und Ingenieurwissenschaften kann häufig auf die Lösung eines Gleichungssystems zurückgeführt werden. Die Formulierung von Gleichungssystemen erfolgt auf der Grundlage physikalischer oder mathematischer Modelle zur Abstraktion von Ausschnitten der realen Welt. Gleichungssysteme, deren Lösung von den Eigenschaften des Modells und nicht von den Einwirkungen auf das Modell abhängen, werden als Eigenwertaufgaben bezeichnet.

Eigenwertaufgaben sind eine natürliche Folge von Aufgabenstellungen aus dem Bereich der Strukturdynamik, der Elektrotechnik, der technischen Chemie, der Genetik, der Quantenmechanik oder der Ökonomie. Ihre Formulierung und Lösung dient der Untersuchung von Schwingungs- und Stabilitätsverhalten von Tragstrukturen in Bauwerken, Flugzeugen, Turbinen u.a., der Spektroanalyse von Gasen, der Zustandsbeschreibung von Elementarteilchen oder der Fragestellungen makroökonomischer Prozesse nach ausgeglichenem Wachstum oder balanzierter Profitabilität. Insbesondere Schwingungs- und Stabilitätsanalysen von Bauwerken stehen im Mittelpunkt der Aufgabenstellungen des Bauwesens, die eine Bearbeitung von Eigenwertaufgaben erfordern.

Eigenwertaufgaben sind von Natur aus nichtlinear und bedürfen in der Regel einer iterativen Lösung. Der Aufwand moderner Lösungsverfahren ist um ein Vielfaches größer als für lineare Gleichungssysteme, weshalb der Größe der Aufgabe, der Struktur und der Belegung des Gleichungssystems eine zentrale Rolle zukommt. Trotz eines stetigen Entwicklungsprozesses des Computers als zentrales Medium bei der Speicherung und Berechnung großer Gleichungssysteme sind der Größe der Aufgaben noch immer Grenzen gesetzt. Die Leistungssteigerung der Rechnerentwicklung geht einher mit einem steten Anwachsen der Anforderungen. Die heute gesetzten Grenzen werden nur teilweise durch effiziente Speicherstrukturen und Algorithmen aufgeweicht.

Eine Vielzahl von Entwicklungen zur Lösung von Eigenwertaufgaben tragen der besonderen Eigenschaften der Aufgabenstellung und ihrer Matrizen Rechnung, um eine möglichst wirtschaftliche Bearbeitung der Aufgabe zu erzielen. In der vorliegenden Arbeit wird die für die Aufgabenstellungen des Bauingenieurwesens typische reelle, symmetrische Eigenwertaufgabe behandelt.

1.1 Problemstellung

Der Entwurf und die Konstruktion von Bauwerken sind eng verknüpft mit der Frage nach ihrer Stabilität und ihrem dynamischen Verhalten. Komplizierte Tragstrukturen mit immer schlankeren Bauteilen sind die Vorgaben zeitgenössischer Architektur, die hinsichtlich ihrer Sicherheit und Tragfähigkeit, aber auch hinsichtlich dem Wohlbefinden deren Nutzer und weiterer Aspekte beurteilt und bemessen werden müssen. Häufig ist dabei eine Eigenwertaufgabe so zu lösen, daß eine Reihe aufeinanderfolgender Eigenwerte und zugehöriger Eigenvektoren zu bestimmen ist.

Die Bestimmungsgleichungen der typischen Aufgabenstellungen aus dem Bereich der Stabilitäts- und Schwingungsanalyse oder der Wärmeströmung stammen aus Finite-Element-Formulierungen für partielle Differentialgleichungen zur Bestimmung physikalischer Feldfunktionen wie Verschiebungs-, Spannungs- oder Temperaturverläufe. Durch die geometrische bzw. zeitliche Diskretisierung des Lösungsgebietes zur Lösung des mathematischen Modells der Finite-Element-Methode ergeben sich in Abhängigkeit der gewählten Elementform und der eingeführten Freiheitsgrade mehrere hundert bis mehrere hunderttausend Systemgleichungen. Eine Matrix-Vektornotation der Systemgleichungen führt, eine zweckmäßige Diskretisierung vorausgesetzt, auf reelle symmetrische Koeffizientenmatrizen mit ausgeprägter Profilstruktur.

Die Struktur der Systemmatrix entscheidet wesentlich über den erforderlichen Speicherbedarf und den numerischen Aufwand zur Lösung der Aufgabe. Die Ausnutzung der Symmetrie reduziert den Speicherbedarf um den Faktor 2, die Ausnutzung einer Profilstruktur reduziert den Speicherbedarf um ein Vielfaches von 2. Der numerische Aufwand der Berechnung steht in Abhängigkeit der Dimension und der Form der Profilstruktur. Beide Aspekte, Speicherbedarf und Komplexität, sind trotz gestiegener Rechnerressourcen noch immer mitentscheidende Faktoren bei der Leistungsbeurteilung eines Lösungsverfahrens für Eigenwertaufgaben.

Sowohl die modernen großen Softwarepakete der Linearen Algebra als auch die Softwarelösungen der etablierten Bemessungssoftware im Bauwesen stützen sich im wesentlichen auf zwei Lösungsstrategien : Verfahren, die auf Ähnlichkeitstransformationen basieren und Subraummethoden.

Für die Bestimmung einer großen Anzahl von Eigenzuständen bzw. für die vollständige Lösung der Eigenwertaufgabe ist das auf Ähnlichkeitstransformationen basierende QR-Verfahren die Standardlösungsmethode. Der praktische Einsatz des QR-Verfahrens ist eine Kombination aus drei Lösungsschritten. Mit Hilfe orthogonaler Householder-Transformationen wird die Eigenwertaufgabe zunächst auf Tridiagonalgestalt transformiert und anschließend mit dem QR-Verfahren für tridiagonale Matrizen gelöst. Die Eigenvektoren werden anschließend mittels inverser Vektoriteration aus der spektralverschobenen Tridiagonalmatrix ermittelt und mit Hilfe der gespeicherten Informationen zur Rekonstruktion der Householder-Transformationsmatrizen zurück transformiert. Die Methode wird häufig als HQRI-Verfahren bezeichnet. Der numerische Hauptaufwand dieser Methode liegt in den Householder-Transformationen und bei größerer Anzahl zu berechnender Eigenvektoren bei der Rücktransformation der Eigenvektoren in die ursprüngliche Aufgabe. Die Bestimmung der Eigenzustände ist zuverlässig, aber wegen der Householder-Transformationen numerisch teuer.

Für große Matrizen mit ausgeprägter Bandstruktur erweist es sich als zweckmäßig, die Reduktion auf Tridiagonalgestalt durch Rotationsmatrizen vorzunehmen. Auf die Bildung des Produkts der Rotationsmatrizen zur Bestimmung der Eigenvektoren wird verzichtet. Die Eigenvektoren werden mit der inversen Vektoriteration an der ursprünglichen Matrix bestimmt.

Die zweite, häufig eingesetzte Methodik sind Subraumverfahren, insbesondere Verfahren, die auf der Methode von Lanczos basieren. Diese Berechnungsverfahren approximieren die gesuchten Eigenzustände durch Abbildung der Eigenwertaufgabe auf einen Subraum wesentlich kleinerer Dimension. Das Bild der Systemmatrix im Subraum ist im reellen symmetrischen Fall eine Tridiagonalmatrix kleiner Dimension, deren Eigenwerte mit dem QR-Verfahren für tridiagonale Matrizen bestimmt werden. Die Eigenvektoren der Tridiagonalmatrix werden mit Hilfe der orthonormalen Subraumbasis in die ursprüngliche Aufgabe transformiert.

Lanczos-Methoden sind durch die endliche Zahlendarstellung im Rechner hochgradig numerisch instabil und bedürfen eines großen numerischen Aufwands zur Beseitigung dieser Instabilität. Darüberhinaus werden die im Subraum approximierten Eigenwerte häufig nicht in aufeinanderfolgender Reihenfolge bestimmt, und stehen in direkter Abhängigkeit zum gewählten Startvektor der Iteration. Mehrfache Eigenwerte können als natürliche Folge des verwendeten Krylov-Subraums nicht ohne zusätzliche Maßnahmen approximiert werden. Lanczos-Methoden erfordern damit die Bestimmung einer größeren Anzahl von Eigenwerten, als dies zur Lösung der eigentlichen Aufgabe mit ausreichender Genauigkeit erforderlich ist. Die numerische Zuverlässigkeit der Methode muß nach der Eigenwertapproximation durch zusätzliche Maßnahmen sichergestellt werden. Mit zunehmender Anzahl zu bestimmender Eigenzustände steigt insbesondere der Speicheraufwand der Subraumbasis und damit auch der numerische Aufwand der Verfahren stark an. Lanczos-Verfahren nutzen die Symmetrie und Profilstruktur der Matrix. Damit sind sie insbesondere für die Berechnung sehr großer Matrizen geeignet.

Intensive Forschungen auf dem Gebiet der Subraummethoden haben dazu geführt, daß Eigenwerte für viele Anwendungen im Bauwesen zuverlässig bestimmt werden können. Die Zuverlässigkeit und Wirtschaftlichkeit der Eigenwertbestimmung an großen Profilmatrizen ist aber nach wie vor stark von der Anzahl gesuchter Eigenzustände abhängig und bedarf häufig einer ingenieurmäßigen Beurteilung.

1.2 Entwicklung und Stand der Forschung

Die Vielzahl von Verfahren zur Lösung der linearen Eigenwertaufgabe werden im wesentlichen von zwei grundlegenden Entwicklungsrichtungen dominiert. Verfahren zur Lösung von Aufgaben kleiner bis moderater Größe bilden eine Gruppe, welche die Eigenwertaufgabe mit Hilfe von Ähnlichkeitstransformationen in eine Schur-Normalform überführen, und damit eine Ermittlung der Eigenzustände per Inspektion erlauben. Diese Methoden überzeugen durch eine hohe Genauigkeit und numerische Stabilität. Die Hauptanstrengungen der Forschung zur Lösung der Eigenwertaufgabe konzentrieren sich jedoch auf die zweite Entwicklungsrichtung, die Subraummethoden. Seit Mitte der 80er Jahre wird die Lösung der speziellen symmetrischen Eigenwertaufgabe für große Matrizen als weitgehend gelöst betrachtet [14]. Der Forschungsschwerpunkt hat sich seitdem wesentlich auf die nichtsymmetrische Eigenwertaufgabe und Eigenwertprobleme höherer Ordnung verlagert [14],[35]. Die vorhandenen Verfahren zur Lösung der speziellen symmetrischen Eigenwertaufgabe eigenen sich jedoch häufig nicht für einen Einsatz in der Ingenieurpraxis. Mangelnde Zuverlässigkeit und Effektivität bei der Lösung der Eigenwertaufgabe unterschiedlicher Ingenieuraufgaben erfordern zur Beurteilung der Berechnungsergebnisse ein Spezialwissen, das nicht Teil der Ingenieurausbildung ist.

Verfahren nach Givens und Householder : Einen wichtigen und für die folgende Entwicklung grundlegenden Schritt gelang *W. Givens* (1954) und *A.S. Householder* (1958) mit der Erkenntnis, daß Matrizen durch Orthogonaltransformationen in einer endlichen Anzahl von Schritten auf eine einfachere Form reduziert werden können [14]. Givens setzte ebene Rotationen ein, um symmetrische Matrizen auf Tridiagonalgestalt zu transformieren, Householder reduzierte den numerischen Aufwand der Transformationen durch Einsatz von Spiegelmatrizen. Die Householder-Methode ist bis heute die zur Reduktion einer vollbesetzten symmetrischen Matrix auf Tridiagonalform angewandte Methode [3],[14].

Für Matrizen großer Dimension mit ausgeprägter Profilstruktur ist die Householder-Methode nur bedingt tauglich, da ihre Transformation nicht profilerhaltend ist. Lang [18] entwickelt einen Algorithmus, der den Profilverlust durch einen *bulge-chasing*-Prozess erfolgreich bekämpft. Der zusätzliche Speicherbedarf für diese Methode ist jedoch wesentlich größer als bei der auf Givens-Rotationen basierenden Standardmethode nach Schwarz [4]. Die Householder-Methode eignet sich gut für eine Parallelisierung der Berechnung [18]. Auch bei der Methode von Schwarz ist ein *bulge-chasing*-Prozess notwendig, um dem Verlust der Bandstruktur entgegenzuwirken. Der Algorithmus reduziert die Bandbreite jedoch element- statt blockweise und ist damit sowohl im Speicheraufwand als auch im operativen Aufwand wesentlich günstiger [18], [4].

Klassisches QR-Verfahren : In [14] wird das 1961 von *J.F.G. Francis* entwickelte QR-Verfahren als “eines der beliebtesten und mächtigsten Verfahren zur Lösung von Eigenwertproblemen für dichtbesiedelte symmetrische Matrizen am Ende des 20. Jahrhundert” bezeichnet. Moderne Computerimplementierungen des QR-Verfahrens für vollbesetzte Matrizen finden sich in allen großen Softwarepaketen der Linearen Algebra. 1971 wurde der Erhalt einer Bandstruktur von Wilkinson und Reinsch [49] und später von Parlett [30] dokumentiert.

Eine Anwendung und Weiterentwicklung dieser Entdeckung für große Profilmatrizen wurde aber

in der folgenden Forschung zur Lösung der speziellen Eigenwertaufgabe nicht weiter verfolgt. Stattdessen hat sich das QR-Verfahren als vorrangige Lösungsmethode für Tridiagonalmatrizen etabliert, die durch die Anwendung verschiedener Reduktionsmethoden auf große Profilmatrizen erzeugt werden [30],[35]. Eine implizite *Shift*-Strategie mit Wilkinson-Shift garantiert für tridiagonale Matrizen Konvergenz mit kubischer Konvergenzrate [30]. Dhillon [11] verbessert den Aufwand zur Eigenvektorbestimmung der Tridiagonalmatrizen von $O(n^3)$ auf $O(n^2)$ und verbessert zusätzlich die Orthogonalität der Vektoren untereinander. Gemeinsam mit dem Algorithmus von Schwarz zur Tridiagonalisierung großer Bandmatrizen ist das Verfahren unter der Bezeichnung *Givens-QR*-Verfahren bekannt und ist in verschiedenen Softwarepaketen Stand der Technik zur Bestimmung einer größeren Anzahl von Eigenwerten.

Subraumverfahren : Die durch die Vektoriteration erzeugte Vektorfolge $\{\mathbf{x}, \mathbf{A}\,\mathbf{x}, \mathbf{A}^2\,\mathbf{x}, \ldots\}$ wurde 1931 durch *A.N. Krylov* zur Bestimmung der Koeffizienten des charakteristischen Polynoms eingesetzt. Die generierten Vektoren sind linear unabhängige Vektoren von Subräumen mit zunehmender Dimension. Als Transformationsbasis sind sie jedoch nur wenig geeignet, da ihre Verwendung zu hochgradig schlecht konditionierten Systemen führt und die Eigenwerte der Transformation somit anfällig gegen kleine Störungen sind [48]. Damit scheiterte die Methode von Krylov, die so erzeugten Subräume sind seither jedoch mit dem Namen Krylovs belegt [14],[30], [35].

Lanczos-Verfahren : Die erfolgreichsten und am besten entwickelten Verfahren zur Bestimmung einer kleinen Anzahl von Eigenwerten großer Profilmatrizen sind das Verfahren von Lanczos für symmetrische Matrizen und das Arnoldi-Verfahren für den unsymmetrischen Fall. Beide Verfahren gehen zurück auf eine Entwicklung von *C. Lanczos*, 1950, bei der die Schwächen der Krylov-Vektoren durch eine Orthogonalisierung jedes neu generierten Vektors zu den vorhandenen Krylov-Vektoren überwunden werden [14], [30], [35]. Lanczos erkennt schnell, daß die verwendete Orthogonalbasis zur Abbildung der Eigenwertaufgabe auf einen wesentlich kleineren Subraum infolge Rundungsfehler nicht die theoretische Orthogonalität besitzt. Als Folge davon tendieren die erzeugten Basisvektoren immer stärker in Richtung des dominanten Eigenvektors der Abbildung auf einen Subraum. Die durch die Transformation mit der Teilbasis gewonnene, wesentlich kleinere Eigenwertaufgabe ist dadurch schlecht konditioniert und verhindert eine zuverlässige Bestimmung der gesuchten Eigenwerte. Beide Verfahren scheiterten zunächst und wurden erst zu Beginn der 70er Jahre wiederentdeckt, als *C.C. Paige* 1971 in seiner Dissertation *The computation of eigenvalues and eigenvectors of very large matrices* die Ursachen für den Orthogonalitätsverlust systematisch analysierte und aufdeckte. Seit dieser Zeit hat sich ein Großteil der Forschung auf die Behebung des Orthogonalitätsverlustes zwischen den Basisvektoren konzentriert und eine Vielzahl von Veröffentlichungen hervorgebracht, die verschiedene Reorthogonalisierungsmethoden zur Begrenzung des numerischen Mehraufwands behandeln [35],[30],[44].

Ein weiterer Schwerpunkt der Forschung behandelt numerisch stabile Deflationsstrategien, die zur Bestimmung mehrfacher Eigenwerte mit dem Lanczos-Verfahren erforderlich sind [19], [38]. Eine andere erfolgreiche Strategie zur Bestimmung mehrfacher Eigenwerte wird durch sogenannte Block-Lanczos-Verfahren verfolgt. Anstelle eines Iterationsvektors arbeiten diese Me-

thoden mit vergrößerten Subräumen [40], [42].

Erweiterungen der Lanczos-Verfahren : Eine wichtige Entwicklung bei der teilweisen Lösung der symmetrischen Eigenwertaufgabe, die eine Neuerung und Verbesserung der Lanczos-Verfahren erreichte, ist die implizite Neustartmethode (*Implicit Restart Technique*). Mit ihr gelingt es, unerwünschte Eigeninformationen aus dem Iterationsprozess herauszufiltern. Ein wiederholter Neustart der Methode verringert drastisch den Speicheraufwand und beschränkt insbesondere den numerischen Aufwand für den Orthogonalitätserhalt der Transformationsbasis. Der heutige Entwicklungsstand dieser Lanczos-Methode ist in der Theorie und in den Implementierungen des IRLES-Verfahrens (Implicitly Restarted Lanczos Method with Exact Shifts) umfassend dokumentiert und realisiert [5], [19].

Erweiterungen der Subraummethoden : Eine der jüngsten Entwicklungen der Subraummethoden zur Berechnung von Eigenzuständen großer Matrizen hatte ihren Ursprung 1975 in einer von *E.R. Davidson* entwickelten Methode. Davidson nutzte die Diagonaldominanz seiner Matrizen um Informationen über die gesuchten Eigenzustände in den anwachsenden Subraum einzutragen. Die Methode wurde für bestimmte Anwendungen sehr beliebt, führt aber zu Konvergenzproblemen sowohl für nicht-diagonaldominante Matrizen, als auch für Diagonalmatrizen [14]. Erst Mitte der 90er Jahre wurde eine vielversprechende Variante des Verfahrens als Kombination mit einem von Jacobi stammenden Verfahren entwickelt. Das Jacobi-Davidson-Verfahren stellt eine leistungsfähige Alternative zum Lanczos- und Arnoldi-Verfahren dar, sofern Eigeninformationen bereits vorhanden sind. Das Verfahren ist nach wie vor Bestandteil der aktiven Forschung.

1.3 Ziel der Arbeit

Wünschenswert wäre eine Methode, die die Bestimmung einer beliebigen Anzahl von Eigenzuständen, unabhängig von der Verteilung der Eigenwerte im Gesamtspektrum und unabhängig von a priori Informationen, in sortierter Reihenfolge zuverlässig ermöglicht. Die Methode sollte numerische Stabilität, hohe Genauigkeit, eine ansteigende Konvergenzrate und akzeptablen numerischen Aufwand für die Berechnung großer Profilmatrizen vereinen.

Das Ziel dieser Arbeit ist die Entwicklung und Erprobung eines stabilen Berechnungsverfahrens zur teilweisen oder vollständigen Lösung der speziellen symmetrischen Eigenwertaufgabe. Die Theorie des entwickelten Verfahrens wird systematisch hergeleitet und dokumentiert. Für eine Analyse des Konvergenzverhaltens und die darauf aufbauenden Entwicklungen zur Verbesserung und Begünstigung der Konvergenz wird eine vollständige Beweisführung vorgenommen. Ein Algorithmus für das Verfahren wird schrittweise entwickelt und implementiert. Mit der Implementierung werden verschiedene Untersuchungen zum Konvergenzverhalten, zur numerischen Stabilität und zum numerischen Aufwand vorgenommen. Die Untersuchungsergebnisse werden den Ergebnissen anderer Verfahren gegenübergestellt und bewertet. Die implementierten Vergleichsverfahren sind in ihrer Theorie und praktischen Umsetzung dokumentiert und spiegeln in ihrer Leistungsfähigkeit und Anwendbarkeit den aktuellen Entwicklungsstand wider. Die Anwendbarkeit der entwickelten Methode wird auf typische Problemstellungen aus dem Bauwesen angewandt und hinsichtlich ihrer Tauglichkeit bewertet und eingeordnet.

1.4 Vorgehensweise

Formulierung von Eigenwertaufgaben im Bauwesen : Im folgenden Kapitel 2 werden verschiedene Aufgabenstellungen des Bauwesens vorgestellt, die eine Bearbeitung einer Eigenwertaufgabe erfordern. Am Beispiel der Schwingungsaufgabe werden die mathematischen Grundlagen zur Formulierung einer Ingenieuraufgabe vorgestellt. Auf dieser Grundlage wird die Eigenwertaufgabe abgeleitet und die vorteilhafte Nutzung ihrer Ergebnisse als Teil der Ingenieuraufgabe aufgezeigt.

Inverse Matrixiteration : In Kapitel 3 werden aufbauend auf das QR-Verfahren für vollbesetzte Matrizen die Erweiterungen der Methode dokumentiert. Das QR-Verfahren wird in seiner ursprünglichen Form mathematisch dargestellt. Das Konvergenzverhalten des Verfahrens und die Verbesserung der Konvergenz durch den Einsatz von Spektralverschiebung und Deflation wird systematisch aufgezeigt und bewiesen. Die neuen Erweiterungen der Methode hinsichtlich dem Profilerhalt, der Konvergenzverbesserung durch Prekonditionierung und Jacobi-Randkorrektur und der Bestimmung der Eigenvektoren werden schrittweise eingeführt und beschrieben. Die Komplexität der entwickelten Methode wird abgeschätzt und durch Beispielrechnungen verifiziert. Für das Verfahren wird ein Algorithmus entwickelt und in der Programmiersprache *JAVA* implementiert. Seine numerische Stabilität und Zuverlässigkeit wird durch verschiedene Beispielrechnungen dokumentiert. Als Beispielmatrizen werden Systemmatrizen symmetrisch

elementierter, gelagerter und ungelagerter Quadratplatten mit knotenkonzentrierter Massenbelegung verwendet. Infolge der Symmetrie besitzen diese Matrizen eine hohe Anzahl mehrfacher Eigenwerte und stellen damit den schwierigsten Berechnungsfall der im Bauwesen relevanten Eigenwertaufgaben dar.

Vergleichsverfahren : Das entwickelte Verfahren wird in Kapitel 4 etablierten Verfahren zur Lösung der Eigenwertaufgabe gegenübergestellt. Als Vergleichsmethoden werden das QR-Verfahren für Matrizen mit ausgeprägter Bandstruktur (*Givens-QRI*-Verfahren) und eine leistungsfähige Implementierung der Lanczos-Methode (*IRL-Implicitly-Restarted-Lanczos*) gewählt. Beide Verfahren spiegeln den Stand der Technik zur Lösung der Eigenwertaufgabe im Ingenieurwesen wider. Sie werden in ihrer Theorie und praktischen Umsetzung beschrieben und in der Programmiersprache *JAVA* implementiert. Die Leistungsunterschiede der Vergleichsverfahren zur Inversen Matrixiteration werden aufgezeigt, mit Beispielrechnungen verifiziert und anschließend beurteilt. Das Verfahren der Inversen Matrixiteration wird bezüglich seiner Genauigkeit, Zuverlässigkeit und Gebrauchstauglichkeit eingeordnet.

Schwingungsanalyse ebener Tragwerke : Die Anwendung des entwickelten Verfahrens im Bauwesen wird am Beispiel der Schwingungsanalyse von Tragwerken aufgezeigt. Die Schwingungsanfälligkeit einer dynamisch belasteten Deckenplatte wird durch eine Spektralanalyse bestimmt. Die kritischen Schwingungsfrequenzen und die dazugehörigen Schwingformen werden ermittelt. Die Auswirkungen der Deckenschwingungen auf das Wohlbefinden der betroffenen Personen und auf die Tragfähigkeit der Konstruktion werden bestimmt und bewertet.

Kapitel 2

Formulierung von Eigenwertaufgaben im Bauwesen

2.1 Einleitung

Die typischen Formulierungen von Eigenwertaufgaben im Bauwesen haben ihren Ursprung in der Struktur- und Festigkeitsanalyse von Tragwerken. Sie sind Teil der Bestimmung struktureller Eigenschaften von Tragwerken zur Einschätzung und Beurteilung extremaler Last- und Verformungszustände und zur Prognose zeitabhängigen Verhaltens. Nachfolgende Aufgabenstellungen sind typische Beispiele für eine zwangsläufige Formulierung von Eigenwertaufgaben :

- *Stabilitätsuntersuchungen :* Die Analyse des Tragverhaltens schlanker Strukturen ist durch Spannungs- und Stabilitätsuntersuchungen gekennzeichnet. Während bei Spannungsproblemen jedem Belastungszustand eindeutig ein Verformungszustand zugeordnet ist, existieren bei Stabilitätsproblemen für ein bestimmtes Lastniveau mehrere Verformungszustände. Als kritisches Lastniveau wird ein Zustand bezeichnet bei dem die Struktur eine stabile Gleichgewichtslage verläßt und sich immer weiter von der Ausgangslage entfernt. Die Bestimmung kritischer Lastzustände zur Einschätzung der Stabilität und zur Ermittlung weiterer Gleichgewichtslagen im verformten Zustand erfordert die Bearbeitung einer allgemeinen Eigenwertaufgabe.

- *Schwingungsanalysen :* Untersuchungen zum Schwingungsverhalten von Bauwerken unter dynamischer Belastung erfordern die Formulierung und Lösung einer allgemeinen Eigenwertaufgabe. Die Lösung der Aufgabe liefert das Frequenzspektrum des diskretisierten Tragwerks, sowie die zugehörigen Schwingformen.

  Die Kenntnis über das Frequenzspektrum ermöglicht eine Prognose über die Stärke und die Wahrnehmung von Erschütterungen infolge unterschiedlicher Erregerfrequenzen, beispielsweise menscheninduzierter Schwingungen in Gebäuden, Schwingungen infolge Ver-

kehrsbelastung, Bautätigkeiten und Schallübertragung oder Erschütterungen infolge von Erdbeben und anderer Naturkatastrophen.

Eine Analyse des dynamischen Verhaltens von Tragstrukturen kann über direkte Integrationsschemata oder modale Methoden erfolgen. Direkte Integrationsmethoden lösen die Schwingungsaufgabe im Zeitbereich. Die Lösung erfolgt schrittweise in diskreten Zeitintervallen [3]. Der Aufwand für direkte Integrationsmethoden ist proportional zur Anzahl der verwendeten Zeitschritte und damit nur effektiv für die Berechnung einer kleinen Zeitspanne. Ist die Betrachtung einer größeren Zeitspanne erforderlich, so erweist sich die modale Analyse als effektiver. Eine selektive Berechnung der Eigenmoden [1] ist Teil der Bearbeitung der Eigenwertaufgabe und Voraussetzung für die modale Analyse.

- *Bestimmung von Hauptspannungszuständen :* Die Bestimmung der Hauptspannungen und Hauptspannungsrichtungen für einen bekannten Spannungszustand erfordert die Lösung einer speziellen Eigenwertaufgabe. Durch einen Basiswechsel mit Hilfe der Eigenvektoren der Aufgabe wird der Spannungzustand in ein Koordinatensystem mit maximalen Normalspannungskoeffizienten transformiert.

- *Wärmeübertragungsphänomene :* Zur Beurteilung des Wärmeverhaltens von Bauwerken wird die Lösung der Anfangswertaufgabe der instationären Wärmeströmung betrachtet. Unter der Voraussetzung nicht-temperaturabhängigen Materialverhaltens ist für eine aussagekräftige Einschätzung des Wärmeverhaltens die Betrachtung eines großen Zeitintervalls erforderlich. Die zu lösenden Bestimmungsgleichungen sind formal analog zu den Bestimmungsgleichungen der Strukturdynamik und werden ebenso vorteilhaft durch modale Methoden gelöst. Die Bestimmung der Eigenmoden zu den thermalen Eigenwerten als Bestandteil der modalen Analyse erfordert die Lösung einer allgemeinen Eigenwertaufgabe.

Im folgenden wird die Formulierung der Eigenwertaufgabe beispielhaft für die Schwingungsanalyse von Tragwerken gezeigt. Ausgehend von der differentiellen Beschreibung des dynamischen Verhaltens materieller Punkte eines Kontinuums im $\mathbb{R}^3$ wird nach der Methode der gewichteten Reste eine Integralform der dynamischen Bestimmungsgleichungen abgeleitet. Durch die Wahl lokaler Elementansätze für das diskretisierte Tragwerk wird die algebraische Form des Prinzips der virtuellen Verschiebungen der Dynamik formuliert.

Im Rahmen einer Modalanalyse wird eine allgemeine Eigenwertaufgabe formuliert. Die Berechnungsergebnisse sind zur Frequenzanalyse der Tragstruktur geeignet und werden zur effektiven Berechnung des Verschiebungsverlaufs mit der Zeit verwendet.

[1] = Eigenschwingformen

2.2 Dynamische Bestimmungsgleichungen

2.2.1 Differentialform

Betrachtet werden die materiellen Punkte eines Kontinuums im $\mathbb{R}^3$. Die Massenbelegung an jedem Punkt des Kontinuums sei unabhängig von der Zeit. Als äußere Einwirkungen des Kontinuums seien Massenkräfte und Reibungskräfte im Innern, sowie eingeprägte Verschiebungen und eingeprägte Spannungen auf der Oberfläche bekannt.

Für das Kontinuum sind folgende dynamischen Feldfunktionen in der Momentankonfiguration $\hat{C}$ zur Zeit t zu bestimmen :

$\mathbf{u}$	$=$	$\mathbf{u}(\mathbf{x}, t)$	Verschiebungsverlauf
$\mathbf{T}$	$=$	$\mathbf{T}(\mathbf{x}, t)$	Spannungszustand

Mit der linearen Elastizitätstheorie wird das Gleichgewicht für das Kontinuum in der bekannten, last- und spannungsfreien Referenzkonfiguration C zur Zeit $t = 0$ aufgestellt [27]. Das statische Gleichgewicht des Kontinuums wird nach *d'Alembert* um die Volumenlasten $\mathbf{p}_{VI}$[2] und $\mathbf{p}_{VD}$[3] infolge Massenträgheit und innerer Reibung erweitert.

Massenträgheitskraft $\mathbf{p}_{VI}$ Der materielle Punkt $P(\mathbf{x}, t)$ des Kontinuums wird zur Zeit t durch die infinitesimale Masse dm mit der Massendichte ρ des Kontinuums belegt.

$$dm = \rho \, dv \tag{2.1}$$

Mit der Beschleunigung $\ddot{\mathbf{x}}$ folgt die Massenträgheitskraft des materiellen Punktes $P(\mathbf{x}, t)$ entgegengesetzt der Bewegungsrichtung. Die Lage des Punktes $P(\mathbf{x}, t)$ zur Zeit t ist die Summe seiner Referenzlage in C und seiner Verschiebung $\mathbf{u}$ zur Zeit t. Da die Referenzlage des Punktes nicht zeitabhängig ist folgt für die Beschleunigung $\ddot{\mathbf{x}} = \ddot{\mathbf{u}}$ bzw. für die Geschwindigkeit $\dot{\mathbf{x}} = \dot{\mathbf{u}}$.

$$\mathbf{p}_{VI} \, dv = -\rho \, \ddot{\mathbf{u}} \, dv \qquad \mathbf{x} \in C \qquad \text{(Massenträgheit)} \tag{2.2}$$

Dämpfungskraft $\mathbf{p}_{VD}$ Als maßgebliche Ursache für die Dämpfungseffekte des Kontinuums wird Reibung im Innern und auf der Oberfläche angenommen. Die schwierig zu erfassenden Dämpfungskräfte werden durch ein einfach darstellbares viskoses Dämpfungsmodell approximiert. Mit der Annahme isotroper Materialeigenschaften ist die Dämpfungskraft $\mathbf{p}_{VD}$ unabhängig von der Richtung des Geschwindigkeitsvektors [2]. Die Wirkungsrichtung von $\mathbf{p}_{VD}$ und $\mathbf{p}_{tD}$ ist analog zur Massenträgheitskraft entgegengesetzt der Bewegungsrichtung. Die Koeffizienten μ_v und μ_a werden als Reibungskoeffizienten bezeichnet.

$$\mathbf{p}_{VD} \, dv = -\mu_v \, \dot{\mathbf{u}} \, dv \qquad \mathbf{x} \in C \qquad \text{(Reibung im Innern)} \tag{2.3}$$

$$\mathbf{p}_{tD} \, da = -\mu_a \, \dot{\mathbf{u}} \, da \qquad \mathbf{x} \in \partial C \qquad \text{(Reibung auf der Oberfläche)} \tag{2.4}$$

[2] I : inertia (engl.) = Trägheit
[3] D : damping (engl.) = Dämpfung

Die Menge der materiellen Punkte $\mathbf{x}$ auf der Oberfläche des Kontinuums wird in die disjunkten Teilmengen ∂C_u, ∂C_t und ∂C_{t_D} zerlegt. Auf dem Rand ∂C_u sind eingeprägte Verschiebungen, auf dem Rand ∂C_t die eingeprägten Spannungen bekannt. Der Rand ∂C_{t_D} ist ein Spannungsrand infolge eingeprägter Reibungskräfte. Die Richtung des Randes ist durch seinen Normalenvektor $\mathbf{n}$ eindeutig festgelegt.

Zur Beschreibung des dynamischen Verhaltens des Kontinuums liefert die lineare Elastizitätstheorie folgende Bestimmungsgleichungen :

$$\boldsymbol{\epsilon} = \mathbf{D}\,\mathbf{u} \qquad \mathbf{x} \in C \qquad \text{Dehnung-Verschiebung} \tag{2.5}$$

$$\boldsymbol{\sigma} = \mathbf{C}_M\,\boldsymbol{\epsilon} \qquad \mathbf{x} \in C \qquad \text{Materialgesetz, linear} \tag{2.6}$$

$$\mathbf{0} = \mathbf{D}^T\,\boldsymbol{\sigma} + \mathbf{p}_V - \mathbf{p}_{VI} - \mathbf{p}_{VD} \qquad \mathbf{x} \in C \qquad \text{Dynamisches Gleichgewicht} \tag{2.7}$$

$$\mathbf{t} = \mathbf{S}\,\mathbf{n} \qquad \mathbf{x} \in \partial C \qquad \text{Spannungsvektor auf dem Rand} \tag{2.8}$$

$$u_i = u_{i0} \qquad \mathbf{x} \in \partial C_u \qquad \text{Verschiebungsrand} \tag{2.9}$$

$$t_i = t_{i0} \qquad \mathbf{x} \in \partial C_t \qquad \text{Spannungsrand} \tag{2.10}$$

$$t_{i_D} = t_{i0_D} \qquad \mathbf{x} \in \partial C_{t_D} \qquad \text{Reibungsrand} \tag{2.11}$$

2.2.2 Integralform

Die Bestimmungsgleichungen (2.5) bis (2.11) werden numerisch approximiert. Zur Bestimmung geeigneter Arbeitsgleichungen wird ein Ansatz für die Verschiebungen $\mathbf{u}(\mathbf{x}, t) \in C$ gewählt. Die Dehnungen in (2.5) werden direkt durch Differentiation aus den Verschiebungen bestimmt. Die Spannungen in (2.6) sind eine lineare Funktion der Dehnungen. Beide Gleichungen sind damit a priori befriedigt.

Die Gleichungen (2.7) bis (2.11) werden durch den Ansatz nicht a priori befriedigt und liefern Reste, die nach der Methode der gewichteten Reste minimiert werden. Als Gewichtsfunktion wird nach der Methode von Galerkin die Variation $\delta\mathbf{u}$ des Verschiebungsvektors und die Variation $\delta\mathbf{t}$ des Spannungsvektors auf dem Rand ∂C gewählt.

$$\begin{aligned}\int_C \delta\mathbf{u}^T \frac{\partial\boldsymbol{\sigma}}{\partial\mathbf{x}}\,dv + \int_C \delta\mathbf{u}^T\,\mathbf{p}_V\,dv - \int_C \delta\mathbf{u}^T\,\mathbf{p}_{VI}\,dv - \int_C \delta\mathbf{u}^T\,\mathbf{p}_{VD}\,dv + \\ \int_{\partial C} \delta\mathbf{u}^T\,\boldsymbol{\sigma}\,\mathbf{n}\,da + \int_{\partial C_t} \delta\mathbf{t}^T\,(\mathbf{u}-\mathbf{u}_0)\,da + \int_{\partial C_u} \delta\mathbf{u}^T\,(\mathbf{t}-\mathbf{t}_0)\,da + \\ \int_{\partial C_{t_D}} \delta\mathbf{u}^T\,(\mathbf{t}_F - \mathbf{t}_{0F})\,da = 0\end{aligned} \tag{2.12}$$

Die Anwendung des Satzes von Gauss auf den ersten Term aus (2.12) liefert die gewünschte

Integralform der dynamischen Bestimmungsgleichung :

$$\begin{aligned}
&\int_C \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma}\, dv + \int_C \delta\mathbf{u}^T\, \mathbf{p}_{VI}\, dv + \int_C \delta\mathbf{u}^T\, \mathbf{p}_{VD}\, dv \;=\\
&\int_C \delta\mathbf{u}^T\, \mathbf{p}_V\, dv + \int_{\partial C_u} \delta\mathbf{u}^T\, \mathbf{t}\, da - \int_{\partial C_t} \delta\mathbf{u}^T\, \mathbf{t}_0\, da - \int_{\partial C_{t_D}} \delta\mathbf{u}^T\, \mathbf{t}_{0_D}\, da \qquad (2.13)\\
&u_i \;=\; u_{i0} \qquad \mathbf{x} \in \partial C_u
\end{aligned}$$

2.2.3 Algebraische Bestimmungsgleichungen

Die Integralgleichung (2.13) wird in eine algebraische Bestimmungsgleichung überführt. Der Ansatz über das Lösungsgebiet wird durch eine Summe lokaler Elementansätze ersetzt.

$$\begin{aligned}
\mathbf{u}(\mathbf{x},t) &= \sum_e \mathbf{u}_e^T(t)\, \mathbf{s}(\mathbf{z}) \qquad \mathbf{x} \in C \qquad \mathbf{z} \in C_e \qquad (2.14)\\
\mathbf{u}_e(t) &= \mathbf{R}_e\, \mathbf{u}_s(t)
\end{aligned}$$

Die Substitution der Elementansätze (2.14) mit Hilfe der Topologiematrizen $\mathbf{R}_e$ in die Bestimmungsgleichung (2.13) liefert folgende Systembeiträge :

$$\int_C \delta\mathbf{u}^T\, \mathbf{p}_{VD}\, dv = \delta\mathbf{u}_s^T\, \mathbf{M}_s\, \ddot{\mathbf{u}}_s(t) \qquad \text{Massenmatrix} \quad \mathbf{M} \qquad (2.15)$$

$$\int_C \delta\mathbf{u}^T\, \mathbf{p}_{VI}\, dv = \delta\mathbf{u}_s^T\, \mathbf{C}_s\, \dot{\mathbf{u}}_s(t) \qquad \text{Dämpfungsmatrix} \quad \mathbf{C} \qquad (2.16)$$

$$\int_C \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma}\, dv = \delta\mathbf{u}_s^T\, \mathbf{K}_s\, \mathbf{u}_s(t) \qquad \text{Steifigkeitsmatrix} \quad \mathbf{K} \qquad (2.17)$$

$$\int_C \delta\mathbf{u}^T\, \mathbf{p}_V\, dv = \delta\mathbf{u}_s^T\, \mathbf{p}_s(t) \qquad \text{Volumenlastvektor} \quad \mathbf{p}(t) \qquad (2.18)$$

$$\int_{\partial C_t/\partial C_{t_D}} \delta\mathbf{u}^T\, \mathbf{t}_0\, da = \delta\mathbf{u}_s^T\, \mathbf{q}_{s0}(t) \qquad \text{Flächenlastvektor} \quad \mathbf{q}_0(t) \qquad (2.19)$$

$$\int_{\partial C_u} \delta\mathbf{u}^T\, \mathbf{t}\, da = \delta\mathbf{u}_s^T\, \mathbf{q}_s(t) \qquad \text{Randspannungsvektor} \quad \mathbf{q}(t) \qquad (2.20)$$

Das Gleichgewicht der Arbeitsgleichung (2.13) wird zu einem festen Zeitpunkt t betrachtet. Die Variation $\delta\mathbf{u}_s$ nach den freien Parametern $u_i (i = 1, \ldots, n)$ des Ansatzes ist damit unabhängig von der Zeit und liefert die n erforderlichen Bestimmungsgleichungen zur Lösung der Aufgabe (2.13).

Die unbekannten Verschiebungen u_i des resultierenden Gleichungssystems (2.21) sind mit dem Ansatz (2.14) diskrete Funktionen der Geometrie, aber stetige Funktionen der Zeit. Anstelle einer zusätzlichen zeitlichen Diskretisierung des Gleichungssystems mit anschließender schrittweiser Zeitintegration, wird eine Lösung entkoppelter Gleichungen, durch eine Modalanalyse angestrebt.

2.3 Modalanalyse

2.3.1 Modale Form der Bestimmungsgleichungen

Das Verhalten eines linear elastischen Tragwerks im Momentanzustand $\hat{C}$ zur Zeit t wird durch die Bestimmungsgleichungen (2.21) für die Knotenverschiebungen des diskretisierten Tragwerks beschrieben.

$$\mathbf{M}\,\ddot{\mathbf{u}}(t) + \mathbf{C}\,\dot{\mathbf{u}}(t) + \mathbf{K}\,\mathbf{u}(t) \;=\; \mathbf{p}(t) \tag{2.21}$$

$\mathbf{M}$	Massenmatrix
$\mathbf{C}$	Dämpfungsmatrix
$\mathbf{K}$	Steifigkeitsmatrix
$\mathbf{u}(t)$	Vektor der relativen Verschiebungen
$\mathbf{p}(t)$	Lastvektor

Zur Lösung des Gleichungssystems (2.21) wird ein Ansatz (2.22) für den Verschiebungsvektor $\mathbf{u}(t)$ gewählt, der einen Basiswechsel zu generalisierten Verschiebungen $g_i(t)$ bewirkt und die Gleichungen (2.21) in eine einfachere Form transformiert.

$$\mathbf{u}(t) \;=\; \sum_{i=1}^{n} g_i(t)\,\mathbf{q}_i \;=\; \mathbf{Q}\,\mathbf{g}(t) \tag{2.22}$$

Eine effektive Ermittlung der Transformationsmatrix $\mathbf{Q}$ erfolgt durch die Lösung der homogenen Bestimmungsgleichungen für freie, ungedämpfte Schwingungen des Tragwerks (2.23). Als Lösungsansatz werden harmonische Schwingungen gewählt (2.24).

$$\mathbf{M}\,\ddot{\mathbf{u}}(t) + \mathbf{K}\,\mathbf{u}(t) \;=\; \mathbf{0} \tag{2.23}$$

$$\mathbf{u}(t) \;=\; \mathbf{x}_i\; sin(\omega_i\, t) \tag{2.24}$$

$\mathbf{x}_i$	Form der i-ten Eigenschwingung
ω_i	Kreisfrequenz der i-ten Eigenschwingung

Die Substitution des Lösungsansatzes in die Gleichungen (2.23) führen auf die allgemeine Eigenwertaufgabe (2.25). Die Steifigkeitsmatrix $\mathbf{K}$ und die Massenmatrix $\mathbf{M}$ sind in der vorliegenden Aufgabe symmetrisch und positiv definit. Die Aufgabe besitzt damit n Eigenlösungen $(\omega_i^2, \mathbf{x}_i)$ für die n massebehafteten Freiheitsgrade des diskretisierten Tragwerks.

$$\mathbf{K}\,\mathbf{x}_i \;=\; \omega_i^2\,\mathbf{M}\,\mathbf{x}_i \qquad\qquad i = 1, \ldots, n \tag{2.25}$$

Die Eigenvektoren $\mathbf{x}_i$ sind $\mathbf{K}$-orthogonal (2.26) und $\mathbf{M}$-orthonormal (2.27). Die Eigenmatrix $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]$ ist als Transformationsbasis des Ansatzes (2.22) geeignet.

$$\mathbf{x}_i^T \mathbf{K}\, \mathbf{x}_m = \omega_i^2\, \delta_{im} \qquad i, m = 1, \dots, n \tag{2.26}$$

$$\mathbf{x}_i^T \mathbf{M}\, \mathbf{x}_m = \delta_{im} \qquad i, m = 1, \dots, n \tag{2.27}$$

Mit der Eigenmatrix $\mathbf{X}$ folgt der Ansatz der Modalanalyse (2.28) als Linearkombination der Eigenvektoren $\mathbf{x}_i$ mit den generalisierten Verschiebungen $g_i(t)$. Der Ansatz wird in (2.21) substituiert.

$$\mathbf{u}(t) = \sum_{i=1}^{n} g_i(t)\, \mathbf{x}_i = \mathbf{X}\, \mathbf{g}(t) \tag{2.28}$$

$$\mathbf{M}\, \mathbf{X}\, \ddot{\mathbf{g}}(t) + \mathbf{C}\, \mathbf{X}\, \dot{\mathbf{g}}(t) + \mathbf{K}\, \mathbf{X}\, \mathbf{g}(t) = \mathbf{p}(t) \tag{2.29}$$

Wird Gleichung (2.29) mit $\mathbf{X}^T$ multipliziert, so folgen mit den Eigenschaften (2.26) bis (2.27) die Bestimmungsgleichungen der generalisierten Verschiebungen des Tragwerks.

$$\mathbf{X}^T \mathbf{M}\, \mathbf{X}\, \ddot{\mathbf{g}}(t) + \mathbf{X}^T \mathbf{C}\, \mathbf{X}\, \dot{\mathbf{g}}(t) + \mathbf{X}^T \mathbf{K}\, \mathbf{X}\, \mathbf{g}(t) = \mathbf{X}^T \mathbf{p}(t) \tag{2.30}$$

$$m_i\, \ddot{g}_i(t) + c_i\, \dot{g}_i(t) + k_i\, g_i(t) = p_i(t) \qquad i = 1, \dots, n \tag{2.31}$$

$$m_i = \mathbf{x}_i^T \mathbf{M}\, \mathbf{x}_i = 1.0 \qquad \text{generalisierte Masse} \tag{2.32}$$

$$c_i = \mathbf{x}_i^T \mathbf{C}\, \mathbf{x}_i \qquad \text{generalisierte Dämpfung} \tag{2.33}$$

$$k_i = \mathbf{x}_i^T \mathbf{K}\, \mathbf{x}_i = \omega_i^2 \qquad \text{generalisierte Steifigkeit} \tag{2.34}$$

$$p_i = \mathbf{x}_i^T \mathbf{p} \qquad \text{generalisierte Last} \tag{2.35}$$

Beteiligungsfaktor Der Lastvektor $\mathbf{p}(t)$ wird in einen konstanten, vom Wirkungsort der Last abhängigen Anteil, und eine Zeitfunktion zerlegt. Für die generalisierte Last $p_i(t)$ der entkoppelten Bewegungsgleichung folgt :

$$p_i(t) = \mathbf{x}_i^T\, \mathbf{p}(t) = \mathbf{x}_i^T\, \mathbf{p}_0\, f(t) \tag{2.36}$$

$\mathbf{p}_0$: Last, als Funktion des Ortes
$f(t)$: Zeitfunktion

Der Skalar $(\mathbf{x}_i^T\, \mathbf{p}_0)$ wird als Beteiligungsfaktor b_i bezeichnet. Er gibt Auskunft über die Beteiligung der i-ten Eigenschwingung am Gesamtschwingverhalten des durch $f(t)$ erregten Tragwerks. Mit b_i folgt für die Lastfunktion der i-ten generalisierten Schwingungsgleichung :

$$p_i(t) = \mathbf{x}_i^T\, \mathbf{p}(t) = b_i\, f(t) \tag{2.37}$$

Anfangswerte Die Anfangswerte zur Lösung der Gleichungen (2.31) werden mit dem Verschiebungsansatz (2.28) und mit Hilfe der M-Orthonormalität der Eigenvektoren für geeignete Anfangsverschiebungen $\mathbf{u}_0$ ermittelt.

$$t = 0: \quad \mathbf{u} = \mathbf{u}_0 \tag{2.38}$$

$$\dot{\mathbf{u}} = \dot{\mathbf{u}}_0 \tag{2.39}$$

$$\mathbf{u}_0 = \sum_{k=1}^{n} g_{k_0}\, \mathbf{x}_k \tag{2.40}$$

$$\mathbf{x}_i^T\, \mathbf{M}\, \mathbf{u}_0 = \sum_{k=1}^{n} g_{k_0}\, \mathbf{x}_i^T\, \mathbf{M}\, \mathbf{x}_k \quad \Longrightarrow \quad g_{i_0} = \frac{1}{m_i} \mathbf{x}_i^T \mathbf{M}\, \mathbf{u}_0 \tag{2.41}$$

$$\dot{\mathbf{u}}_0 = \sum_{k=1}^{n} \dot{g}_{k_0}\, \mathbf{x}_k \tag{2.42}$$

$$\mathbf{x}_i^T\, \mathbf{M}\, \dot{\mathbf{u}}_0 = \sum_{k=1}^{n} \dot{g}_{k_0}\, \mathbf{x}_i^T\, \mathbf{M}\, \mathbf{x}_k \quad \Longrightarrow \quad \dot{g}_{i_0} = \frac{1}{m_i} \mathbf{x}_i^T \mathbf{M}\, \dot{\mathbf{u}}_0 \tag{2.43}$$

Dämpfung Im allgemeinen kann die Dämpfung von Tragwerken nicht vernachlässigt werden, da sie den Energieverlust der Schwingungsantwort erfaßt [3]. Die Konstruktion der Dämpfungsmatrix $\mathbf{C}$ der allgemeinen Schwingungsgleichung (2.21) erfolgt entweder über phänomenologische Modelle zur Modellierung physikalischer Mechanismen wie Strukturdämpfung oder über Spektraldämpfungsmodelle, die auf Ersatzmodellen mit viskoser Dämpfung basieren [2][7].

Letzteres findet hier Anwendung in Form sogenannter Rayleigh-Dämpfung (Gleichung (2.44)). Die zu erfassende Dämpfung wird mit Hilfe frei wählbarer Dämpfungskonstanten α und β proportional zur Steifigkeit und Masse des Tragwerks erfaßt.

$$\mathbf{C} = \alpha\, \mathbf{K} + \beta\, \mathbf{M} \tag{2.44}$$

Die Konstanten α und β werden in Abhängigkeit von Dämpfungsraten ξ_i gewählt, die auf die kritische Dämpfung $(2\, m_i\, \omega_i)$ der entkoppelten Gleichung (2.31) bezogen sind. Mit (2.44) folgt für Gleichung (2.33)

$$c_i = \alpha\, \omega_i^2 + \beta = 2\, \xi_i\, \omega_i \tag{2.45}$$

Gesamtlösung

Verschiebungsverlauf Die Gesamtlösung der Bestimmungsgleichungen (2.21) ergibt sich nach der Ermittlung der generalisierten Verschiebungen mit den Gleichungen (2.31) und Substitution in den Ansatz (2.28). Die Lösung der Gleichungen (2.31) kann durch numerische Auswertung des Duhamel-Integrals (2.46) erfolgen. Für den eingeschwungenen Zustand folgt :

$$g_i(t) = \frac{b_i}{k_i}\frac{\omega_i^2}{\omega_{d_i}}\int_0^t f_i(\tau)\, sin\,\omega_{d_i}(t-\tau)\, e^{-\xi_i\,\omega_i(t-\tau)} d\tau \tag{2.46}$$

$$= u_{s_i}\, D_i(t) \tag{2.47}$$

$$u_{s_i} = \frac{b_i}{k_i} \qquad \text{Statische Verschiebung} \tag{2.48}$$

$$D_i(t) = \frac{\omega_i^2}{\omega_{d_i}}\int_0^t f_i(\tau)\, sin\,\omega_{d_i}(t-\tau)\, e^{-\xi_i\,\omega_i(t-\tau)} d\tau \qquad \text{Dynamischer Lastfaktor} \tag{2.49}$$

Dynamischer Lastfaktor Der dynamische Lastfaktor $D_i(t)$ beinhaltet den Lasttyp (harmonisch, periodisch, impulsförmig etc.) der Erregung. Er variiert mit dem Frequenzverhältnis $\eta_i = \Omega_i/\omega_i$ (Erregerfrequenz/Eigenfrequenz) und der Dämpfungsrate ξ_i. Der Fall $\eta_i = 1.0$ wird als Resonanzfall bezeichnet. Im ungedämpften Fall $(\xi_i = 0.0)$ strebt der dynamische Lastfaktor dann gegen ∞. Im gedämpften Fall $(\xi_i \neq 0.0)$ besitzt der dynamische Lastfaktor für $\eta_i = 1.0$ den Maximalwert $max\, D_i(t)$. Wegen der geringen Dämpfungsrate im Bauwesen (i.a $\xi < 10\%$) ist dieser Fall relevant für die Beurteilung der Schwingungsantwort einer Tragstruktur.

Mit Gleichung (2.47) dominieren Eigenschwingungen mit einem Frequenzverhältnis η nahe Eins und großem Beteiligungsfaktor b_i die Schwingungsantwort (2.28) des Systems. Die maximale Verschiebung $max\, u_m(t)$ an der Stelle m des Systems ergibt sich aus der Überlagerung der maximalen Verschiebungen $max\, u_{m_i}$ der i-ten Eigenschwingung an der Stelle m. Im ungünstigsten Fall treten die maximalen Verschiebungen aller Eigenschwingungen gleichzeitig auf. Die maximale Verschiebung $max\, u_m(t)$ ergibt sich dann nach (2.50).

$$max\, u_m = \sum_{i=1}^{n} u_{m_i} \tag{2.50}$$

Dieser Fall ist unwahrscheinlich. Im allgemeinen treten die maximalen Verschiebungen $max\, u_{m_i}$ zu verschiedenen Zeitpunkten auf. Als hinreichend genaue Überlagerungsformel zur Berücksichtigung dieses Umstands wird häufig die Wurzel der Quadratsumme nach Gleichung (2.51) verwendet [6] :

$$max\, u_m \doteq \sqrt{\sum_{i=1}^{n} u_{m_i}^2} \tag{2.51}$$

Geschwindigkeitsverlauf Durch Differentiation des Integrals (2.46) nach der Zeit t folgt für die generalisierte Geschwindigkeit :

$$\dot{g}_i(t) \;=\; \frac{b_i}{k_i}\,\omega_i^2 \int_0^t f(\tau)\, cos\,\omega_{d_i}(t-\tau)\, e^{-\xi_i\,\omega_i(t-\tau)} d\tau - \xi_i\,\omega_i\, g_i(t) \tag{2.52}$$

Der Geschwindigkeitsvektor $\dot{\mathbf{u}}(t)$ zur Zeit t folgt nach Substitution von (2.52) in (2.28).

Beschleunigungsverlauf Mit dem bekannten Verschiebungs- und Geschwindigkeitsverlauf wird die generalisierte Beschleunigung aus (2.31) ermittelt.

$$\ddot{g}_i(t) + 2\,\xi_i\,\omega_i\,\frac{1}{m_i}\,\dot{g}_i(t) + \omega_i^2\,\frac{1}{m_i}\,g_i(t) \;=\; \frac{1}{m_i}\,p_i(t) \tag{2.53}$$

$$\ddot{g}_i(t) \;=\; \frac{1}{m_i}\,\left(\,p_i(t) \;-\; 2\,\xi_i\,\omega_i\,\dot{g}_i(t) \;-\; \omega_i^2\,g_i(t)\,\right) \tag{2.54}$$

Der Beschleunigungsvektor $\ddot{\mathbf{u}}(t)$ zur Zeit t ermittelt sich analog zu $\dot{\mathbf{u}}(t)$.

Kapitel 3

Inverse Matrixiteration

3.1 Einleitung

Die Inverse Matrixiteration ist ein Verfahren zur teilweisen und vollständigen Bestimmung der Eigenzustände großer, reeller, symmetrischer Profilmatrizen. Das Verfahren ist eine Erweiterung des klassischen QR-Verfahrens nach Francis. Die Erweiterungen dieser Methode umfassen den Erhalt der Profilstruktur und Verfahren zur Verbesserung des Konvergenzverhaltens.

Die Methode bestimmt die Eigenzustände in der Reihenfolge der Beträge ihrer Eigenwerte, beginnend mit dem betragskleinsten Eigenwert. Eine konvexe Profilstruktur der Matrix bleibt während der Iteration erhalten und reduziert dadurch den Speicherbedarf und den numerischen Aufwand der Berechnung. Das Verfahren konvergiert schrittweise und ermöglicht damit den konvergenzbeschleunigenden Einsatz einer Spektralverschiebung in Verbindung mit einer kontinuierlichen Verkleinerung der Aufgabe durch Deflation. Die Spektralverschiebung ermöglicht darüberhinaus die lückenlose Bestimmung von Eigenzuständen in vorgegebenen Bereichen des Gesamteigenwertspektrums. Zusätzliche Prekonditionierungsmethoden fördern die rasche Diagonaldominanz der iterierten Matrix und begünstigen dadurch die Konvergenz der Iteration. Die Methode ist in der Lage mehrfache Eigenwerte und betragsgleiche Eigenwerte verschiedener Vorzeichen zu bestimmen. Als natürliche Eigenschaft des Verfahrens steigt die Effizienz der Iteration mit zunehmender Anzahl der bestimmten Eigenwerte. Für große Matrizen entspricht der numerische Aufwand pro berechnetem Eigenwert etwa dem Aufwand einer halben QR-Zerlegung. Der Profilerhalt ermöglicht eine unabhängige Bestimmung der Eigenvektoren mit Eigenwerten beliebiger Multiplizität.

Das entwickelte und implementierte Verfahren wird zur klaren Abgrenzung vom klassischen QR-Verfahren und Givens-QRI-Verfahren als *Inverse Matrixiteration* bezeichnet. Die Namensgebung trägt der engen Verwandtschaft, hinsichtlich der Konvergenzeigenschaften, zur Inversen Vektoriteration Rechnung.

3.2 Eigenwertaufgabe

3.2.1 Aufgabenstellung

Gegeben sei die allgemeine Eigenwertaufgabe (3.2.1) mit quadratischen Koeffizientenmatrizen $\mathbf{A}$ und $\mathbf{B}$ der Dimension N. Gesucht werden die nichttrivialen Lösungen $(\lambda_i, \mathbf{x}_i)$ mit $\mathbf{x}_i \neq \mathbf{0}$.

$$\mathbf{A}\,\mathbf{x}_i \;=\; \lambda_i\,\mathbf{B}\,\mathbf{x}_i \tag{3.2.1}$$

Gibt es wenigstens einen Vektor $\mathbf{x}_i \in \mathbb{C}, \mathbf{x} \neq \mathbf{0}$ der mit λ_i die nichtlineare Gleichung (3.2.1) erfüllt, so ist λ_i ein Eigenwert des Matrixpaars $(\mathbf{A}, \mathbf{B})$. Die Menge aller Eigenwerte bildet das Spektrum von $(\mathbf{A}, \mathbf{B})$ und wird mit $\sigma(\mathbf{A}, \mathbf{B})$ bezeichnet. Die Vektoren $\mathbf{x}_i$ heißen Eigenvektoren des Matrixpaars $(\mathbf{A}, \mathbf{B})$. Die Lösung $(\lambda_i, \mathbf{x}_i)$ wird als i-ter Eigenzustand der Aufgabe (3.2.1) bezeichnet.

Ist die Koeffizientenmatrix $\mathbf{B}$ gleich der Identitätsmatrix $\mathbf{I}$, so stellt dies ein Sonderfall der allgemeinen Eigenwertaufgabe dar. Die Gleichung (3.2.1) geht für diesen Fall in das spezielle Eigenwertproblem (3.2.2) über.

$$\mathbf{A}\,\mathbf{x}_i \;=\; \lambda_i\;\mathbf{x}_i \tag{3.2.2}$$

Ist eine der Matrizen $\mathbf{A}, \mathbf{B}$ regulär, kann die Aufgabe (3.2.1) in eine spezielle Eigenwertaufgabe überführt werden. Existiert beispielsweise $\mathbf{B}^{-1}$ und damit die Gauss-Zerlegung $\mathbf{B} = \mathbf{L}\,\mathbf{R}$, folgt für (3.2.1) nach Transformation die spezielle Eigenwertaufgabe (3.2.4).

$$\mathbf{A}\,\mathbf{x}_i \;=\; \lambda_i\,\mathbf{L}\,\mathbf{R}\,\mathbf{x}_i \tag{3.2.3}$$

$$\mathbf{C}\,\mathbf{y}_i \;=\; \lambda_i\,\mathbf{y}_i \tag{3.2.4}$$

$$\mathbf{C} \;:=\; \mathbf{L}^{-1}\mathbf{A}\,\mathbf{R}^{-1} \tag{3.2.5}$$

$$\mathbf{y}_i \;:=\; \mathbf{R}\mathbf{x}_i \tag{3.2.6}$$

Der Aufwand für die Lösung der Aufgabe (3.2.2) ist wesentlich geringer als für die Lösung der allgemeinen Eigenwertaufgabe.

3.2.2 Mathematische Grundlagen

Charakteristisches Polynom

Zur Lösung der speziellen Eigenwertaufgabe wird das Gleichungssystem (3.2.2) in der homogenen Form dargestellt. Das Gleichungssystem (3.2.7) besitzt nichttriviale Lösungen $(\lambda_i, \mathbf{x}_i), \mathbf{x}_i \neq$

$\mathbf{0}$, wenn $(\mathbf{A} - \lambda\mathbf{I})$ singulär ist.

$$(\mathbf{A} - \lambda_i\mathbf{I})\,\mathbf{x}_i = \mathbf{0} \tag{3.2.7}$$

$$det(\mathbf{A} - \lambda_i\mathbf{I}) = 0 \qquad \text{mit } \mathbf{x}_i \neq \mathbf{0} \tag{3.2.8}$$

Gleichung (3.2.8) kann als Funktion $p(\lambda)$ der Unbekannten λ dargestellt werden. Die Funktion $p(\lambda)$ ist ein Polynom n-ten Grades. Sie wird als *charakteristisches Polynom* der Eigenwertaufgabe bezeichnet. Die Nullstellen des Polynoms sind die Eigenwerte von $\mathbf{A}$.

$$p(\lambda) := det(\mathbf{A} - \lambda_i\mathbf{I}) = (-1)^n\,(\lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_0) \tag{3.2.9}$$

$$= (-1)^n\,(\lambda - \lambda_1)^{\mu_1} \ldots (\lambda - \lambda_m)^{\mu_m} \tag{3.2.10}$$

Der Exponent μ_k kennzeichnet hierbei die algebraische Multiplizität[1] des k-ten Eigenwerts λ_k.

Eigenzustände reeller symmetrischer Matrizen

Mit Gleichung (3.2.9) besitzt die quadratische Matrix $\mathbf{A} \in \mathbb{C}^{(N\times N)}$ genau N Eigenwerte $\lambda_i \in \mathbb{C}, (i = 1, \ldots, N)$. Im allgemeinen bilden die N Eigenvektoren $\mathbf{x}_i \in \mathbb{C}$ von $\mathbf{A}$ keine vollständige Basis in $\mathbb{C}^{(N\times N)}$. Die Eigenvektoren sind nicht vollständig linear unabhängig. Erfüllt $\mathbf{A}$ die Eigenschaft (3.2.11), so heißt $\mathbf{A}$ normal und besitzt N linear unabhängige Eigenvektoren $\mathbf{x}_i \in \mathbb{C}^{(N\times N)}$ [42].

$$\mathbf{A}\,\mathbf{A}^H = \mathbf{A}^H\,\mathbf{A} \qquad \mathbf{A} \text{ ist normal} \tag{3.2.11}$$

Symmetrische Matrizen $\mathbf{M}$ erfüllen die Eigenschaft (3.2.11) und besitzen damit N linear unabhängige Eigenvektoren $\mathbf{u}_i, (i = 1, \ldots, N)$. Die Eigenvektoren $\mathbf{u}_i$ bilden eine unitäre Transformationsbasis $\mathbf{U}$, die $\mathbf{M}$ diagonalisiert [35].

$$\mathbf{U}^H\mathbf{M}\mathbf{U} = \mathbf{D} \qquad \mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I} \;\wedge\; \mathbf{D} = diag[d_{ii}] \tag{3.2.12}$$

Erfüllt $\mathbf{M}$ die Eigenschaft (3.2.13), so heißt $\mathbf{M}$ hermitesch. Die Diagonalkoeffizienten hermitescher Matrizen sind reell [42]. Wegen der Invarianz der Spur $sp(\mathbf{M})$ in (3.2.12), sind die Diagonalkoeffizienten von $\mathbf{D}$ und damit die Eigenwerte von $\mathbf{M}$ reell.

$$\mathbf{A}^H = \mathbf{A} \qquad \mathbf{A} \text{ ist hermitesch} \tag{3.2.13}$$

Besitzt $\mathbf{M}$ reelle Koeffizienten, so sind auch die Koeffizienten in $(\mathbf{M} - \lambda_i\mathbf{I})$ reell. Mit der homogenen Gleichung (3.2.7) folgt, daß damit auch die Koeffizienten der Eigenvektoren $\mathbf{u}_i$ reell sind.

Reelle symmetrische Matrizen der Dimension N sind hermitesch und damit normal. Sie besitzen N reelle Eigenzustände $(\lambda_i, \mathbf{x}_i) \in \mathbb{R}, (i = 1, \ldots, N)$. Die N Eigenvektoren $\mathbf{x}_i$ sind linear unabhängig und bilden eine orthonormale Basis im Raum $\mathbb{R}^{(N\times N)}$.

[1] algebraische Multiplizität = algebraische Vielfachheit : Anzahl gleicher Eigenwerte

Ähnlichkeitstransformation

$\mathbf{A}$ und $\mathbf{B}$ seien quadratische Matrizen der Dimension N. $\mathbf{A}$ und $\mathbf{B}$ sind ähnlich zueinander, falls sie dasselbe Eigenwertspektrum besitzen, $\sigma(\mathbf{A}) = \sigma(\mathbf{B})$. Sind $\mathbf{A}$ und $\mathbf{B}$ ähnlich zueinander, so existiert eine reguläre Matrix $\mathbf{P}$, die $\mathbf{A}$ durch die Transformation (3.2.14) in $\mathbf{B}$ überführt.

$$\mathbf{B} \;=\; \mathbf{P}^{-1}\,\mathbf{A}\,\mathbf{P} \tag{3.2.14}$$

Gleichung (3.2.14) heißt *Ähnlichkeitstransformation*.

Der Beweis der Ähnlichkeit von $\mathbf{A}$ und $\mathbf{B}$ wird durch Vergleich ihres charakteristischen Polynoms geführt :

$$\begin{aligned} det(\mathbf{B}-\lambda\mathbf{I}) &= det(\mathbf{P}^{-1}\mathbf{A}\,\mathbf{P} - \lambda\,\mathbf{P}^{-1}\mathbf{P}) && (3.2.15)\\ &= det(\mathbf{P}^{-1}(\mathbf{A}-\lambda\mathbf{I})\mathbf{P}) && (3.2.16)\\ &= det(\mathbf{P}^{-1})\,det(\mathbf{A}-\lambda\mathbf{I})\,det(\mathbf{P}) && (3.2.17)\\ &= det(\mathbf{A}-\lambda\mathbf{I}) && (3.2.18) \end{aligned}$$

□

Eigendarstellung

Die N Lösungen der Aufgabe (3.2.2) werden als Gleichungssystem dargestellt. Die Eigenwerte λ_i von $\mathbf{A}$ werden dabei in der Diagonalmatrix $\mathbf{\Lambda}$ angeordnet. Die zugeordneten Eigenvektoren $\mathbf{x}_i$ werden in gleicher Reihenfolge spaltenweise in der Eigenmatrix $\mathbf{X}$ angeordnet.

$$\mathbf{A}\,\mathbf{X} \;=\; \mathbf{X}\,\mathbf{\Lambda} \tag{3.2.19}$$

$$\mathbf{\Lambda} \;=\; \begin{pmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \lambda_3 & & \\ & & & \ddots & \\ & & & & \lambda_n \end{pmatrix} \qquad \mathbf{X} \;=\; \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_n \end{pmatrix}$$

Bild 3.1: Eigenwertmatrix $\mathbf{\Lambda}$ und Eigenmatrix $\mathbf{X}$

Wegen der Orthonormalität der Eigenvektoren $\mathbf{x}_i$ ist auch die Eigenmatrix $\mathbf{X}$ orthonormal. Aus (3.2.19) folgt damit die Eigendarstellung von $\mathbf{A}$:

$$\mathbf{A} \;=\; \mathbf{X}\,\mathbf{\Lambda}\,\mathbf{X}^T \qquad\qquad \text{mit} \quad \mathbf{X}\,\mathbf{X}^T \;=\; \mathbf{X}^T\mathbf{X} \;=\; \mathbf{I} \tag{3.2.20}$$

Mit der Eigendarstellung (3.2.20) ist $\mathbf{A}$ als Summe der N Eigenzustände darstellbar :

$$\mathbf{A} \;=\; \lambda_1\mathbf{x}_1\,\mathbf{x}_1^T + \ldots + \lambda_n\mathbf{x}_n\,\mathbf{x}_n^T \tag{3.2.21}$$

3.3 QR-Verfahren

3.3.1 Lösungsansatz

Die spezielle Eigenwertaufgabe (3.3.1) wird durch ein Reihe von Ähnlichkeitstransformationen in die Diagonalform (3.3.2) überführt. Die Diagonalkoeffizienten von $\mathbf{D}$ sind die Eigenwerte der Aufgabe (3.3.1). Die Eigenvektoren der Gleichung (3.3.2) sind die Einheitsvektoren $\mathbf{e}_i$. Die Eigenvektoren von $\mathbf{A}$ sind spaltenweise in der Transformationsmatrix $\mathbf{Q}$ angeordnet.

$$\mathbf{A}\,\mathbf{x}_i \;=\; \lambda_i\;\mathbf{x}_i \tag{3.3.1}$$

$$\mathbf{D}\,\mathbf{e}_i \;=\; \lambda_i\;\mathbf{e}_i \tag{3.3.2}$$

$$\mathbf{D} \;:=\; \mathbf{Q}^T\,\mathbf{A}\,\mathbf{Q} \qquad \text{Diagonalmatrix mit den Eigenwerten von } \mathbf{A} \tag{3.3.3}$$

$$\mathbf{x}_i \;:=\; \mathbf{Q}\,\mathbf{e}_i \qquad \text{Eigenvektor zum Eigenwert } \lambda_i \tag{3.3.4}$$

Die Transformationsmatrix $\mathbf{Q}$ wird schrittweise bestimmt. In jedem Schritt s wird die Matrix $\mathbf{A}$ in eine orthonormale Matrix $\mathbf{Q}_s$ und eine Rechtsdreiecksmatrix $\mathbf{R}_s$ zerlegt. Die iterierte Matrix $\mathbf{A}_{s+1}$ wird invers aus dem Zerlegungsprodukt berechnet (3.3.5).

$$\mathbf{Q}_s\,\mathbf{R}_s \;:=\; \mathbf{A}_s \qquad \text{mit } \mathbf{Q}_s\,\mathbf{Q}_s^T \;=\; \mathbf{Q}_s^T\,\mathbf{Q}_s \;=\; \mathbf{I} \tag{3.3.5}$$

$$\mathbf{A}_{s+1} \;=\; \mathbf{R}_s\,\mathbf{Q}_s \;=\; \mathbf{Q}_s^T\,\mathbf{A}_s\,\mathbf{Q}_s \tag{3.3.6}$$

Die iterierte Matrix $\mathbf{A}_{s+1}$ konvergiert mit fortschreitender Iteration mit den Gleichungen (3.3.5) und (3.3.6) zur Diagonalform (3.3.3). Die Diagonalmatrix $\mathbf{D}$ enthält die Eigenwerte von $\mathbf{A}$ in absteigender Reihenfolge ihrer Beträge.

$$|d_1| \geq |d_2| \geq \ldots \geq |d_{k-1}| \geq |d_k| \geq |d_{k+1}| \geq \ldots \geq |d_n| \tag{3.3.7}$$

QR-Zerlegung

Im Schritt s der Iteration wird die Matrix $\mathbf{A}_s$ in das Produkt einer orthonormalen Matrix $\mathbf{Q}_s$ und einer Rechtsdreiecksmatrix $\mathbf{R}_s$ zerlegt. Die Zerlegung erfolgt durch die schrittweise Reduktion von $\mathbf{A}_s$ auf Dreiecksform mit Hilfe ebener Rotationsmatrizen $\mathbf{P}_{ik}$. Dabei werden die Koeffizienten a_{ik} unterhalb der Hauptdiagonalen von $\mathbf{A}_s$ durch Multiplikation mit $\mathbf{P}_{ik}^T$ von links spaltenweise auf Null getrieben, beginnend jeweils mit dem Subdiagonalelement $a_{i+1,i}$. Die erzeugten Nullelemente bleiben bei den nachfolgenden Rotationen erhalten.

$$\mathbf{R}_s \;=\; \mathbf{P}_{n,n-1}^T \ldots \mathbf{P}_{ik}^T \ldots \mathbf{P}_{32}^T\,\mathbf{P}_{n1}^T \ldots \mathbf{P}_{31}^T\,\mathbf{P}_{21}^T\,\mathbf{A}_s \;=\; \mathbf{Q}_s^T\mathbf{A}_s \tag{3.3.8}$$

$$\mathbf{Q}_s \;=\; \mathbf{P}_{21}\,\mathbf{P}_{31} \ldots \mathbf{P}_{n1}\,\mathbf{P}_{32} \ldots \mathbf{P}_{ik} \ldots \mathbf{P}_{n,n-1} \tag{3.3.9}$$

Die Multiplikation der Matrix $\mathbf{A}_s$ mit der Rotationsmatrix $\mathbf{P}_{ik}^T$ von links ändert nur die Koeffizienten a_{ik} in den Zeilen i und k.

$$\mathbf{P}_{ik}^T\,\mathbf{A}_s \;=\; \hat{\mathbf{A}}_s \qquad \text{mit } \hat{a}_{ik} \;=\; 0, \quad i > k \tag{3.3.10}$$

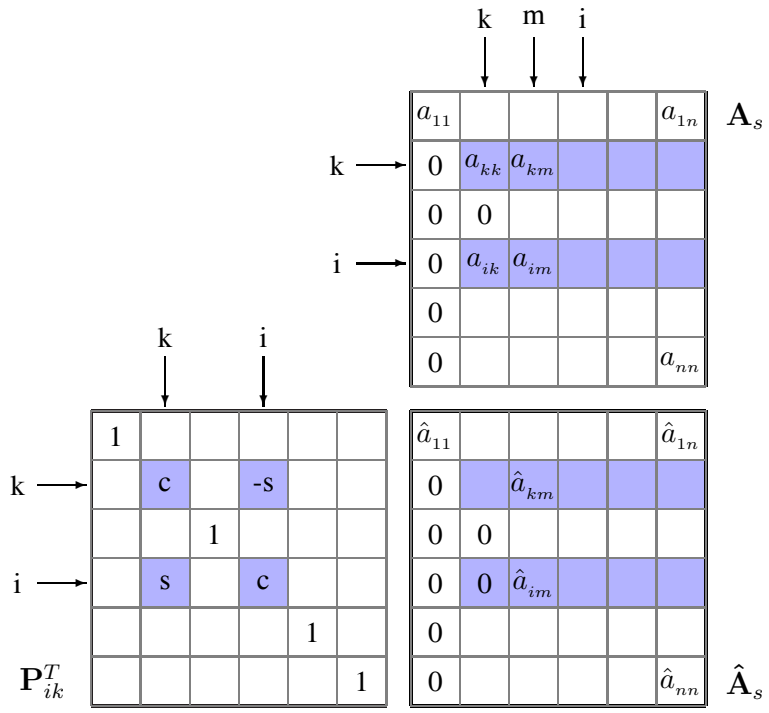


Bild 3.2: QR-Zerlegung : Linksrotation

$$\hat{a}_{km} = a_{km}\, cos(\theta) - a_{im}\, sin(\theta) \qquad m = k, \ldots, n \tag{3.3.11}$$

$$\hat{a}_{im} = a_{km}\, sin(\theta) + a_{im}\, cos(\theta) \tag{3.3.12}$$

RQ-Rekombination

Zur Berechnung der iterierten Matrix $\mathbf{A}_{s+1}$ wird die Ähnlichkeitstransformation durch Multiplikation des Zerlegungsproduktes $\mathbf{Q}_s\mathbf{R}_s$ in umgekehrter Reihenfolge vervollständigt. Die Berechnung von $\mathbf{A}_{s+1}$ beginnt mit der Rechtsdreiecksmatrix $\mathbf{R}_s$. Die Multiplikation mit $\mathbf{P}_{21}$ zerstört nur das Nullelement an Position $(2,1)$. Allgemein zerstört die Multiplikation mit $\mathbf{P}_{ik}$ nur das Nullelement an Position (i,k). Alle anderen Nullelemente bleiben erhalten.

$$\mathbf{A}_{s+1} = \mathbf{R}_s\mathbf{Q}_s = \mathbf{R}_s\, \mathbf{P}_{21}\, \mathbf{P}_{31} \ldots \mathbf{P}_{n1}\, \mathbf{P}_{32} \ldots \mathbf{P}_{ik} \ldots \mathbf{P}_{n,n-1} \tag{3.3.13}$$

Die Multiplikation der Matrix $\mathbf{A}_s$ mit der Rotationsmatrix $\mathbf{P}_{ik}$ von rechts ändert nur die Koeffizienten a_{ik} in den Spalten i und k. Die Reihenfolge der Multiplikationen in (3.3.13) entspricht der Reihenfolge der Multiplikationen in (3.3.8), spaltenweise, beginnend mit dem Subdiagonalelement $a_{i+1,i}$.

$$\check{a}_{mk} = \hat{a}_{mk}\, cos(\theta) - \hat{a}_{mi}\, sin(\theta) \qquad m = k, \ldots, n \tag{3.3.14}$$

$$\check{a}_{mi} = \hat{a}_{mk}\, sin(\theta) + \hat{a}_{mi}\, cos(\theta) \tag{3.3.15}$$

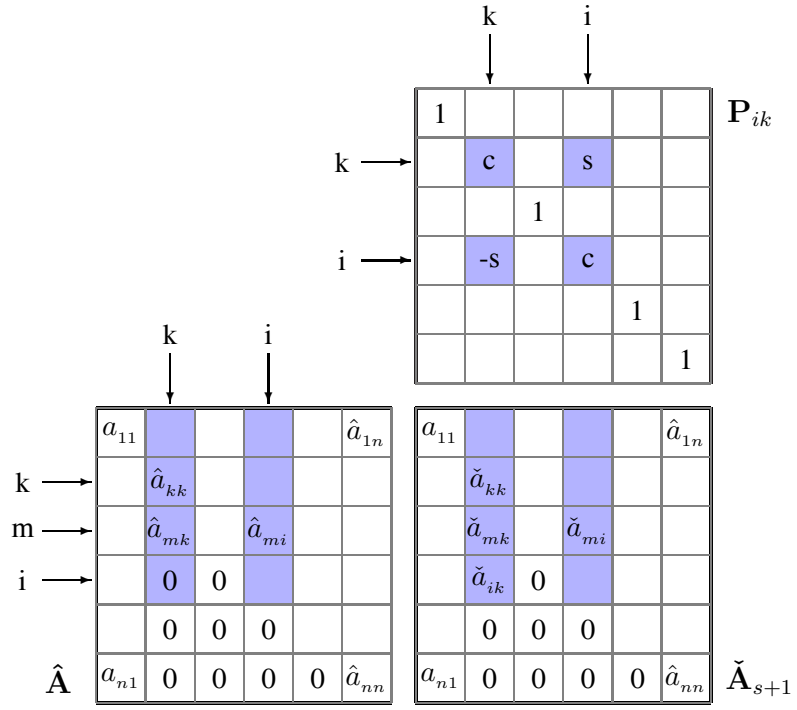


Bild 3.3: RQ-Rekombination : Rechtsrotation

Konstruktion von Q

Die Transformationsmatrix $\mathbf{Q}$ aus (3.3.3) wird aus dem Produkt der Matrizen $\mathbf{Q}_s$ der Ähnlichkeitstransformationen in den Schritten $s = 1, 2, \ldots$ bestimmt. Sie wird somit aus allen Rotationsmatrizen $\mathbf{P}_{ik}$ der Zerlegung (3.3.8) in den Schritten $s = 1, 2, \ldots$ gebildet. Die Koeffizienten von $\mathbf{Q}$ sind in Analogie zu (3.3.9) mit den Formeln (3.3.14) und (3.3.15) bestimmbar.

$$\mathbf{Q} \;=\; \lim_{s\to\infty} \mathbf{Q}_s \;=\; \prod_{i=1}^{s-1} \mathbf{Q}_i \cdot \mathbf{P}_{21}\,\mathbf{P}_{31}\ldots\mathbf{P}_{n1}\,\mathbf{P}_{32}\ldots\mathbf{P}_{ik}\ldots\mathbf{P}_{n,n-1} \tag{3.3.16}$$

Wahl des Rotationswinkels

Der Rotationswinkel θ der Transformation wird durch die Bestimmungsgleichung (3.3.12) für das Nichtdiagonalelement $\hat{a}_{ik} = 0.0$ festgelegt (Bild 3.2).

Für $a_{kk} = 0.0$ wird der Rotationswinkel so gewählt, daß die Transformationsmatrix $\mathbf{P}_{ik}$ einen Tausch der Zeilen und Spalten i und k bewirkt.

$$a_{kk} \;\neq\; 0.0 \;:\qquad \theta \;=\; atan\left(-\frac{a_{ik}}{a_{kk}}\right) \tag{3.3.17}$$

$$a_{kk} \;=\; 0.0 \;:\qquad \theta \;=\; \frac{\pi}{2} \tag{3.3.18}$$

Beispiel 1 : QR-Verfahren

Gegeben sei die reelle, symmetrische Matrix $\mathbf{A}$ mit den Eigenwerten $\lambda_1 = 0.0$, $\lambda_2 = 1.0$ und $\lambda_3 = 3.0$. Mit dem QR-Verfahren werden Näherungslösungen zu den Eigenwerten und Eigenvektoren von $\mathbf{A}$ bestimmt.

$$\begin{array}{|ccc|} \hline 1.0 & -1.0 & 0.0 \\ -1.0 & 2.0 & -1.0 \\ 0.0 & -1.0 & 1.0 \\ \hline \end{array} \quad \mathbf{A}$$

$$\mathbf{P}_{21}^T \begin{array}{|ccc|} \hline \frac{1.}{\sqrt{2.}} & -\frac{1.}{\sqrt{2.}} & 0.0 \\ \frac{1.}{\sqrt{2.}} & \frac{1.}{\sqrt{2.}} & 0.0 \\ 0.0 & 0.0 & 1.0 \\ \hline \end{array} \quad \begin{array}{|ccc|} \hline \sqrt{2.} & -\frac{3.}{\sqrt{2.}} & \frac{1.}{\sqrt{2.}} \\ 0.0 & \frac{1.}{\sqrt{2.}} & -\frac{1.}{\sqrt{2.}} \\ 0.0 & -1.0 & 1.0 \\ \hline \end{array} \quad \mathbf{P}_{21}^T \mathbf{A}$$

$$\mathbf{P}_{32}^T \begin{array}{|ccc|} \hline 1.0 & 0.0 & 0.0 \\ 0.0 & \frac{1.}{\sqrt{3.}} & -\frac{\sqrt{2.}}{\sqrt{3.}} \\ 0.0 & \frac{\sqrt{2.}}{\sqrt{3.}} & \frac{1.}{\sqrt{3.}} \\ \hline \end{array} \quad \begin{array}{|ccc|} \hline \sqrt{2.} & -\frac{3.}{\sqrt{2.}} & \frac{1.}{\sqrt{2.}} \\ 0.0 & \frac{3.}{\sqrt{2.}} & -\frac{3.}{\sqrt{2.}} \\ 0.0 & 0.0 & 0.0 \\ \hline \end{array} \quad \mathbf{P}_{32}^T \mathbf{P}_{21}^T \mathbf{A} = \mathbf{R}_1$$

Bild 3.4: Zerlegung $\mathbf{A} = \mathbf{Q}_1\mathbf{R}_1$ im Zyklus 1

Der erste Eigenzustand $(\lambda_1, \mathbf{x}_1)$ ist nach dem ersten Iterationszyklus bestimmt (Bild 3.5). Zur Bestimmung der Eigenvektoren wird die orthonormale Transformationsmatrix $\mathbf{Q}_1$ aus dem Produkt der Rotationsmatrizen $\mathbf{P}_{21}$ und $\mathbf{P}_{32}$ explizit gebildet (Bild 3.6).

Nach vier weiteren Zerlegungen liefert das Verfahren folgende Näherungslösungen zu den Eigenwerten und Eigenvektoren :

$\hat{\lambda}_1 = 0.000000$	$\hat{\lambda}_2 = 1.000102$	$\hat{\lambda}_3 = 2.999898$
0.577350	−0.709999	0.403198
0.577350	0.005820	−0.816476
0.577350	0.704179	0.413278
$\hat{\mathbf{x}}_1$	$\hat{\mathbf{x}}_2$	$\hat{\mathbf{x}}_3$

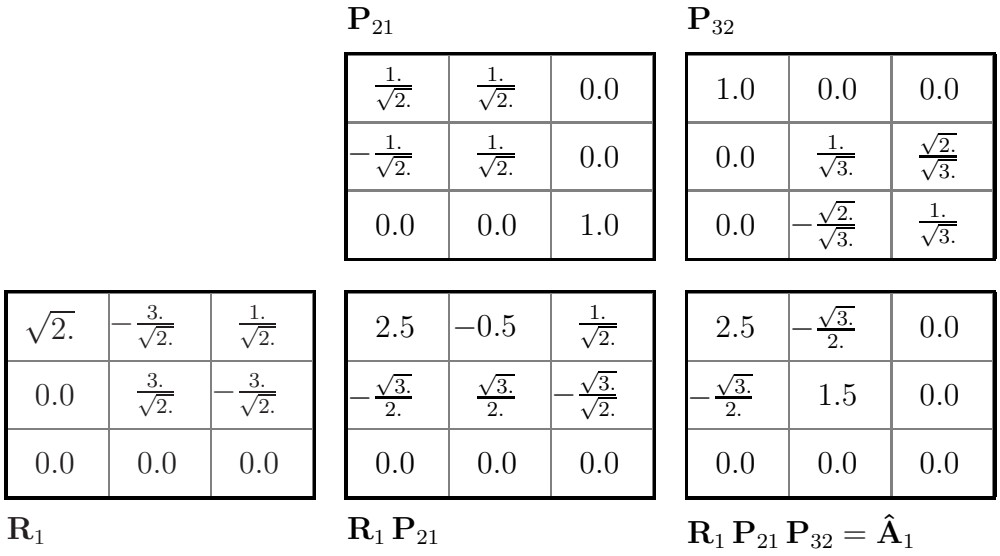


Bild 3.5: Rekombination $\hat{\mathbf{A}}_1 = \mathbf{R}_1\mathbf{Q}_1$ im Zyklus 1

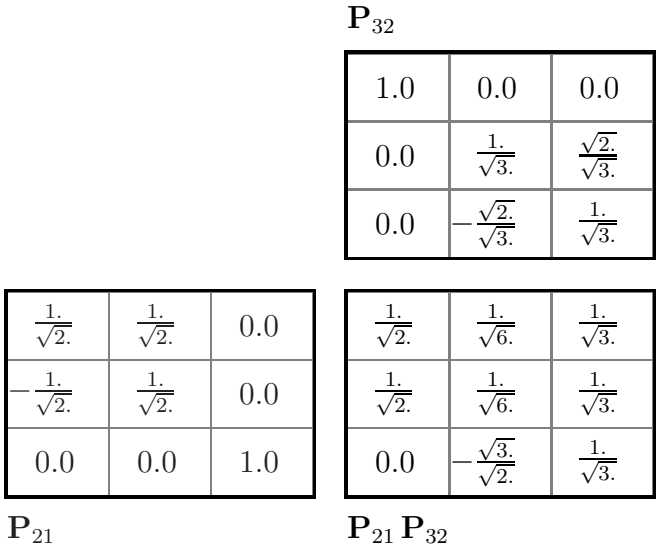


Bild 3.6: Konstruktion von $\mathbf{Q}_1$

3.3.2 Konvergenz der Iteration

Um Aufschluß über das Konvergenzverhalten und damit verbundene Problemstellungen bei der Entwicklung eines Algorithmus für die in 3.3.1 beschriebene Methode zu erhalten, wird im folgenden ein vollständiger Konvergenzbeweis vorgenommen. Die Beweisführung erfolgt unter der Annahme reeller Koeffizienten und Eigenwerte von $\mathbf{A}$, in Anlehnung an [24],[28] und [48]. Der Konvergenzbeweis wird unter Berücksichtigung verschiedener Sonderfälle in mehreren Schritten vollzogen :

- *Getrennte Eigenwerte :* Im ersten Schritt der Beweisführung wird das Konvergenzverhalten der Iteration für getrennte Eigenwerte von $\mathbf{A}$ aufgezeigt. Es wird bewiesen, daß die QR-Zerlegung einer beliebigen quadratischen, reellen Matrix nicht eindeutig ist und es somit für die Untersuchung des Konvergenzverhaltens einer allgemeineren Form der Zerlegung bedarf. Mit dieser Zerlegung wird der Zusammenhang zwischen der iterierten Matrix $\mathbf{A}_s$ im Schritt s und der Ausgangsmatrix $\mathbf{A}$ im Schritt 1 der Iteration aufgezeigt. Dies führt zu einer Zerlegung der s-ten Potenz von $\mathbf{A}$ im Schritt s mit der schließlich das Konvergenzverhalten der Iteration in Abhängigkeit der aufeinanderfolgenden Eigenwerte der Aufgabe dargestellt werden kann.

- *Zerlegung der Eigenmatrix :* Voraussetzung für den Konvergenzbeweis ist u.a. die Existenz einer Zerlegung der Eigenmatrix in das Produkt einer Linksdreiecksmatrix $\mathbf{L}$ mit Diagonalkoeffizienten $l_{ii} = 1.0$ und einer Rechtsdreiecksmatrix $\mathbf{U}$. Der Beweis dafür schließt direkt an.

- *Nulleigenwerte :* Im zweiten Schritt der Beweisführung wird der Sonderfall 'Nulleigenwerte' untersucht. Mit diesem Beweis wird die Bestimmung der Nulleigenwerte mit nur einer QR-Zerlegung gezeigt. Dieser Teil ist eine der Grundlagen für die Erweiterungen des QR-Verfahrens im nachfolgenden Kapitel.

- *Betragsgleiche Eigenwerte :* Im dritten Schritt der Beweisführung wird der Sonderfall 'Betragsgleiche Eigenwerte' untersucht. Die Konvergenz der Iteration wird dabei für betragsgleiche Eigenwerte mit gleichen Vorzeichen und betragsgleiche Eigenwerte mit verschiedenen Vorzeichen analysiert und nachgewiesen.

Getrennte Eigenwerte $|\lambda_i| \neq |\lambda_k| \neq 0$

Satz 3.3.1 *Sei* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *eine reelle symmetrische Matrix mit den beiden QR-Zerlegungen* $\mathbf{A} = \mathbf{N}_1\mathbf{C}_1$ *und* $\mathbf{A} = \mathbf{N}_2\mathbf{C}_2$. *Die Matrizen* $\mathbf{N}_1$ *und* $\mathbf{N}_2$ *sind orthonormal, die Matrizen* $\mathbf{C}_1$ *und* $\mathbf{C}_2$ *besitzen Rechtsdreiecksgestalt. Ist* $\mathbf{N}_1 \neq \mathbf{N}_2$ *und* $\mathbf{C}_1 \neq \mathbf{C}_2$, *dann ist die QR-Zerlegung von* $\mathbf{A}$ *nicht eindeutig.*

Beweis : Wegen $|\lambda_i| \neq |\lambda_k| \neq 0$ sind die Matrizen $\mathbf{C}_1$ und $\mathbf{C}_2$ nicht singulär und besitzen eine Inverse. $\mathbf{C}_1^{-1}$ ist eine Rechtsdreiecksmatrix, ebenso das Produkt der Rechtsdreiecksmatrizen $\mathbf{C}_2$ und $\mathbf{C}_1^{-1}$.

$$\mathbf{A} \;=\; \mathbf{N}_1\mathbf{C}_1 \;=\; \mathbf{N}_2\mathbf{C}_2 \qquad\qquad \text{mit}\;\; \mathbf{N}_i^T\mathbf{N}_i \;=\; \mathbf{I} \tag{3.3.19}$$

$$\mathbf{N}_2^T\mathbf{N}_1 \;=\; \mathbf{C}_2\mathbf{C}_1^{-1} \;:=\; \mathbf{E} \tag{3.3.20}$$

Das Produkt der orthonormalen Matrizen $\mathbf{N}_1$ und $\mathbf{N}_2$ ergibt eine orthonormale Matrix $\mathbf{E}$. Da das Produkt der Matrizen $\mathbf{C}_2$ und $\mathbf{C}_1^{-1}$ ebenfalls orthonormal ist folgt, daß $\mathbf{E}$ eine Diagonalmatrix mit beliebigen Koeffizienten $e_{ii} = +1$ bzw. $e_{ii} = -1$ ist.

$$\mathbf{E}^T\mathbf{E} \;=\; (\mathbf{N}_2^T\mathbf{N}_1)^T(\mathbf{N}_2^T\mathbf{N}_1) \;=\; \mathbf{I} \;:\qquad e_{ii} \;=\; \pm 1 \tag{3.3.21}$$

$$\mathbf{A} \;=\; \mathbf{N}_1\mathbf{C}_1 \;=\; (\mathbf{N}_2\,\mathbf{E})(\mathbf{E}\,\mathbf{C}_2) \tag{3.3.22}$$

Die QR-Zerlegung von $\mathbf{A}$ ist somit wegen $\mathbf{E}$ nicht eindeutig. □

Satz 3.3.1 zeigt, daß in der QR-Zerlegung einer Matrix das Vorzeichen der Spalten von $\mathbf{Q}$ und der entsprechenden Zeilen von $\mathbf{R}$ wegen der Phasenmatrix $\mathbf{E}$ beliebig ist. Das Vorzeichen der Diagonalkoeffizienten von $\mathbf{R}$ kann damit beliebig gewählt werden. Die nachfolgende Beweisführung zeigt, daß das Ergebnis der Iteration durch die Phasenmatrix $\mathbf{E}$ nicht beeinflußt wird.

Im folgenden wird die Zerlegung (3.3.5) durch ihre allgemeinere Form (3.3.22) ersetzt. Im Schritt s der Iteration wird das Produkt $\mathbf{P}_s$ der Orthonormalmatrizen $\mathbf{Q}_s\mathbf{E}_s$ sowie das Produkt $\mathbf{U}_s$ der Rechtsdreiecksmatrizen $\mathbf{E}_s\mathbf{R}_s$ gebildet.

$$\mathbf{A}_s \;=\; (\mathbf{Q}_s\mathbf{E}_s)(\mathbf{E}_s\mathbf{R}_s) \tag{3.3.23}$$

$$\mathbf{P}_s \;=\; \mathbf{Q}_1\mathbf{E}_1 \ldots \mathbf{Q}_s\mathbf{E}_s \tag{3.3.24}$$

$$\mathbf{U}_s \;=\; \mathbf{E}_s\mathbf{R}_s \ldots \mathbf{E}_1\mathbf{R}_1 \tag{3.3.25}$$

Die Orthonormalmatrix $\mathbf{P}_s$ transformiert die gegebene Matrix $\mathbf{A}$ im Schritt s in die iterierte Matrix $\mathbf{A}_{s+1}$. Mit dieser Beziehung wird der Zusammenhang zwischen den Matrizen $\mathbf{A}$ im Schritt 1 und der Matrix $\mathbf{A}_s$ im Schritt s aufgezeigt.

Im Schritt s der Iteration ergibt das Produkt der Matrizen $\mathbf{P}_s$(3.3.24) und $\mathbf{U}_s$(3.3.25) die s-te Potenz der Matrix $\mathbf{A}$.

$$\mathbf{A}_{s+1} = \mathbf{E}_s\mathbf{R}_s\mathbf{Q}_s\mathbf{E}_s = \mathbf{E}_s\mathbf{Q}_s^T\mathbf{A}_s\mathbf{Q}_s\mathbf{E}_s \tag{3.3.26}$$

$$\mathbf{A}_{s+1} = \mathbf{E}_s\mathbf{Q}_s^T \ldots \mathbf{E}_1\mathbf{Q}_1^T\,\mathbf{A}\,\mathbf{Q}_1\mathbf{E}_1 \ldots \mathbf{Q}_s\mathbf{E}_s = \mathbf{P}_s^T\mathbf{A}\mathbf{P}_s \tag{3.3.27}$$

Mit den Gleichungen (3.3.23) bis (3.3.25) und Gleichung (3.3.27) läßt sich das Produkt $\mathbf{P}_s\mathbf{U}_s$ wie folgt darstellen :

$$\mathbf{P}_s\mathbf{U}_s = \mathbf{P}_{s-1}\mathbf{A}_s\mathbf{U}_{s-1} = \mathbf{A}\,\mathbf{P}_{s-1}\mathbf{U}_{s-1} = \mathbf{A}^2\,\mathbf{P}_{s-2}\mathbf{U}_{s-2} = \ldots \tag{3.3.28}$$

$$\mathbf{P}_s\mathbf{U}_s = \mathbf{A}^s \tag{3.3.29}$$

Im folgenden wird die Konvergenz der iterierten Matrix $\mathbf{A}_s$ zur Diagonalform aufgezeigt. Dafür wird der Zusammenhang zwischen der Zerlegung (3.3.29) im Schritt s und den Eigenzuständen von $\mathbf{A}$ untersucht.

Die reelle symmetrische Matrix $\mathbf{A}^s$ besitzt n reelle Eigenwerte λ_i und n linear unabhängige, reelle Eigenvektoren $\mathbf{x}_i$. Die Eigenwerte λ_i werden in beliebiger Reihenfolge in der Diagonalmatrix $\mathbf{\Lambda}$ angeordnet. Die Eigenvektoren werden in gleicher Reihenfolge in der Eigenmatrix $\mathbf{X}$ angeordnet. Dann kann im Schritt s der Iteration $\mathbf{A}^s$ in der Form (3.3.30) zerlegt werden.

$$\mathbf{A}^s = \mathbf{X}\,\mathbf{E}\,\mathbf{\Lambda}^s\,\mathbf{E}\,\mathbf{X}^T \qquad \text{mit } \mathbf{X}^T\mathbf{X} = \mathbf{X}\,\mathbf{X}^T = \mathbf{I} \tag{3.3.30}$$

Die Diagonalkoeffizienten der Diagonalmatrix $\mathbf{E}$ sind wahlweise $+1$ oder -1.

Die Zerlegung (3.3.30) wird im folgenden dazu verwendet, um aufzuzeigen, daß die Konvergenz im Schritt s der Matrix $\mathbf{A}^s$ zur Diagonalform $\mathbf{\Lambda}$ vom Verhältnis der Eigenwerte (λ_i/λ_k) abhängig ist. Für den Konvergenzbeweis werden zunächst folgende Voraussetzungen vereinbart :

1. Das Produkt $\mathbf{E}\,\mathbf{X}^T$ sei regulär. Es besitzt damit die Zerlegung (3.3.31) in eine Linksdreiecksmatrix $\mathbf{L}$ und eine Rechtsdreiecksmatrix $\mathbf{U}$.

$$\mathbf{E}\,\mathbf{X}^T = \mathbf{L}\,\mathbf{U} \qquad \text{mit } l_{ii} = 1 \tag{3.3.31}$$

2. Die Matrix $\mathbf{A}^s$ sei definit. Die Eigenwertmatrix $\mathbf{\Lambda}$ besitzt somit ausschließlich Diagonalkoeffizienten $\lambda_i \neq 0$. Sie wird in das Produkt der Einheitsmatrix $\mathbf{F}$ und der Diagonalmatrix $\mathbf{\Lambda}_+$ zerlegt. Die Diagonalmatrix $\mathbf{\Lambda}_+$ besitzt ausschließlich positive Diagonalelemente.

$$\mathbf{\Lambda} = \mathbf{F}\,\mathbf{\Lambda}_+ \qquad \text{mit } \lambda_{+ii} \geq 0 \tag{3.3.32}$$

$$\mathbf{C}_s \;=\; \begin{array}{|c|c|c|c|}\hline 1 & 0 & 0 & \cdots \\ \hline l_{21}\left[\frac{\lambda_2}{\lambda_1}\right]^s & 1 & 0 & \cdots \\ \hline l_{31}\left[\frac{\lambda_3}{\lambda_1}\right]^s & l_{32}\left[\frac{\lambda_3}{\lambda_2}\right]^s & 1 & \cdots \\ \hline \vdots & \vdots & \vdots & \ddots \\ \hline \end{array}$$

Bild 3.7: Konvergenzbestimmende Linksdreiecksmatrix $\mathbf{C}_s$

Mit (3.3.31) und (3.3.32) in (3.3.30) folgt :

$$\mathbf{A}^s = \mathbf{X}\,\mathbf{E}\,(\mathbf{F}\,\mathbf{\Lambda}_+)^s\,\mathbf{L}\,\mathbf{U} \tag{3.3.33}$$

$$= \mathbf{X}\,\mathbf{E}\,\mathbf{F}^s\,\mathbf{C}_s\,\mathbf{\Lambda}_+^s\,\mathbf{U} \tag{3.3.34}$$

$$\mathbf{C}_s = \mathbf{\Lambda}_+^s\,\mathbf{L}\,\mathbf{\Lambda}_+^{-s} \tag{3.3.35}$$

Das Produkt $(\mathbf{X}\,\mathbf{E}\,\mathbf{F}^s\,\mathbf{C}_s)$ in (3.3.33) ist im allgemeinen keine Orthonormalmatrix, da $\mathbf{C}_s$ nicht orthonormal ist. Bei willkürlicher Anordnung der Eigenwerte λ_i in $\mathbf{\Lambda}$ konvergiert $\mathbf{C}_s$ auch nicht zu einer Orthonormalmatrix. Setzt man jedoch voraus, daß die Eigenwerte λ_i im Betrag verschieden und in $\mathbf{\Lambda}$ nach abnehmendem Betrag geordnet sind (3.3.36), so konvergiert $\mathbf{C}_s$ zu einer Identitätsmatrix.

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_{k-1}| \geq |\lambda_k| \geq |\lambda_{k+1}| \geq \ldots \geq |\lambda_n| \tag{3.3.36}$$

$$\lim_{s\to\infty} \mathbf{C}_s = \mathbf{I} \tag{3.3.37}$$

Mit Gleichung (3.3.37) folgt für die Zerlegung (3.3.33), daß $\mathbf{A}^s$ das Produkt einer Orthonormalmatrix $(\mathbf{X}\,\mathbf{E}\,\mathbf{F}^s)$ und einer Rechtsdreiecksmatrix $(\mathbf{\Lambda}_+^s\,\mathbf{U})$ ist. Der Vergleich mit der Zerlegung (3.3.23) zeigt, daß die iterierte Matrix $\mathbf{A}_s$ zur Diagonalform $\mathbf{\Lambda}$, mit der Eigenschaft (3.3.36) konvergiert.

$$\mathbf{P}_s = \mathbf{P}_{s-1}\,\mathbf{Q}_s\,\mathbf{E}_s \quad : \quad \mathbf{X}\,\mathbf{E}\,\mathbf{F}^s = \mathbf{X}\,\mathbf{E}\,\mathbf{F}^{s-1}\,\mathbf{Q}_s\,\mathbf{E}_s \tag{3.3.38}$$

$$\mathbf{Q}_s\,\mathbf{E}_s = \mathbf{F} \tag{3.3.39}$$

$$\mathbf{U}_s = \mathbf{E}_s\,\mathbf{R}_s\,\mathbf{U}_{s-1} \quad : \quad \mathbf{\Lambda}_+\,\mathbf{U} = \mathbf{E}_s\,\mathbf{R}_s\,\mathbf{\Lambda}_+^{s-1}\mathbf{U} \tag{3.3.40}$$

$$\mathbf{E}_s\,\mathbf{R}_s = \mathbf{\Lambda}_+ \tag{3.3.41}$$

$$\mathbf{A}_s = \mathbf{Q}_s\,\mathbf{E}_s\,\mathbf{E}_s\,\mathbf{R}_s = \mathbf{F}\,\mathbf{\Lambda}_+ = \mathbf{\Lambda} \tag{3.3.42}$$

□

Zerlegung der Eigenmatrix X

Der Konvergenzbeweis in (3.3.31) bis (3.3.42) wurde unter der Voraussetzung geführt, daß für die Transponierte der Eigenmatrix $\mathbf{X}$ eine Dreieckszerlegung der Form (3.3.31) existiert. Diese Zerlegung ist nicht immer möglich. Beispielsweise sind Permutationsmatrizen $\mathbf{T}$ nicht als Produkt $\mathbf{L}\,\mathbf{U}$ zerlegbar.

Satz 3.3.2 *Gegeben sei die transponierte Eigenmatrix* $\mathbf{X}^T$ *aus* (3.3.30). *Die Zeilen von* $\mathbf{X}^T$ *sind die Eigenvektoren der symmetrischen Matrix* $\mathbf{A}$ *und damit linear unabhängig. Die Eigenmatrix* $\mathbf{X}^T$ *ist somit nicht singulär. Ist ihre Zerlegung in eine Linksdreiecksmatrix* $\mathbf{L}$ *mit Diagonalkoeffizienten* $l_{ii} = 1$ *und einer Rechtsdreiecksmatrix* $\mathbf{U}$ *nicht möglich, so existiert eine Permutation* $(\mathbf{T}\,\mathbf{X}^T)$ *von* $\mathbf{X}^T$ *mit der Zerlegung :*

$$\mathbf{T}\,\mathbf{X}^T \;=\; \mathbf{T}\,\mathbf{L}\,\mathbf{U} \tag{3.3.43}$$

Die Konvergenz der Iteration bleibt durch die Permutationen in der Zerlegung (3.3.43) *unbeeinflußt.*

Beweis : Die ersten $i-1$ Spalten bzw. Zeilen der Matrizen $\mathbf{L}$ und $\mathbf{U}$ sind berechnet. Im Schritt i der Zerlegung wird das Diagonalelement u_{ii} der Rechtsdreiecksmatrix $\mathbf{U}$ Null.

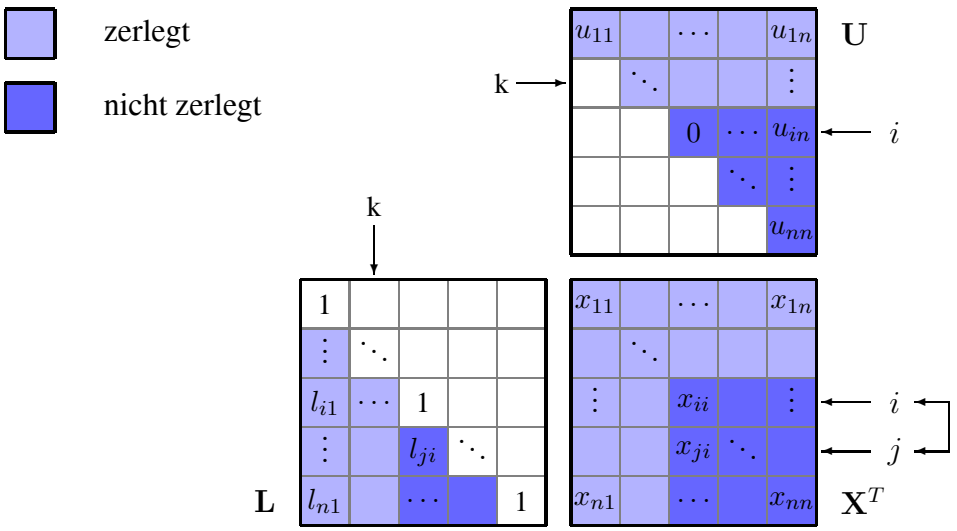


Bild 3.8: Abbruch der Zerlegung in Zeile i mit $u_{ii} = 0$: $\quad \sum_{k=1}^{i-1} l_{ik} u_{ki} \;=\; x_{ii}$

Die Gleichung für das Element $x_{i+1,i}$ liefert einen Widerspruch, deshalb wird Zeile i von $\mathbf{X}^T$ mit der ersten Zeile $j > i$ getauscht, für die $u_{ii} \neq 0$. Aus (3.3.43) folgt, daß die Rechtsdreiecksmatrix

$\mathbf{U}$ vom Zeilentausch unberührt bleibt. In der Linksdreiecksmatrix $\mathbf{L}$ werden die Zeilen i und j getauscht. Die Zerlegung wird mit den permutierten Matrizen $\mathbf{T}\,\mathbf{L}$ und $\mathbf{T}\,\mathbf{X}^T$ fortgesetzt. Das Diagonalelement u_{ii} berechnet sich aus den getauschten Zeilen j :

$$\sum_{k=1}^{i-1} l_{jk}u_{ki} + u_{ii} \;=\; x_{ji} \qquad \text{mit} \quad u_{ii} \neq 0 \tag{3.3.44}$$

Element l_{ij} wird aus den getauschten Zeilen i berechnet und nimmt den Wert Null an :

$$\sum_{k=1}^{i-1} l_{ik}u_{ki} + l_{ji}u_{ii} \;=\; x_{ii} \qquad \text{mit} \quad l_{ij} = 0 \tag{3.3.45}$$

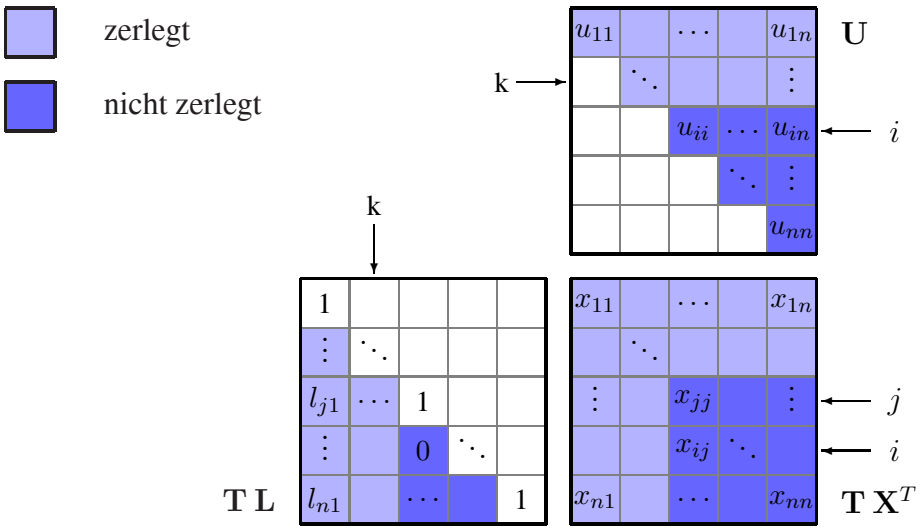


Bild 3.9: Zerlegung der permutierten Eigenmatrix $\mathbf{T}\,\mathbf{X}^T$

Mit den Gleichungen (3.3.43) und (3.3.32) substituiert in (3.3.30) und der Identität $\mathbf{T}\,\mathbf{T}^T = \mathbf{I}$ ändert sich die Iterationsmatrix $\mathbf{C}_s$ wie folgt :

$$\begin{aligned} \mathbf{A}_s &= \mathbf{X}\,\mathbf{E}\,\mathbf{\Lambda}^s\,\mathbf{E}\,\mathbf{X}^T \\ &= \mathbf{X}\,\mathbf{E}\,\mathbf{F}^s\,\mathbf{T}\,\mathbf{T}^T\,\mathbf{\Lambda}_+^s\,\mathbf{T}\,\mathbf{L}\,\mathbf{U} \\ &= \mathbf{X}\,\mathbf{E}\,\mathbf{F}^s\,\mathbf{T}\,\hat{\mathbf{C}}_s\,\mathbf{\Lambda}_+^s\,\mathbf{U} \end{aligned} \tag{3.3.46}$$

$$\hat{\mathbf{C}}_s \;=\; \hat{\mathbf{\Lambda}}_+^s\,\mathbf{L}\,\hat{\mathbf{\Lambda}}_+^{-s} \tag{3.3.47}$$

$$\hat{\mathbf{\Lambda}}_+^s \;=\; \mathbf{T}^T\,\mathbf{\Lambda}_+^s\,\mathbf{T} \tag{3.3.48}$$

$$\hat{\mathbf{C}}_s = \begin{array}{|c|c|c|c|c|} \hline & \overset{k}{\downarrow} & & & \\ \hline 1 & 0 & 0 & 0 & \cdots \\ \hline \ddots & \ddots & 0 & 0 & \cdots \\ \hline \ddots & l_{jk}\left[\frac{\lambda_j}{\lambda_k}\right]^s & 1 & 0 & \cdots \leftarrow j \\ \hline \cdots & l_{ik}\left[\frac{\lambda_i}{\lambda_k}\right]^s & l_{ij}\left[\frac{\lambda_i}{\lambda_j}\right]^s & 1 & \ddots \leftarrow i \\ \hline \vdots & \vdots & \ddots & \ddots & \ddots \\ \hline \end{array}$$

Bild 3.10: Konvergenzbestimmende Linksdreiecksmatrix $\hat{\mathbf{C}}_s$

Mit der Voraussetzung $j > i$ für den Zeilentausch während der Zerlegung von $\mathbf{X}^T$ und $k < i$ gilt auch $k < j$. Die Koeffizienten c_{ik} und c_{jk} der iterierten Matrix $\hat{\mathbf{C}}_s$ streben damit nach Null. Für $k = j$ in der getauschten Zeile i ist l_{ij} gemäß (3.3.45) gleich Null. $\hat{\mathbf{C}}_s$ strebt damit zu einer Identitätsmatrix. Die Permutationen bei der Zerlegung ändern die Reihenfolge der konvergierenden Eigenwerte, sie beeinflussen jedoch nicht das Konvergenzverhalten der Iteration.

□

Beispiel 2 : Dreieckszerlegung der Eigenmatrix mit Permutation

Gegeben sei eine reelle, symmetrische Matrix $\mathbf{A}$ mit ihren Eigenzuständen (Bild 3.11).

0.6	−0.35	−0.25	0.1
−0.35	0.6	0.1	−0.25
−0.25	0.1	0.6	−0.35
0.1	−0.25	−0.35	0.6

$\mathbf{A}$

$\lambda_1 = 1.3$	$\lambda_2 = 0.6$	$\lambda_3 = 0.4$	$\lambda_4 = 0.1$
0.5	−0.5	−0.5	0.5
−0.5	0.5	−0.5	0.5
−0.5	−0.5	0.5	0.5
0.5	0.5	0.5	0.5
$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$

Bild 3.11: Beispielmatrix $\mathbf{A}$

Die Dreieckszerlegung der Eigenmatrix $\mathbf{X}^T = \mathbf{LU}$ mit $\mathbf{X}^T = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]^T$ bricht mit dem Diagonalkoeffizienten $u_{22} = 0.0$ ab (Bild3.12). Eine vollständige Zerlegung der Eigenmatrix ist nur mit Hilfe der Permutation $\mathbf{TX}^T$ (3.3.49) möglich. Die Permutationsmatrix $\mathbf{T}$ tauscht die Zeilen 2 und 3.

$$\mathbf{T}\,\mathbf{X}^T \;:=\; \mathbf{T}\,\mathbf{L}\,\mathbf{U} \tag{3.3.49}$$

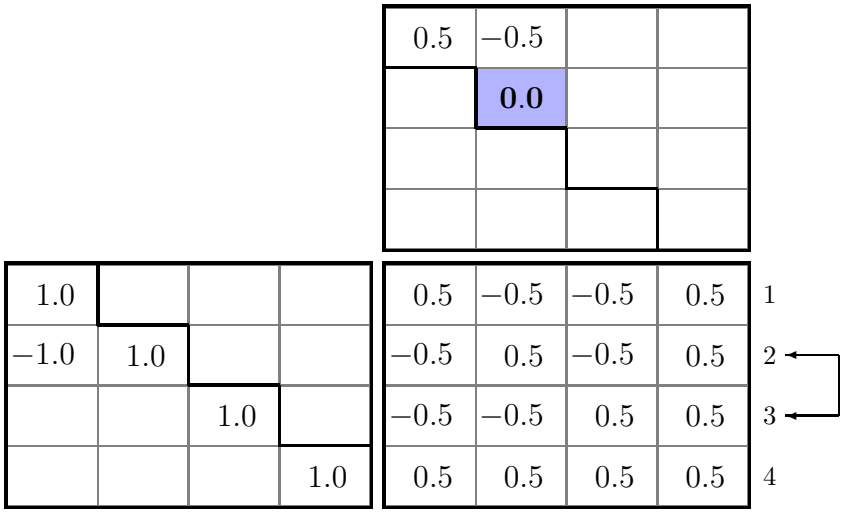


Bild 3.12: Abbruch der Dreieckszerlegung $\mathbf{X}^T = \mathbf{LU}$

Nach dem Zeilentausch $\mathbf{T}\mathbf{X}^T$ besitzt $\mathbf{A}$ die Zerlegung (3.3.50). Die Permutationsmatrix ändert die Bestimmungsreihenfolge der Eigenwerte (Bild 3.13).

$$\mathbf{T}\,\mathbf{D}\,\mathbf{T}^T \;=\; \mathbf{T}\,\mathbf{X}^T\mathbf{A}\,\mathbf{X}\,\mathbf{T}^T \tag{3.3.50}$$

$\mathbf{A}$

0.6	−0.35	−0.25	0.1
−0.35	0.6	0.1	−0.25
−0.25	0.1	0.6	−0.35
0.1	−0.25	−0.35	0.6

$\mathbf{X}\,\mathbf{T}^T$

0.5	−0.5	−0.5	0.5
−0.5	−0.5	0.5	0.5
−0.5	0.5	−0.5	0.5
0.5	0.5	0.5	0.5

$\mathbf{T}\,\mathbf{X}^T$

0.5	−0.5	−0.5	0.5
−0.5	−0.5	0.5	0.5
−0.5	0.5	−0.5	0.5
0.5	0.5	0.5	0.5

$\mathbf{T}\,\mathbf{X}^T\mathbf{A}$

0.65	−0.65	−0.65	0.65
−0.2	−0.2	0.2	0.2
−0.3	0.3	−0.3	0.3
0.05	0.05	0.05	0.05

$\mathbf{T}\,\mathbf{D}\,\mathbf{T}^T$

1.3	0.0	0.0	0.0
0.0	0.4	0.0	0.0
0.0	0.0	0.6	0.0
0.0	0.0	0.0	0.1

Bild 3.13: Schurzerlegung mit Permutation

Nulleigenwerte $\lambda_i = 0$

Satz 3.3.3 *Matrix* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *sei semidefinit. Sie besitzt damit einen* p*-fachen Eigenwert* $\lambda = 0$*. Dann sind die* p *Eigenzustände* $(0, \mathbf{x}_i)$ *mit einer einzigen QR-Zerlegung bestimmbar.*

Beweis : In der QR-Zerlegung $\mathbf{A} = \mathbf{QR}$ ist $\mathbf{A}$ singulär. Die Orthonormalmatrix $\mathbf{Q}$ besitzt linear unabhängige Spaltenvektoren und ist damit nicht singulär. Folglich ist die Rechtsdreiecksmatrix $\mathbf{R}$ singulär und besitzt p Nullzeilen. Tritt während der Zerlegung in Zeile $i < n$ ein Diagonalelement $r_{ii} = 0$ auf, so führt die Transformation (3.3.10) zu einem Zeilentausch der Zeilen i und k. Nach Abschluß der Zerlegung sind die letzten p Zeilen von $\mathbf{R}$ Nullzeilen.

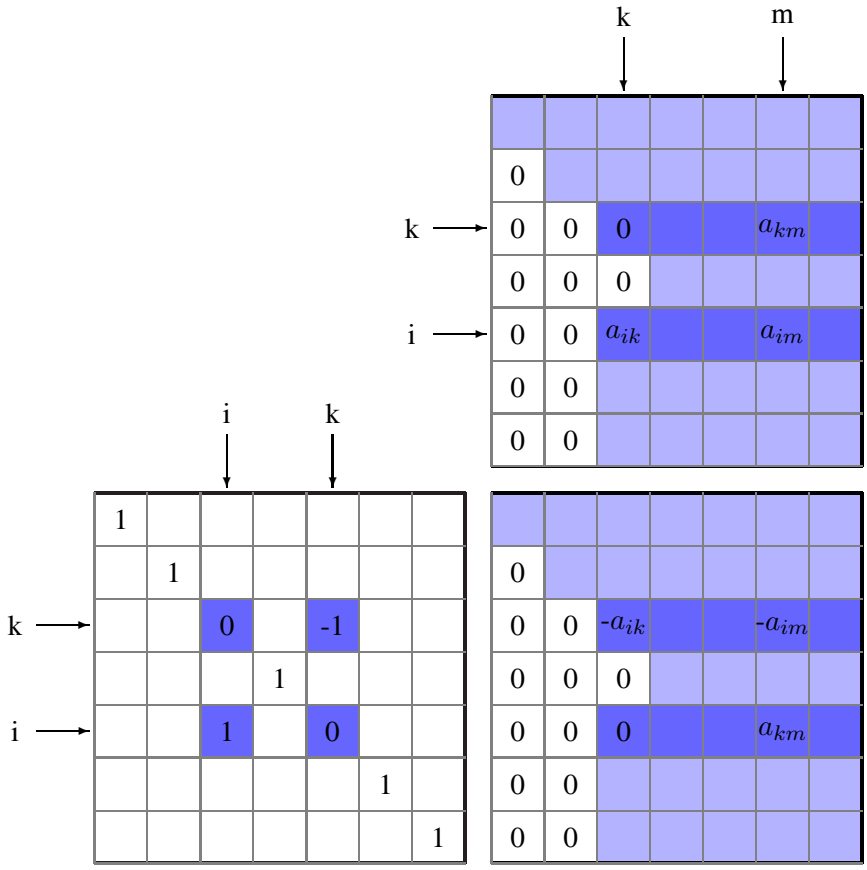


Bild 3.14: Zeilentausch in der Zerlegung der singulären Matrix $\mathbf{A}$

Die letzten p Zeilen und Spalten der symmetrischen Rekombination $\mathbf{RQ} = \mathbf{Q}^T\mathbf{AQ}$ sind daher Null. Die letzten p Spalten der orthonormalen Transformationsmatrix $\mathbf{Q}$ sind die Eigenvektoren zu den p Eigenwerten $\lambda_i = 0$.

□

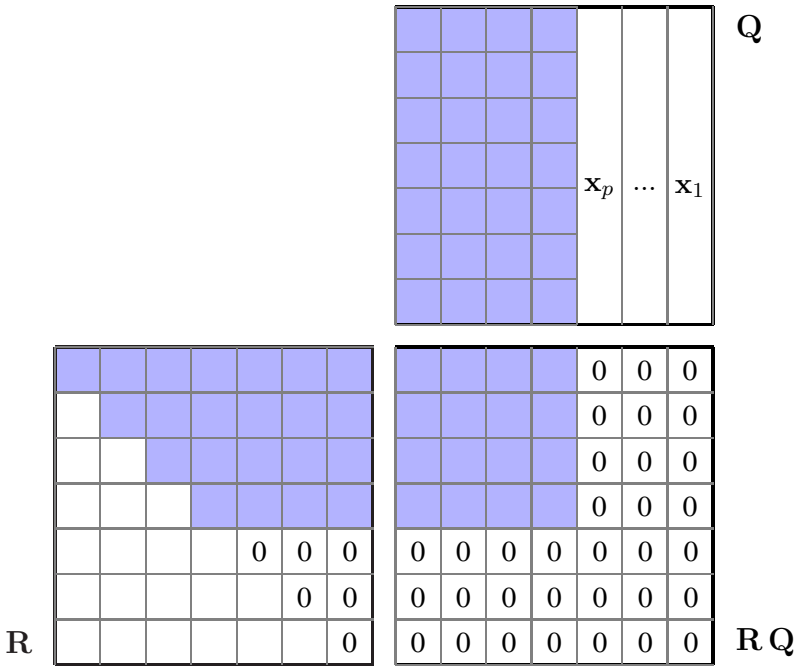


Bild 3.15: p Eigenzustände $(0, \mathbf{x}_i)$ nach der Transformation $\mathbf{Q}^T\mathbf{A}\mathbf{Q}$ im Schritt 1

Die Linksdreiecksmatrix $\mathbf{C}_s$ aus (3.3.35) ist für $\lambda = 0$ nicht definiert. Der Konvergenzbeweis in (3.3.31) bis (3.3.42) ist damit nur gültig für die um p Zeilen und Spalten reduzierte Matrix $\mathbf{A}^{(n-p)\times(n-p)}$.

Betragsgleiche Eigenwerte $|\lambda_i| = |\lambda_k|$

Satz 3.3.4 *Matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ besitze einen p-fachen Eigenwert λ und einen q-fachen Eigenwert $-\lambda$. Alle anderen Eigenwerte seien im Betrag verschieden. Dann konvergiert die iterierte Matrix $\mathbf{A}_s$ zu einer Diagonalmatrix mit Blockstruktur. Die Diagonalböcke sind orthonormal und besitzen die Dimension $(p+q)\times(p+q)$.*

Beweis : Unter der Voraussetzung (3.3.36) besitzen die Matrizen $\mathbf{\Lambda}$ und $\mathbf{\Lambda}_+$ folgende Belegung :

$$\mathbf{\Lambda} \;=\; \mathbf{F}\mathbf{\Lambda}_+ \tag{3.3.51}$$

$$\mathbf{\Lambda} = \begin{array}{|c|c|c|c|}\hline \mathbf{\Lambda}_1 & & & \\ \hline & \lambda\mathbf{I} & & \\ \hline & & -\lambda\mathbf{I} & \\ \hline & & & \mathbf{\Lambda}_2 \\ \hline \end{array} \qquad \mathbf{\Lambda}_+ = \begin{array}{|c|c|c|c|}\hline \mathbf{\Lambda}_{1_+} & & & \\ \hline & \lambda\mathbf{I} & & \\ \hline & & \lambda\mathbf{I} & \\ \hline & & & \mathbf{\Lambda}_{2_+} \\ \hline \end{array}$$

Bild 3.16: Belegung der Eigenwertmatrizen bei mehrfachen betragsgleichen Eigenwerten

Die Iterationsmatrix $\mathbf{C}_s$ aus (3.3.35) konvergiert nicht mehr zu einer Diagonalmatrix, sondern zu einer Linksdreiecksmatrix $\tilde{\mathbf{L}}$ mit Diagonalkoeffizienten 1.

$$\begin{aligned} |\lambda_1| \geq \ldots \geq |\lambda_{k-1}| \geq |\lambda_{k_1}| = \ldots = |\lambda_{k_p}| = |-\lambda_{k_1}| = \ldots = |-\lambda_{k_q}| \\ \geq |\lambda_{k+1}| \geq \ldots \geq |\lambda_n| \end{aligned} \tag{3.3.52}$$

$$\lim_{s\to\infty} \mathbf{C}_s \;=\; \tilde{\mathbf{L}} \tag{3.3.53}$$

Wegen $\tilde{\mathbf{L}}$ ist das Produkt $(\mathbf{X}\;\mathbf{E}\;\mathbf{F}^s\,\tilde{\mathbf{L}})$ aus (3.3.33) keine Orthonormalmatrix. Es wird deshalb in das Produkt einer Orthonormalmatrix $\hat{\mathbf{Q}}$ und einer Rechtsdreiecksmatrix $\hat{\mathbf{R}}$ zerlegt, die bis auf die Einheitsmatrix $\hat{\mathbf{E}}$ eindeutig bestimmbar sind.

$$\mathbf{X}\;\mathbf{E}\;\mathbf{F}^s\,\tilde{\mathbf{L}} \;=\; (\hat{\mathbf{Q}}\;\hat{\mathbf{E}})\;(\hat{\mathbf{E}}\;\hat{\mathbf{R}}) \tag{3.3.54}$$

Substitution von (3.3.54) in (3.3.30) führt zu der Zerlegung von $\mathbf{A}^s$ in das Produkt einer Orthonormalmatrix mit einer Rechtsdreiecksmatrix und damit zu den Faktoren $\mathbf{Q}_s$ und $\mathbf{R}_s$ der iterierten Matrix $\mathbf{A}_s$ (3.3.23) :

$$\mathbf{A}^s \;=\; (\mathbf{X}\;\mathbf{E}\;\mathbf{F}^s\,\mathbf{X}^T\,\hat{\mathbf{Q}}\;\hat{\mathbf{E}})(\hat{\mathbf{E}}\;\hat{\mathbf{R}}\;\mathbf{\Lambda}_+^s\,\mathbf{U}) \tag{3.3.55}$$

Der Vergleich mit der Zerlegung (3.3.23) gibt Aufschluß über das Konvergenzverhalten der iterierten Matrix $\mathbf{A}_s$ mit der Eigenschaft (3.3.52).

$$\mathbf{P}_s = \mathbf{P}_{s-1}\,\mathbf{Q}_s\,\mathbf{E}_s : \quad \mathbf{X}\,\mathbf{E}\,\mathbf{F}^s\,\mathbf{X}^T\hat{\mathbf{Q}}\,\hat{\mathbf{E}} = \mathbf{X}\,\mathbf{E}\,\mathbf{F}^{s-1}\,\mathbf{X}^T\hat{\mathbf{Q}}\,\hat{\mathbf{E}}\,\mathbf{Q}_s\,\mathbf{E}_s \tag{3.3.56}$$

$$\mathbf{Q}_s\,\mathbf{E}_s = \hat{\mathbf{E}}\,\mathbf{W}\,\mathbf{F}\,\mathbf{W}^T\hat{\mathbf{E}} \tag{3.3.57}$$

$$\mathbf{W} = \hat{\mathbf{Q}}^T\mathbf{X} \qquad \text{mit } \mathbf{W}^T\mathbf{W} = \mathbf{I} \tag{3.3.58}$$

$$\mathbf{U}_s = \mathbf{E}_s\,\mathbf{R}_s\,\mathbf{U}_{s-1} : \quad \hat{\mathbf{E}}\,\hat{\mathbf{R}}\,\mathbf{\Lambda}_+^s\,\mathbf{U} = \mathbf{E}_s\,\mathbf{R}_s\,\hat{\mathbf{E}}\,\hat{\mathbf{R}}\,\mathbf{\Lambda}_+^{s-1}\mathbf{U} \tag{3.3.59}$$

$$\mathbf{E}_s\,\mathbf{R}_s = \hat{\mathbf{E}}\,\hat{\mathbf{R}}\,\mathbf{\Lambda}_+\,\hat{\mathbf{R}}^{-1}\,\hat{\mathbf{E}} \tag{3.3.60}$$

$$\begin{aligned}\mathbf{A}_s = \mathbf{Q}_s\,\mathbf{E}_s\,\mathbf{E}_s\,\mathbf{R}_s &= (\hat{\mathbf{E}}\;\mathbf{W}\;\mathbf{F}\;\mathbf{W}^T\;\hat{\mathbf{E}})(\hat{\mathbf{E}}\,\hat{\mathbf{R}}\,\mathbf{\Lambda}_+\,\hat{\mathbf{R}}^{-1}\,\hat{\mathbf{E}}) \\ &= \hat{\mathbf{E}}\;\mathbf{W}\;\mathbf{F}\;\tilde{\mathbf{L}}\;\mathbf{\Lambda}_+\tilde{\mathbf{L}}^{-1}\;\mathbf{W}^T\;\hat{\mathbf{E}}\end{aligned} \tag{3.3.61}$$

Mit der besonderen Belegung der Matrizen $\tilde{\mathbf{L}}$ und $\mathbf{\Lambda}_+$ ergibt sich für das Produkt $(\mathbf{F}\,\tilde{\mathbf{L}}\,\mathbf{\Lambda}_+\,\tilde{\mathbf{L}})$ die Diagonalmatrix $\mathbf{\Lambda}$ aus (3.3.32). Matrix $\mathbf{A}_s$ vereinfacht sich damit zu :

$$\mathbf{A}_s = \hat{\mathbf{E}}\;\mathbf{W}\;\mathbf{\Lambda}\;\mathbf{W}^T\,\hat{\mathbf{E}} \tag{3.3.62}$$

$\mathbf{\Lambda} =$

$\mathbf{I}$			
	$\mathbf{L}_1$		
	$\mathbf{L}_3$	$\mathbf{L}_2$	
			$\mathbf{I}$

$\mathbf{W} =$

$\mathbf{I}$			
	$\mathbf{W}_{11}$		
	$\mathbf{W}_{21}$	$\mathbf{W}_{22}$	
			$\mathbf{I}$

$\hat{\mathbf{E}} =$

$\hat{\mathbf{E}}_1$			
	$\hat{\mathbf{E}}_3$		
		$\hat{\mathbf{E}}_3$	
			$\hat{\mathbf{E}}_4$

Bild 3.17: Belegung der Matrizen $\tilde{\mathbf{L}}, \mathbf{W}, \hat{\mathbf{E}}$

$$\mathbf{A}_{11} = \lambda\,\hat{\mathbf{E}}_2\,(\mathbf{W}_{11}\,\mathbf{W}_{11}^T - \mathbf{W}_{12}\,\mathbf{W}_{12}^T)\,\hat{\mathbf{E}}_2 \tag{3.3.63}$$

$$\mathbf{A}_{12} = \lambda\,\hat{\mathbf{E}}_2\,(\mathbf{W}_{11}\,\mathbf{W}_{21}^T - \mathbf{W}_{12}\,\mathbf{W}_{22}^T)\,\hat{\mathbf{E}}_2 \tag{3.3.64}$$

$$\mathbf{A}_{22} = \lambda\,\hat{\mathbf{E}}_3\,(\mathbf{W}_{21}\,\mathbf{W}_{21}^T - \mathbf{W}_{22}\,\mathbf{W}_{22}^T)\,\hat{\mathbf{E}}_3 \tag{3.3.65}$$

$\mathbf{\Lambda}_1$			
	$\lambda\mathbf{I}$		
		-$\lambda\mathbf{I}$	
			$\mathbf{\Lambda}_2$

$\hat{\mathbf{E}}_1$			
	$\mathbf{W}_{11}^T\hat{\mathbf{E}}_2$	$\mathbf{W}_{21}^T\hat{\mathbf{E}}_3$	
	$\mathbf{W}_{12}^T\hat{\mathbf{E}}_2$	$\mathbf{W}_{22}^T\hat{\mathbf{E}}_3$	
			$\hat{\mathbf{E}}_4$

$\hat{\mathbf{E}}_1$			
	$\hat{\mathbf{E}}_2\mathbf{W}_{11}$	$\hat{\mathbf{E}}_2\mathbf{W}_{12}$	
	$\hat{\mathbf{E}}_3\mathbf{W}_{21}$	$\hat{\mathbf{E}}_3\mathbf{W}_{22}$	
			$\hat{\mathbf{E}}_4$

$\hat{\mathbf{E}}_1\mathbf{\Lambda}_1$			
	$\lambda\hat{\mathbf{E}}_2\mathbf{W}_{11}$	-$\lambda\hat{\mathbf{E}}_2\mathbf{W}_{12}$	
	$\lambda\hat{\mathbf{E}}_3\mathbf{W}_{21}$	-$\lambda\hat{\mathbf{E}}_3\mathbf{W}_{22}$	
			$\hat{\mathbf{E}}_4\mathbf{\Lambda}_2$

$\mathbf{\Lambda}_1$			
	$\mathbf{A}_{11}$	$\mathbf{A}_{12}$	
	$\mathbf{A}_{12}^T$	$\mathbf{A}_{22}$	
			$\mathbf{\Lambda}_2$

Bild 3.18: Produkt $(\hat{\mathbf{E}}\,\mathbf{W}\,\mathbf{\Lambda}\,\mathbf{W}^T\hat{\mathbf{E}})$

Die Belegung der Submatrizen $\mathbf{A}_{mn}$ und damit der Matrix $\mathbf{A}_s$ ist abhängig von den Eigenwerten von $\mathbf{A}$. Für das Konvergenzverhalten von $\mathbf{A}_s$ zur Diagonalform sind folgende Fälle zu unterscheiden :

1. *Alle Eigenwerte λ_i sind im Betrag verschieden* : $\mathbf{A}_s$ konvergiert zur Diagonalform $\mathbf{\Lambda}$ mit den Eigenwerten von $\mathbf{A}$. Matrix $\mathbf{A}_s$ ist nur mit den Submatrizen $\mathbf{\Lambda}_1$ und $\mathbf{\Lambda}_2$ belegt.

2. $\mathbf{A}$ *besitzt einen p-fachen Eigenwert λ* : $\mathbf{A}_s$ konvergiert zur Diagonalform $\mathbf{\Lambda}$ mit den Eigenwerten von $\mathbf{A}$. Matrix $\mathbf{A}_s$ ist nur mit den Submatrizen $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$ und $\mathbf{A}_{11}$ belegt.

$$\mathbf{A}_{11} \;=\; \lambda\,\hat{\mathbf{E}}_2\,\mathbf{W}_{11}\,\mathbf{W}_{11}^T\,\hat{\mathbf{E}}_2 \;=\; \lambda\mathbf{I}$$

3. $\mathbf{A}$ *besitzt einen q-fachen Eigenwert $-\lambda$* : $\mathbf{A}_s$ konvergiert zur Diagonalform $\mathbf{\Lambda}$ mit den Eigenwerten von $\mathbf{A}$. Matrix $\mathbf{A}_s$ ist nur mit den Submatrizen $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$ und $\mathbf{A}_{22}$ belegt.

$$\mathbf{A}_{22} \;=\; -\lambda\,\hat{\mathbf{E}}_3\,\mathbf{W}_{22}\,\mathbf{W}_{22}^T\,\hat{\mathbf{E}}_3 \;=\; -\lambda\mathbf{I}$$

4. $\mathbf{A}$ *besitzt einen p-fachen Eigenwert λ und einen q-fachen Eigenwert $-\lambda$* : $\mathbf{A}_s$ konvergiert im Bereich der Submatrizen $\mathbf{\Lambda}_1$ und $\mathbf{\Lambda}_2$ zur Diagonalform mit den im Betrag verschiedenen Eigenwerten von $\mathbf{A}$. Dazwischen konvergiert $\mathbf{A}_s$ zur Blockdiagonalform $\mathbf{A}_D$ mit den orthonormalen Submatrizen $\mathbf{A}_{11}, \mathbf{A}_{12}, \mathbf{A}_{22}$.

5. $\mathbf{A}$ *besitzt m verschiedene Gruppen von betragsgleichen Eigenwerten mit unterschiedlichem Vorzeichen* : $\mathbf{A}_s$ konvergiert zur Diagonal- bzw. Blockdiagonalform mit m Blockdiagonalmatrizen.

□

3.3.3 Fehlerschranken

Die Theorie zur Bestimmung von Eigenwerten und Eigenvektoren wird in ihrer praktischen Umsetzung durch zwei Einschränkungen maßgeblich beeinflußt. Durch die endliche Darstellung von Gleitpunktzahlen im Rechner unterliegen die Algorithmen i.d.R. Rundungsfehlern der Ordnung $O(\epsilon)$ (mit $\epsilon :=$ Maschinengenauigkeit). Zusätzlich erfordert der iterative Charakter der Lösungsmethoden ein Abbruchkriterium, das nach einer endlichen Anzahl von Berechnungsschritten zu einer Terminierung des verwendeten Algorithmus führt. Für eine zuverlässige Abschätzung der Genauigkeit der berechneten Eigenwert- bzw. Eigenvektornäherungen werden Fehlerschranken formuliert :

- *A priori*-Schranken erlauben eine Aussage über die zu erwartende Genauigkeit eines Algorithmus und werden häufig durch eine Rückwärtsanalyse bestimmt. Untersucht wird dabei die Fragestellung nach den Änderungen in den Eigenwerten und Eigenvektoren, wenn die Matrizen der Eigenwertaufgabe kleinen Störungen unterworfen werden.
- Fehlerschranken, die auf der Bestimmung der Norm des Restvektors $\mathbf{r} = \mathbf{A}\mathbf{x} - \lambda\mathbf{x}$ basieren werden als *a posteriori*-Schranken bezeichnet. Diese Form der Fehleranalyse gibt Auskunft über die erzielte Genauigkeit einer berechneten Näherungslösung.

Die Theorie zur Bestimmung von Fehlerschranken des folgenden Kapitels beschränkt sich auf die in den Implementierungen zur Anwendung kommenden Methoden. Eine umfassende Behandlung der Störungstheorie und Fehleranalyse ist in [48], [44], [35] oder [13] gegeben.

Genauigkeitsanalyse

Zur Abschätzung des Fehlerniveaus der durch die Ähnlichkeitstransformation des QR-Algorithmus ermittelten Eigenwertnäherungen wird die Fehlermatrix der Transformation bestimmt.

Untersucht wird der Fehler der Transformation (3.3.66) im Schritt k. Die orthonormale Transformationsmatrix $\mathbf{Q}_k$ wird im Schritt k aus $\mathbf{A}_k$ bestimmt.

$$\mathbf{A}_{k+1} \;=\; \mathbf{Q}_k^T\,\mathbf{A}_k\,\mathbf{Q}_k \qquad\qquad k \;=\; 1, 2, 3, \ldots, m \tag{3.3.66}$$

Zur Unterscheidung zwischen exakter und durch Gleitpunktzahlendarstellung im Rechner genäherten Rechnung wird folgende Notation eingeführt :

1. Exakte Größen werden ohne Überstreichungen dargestellt ($\mathbf{A}, \mathbf{Q}, \ldots$).
2. Näherungen, gleichgültig ob aus exakten Größen oder Näherungsgrößen bestimmt, werden durch das Symbol $\hat{}$ gekennzeichnet ($\hat{\mathbf{A}}, \hat{\mathbf{Q}}, \ldots$).

3. Exakte Größen, aus einer Näherung bestimmt, werden durch das Symbol $^-$ gekennzeichnet ($\bar{\mathbf{Q}}, \ldots$)

Im Schritt k wird die Transformationsmatrix $\hat{\mathbf{Q}}_k$ im Rechner aus der Näherung $\hat{\mathbf{A}}$ bestimmt. Die Orthogonalität von $\hat{\mathbf{Q}}_k$ ist durch die Fehlermatrix $\mathbf{U}_k$ gestört. Bei exakter Transformation ergibt sich mit (3.3.67) die Transformation (3.3.68).

$$\hat{\mathbf{Q}}_k = \bar{\mathbf{Q}}_k + \mathbf{U}_k \tag{3.3.67}$$

$$\bar{\mathbf{A}}_{k+1} = (\bar{\mathbf{Q}}_k + \mathbf{U}_k)^T \, \hat{\mathbf{A}}_k \, (\bar{\mathbf{Q}}_k + \mathbf{U}_k) \tag{3.3.68}$$

Der Fehler der Transformation im Rechner wird durch die Fehlermatrix $\mathbf{F}_k$ berücksichtigt und ergibt für $\hat{\mathbf{A}}_{k+1}$:

$$\hat{\mathbf{A}}_{k+1} = (\bar{\mathbf{Q}}_k + \mathbf{U}_k)^T \, \hat{\mathbf{A}}_k \, (\bar{\mathbf{Q}}_k + \mathbf{U}_k) + \mathbf{F}_k \tag{3.3.69}$$

$$= \bar{\mathbf{Q}}_k^T \, \hat{\mathbf{A}}_k \, \bar{\mathbf{Q}}_k + \mathbf{E}_k \tag{3.3.70}$$

$$\mathbf{E}_k := \bar{\mathbf{Q}}_k^T \, \hat{\mathbf{A}}_k \, \mathbf{U}_k + \mathbf{U}_k^T \, \hat{\mathbf{A}}_k \, \bar{\mathbf{Q}}_k + \mathbf{U}_k^T \, \hat{\mathbf{A}}_k \, \mathbf{U}_k + \mathbf{F}_k \tag{3.3.71}$$

Nach $k = m, m-1, \ldots, 1$ Schritten folgt für $\hat{\mathbf{A}}_{m+1}$:

$$\hat{\mathbf{A}}_{m+1} = \bar{\mathbf{Q}}_m^T \, \hat{\mathbf{A}}_m \, \bar{\mathbf{Q}}_m + \mathbf{E}_m \tag{3.3.72}$$

$$= \bar{\mathbf{Q}}_m^T \, (\bar{\mathbf{Q}}_{m-1}^T \, \hat{\mathbf{A}}_{m-1} \, \bar{\mathbf{Q}}_{m-1} + \mathbf{E}_{m-1}) \, \bar{\mathbf{Q}}_m + \mathbf{E}_m \tag{3.3.73}$$

$$= \mathbf{H}_1^T \, \mathbf{A} \, \mathbf{H}_1 + \sum_{i=2}^{m+1} \left(\mathbf{H}_i^T \, \mathbf{E}_i \, \mathbf{H}_i \right) \tag{3.3.74}$$

$$\mathbf{H}_j := \bar{\mathbf{Q}}_m \bar{\mathbf{Q}}_{m-1} \ldots \bar{\mathbf{Q}}_j \qquad \mathbf{H}_{m+1} = \mathbf{I} \tag{3.3.75}$$

Der Gesamtfehler der Transformationen wird durch die Fehlermatrix $\mathbf{M}$ (3.3.77) beschrieben. Das Ergebnis $\hat{\mathbf{A}}_{m+1}$ entspricht einer exakten Transformation der mit $\mathbf{M}$ gestörten Ausgangsmatrix $\mathbf{A}$.

$$\hat{\mathbf{A}}_{m+1} = \mathbf{H}_1^T \, (\mathbf{A} + \mathbf{M}) \, \mathbf{H}_1 \tag{3.3.76}$$

$$\mathbf{M} := \mathbf{H}_1^T \left(\sum_{i=2}^{m+1} \mathbf{H}_i^T \, \mathbf{E}_i \, \mathbf{H}_i \right) \mathbf{H}_1 \tag{3.3.77}$$

Mit der Orthonormalität der Matrizen $\bar{\mathbf{Q}}_j$ und $\mathbf{H}_j$ wird die Ordnung der Fehlermatrix mit (3.3.71) wie folgt abgeschätzt :

$$\|\mathbf{M}\|_2 \leq \sum_{i=2}^{m+1} \|\mathbf{E}_i\|_2 \leq \sum_{i=2}^{m+1} \left(\|\mathbf{F}_i\|_2 + \|\mathbf{A}_i\|_2 (2 \, \|\mathbf{U}_i\|_2 + \|\mathbf{U}_i\|_2^2) \right) \tag{3.3.78}$$

$\|\mathbf{M}\|_2$ wird als Fehler der Rückwärtsanalyse bezeichnet. Ist $\|\mathbf{M}\|_2 \leq c\,\epsilon\|\mathbf{A}\|_2$, so wird der Algorithmus als stabil (*backward stable*) bezeichnet. Der Faktor c ist ein langsam anwachsendes Polynom in Abhängigkeit der Dimension N, so daß $(c\,\epsilon)$ nur unwesentlich größer ϵ ist. Der QR-Algorithmus besitzt diese Eigenschaft [13], [30], [42].

Die Transformationsmatrix $\bar{\mathbf{Q}}_i$ entspricht im vorgestellten Verfahren einer Rotationsmatrix $\mathbf{R}_{ik}$ zur Eliminierung einzelner Koeffizienten aus $\hat{\mathbf{A}}_i$. Der Fehler in $\mathbf{U}_i$ stammt aus der Bestimmung der Rotationsparameter $cos\ \theta$ und $sin\ \theta$ und verursacht, daß $sin^2\theta + cos^2\theta \neq 1.0$. Die Fehlermatrix $\mathbf{F}_i$ der Transformation folgt aus den Operationen auf die Elemente der Zeilen und Spalten von $\hat{\mathbf{A}}_i$.

Die Diagonalkoeffizienten der Matrix $\mathbf{D}$ aus Gleichung (3.3.2) sind damit die Eigenwerte der um $\mathbf{M}$ gestörten Matrix $\mathbf{A}$. Jeder Eigenwert aus (3.3.79) ist mit einem maximalen Fehler der Ordnung $(\epsilon\, c\, \|\mathbf{A}\|_2)$ behaftet.

$$\hat{\mathbf{D}} = \mathbf{Q}^T (\mathbf{A} + \mathbf{M})\, \mathbf{Q} \tag{3.3.79}$$

$$\hat{\lambda}_i = \mathbf{q}_i^T (\mathbf{A} + \mathbf{M})\, \mathbf{q}_i \tag{3.3.80}$$

$$\leq \lambda_i + \mathbf{q}_i^T \mathbf{M}\, \mathbf{q}_i \tag{3.3.81}$$

$$\leq \lambda_i + c\,\epsilon\, \|\mathbf{A}\|_2 \leq \lambda_i + c\,\epsilon\, \max_i |\lambda_i| \tag{3.3.82}$$

Schranken für Eigenwerte

Im folgenden werden Fehlerschranken eingeführt, die zur Abschätzung der Qualität der berechneten Näherungslösungen in Kapitel 4.4 bzw. für eine Formulierung geeigneter Abbruchschranken für die Iteration eingesetzt werden. Eine Beweisführung für die Richtigkeit der Schranken wird nur vorgenommen, falls der Beweis für nachfolgende Betrachtungen von Bedeutung ist.

Bauer - Fike Die Schranke nach Bauer-Fike stellt eine einfache, wenngleich nicht strenge Eigenwertschranke dar, die jedoch wegen ihrer einfachen Berechnung von großer praktischer Bedeutung ist.

Satz 3.3.5 *Gegeben sei die Näherung* $(\hat{\lambda}, \hat{\mathbf{x}})$ *mit* $\hat{\mathbf{x}}^T\hat{\mathbf{x}} = 1.0$ *zum Eigenpaar* $(\lambda, \mathbf{x})$ *der Matrix* $\mathbf{A}$ *mit der Zerlegung* $\mathbf{A} = \mathbf{X}^{-1}\mathbf{D}\,\mathbf{X}$. *Dann existiert ein Eigenwert* λ *aus dem Eigenwertspektrum* $\sigma(\mathbf{A})$ *von* $\mathbf{A}$ *für den gilt :*

$$\min_{\lambda \in \sigma(\mathbf{A})} |\lambda - \hat{\lambda}| \leq \|\mathbf{r}\|_2 \qquad \textit{mit}\ \ \mathbf{r} = \mathbf{A}\hat{\mathbf{x}} - \hat{\lambda}\hat{\mathbf{x}} \tag{3.3.83}$$

Beweis : siehe [35]

Gerschgorin Das vorgestellte Verfahren aus 3.2 zur Eigenwertbestimmung überführt die Eigenwertaufgabe (3.2.2) durch eine Reihe von Ähnlichkeitstransformationen in die einfachere Diagonalform (3.3.2). Der iterative Berechnungsprozess erfordert eine Entscheidung darüber, ob der Grad der Diagonalisierung ausreicht, um die spezielle Eigenwertaufgabe mit ausreichender Genauigkeit zu approximieren. Der folgende Satz nach Gerschgorin liefert berechenbare Schranken zur Abschätzung der Eigenwerte einer diagonalisierbaren Matrix $\mathbf{A}$. Die Genauigkeit der Abschätzung steigt mit dem Grad der Diagonalisierung von $\mathbf{A}$ [22], [35], [43].

Satz 3.3.6 *Jeder Eigenwert λ_i einer normalen Matrix $\mathbf{A}$ liegt im Zentrum eines Kreises der komplexen Ebene mit Mittelpunkt a_{ii} und Radius r_i und befriedigt damit mindestens eine der Ungleichungen* (3.3.84).

$$|\lambda - a_{ii}| \;\leq\; \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \;=\; r_i \qquad\qquad i \;=\; 1,\ldots,n \tag{3.3.84}$$

Beweis : Zum Beweis des Gerschgorin Theorems wird die k-te Zeile der speziellen Eigenwertaufgabe $\mathbf{A}\,\mathbf{x} \;=\; \lambda\,\mathbf{x}$ entwickelt :

$$\sum_{j=1}^{n} |a_{kj}|\, x_j \;=\; \lambda\, x_k \tag{3.3.85}$$

Das betragsgrößte Element des Eigenvektors $\mathbf{x}$ sei x_i, so daß für alle $j = 1,\ldots,n$ gilt :

$$\left|\frac{x_j}{x_i}\right| \;\leq\; 1.0 \tag{3.3.86}$$

Allgemein folgt damit für den Diagonalkoeffizienten a_{kk} der Zeile k :

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|\, x_j \qquad = \;\; (\lambda - a_{kk})\, x_k \tag{3.3.87}$$

$$|\lambda - a_{kk}| \left|\frac{x_k}{x_i}\right| \;\leq\; \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{ij}| \left|\frac{x_j}{x_i}\right| \;\leq\; \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{ij}| \tag{3.3.88}$$

Mit (3.3.88) folgt für den Diagonalkoeffizienten a_{ii} als Näherung zum Eigenwert λ :

$$|\lambda - a_{ii}| \;\leq\; \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \left|\frac{x_j}{x_k}\right| \;\leq\; \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \tag{3.3.89}$$

□

In [48] ist gezeigt, daß überschneidungsfreie Gerschgorinsche Kreise jeweils genau einen Eigenwert der Multiplizität m aus dem Spektrum von $\mathbf{A}$ enthalten. Damit lassen sich die Gerschgorinschen Kreise als einfach berechenbare Trennungsregel für schlecht getrennte Eigenwertcluster einsetzten.

Mit Gleichung (3.3.82) substituiert in (3.3.89) wird ein Konvergenzkriterium für die Eigenwertberechnung formuliert. Der Diagonalkoeffizient a_{ii} wird als Näherung $\hat{\lambda}_i$ zum Eigenwert λ_i betrachtet, falls Gleichung (3.3.90) erfüllt ist.

$$|\lambda - \hat{\lambda}| \;=\; |\, c\,\epsilon\,\|\mathbf{A}\|_2\,| \;\geq\; \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \tag{3.3.90}$$

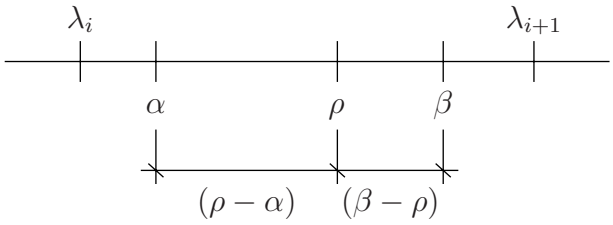
Die Spektralnorm $\|\mathbf{A}\|_2$ wird durch die großzügigere, a priori berechenbare Frobeniusnorm $\|\mathbf{A}\|_F$ ($:= (\sum_i \sum_j a_{ij}^2)^{0.5}$) ersetzt. Für den Faktor c wird die aktuelle Zyklenanzahl gewählt.

Temple - Kato Die Schranken von Temple und Kato für Eigenwerte sind streng im Vergleich zu den Schranken nach Gerschgorin. Als Voraussetzung für die Schranken wird folgendes Lemma formuliert.

Lemma 3.3.1 *Gegeben sei eine Eigenvektornäherung $\hat{\mathbf{x}}$ mit Einheitslänge und der zugeordnete Rayleigh-Quotient $\rho(\mathbf{A}, \hat{\mathbf{x}})$. Ist $[\alpha, \beta]$ ein Intervall, das keinen Eigenwert aus dem Eigenwertspektrum $\sigma(\mathbf{A})$ enthält, so gilt für die Näherung ρ :*

$$(\beta - \rho)\,(\rho - \alpha) \;\leq\; \|\mathbf{r}\|_2^2 \tag{3.3.91}$$

$$mit \qquad \mathbf{r} \;=\; \mathbf{A}\hat{\mathbf{x}} - \rho\hat{\mathbf{x}} \tag{3.3.92}$$



Beweis : siehe [35]

Die Anwendung des Lemmas auf gegebene Intervallgrenzen $[a, b]$ führt direkt auf die Fehlerschranken nach Temple und Kato.

Satz 3.3.7 *(Temple-Kato) Gegeben sei eine Eigenvektornäherung $\hat{\mathbf{x}}$ mit $\hat{\mathbf{x}}^T\hat{\mathbf{x}} = 1.0$ und der dazugehörige Rayleigh-Quotient $\rho(\mathbf{A}, \hat{\mathbf{x}})$. Das Intervall $[a, b]$ enthalte ρ und einen Eigenwert λ aus $\sigma(\mathbf{A})$, dann gilt :*

$$-\frac{\|\mathbf{r}\|_2^2}{\rho - a} \;\leq\; \rho - \lambda \;\leq\; \frac{\|\mathbf{r}\|_2^2}{b - \rho} \tag{3.3.93}$$

Im allgemeinen sind genau einen Eigenwert λ aus $\sigma(\mathbf{A})$ umschließende Intervallgrenzen α und β nicht zuverlässig bestimmbar, da die nächstliegenden Eigenwerte zu λ nur als Näherungen bekannt sind. Die Eigenwertlücke μ (Gl.(3.3.94)) wird deshalb für den praktischen Einsatz der Schranken durch einen Schätzwert δ ersetzt. Die mit dem Schätzwert δ gebildeten Intervallgrenzen umschließen den Rayleigh-Quotienten symmetrisch (3.3.95). Für die Fehlerschranke folgt (3.3.96).

$$\mu \;=\; \min_i |\lambda_i - \rho| \qquad \text{mit} \quad \lambda_i \neq \lambda \tag{3.3.94}$$

$$[\alpha, \beta] \;:=\; [\,\rho - \delta, \rho + \delta\,] \tag{3.3.95}$$

$$|\rho - \lambda| \;\leq\; \frac{\|\mathbf{r}\|_2^2}{\delta} \tag{3.3.96}$$

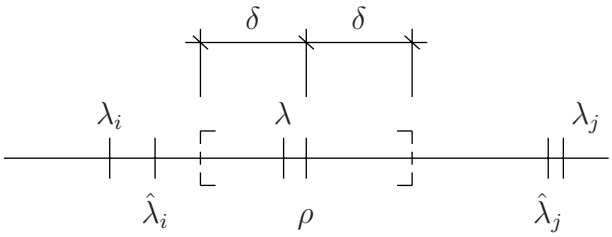


Bild 3.19: Intervallgrenzen nach Temple-Kato

Zur Bestimmung eines geeigneten Schätzwertes δ kommt die Fehlerschranke von Bauer-Fike (Satz 3.3.5) zur Anwendung.

$$\delta \;=\; |\rho - \lambda_i| \;\geq\; |\rho - \hat{\lambda}_i| - |\lambda_i - \hat{\lambda}_i| \tag{3.3.97}$$

$$\geq\; |\rho - \hat{\lambda}_i| - \|\mathbf{r}_i\|_2 \tag{3.3.98}$$

Die rechte Seite von (3.3.98) ist eine berechenbare Größe für den Schätzwert δ, die bei genügend kleiner Restvektornorm $\|\mathbf{r}_i\|_2$ eine zuverlässige obere Fehlerschranke liefert.

Um die Fehlerabschätzung der Eigenwerte unabhängig von der Qualität der berechneten Eigenvektornäherung $\hat{\mathbf{x}}_i$ vornehmen zu können, wird die Restnorm mit der reduzierten Matrix $\hat{\mathbf{D}}$ der Berechnung und den Einheitsvektoren $\mathbf{e}_i$ als Eigenvektornäherung bestimmt.

Schranken für Eigenvektoren

Eine Abschätzung der Güte der Eigenvektornäherung erfolgt in Analogie zu den Fehlerschranken nach Temple-Kato. Betrachtet wird dabei der Winkel θ zwischen der Eigenvektornäherung $\hat{\mathbf{x}}$ und dem Eigenvektor $\mathbf{x}$ zum exakten Eigenwert λ.

Satz 3.3.8 *Gegeben sei die Eigenvektornäherung* $\hat{\mathbf{x}}$ *mit* $\hat{\mathbf{x}}^T\hat{\mathbf{x}} = 1.0$*, der dazugehörige Rayleigh-Quotient* $\rho(\mathbf{A}, \hat{\mathbf{x}})$ *und der Restvektor* $\mathbf{r} = (\mathbf{A} - \rho\,\mathbf{I})\,\hat{\mathbf{x}}$ *.* λ *sei der nächste Eigenwert zu* ρ *und* μ *sei die Lücke zwischen* ρ *und dem nächsten Eigenwert* $\lambda_i \neq \lambda$*. Für den Winkel* θ *zwischen* $\hat{\mathbf{x}}$ *und* $\mathbf{x}$ *gilt :*

$$sin\theta\,(\hat{\mathbf{x}}, \mathbf{x}) \;\leq\; \frac{\|\mathbf{r}\|_2}{\mu} \tag{3.3.99}$$

Beweis : siehe [35]

3.4 Erweiterung des QR-Verfahrens

Es ist der Erhalt der Profilstruktur der Systemmatrizen, der den Einsatz der Methode auf Matrizen großer Dimension überhaupt erst ermöglicht. Gleichungssysteme mit mehreren tausend bis mehreren zehntausend Gleichungen sind bei der Formulierung physikalischer Aufgaben üblich.

Die seit den 60er Jahren stetig anhaltende Weiterentwicklung im Computer-Hardwarebereich hat eine Steigerung von Speicherkapazität und Rechenleistung um etwa den Faktor 100 pro Dekade[2] zur Folge. Trotz dieser rasanten Entwicklung ist der Lösbarkeit großer Gleichungssysteme aus kapazitären und wirtschaftlichen Gründen nach wie vor eine Grenze gesetzt. Diese Grenze wird durch eine effiziente Speicherform unter Ausnutzung von Symmetrie und Struktur der Matrix stark erhöht. Der Faktor, um den der Speicheraufwand und insbesondere der Lösungsaufwand bei Ausnutzung einer Profilstruktur der Matrix verringert werden, wird wesentlich durch die mittlere Bandbreite b der Matrix bestimmt (Bild 3.20). Die Bandbreite ist sehr stark von der geometrischen Dimension der Aufgabe abhängig. Im 2-Dimensionalen beträgt die Bandbreite näherungsweise $\sqrt{N}$, im 3-Dimensionalen näherungsweise $\sqrt[3]{N^2}$ [26].

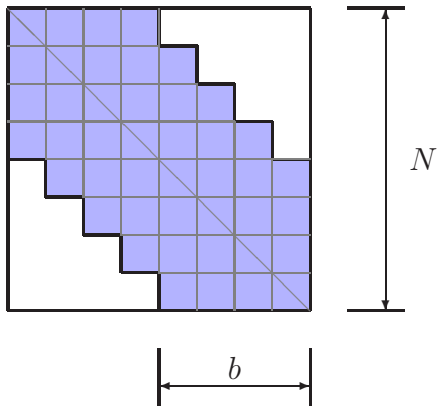


Bild 3.20: Mittlere Bandbreite b

Für Aufgabenstellungen und Lösungsmethoden, die die Lösung eines linearen Gleichungssystems erfordern, haben sich verschiedene Speicherungsmodelle für Profilmatrizen etabliert, die allesamt der Verringerung des Lösungsaufwands Rechnung tragen. Die Verfahren zur Lösung von Eigenwertaufgaben lassen sich im wesentlichen in zwei Gruppen einteilen. Eine Gruppe wird durch die Subraummethoden gebildet, die als wesentlichen Rechenschritt Matrix-Vektor-Multiplikationen erfordern und damit Symmetrie und Matrixstruktur sehr gut ausnutzen können. In der zweiten Gruppe werden Verfahren zusammengefaßt, die auf Ähnlichkeitstransformationen beruhen. Die Matrizeniteration nach Jacobi und das QR-Verfahren bilden die klassischen Vertreter dieser Methoden. Beide Verfahren sind von Natur aus nicht profilerhaltend und zerstören

[2] $\sim 2^{(10 \cdot 12/18)}$ *Moore's Law :* Verdoppelung der Transistorzahl pro Chip, alle 18 Monate

schon nach wenigen Schritten die Struktur der Matrix. Der Speicheraufwand für diese Verfahren beträgt N^2. Der Lösungsaufwand ist proportional zu N^3.

Folgende Erweiterungen des QR-Verfahrens für vollbesetzte Matrizen werden schrittweise eingeführt, untersucht und bewertet :

1. *Erhalt einer konvexen Profilstruktur :* Die Bestimmung der Eigenzustände reeller symmetrischer Profilmatrizen unter Ausnutzung einer konvexen Profilstruktur auf Grundlage des QR-Verfahrens wird gezeigt. Die konvexe Profilstruktur bleibt während der Iteration erhalten. Damit wird der Einsatz dieses Verfahrens für Eigenwertaufgaben mit mehreren tausend Gleichungen aus den typischen strukturmechanischen Problemstellungen des Bauingenieurwesens erst ermöglicht. Die Ausnutzung von Symmetrie und Profil reduziert erheblich den Speicheraufwand des Gleichungssystems und damit den numerischen Aufwand der Berechnung.

2. *Deflation und Spektralverschiebung :* Durch den Einsatz von Spektralverschiebung und der Deflation konvergierter Eigenzustände wird das globale Konvergenzverhalten stark verbessert. Beide Verfahren werden in der Theorie vorgestellt. Ihre positive Auswirkung auf den Berechnungsverlauf wird durch Beweisführung gesichert. Beide Verfahren sind fester Bestandteil aller im folgenden gezeigten Beispielrechnungen.

3. *Prekonditionierung :* Der Konvergenzverlauf der Iteration wird maßgeblich durch die Trennung der Eigenwerte beeinflußt. Zur Verbesserung der Konvergenzrate wird eine Prekonditionierung der iterierten Matrix durch lokale Iterationen in ausgesuchten Matrixbereichen eingeführt. An einem Berechnungsbeispiel werden die Problemstellung, die Lösungsstrategie und das Konvergenzverhalten mit und ohne Prekonditionierung der Matrix analysiert und bewertet.

4. *Jacobi-Randkorrektur :* Matrixbereiche, die von der Prekonditionierung nicht oder nur unzureichend erfaßt werden, stören teilweise stark den Konvergenzfortschritt. Sie werden am unteren Matrixrand durch Jacobi-Transformationen profilerhaltend diagonalisiert. Die Problemstellung, die Berechnungsstrategie, sowie das Konvergenzverhalten mit und ohne Jacobi-Randkorrektur werden an einem Berechnungsbeispiel analysiert und bewertet.

Die Durchführung der Prekonditionierung und der Jacobi-Randkorrektur werden in ihrer Theorie gezeigt und durch eine schematische Dokumentation unterstützt. Die Verbesserung des Konvergenzverhaltens durch die Prekonditionierung und die Jacobi-Randkorrektur werden durch Beispielrechnungen an Matrizen verschiedener Dimensionen verifiziert und diagrammatisch dokumentiert.

3.4.1 Erhalt einer konvexen Profilstruktur

Zur Profilbeschreibung der Matrix werden die Profilvektoren $\mathbf{pl}$ (profile left) und $\mathbf{pr}$ (profile right) eingeführt (Bild 3.21, Seite 53). Mit den beiden profilbeschreibenden Vektoren wird folgende Notation zur Beschreibung der Struktur der Koeffizientenmatrix $\mathbf{A}$ eingeführt :

1. Der Index des ersten Koeffizienten in Zeile i, der von Null verschieden ist wird als linkes Profil $pl[i]$ der Matrix $\mathbf{A}$ bezeichnet.
2. Der Index des letzten Koeffizienten in Zeile i, der von Null verschieden ist wird als rechtes Profil $pr[i]$ der Matrix $\mathbf{A}$ bezeichnet.
3. Wegen der Symmetrie der Koeffizientenmatrix $\mathbf{A}$ entspricht das linke Profil in Zeile i dem oberen Profil $up[i]$ (upper profile) in Spalte i.
4. Wegen der Symmetrie der Koeffizientenmatrix $\mathbf{A}$ entspricht das rechte Profil in Zeile i dem unteren Profil $lp[i]$ (lower profile) in Spalte i. Im folgenden werden die Bezeichnungen $pl[i]$ und $pr[i]$ verwendet.
5. Das Profil von $\mathbf{A}$ wird als konvex bezeichnet, wenn für $m \geq i$ gilt :

   $$pl[m] \geq pl[i] \quad \text{bzw.} \quad pr[m] \geq pr[i]$$

6. Die Anzahl Koeffizienten zwischen $pl[i]$ und i wird als Bandbreite bezeichnet. Der Mittelwert aus der Bandbreite aller Zeilen $i = 1, \ldots, N$ ergibt die mittlere Bandbreite b.

Im allgemeinen ist das Profil einer Matrix nicht vollständig konvex. Die Konvexität der Matrix wird durch eine Erweiterung der Speicherstruktur um wenige Koeffizienten sichergestellt. Im folgenden wird eine konvexe Speicherstruktur von $\mathbf{A}$ vorausgesetzt.

Zerlegung und Rekombination einer konvexen Profilmatrix

Es wird gezeigt, daß wegen der Symmetrie der iterierten Matrix $\mathbf{A}_s$ das Profil von $\mathbf{A}$ in den aufeinanderfolgenden Schritten s und $s+1$ unverändert bleibt.

Im Schritt s wird die Matrix $\mathbf{A}_s$ mit (3.3.8) in das Produkt einer orthonormalen Matrix $\mathbf{Q}$ und einer Rechtsdreiecksmatrix $\mathbf{R}$ zerlegt (Bild 3.22, Seite 54). Die Zerlegung erfolgt durch die schrittweise Reduktion von $\mathbf{A}_s$ auf Dreiecksgestalt mit Hilfe ebener Rotationsmatrizen $\mathbf{P}_{ik}$. Dabei wird Zeile k (blau) dazu verwendet, die Koeffizienten der Spalte k in den Zeilen $i = k+1$ bis $i = pr[k]$ zu eliminieren. Die schrittweise Reduktion wird spaltenweise ausgeführt. Die dabei erzeugten Nullelemente (grün) vorangegangener Spalten bleiben erhalten. Die Transformationen (3.3.10) werden solange fortgesetzt, bis $\mathbf{A}_s$ Rechtsdreiecksgestalt besitzt.

Die Multiplikation von $\mathbf{A}$ mit der Rotationsmatrix $\mathbf{P}_{ik}^T$ zur Elimination des Koeffizienten a_{ik} verändert nur die Koeffizienten der Zeilen i und k (Bild 3.22, Seite 54). Mit $i > k$ erweitert

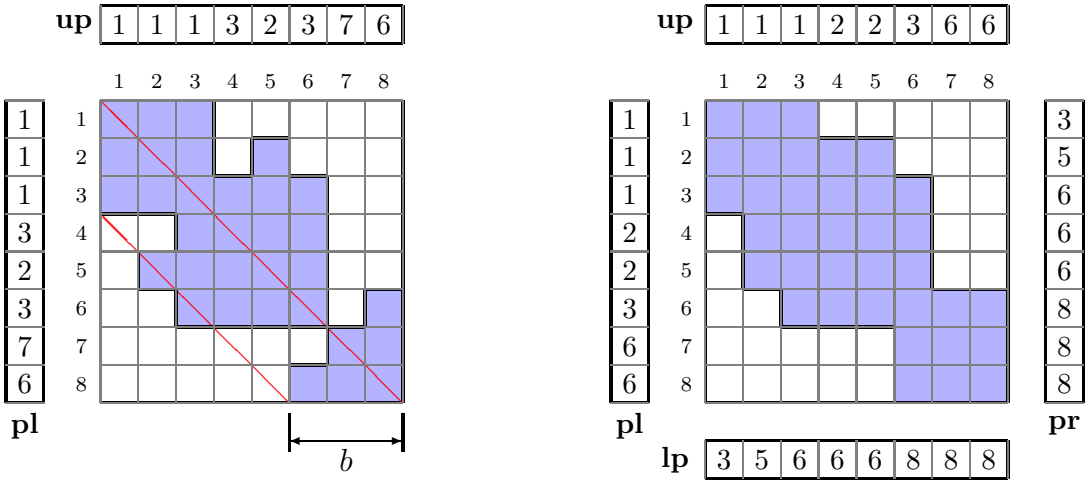


Bild 3.21: Allgemeines und konvexes Profil von $\mathbf{A}$

die Multiplikation $\mathbf{P}_{ik}^T\mathbf{A}$ das Profil in Zeile k von $pr[k]$ auf $pr[pr[k]]$. Wegen der Konvexität des Matrixprofils bleibt das Profil $\mathbf{pr}$ der Zeilen $k+1 \ldots pr[k]$ unverändert. Die Profilerweiterung in Zeile k ist nur für die Dauer der Reduktion von Spalte k erforderlich. Alle nachfolgenden Berechnungsschritte der Zerlegung sind von Zeile und Spalte k unabhängig. Das rechte Matrixprofil wird nicht nachhaltig zerstört.

Für die Transformation von $\mathbf{A}_s$ in $\mathbf{R}_s$ folgt :

$$\mathbf{Q}_s = \mathbf{P}_{21} \ldots \mathbf{P}_{pr[1],1}\mathbf{P}_{32} \ldots \mathbf{P}_{pr[2],2} \ldots \tag{3.4.1}$$

$$\mathbf{R}_s = \ldots \mathbf{P}_{pr[2],2}^T \ldots \mathbf{P}_{32}^T\mathbf{P}_{pr[1],1}^T \ldots \mathbf{P}_{21}^T\mathbf{A}_s \tag{3.4.2}$$

Die Matrix $\mathbf{A}_{s+1}$ bestimmt sich durch Multiplikation von $\mathbf{R}_s$ mit den Rotationsmatrizen $\mathbf{P}_{ik}$ von rechts :

$$\mathbf{A}_{s+1} = \mathbf{R}_s\mathbf{P}_{21} \ldots \mathbf{P}_{pr[1],1}\mathbf{P}_{32} \ldots \mathbf{P}_{pr[2],2} \ldots \tag{3.4.3}$$

Die Berechnung von $\mathbf{A}_{s+1}$ beginnt mit der Rechtsdreiecksmatrix $\mathbf{R}_s$ (Bild 3.23, Seite 55). Die Multiplikation mit $\mathbf{P}_{21}$ zerstört lediglich das Nullelement der Position $(2,1)$ von $\mathbf{R}_s$. Allgemein zerstört die Multiplikation mit $\mathbf{P}_{ik}$ das Nullelement an Position (i,k). Alle anderen Nullelemente bleiben erhalten. Die Reihenfolge der Multiplikationen in (3.4.3) zerstört die Nullelemente $r_{ik}, (k<i)$ spaltenweise, beginnend mit Spalte $k=1$. Das Überschreiben der Nullelemente in Spalte k beginnt in Zeile $k+1$ und endet in Zeile $pr[k]$.

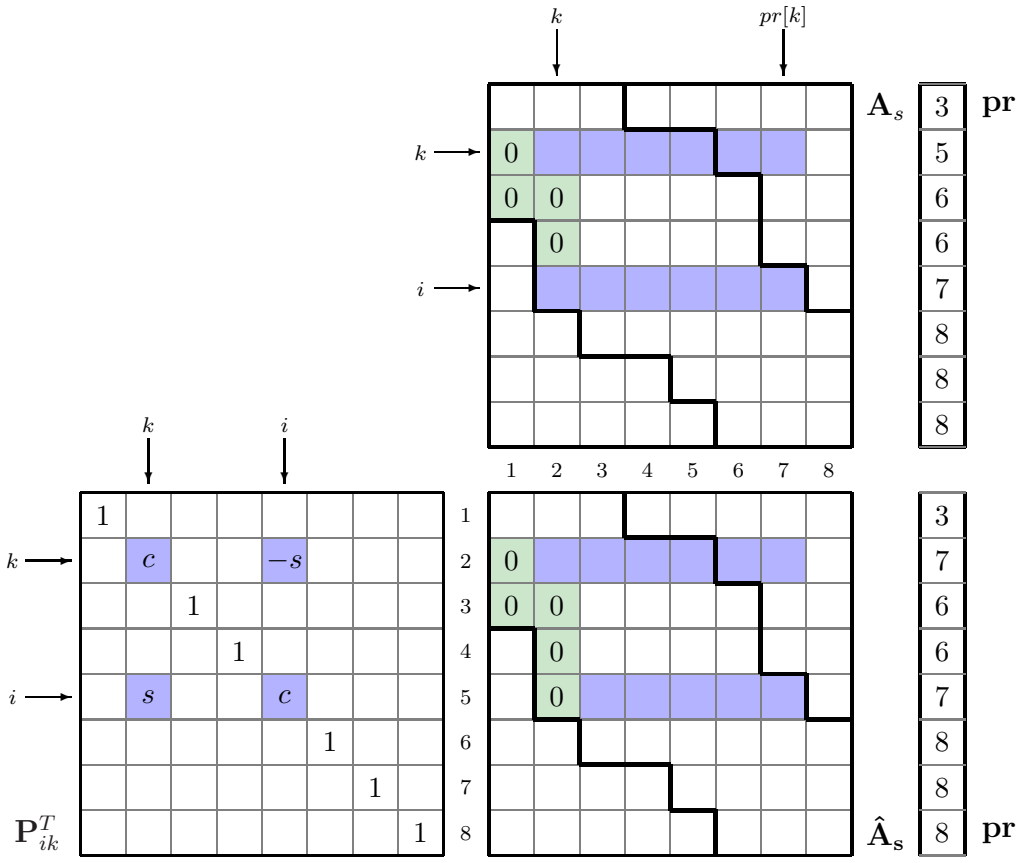


Bild 3.22: Zerlegung der Profilmatrix $\mathbf{A}_s$

Wegen der Symmetrie von $\mathbf{A}_{s+1}$ ist mit der Berechnung der Koeffizienten in Spalte k die explizite Berechnung der Koeffizienten in Zeile k nicht erforderlich. Sie werden mit den Koeffizienten in Spalte k bestimmt : $\hat{a}_{ik} = \hat{a}_{ki}$.

Der hell schattierte Bereich in $\mathbf{R}_s$ (Bild 3.23, Seite 55) markiert die Bereiche der Matrix, in denen zeilenweise und nur temporär Koeffizienten außerhalb des Matrixprofils entstehen, die zur vollständigen Reduktion einer Spalte k erforderlich sind. Sind alle Koeffizienten $a_{mk}, (m > k)$ der Spalte k eliminiert, wird der zusätzliche Speicher in Zeile k wieder frei gegeben. Wegen der Konvexität von $\mathbf{A}$ werden diese Koeffizienten auch nicht zur Berechnung von $\mathbf{A}_{s+1}$ aus dem Zerlegungsprodukt $\mathbf{Q}_s\mathbf{R}_s$ benötigt.

Der temporär zusätzliche Speicheraufwand für die Zerlegung beträgt m Elemente :

$$m \;=\; \max_i \; (pr[pl[i]] - pr[i]) \qquad\qquad i = 1, \ldots, N \tag{3.4.4}$$

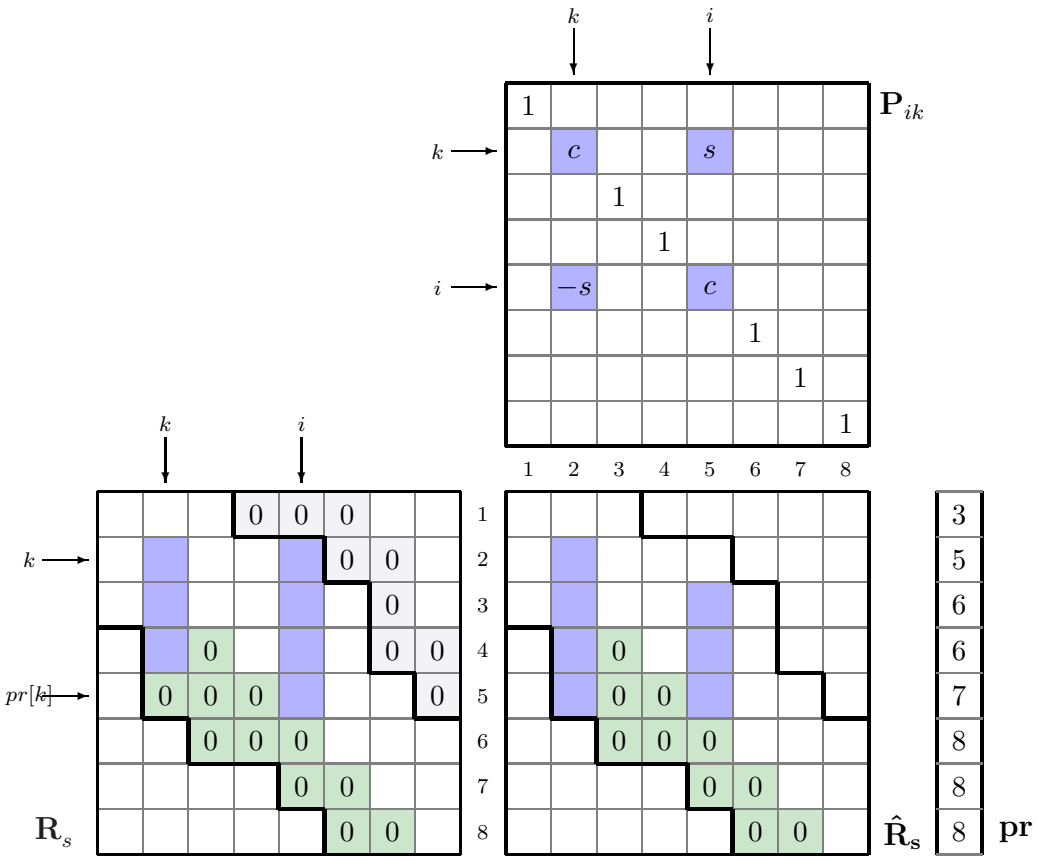


Bild 3.23: Rekombination der Profilmatrix $\mathbf{A}_{s+1}$

3.4.2 Verbesserung des Konvergenzverhaltens

Deflation

Aus dem Konvergenzbeweis (Abschnitt 3.3.2, Abb. 3.7) ist ersichtlich, daß der in 3.3 beschriebene Algorithmus am schnellsten in der letzten Zeile und Spalte der iterierten Matrix $\mathbf{A}_s$ konvergiert, indem die Nichtdiagonalelemente dieser Zeile und Spalte mit zunehmender Schrittzahl s gegen Null streben. Sind die Nichtdiagonalelemente $a_{nk}, n \neq k$ genügend klein, so ist der Diagonalkoeffizient a_{nn} eine gute Näherung zum betragskleinsten Eigenwert $\lambda_n = min(\lambda)$ von $\mathbf{A}$. Eine Spektralverschiebung von $\mathbf{A}_s$ um den Koeffizienten a_{nn} liefert ausschließlich Nullelemente in der letzten Zeile und Spalte. Die Dimension von $\mathbf{A}_s$ wird durch Streichen dieser Zeile und Spalte um eins verringert. Nach Aufhebung der Spektralverschiebung wird die weitere Berechnung an der $(n-1) \times (n-1)$ Matrix $\bar{\mathbf{A}}_s$ fortgeführt. Ist λ_n ein p-facher Eigenwert von $\mathbf{A}$, so wird die Dimension um p Zeilen und Spalten verringert. Die Deflation der iterierten Matrix $\mathbf{A}_s$ ändert nicht das Eigenwertspektrum von $\mathbf{A}$.

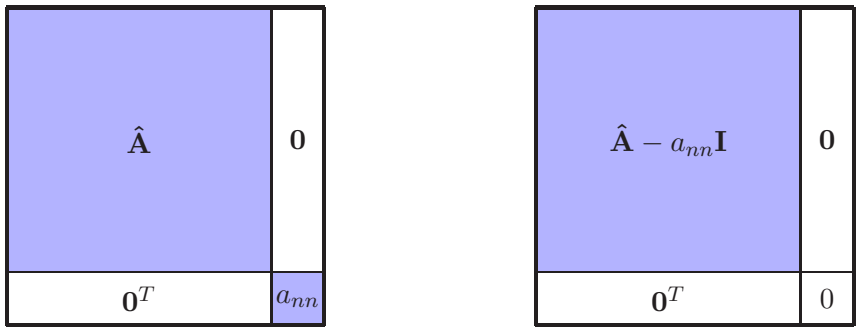


Bild 3.24: Deflation von $\mathbf{A}$ nach Spektralverschiebung um a_{nn}

Satz 3.4.1 *λ_k sei ein Eigenwert von $\mathbf{A}$, $\mathbf{x}_k$ der zugeordnete normierte Eigenvektor und $(\mathbf{A} - \lambda_k \mathbf{x}_k \mathbf{x}_k^T)$ die um den Eigenzustand $(\lambda_k, \mathbf{x}_k)$ reduzierte Matrix $\bar{\mathbf{A}}$. Dann sind die Eigenzustände $(\lambda_i, \mathbf{x}_i)$, $i = 1, \ldots, n, i \neq k$ von $\bar{\mathbf{A}}$ die $n-1$ verbleibenden Eigenzustände von $\mathbf{A}$.*

Beweis : Die reduzierte Matrix $\bar{\mathbf{A}}$ folgt direkt aus Gleichung (3.2.21).

$$\bar{\mathbf{A}} = \mathbf{A} - \lambda_k \, \mathbf{x}_k \, \mathbf{x}_k^T \tag{3.4.5}$$

Mit der Eigendarstellung von $\mathbf{A}$ folgt für die Eigendarstellung der deflationierten Matrix $\bar{\mathbf{A}}$.

$$(\bar{\mathbf{A}} + \lambda_k \, \mathbf{x}_k \, \mathbf{x}_k^T) \, \mathbf{X} = \mathbf{X} \, \mathbf{\Lambda} \tag{3.4.6}$$

$$\bar{\mathbf{A}} \, \mathbf{X} = \mathbf{X} \, \mathbf{\Lambda} - (\lambda_k \, \mathbf{x}_k \, \mathbf{x}_k^T)\mathbf{X} \tag{3.4.7}$$

$$= \mathbf{X} \, (\mathbf{\Lambda} - \lambda_k \, \mathbf{e}_k \, \mathbf{e}_k^T) \tag{3.4.8}$$

Die deflationierte Diagonalmatrix $(\mathbf{\Lambda} - \lambda_k \, \mathbf{e}_k \, \mathbf{e}_k^T)$ enthält die Eigenwerte $\lambda_i, i = 1 \ldots n, i \neq k$ von $\mathbf{A}$ als Diagonalkoeffizienten. Für $i = k$ besitzt λ_i den Wert Null. Die zugeordneten Eigenvektoren $\mathbf{x}_i$ von $\bar{\mathbf{A}}$ sind die Eigenvektoren der Eigenmatrix von $\mathbf{A}$. □

Spektralverschiebung

Aus dem Konvergenzbeweis (3.3.2) folgt, daß die Konvergenzgeschwindigkeit des QR-Verfahrens durch das Verhältnis $|\lambda_{i+1}/\lambda_i|$ bestimmt wird. Für die Konvergenz der Außendiagonalelemente $a_{ik}, k < (i-1)$ kann im allgemeinen eine quadratische Konvergenzrate festgestellt werden. Jedoch führt der Fall schlecht getrennter Eigenwerte bzw. Eigenwerte mit Multiplizität > 1 häufig lokal zu sehr langsamer Konvergenz. In beiden Fällen kann das Konvergenzverhalten lokal und global durch den Einsatz einer geeigneten Spektralverschiebung erheblich verbessert werden. Der Einsatz der Spektralverschiebung führt zur modifizierten Eigenwertaufgabe (3.4.9).

$$(\mathbf{A} - c\,\mathbf{I})\,\mathbf{x} \;=\; \mu\,\mathbf{x} \qquad\qquad \text{mit } \mu \;=\; \lambda - c \tag{3.4.9}$$

Im Schritt s der Iteration besitzt die verschobene Matrix $(\mathbf{A}_s - c\,\mathbf{I})$ die QR-Zerlegung (3.4.10).

$$\mathbf{A}_s - c_s\,\mathbf{I} \;=\; \mathbf{Q}_s\,\mathbf{R}_s \tag{3.4.10}$$

Zur Berechnung von $\mathbf{A}_{s+1}$ wird die Spektralverschiebung wieder aufgehoben. Die Matrizen $\mathbf{A}_s$ und $\mathbf{A}_{s+1}$ sind somit unitär ähnlich zueinander.

$$\mathbf{A}_{s+1} \;=\; \mathbf{R}_s\,\mathbf{Q}_s + c_s\,\mathbf{I} \tag{3.4.11}$$

Ist der Verschiebungsparameter c (*shift*) ein p-facher Eigenwert von $\mathbf{A}$, so ist die Rechtsdreiecksmatrix $\mathbf{R}_s$ singulär. In diesem Fall besitzt $(\mathbf{A}_s - c\,\mathbf{I})$ einen p-fachen Eigenwert Null. Die Zerlegung führt zu der Form aus Bild 3.15 (Seite 39) bzw. Bild 3.24. Die Iteration wird mit reduzierter Dimension $(N - p)$ fortgesetzt.

Zur Untersuchung der Konvergenzverbesserung durch die Spektralverschiebung, wird die verschobene Matrix $(\mathbf{A}_s - c_s\,\mathbf{I})$ in die Zerlegung (3.3.28) substituiert.

$$\begin{aligned}
\mathbf{P}_s\,\mathbf{U}_s &= \mathbf{P}_{s-1}\,(\mathbf{A}_s - c_s\,\mathbf{I})\,\mathbf{U}_{s-1} && (3.4.12)\\
&= (\mathbf{A} - c_s\,\mathbf{I})\,\mathbf{P}_{s-1}\,\mathbf{U}_{s-1} && (3.4.13)\\
&= (\mathbf{A} - c_s\,\mathbf{I})\,(\mathbf{A} - c_{s-1}\,\mathbf{I})\,\mathbf{P}_{s-2}\,\mathbf{U}_{s-2} && (3.4.14)\\
&= \ldots \\
\mathbf{P}_s\,\mathbf{U}_s &= (\mathbf{A} - c_s\,\mathbf{I}) \ldots (\mathbf{A} - c_1\,\mathbf{I}) && (3.4.15)\\
\mathbf{P}_s\,\mathbf{U}_s &= \mathbf{X}\,(\mathbf{\Lambda} - c_s\,\mathbf{I}) \ldots (\mathbf{\Lambda} - c_1\,\mathbf{I})\,\mathbf{X}^T && (3.4.16)
\end{aligned}$$

Mit (3.4.16) werden die Koeffizienten der konvergenzbestimmenden Linksdreiecksmatrix $\mathbf{C}_s$ bestimmt :

$$\mathbf{C}_s = (\mathbf{\Lambda} - c_s\,\mathbf{I}) \ldots (\mathbf{\Lambda} - c_1\,\mathbf{I})\;\mathbf{L}\;(\mathbf{\Lambda} - c_1\,\mathbf{I})^{-1} \ldots (\mathbf{\Lambda} - c_s\,\mathbf{I})^{-1} \tag{3.4.17}$$

$$c_{ik} = l_{ik}\,\frac{(\lambda_i - c_s)\ldots(\lambda_i - c_1)}{(\lambda_k - c_s)\ldots(\lambda_k - c_1)} \tag{3.4.18}$$

Im Schritt s der Iteration wird die Konvergenzgeschwindigkeit nun durch einen neuen, wesentlich kleineren Konvergenzfaktor bestimmt :

$$\left|\frac{\lambda_i - c_s}{\lambda_{i-1} - c_s}\right|^s \ll \left|\frac{\lambda_i}{\lambda_{i-1}}\right|^s \tag{3.4.19}$$

Durch den Einsatz der Spektralverschiebung werden betragsgleiche Eigenwerte mit verschiedenen Vorzeichen von $\mathbf{A}$ mit betragsverschiedenen Eigenwerten von $(\mathbf{A} - c_s\mathbf{I})$ assoziiert und vice versa. Die Konvergenz des Verfahrens ohne Spektralverschiebung zur Diagonalblockform bei einem p-fachen Eigenwert λ_m von $\mathbf{A}$ und einem q-fachen Eigenwert $-\lambda_m$ ist mit einer geeigneten Spektralverschiebung nicht länger problematisch. Nach der Spektralverschiebung sind die Eigenwerte $(\lambda_m - c_s)$ und $(-\lambda_m - c_s)$ im Betrag verschieden, und $(\mathbf{A} - c_s\mathbf{I})$ konvergiert zur Diagonalform. Durch eine ungünstige Wahl des Verschiebungsparameters c_s können jedoch auch betragsgleiche Eigenwerte in $(\mathbf{A} - c_s\mathbf{I})$ induziert werden. Durch die Anpassung des Verschiebungsparameters an die Berechnung in jedem Schritt, bleibt dieser Fall in der Regel ohne größere Auswirkung auf das Konvergenzverhalten.

Die Verbesserung der Konvergenzrate durch die Spektralverschiebung wird wesentlich durch die Wahl des Verschiebungsparameters c_s festgelegt. Aus (3.4.19) ist ersichtlich, daß die beste Konvergenz mit einer Wahl von c_s möglichst nahe am gesuchten Eigenwert λ_i zu erwarten ist.

Die Eigenwerte von $\mathbf{A}$ konvergieren schrittweise in geordneter Reihenfolge (3.3.36). Mit zunehmendem Iterationsverlauf stellt das letzte Diagonalelement a^s_{nn} eine gute Näherung zum betragskleinsten Eigenwert $min|\lambda| = |\lambda_n|$ dar und kann als *shift* eingesetzt werden. Ist a^s_{nn} jedoch eine bessere Näherung zum Eigenwert $|\lambda_k|, k < n$, so wird λ_k vor λ_n konvergieren. Die Sortierung der Eigenwertnäherungen auf der Hauptdiagonalen geht zumindest lokal verloren.

Werden mit der Spektralverschiebung nur wenige $(2m, m \ll \sqrt{N})$ Eigenwerte in der Nähe von λ_k bestimmt, besteht die Gefahr, durch eine schlechte Wahl des *shift*, einen Eigenwert der Folge $\lambda_{k-m},\ \lambda_{k-m+1}, \ldots, \lambda_k, \lambda_{k+1}, \ldots, \lambda_{k+m},$ zu überspringen. Für diese Art der Aufgabenstellung, wie sie beispielsweise bei der Bestimmung des Resonanzbereichs schwingungsgefährdeter Tragwerke eine Rolle spielt, wird eine konservative Shiftstrategie verfolgt. Als Verschiebungsparameter c_s wird jeweils die zuletzt bestimmte Eigenwertnäherung verwendet. Konvergiert innerhalb von 2 Iterationszyklen kein weiterer Eigenwert, wird der Verschiebungsparameter mit dem betragskleinsten Diagonalkoeffizienten neu angepasst. Ein Überspringen von Eigenwerten wird

dadurch in fast allen Fällen vermieden. Die Vollständigkeit des so bestimmten Teilspektrums $\psi(\mathbf{A})$ muß jedoch durch zusätzliche Maßnahmen sichergestellt werden. Mit dieser Shiftstrategie verschlechtert sich teilweise die Konvergenzrate. Mit den Erweiterungen der kommenden Kapitel wird dieser Umstand vollständig aufgehoben.

Durch eine Spektralverschiebung mit dem letzten Diagonalelement $a_{nn}^{(s)}$ in jedem Iterationsschritt $s, s+1, s+2, \ldots$ ist für die Nichtdiagonalelemente $a_{in} = a_{ni} (n \neq i)$ der letzten Zeile und Spalte ein kubisches Konvergenzverhalten gegen Null zu beobachten. Lokal ist die Konvergenz häufig besser als kubisch.

Zum Beweis der kubischen Konvergenz der Nichtdiagonalelemente der letzten Zeile und Spalte n wird die Ähnlichkeitstransformation $\mathbf{Q}_s^T(\mathbf{A}_s - a_{nn}\mathbf{I})\mathbf{Q}_s$ im Schritt s der Iteration betrachtet. Es wird gezeigt, daß die Euklidische Norm $\|\hat{\mathbf{a}}\|_2$ in Zeile und Spalte n der iterierten Matrix kubisch gegen Null strebt.

Die Koeffizienten der letzten Zeile und Spalte seien mit einem Faktor $\epsilon < 1.0$ behaftet.

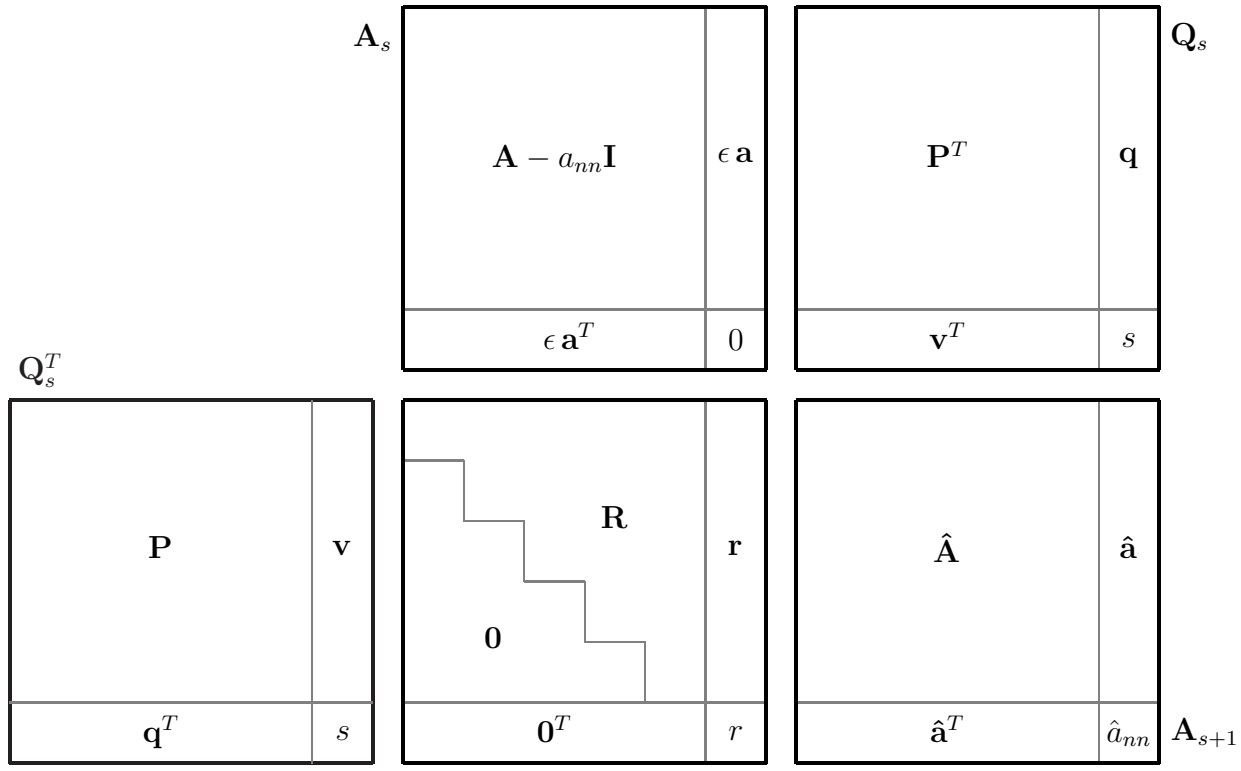


Bild 3.25: Ähnlichkeitstransformation im Schritt s

Wegen $\mathbf{Q}_s\mathbf{Q}_s^T = \mathbf{Q}_s^T\mathbf{Q}_s = \mathbf{I}$, gilt für die letzte Spalte und Zeile von $\mathbf{Q}_s$:

$$\mathbf{q}^T\mathbf{q} + s^2 = \mathbf{v}^T\mathbf{v} + s^2 = 1.0 \tag{3.4.20}$$

$$\|\mathbf{q}\|_2 = \|\mathbf{v}\|_2 \leq 1.0 \tag{3.4.21}$$

Die Koeffizienten von $\hat{\mathbf{a}}$ der iterierten Matrix $\mathbf{A}_{s+1}$ ergeben sich zu :

$$\hat{\mathbf{a}} = r\,\mathbf{v} \tag{3.4.22}$$

Der Diagonalkoeffizient r folgt aus dem Produkt $\mathbf{Q}_s^T\,(\mathbf{A}_s - a_{nn}\mathbf{I})$:

$$\mathbf{q}^T\,\mathbf{a}\,\epsilon = r \tag{3.4.23}$$

Aus der Zerlegung $\mathbf{A}_s = \mathbf{Q}_s\mathbf{R}_s$ folgt für die Norm des Vektors $\mathbf{v}$:

$$\mathbf{v}^T = \epsilon\,\mathbf{a}\,\mathbf{R}^{-1} \tag{3.4.24}$$

$$\|\mathbf{v}\|_2 \leq \epsilon\,\|\mathbf{a}\|_2\,\|\mathbf{R}^{-1}\|_2 \tag{3.4.25}$$

Mit den Gleichungen (3.4.23), (3.4.21) und (3.4.25) folgt für r :

$$r \leq \|\mathbf{q}\|_2\,\|\mathbf{a}\|_2\,\epsilon \tag{3.4.26}$$

$$\leq \epsilon^2\,\|\mathbf{a}\|_2^2\,\|\mathbf{R}^{-1}\|_2 \tag{3.4.27}$$

Mit Gleichung (3.4.27) substituiert in Gleichung (3.4.22) folgt für die Konvergenz der letzten Zeile und Spalte von $\mathbf{A}_{s+1}$ gegen Null :

$$\|\hat{\mathbf{a}}\|_2 \leq \epsilon^3\,\|\mathbf{a}\|_2^3\,\|\mathbf{R}^{-1}\|_2^2 = \epsilon\,\|\mathbf{a}\|_2\,\|\mathbf{v}\|_2^2 \leq \epsilon\,\|\mathbf{a}\|_2 \tag{3.4.28}$$

Für eine ausreichend kleine Norm von $\mathbf{a}$, konvergiert das Verfahren damit, in der letzten Zeile und Spalte der iterierten Matrix $\mathbf{A}_{s+1}$, mindestens kubisch.

□

Prekonditionierung

Die Matrix $\mathbf{C}_s$ (Gleichung (3.3.35), Bild 3.7, Seite 31) zeigt, daß die für das Konvergenzverhalten kritischen Koeffizienten die Subdiagonalelemente $a_{i+k,i}, k \ll i$ sind. Insbesondere im Fall schlecht getrennter Eigenwerte oder betragsgleicher Eigenwerte ist zu beobachten, daß sich trotz Einsatz der Spektralverschiebung Diagonalblöcke geringer Größe ausbilden, für die ein sehr langsamer Konvergenzfortschritt zu verzeichnen ist. Ein monotoner Konvergenzfortschritt wird dadurch lokal gestört. Die Folge dieser Konvergenzverzögerung ist eine hohe Diagonaldominanz, vorwiegend im untersten Drittel der Matrix, jedoch eine schlechte Konvergenzrate der zu bestimmenden Eigenwerte.

Um diesem Verhalten entgegenzuwirken werden die kritischen Koeffizienten der letzten Zeilen und Spalten wiederholt durch eine Prekonditionierung der Matrix auf Null getrieben. Im nachfolgenden Iterationsschritt werden diese Koeffizienten überschrieben, bleiben jedoch im Betrag klein. Die zunehmende Diagonaldominanz der iterierten Matrix $\mathbf{A}_s$ wird durch die Prekonditionierung zusätzlich verstärkt. Die Diagonalkoeffizienten sind in absteigender Reihenfolge als Näherungen zu den gesuchten Eigenwerten identifizierbar. Das Konvergenzverhalten wird damit lokal und global wesentlich verbessert.

Problemanalyse Das Konvergenzverhalten in Gegenwart schlecht getrennter Eigenwerte wird im folgenden an einem Beispiel analysiert. Eine lokale Stagnation der Iteration ist trotz Einsatz der Spektralverschiebung nicht vermeidbar. Es wird gezeigt, daß der beobachtete Effekt in weiten Teilen der Matrix mit dem im folgenden behandelten Verfahren effektiv beseitigt werden kann.

Als Untersuchungsbeispiel dient die Eigenwertaufgabe zur Schwingungsanalyse einer Quadratplatte mit 47 Freiheitsgraden. Eine vollständige Modellbeschreibung ist für eine analoge Modellaufgabe mit 5043 Freiheitsgraden in Beispiel 6 (Seite 94) gezeigt.

Beispiel 3 : Einfach gelagerte Quadratplatte mit 47 Freiheitsgraden

Die Platte besitzt eine knotenkonzentrierte Massenbelegung und ist symmetrisch elementiert. Ein Viertel der Eigenwerte besitzt Multiplizität 2. Tabelle 3.1 (Seite 64) zeigt das vollständige Eigenwertspektrum der Aufgabe. Die Konvergenzschranke für Nichtdiagonalelemente beträgt $4.e-14$. Das statistische Mittel $\bar{\delta}_{rel}$ der relativen Fehlernorm der Eigenzustände und die Standardabweichung $\breve{\delta}_{rel}$ geben einen Eindruck über das Fehlerniveau der Berechnung.

$$\begin{aligned} \bar{\delta}_{rel} &= 1.295e-11 \qquad mit \quad \delta_{rel} = \|\mathbf{r}_i\|_2/\lambda_i \\ \breve{\delta}_{rel} &= 3.146e-11 \end{aligned}$$

Bild 3.26 zeigt den Konvergenzverlauf mit Spektralverschiebung und Deflation. Die vollständige Bestimmung des Eigenwertspektrums erfordert 97 Zyklen.

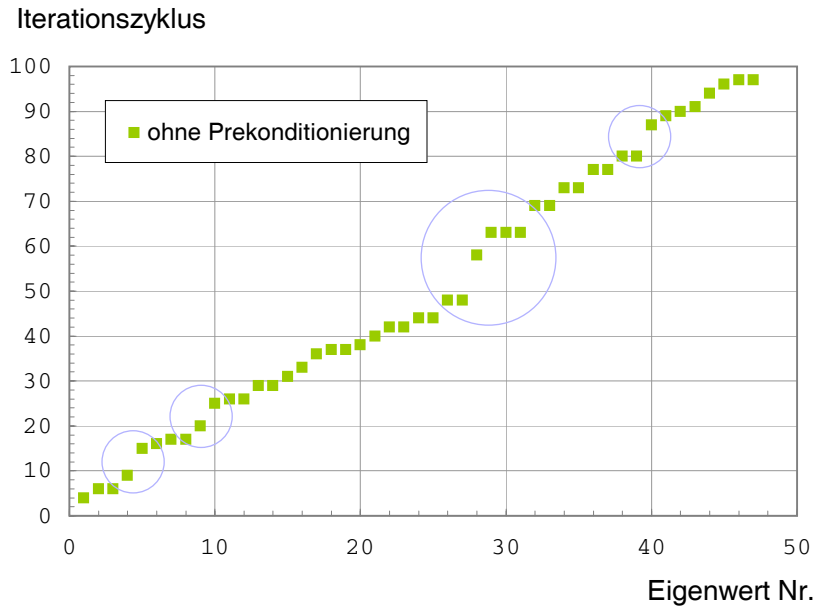


Bild 3.26: Iterationsverlauf ohne Prekonditionierung

Die Mehrzahl der berechneten Eigenwerte wird durch den Einsatz der Spektralverschiebung mit wenigen Zyklen (≤ 3) bestimmt. Für einige wenige Eigenwerte jedoch ist für eine ausreichende Konvergenz eine größere Anzahl von Zyklen erforderlich.

Die ersten vier Eigenwerte sind nach 9 Zyklen bestimmt. Für den fünften Eigenwert sind 6 weitere Zyklen erforderlich. Eine ähnliche Stagnation der Konvergenz ist u.a. für die Bestimmung der Eigenwerte λ_{10}, λ_{28}, λ_{32} und λ_{40} zu beobachten. In allen Fällen ist die Matrix nach der Deflation des zuletzt bestimmten Eigenwerts um den letzten Diagonalkoeffizienten a_{nn} spektralverschoben. Der Verschiebungsparameter ist bereits eine gute Näherung zum gesuchten Eigenwert λ_n. Die schlechte Trennung von λ_n und den nachfolgenden k Eigenwerten (z.B. λ_{32} bis λ_{35}), verhindert jedoch den gewünschten Konvergenzfortschritt. Die Konvergenz der Nichtdiagonalelemente gegen Null in diesen Bereichen stagniert.

Bild 3.27 zeigt die Submatrix $\tilde{\mathbf{A}}$ der letzten 6 Zeilen und Spalten im Zyklus 63 nach der Deflation um den Eigenwert $\lambda_{31} = 304.35696348$. Die Matrix ist um den Parameter $c = 355.31914769$ spektralverschoben. Die Nichtdiagonalkoeffizienten der letzten 4 Zeilen und Spalten sind groß, gemessen an den Diagonalkoeffizienten. Die Submatrix ist nicht diagonaldominant und bedarf 6 Zyklen zur vollständigen Diagonalisierung.

[11]	[12]	[13]	[14]	[15]	[16]
64.47958032	0.00259335	−0.00032174	0.00000883	0.00000002	0.00000005
0.00259335	64.48209773	−0.00025917	−0.00002411	−0.00000008	0.00000007
−0.00032174	−0.00025917	0.01966355	0.01938563	0.00514686	0.01940750
0.00000883	−0.00002411	0.01938563	0.01618533	0.01380275	−0.01596565
0.00000002	−0.00000008	0.00514686	0.01380275	0.03438046	0.01001923
0.00000005	0.00000007	0.01940750	−0.01596565	0.01001923	0.00000000

Bild 3.27: Submatrix $\tilde{\mathbf{A}}$ im Zyklus 63 nach Deflation um λ_{31}

Lösungsstrategie Die Diagonalisierung schlecht konvergierender Matrixbereiche mit dem vorgestellten QR-Verfahren ist numerisch teuer, da der Aufwand für eine vollständige Zerlegung und Rekombination der Matrix einen nur sehr schwachen Konvergenzfortschritt verursacht. Die Gründe für diese Stagnation liegen in der schlechten Trennung von Eigenwerten und sind mit dem konvergenzbestimmenden Faktor $|\frac{\lambda_i - c}{\lambda_k - c}|^s$ (Gleichung 3.4.19) zu erklären.

Zur Beseitigung der Stagnation wird im folgenden eine Prekonditionierung der Matrix durchgeführt. Die Prekonditionierung erfaßt blockweise die Zeilen und Spalten der Matrix und führt zu einer starken Diagonaldominanz der kritischen Bereiche um die Hauptdiagonale. Die Problemkoeffizienten werden in großen Bereichen der Matrix eliminiert und bleiben in den nachfolgenden Schritten im Betrag klein.

[11]	[12]	[13]	[14]	[15]	[16]
64.47958032	0.00259335	0.00015638	0.00016461	0.00022784	0.00001143
0.00259335	64.48209773	0.00010989	0.00014763	0.00018138	0.00003125
0.00015638	0.00010989	−0.02789819	−0.00000000	−0.00000000	0.00000000
0.00016461	0.00014763	−0.00000000	0.05082269	0.00000000	0.00000000
0.00022784	0.00018138	−0.00000000	0.00000000	0.02365242	−0.00000000
0.00001143	0.00003125	0.00000000	0.00000000	−0.00000000	0.02365241

Bild 3.28: Submatrix $\tilde{\mathbf{A}}$ im Zyklus 63 nach Prekonditionierung

Bild 3.28 zeigt die Submatrix $\tilde{\mathbf{A}}$ im Zyklus 63 nach der Prekonditionierung der letzten 4 Zeilen und Spalten. Die Problemkoeffizienten um die Hauptdiagonale sind alle null und stören nicht weiter den Konvergenzfortschritt. Außerhalb des Subblocks der Zeilen und Spalten 13 bis 16 sind die Koeffizienten im Betrag größer, als vor der Prekonditionierung. Diese Elemente stören nicht weiter, da sie im allgemeinen in der nachfolgenden QR-Zerlegung fast vollständig auf Null getrieben werden. Der Faktor $|\frac{\lambda_i - c}{\lambda_k - c}|^s$ (Gleichung 3.4.19) dieser Elemente ist kleiner $1e-4$.

λ_1	=	1.377305498545	λ_{17}	=	92.490330594375	λ_{33}	=	355.342800108939
λ_2	=	7.801599495025	λ_{18}	=	93.589912564165	λ_{34}	=	355.369970379000
λ_3	=	7.801599495025	λ_{19}	=	93.589912564165	λ_{35}	=	355.291249494754
λ_4	=	16.309593950464	λ_{20}	=	106.794056049899	λ_{36}	=	419.800605082411
λ_5	=	24.934276415090	λ_{21}	=	118.912948033697	λ_{37}	=	419.800605082411
λ_6	=	24.288929347640	λ_{22}	=	129.098758033255	λ_{38}	=	419.805138802000
λ_7	=	32.520219682477	λ_{23}	=	129.098758033254	λ_{39}	=	419.793616454650
λ_8	=	32.520219682477	λ_{24}	=	138.581404511926	λ_{40}	=	479.536082421288
λ_9	=	57.378609451969	λ_{25}	=	138.581404511926	λ_{41}	=	482.514950910404
λ_{10}	=	53.650068439340	λ_{26}	=	147.027481620997	λ_{42}	=	482.514950910404
λ_{11}	=	60.466825596977	λ_{27}	=	148.637245094030	λ_{43}	=	484.653973174928
λ_{12}	=	60.466825596977	λ_{28}	=	304.339773278665	λ_{44}	=	494.497234527833
λ_{13}	=	78.226000012205	λ_{29}	=	304.348057996252	λ_{45}	=	496.108866005962
λ_{14}	=	78.226000012205	λ_{30}	=	304.348057996251	λ_{46}	=	496.108866005962
λ_{15}	=	77.762732898434	λ_{31}	=	304.356963480576	λ_{47}	=	498.450360838359
λ_{16}	=	83.142059753463	λ_{32}	=	355.342800108938			

Tabelle 3.1: Eigenwertspektrum $\sigma(\mathbf{K}_{47})$

Durchführung der Prekonditionierung Die Prekonditionierung wird durch Ähnlichkeitstransformationen mit orthonormalen Matrizen $\mathbf{U}_k$ ausgeführt. $\mathbf{U}_k$ unterscheidet sich von einer Identitätsmatrix durch einen orthonormalen Block der Dimension $(p \times p)$ auf der Hauptdiagonalen und ändert durch die Transformation (3.4.29) nur die Koeffizienten a_{im} in den Zeilen und Spalten $i, i+1, \ldots, p$ bzw. $m, m+1, \ldots, p$ der iterierten Matrix $\mathbf{A}_s$.

$$\hat{\mathbf{A}}_s \;=\; \mathbf{U}_k^T\, \mathbf{A}_s\, \mathbf{U}_k \qquad\qquad k \;=\; 1, \ldots, (\ll n) \tag{3.4.29}$$

Zur Bestimmung der Blockdimension p der Transformationsmatrizen $\mathbf{U}_k$ wird $\mathbf{A}_s$ in k Zeilen- und Spaltenbereiche eingeteilt, deren linkes und rechtes Profil bzw. oberes und unteres Profil über p Zeilen und Spalten konstant ist.

Jeder der k Bereiche führt zu einem Diagonalblock $\mathbf{A}_k^{p\times p}$ auf der Hauptdiagonalen von $\mathbf{A}_s$ (Bild 3.29). Die in ihrer Dimension kleine Matrix $\mathbf{A}_k^{p\times p}$ wird mit geringem Aufwand diagonalisiert.

$$\mathbf{A}_k^{p\times p} \;=\; \mathbf{V}_k\, \mathbf{D}_k\, \mathbf{V}_k^T \tag{3.4.30}$$

Die orthonormale Transformationsmatrix $\mathbf{V}_k^{p\times p}$ wird dazu verwendet, die p Zeilen und Spalten des mit $\mathbf{A}_k^{p\times p}$ assoziierten Blocks in $\mathbf{A}_s$ zu transformieren. Die Diagonalisierung von $\mathbf{A}_k^{p\times p}$ wird mit dem QR-Algorithmus für vollbesetzte Matrizen durchgeführt. Eine Diagonaldominanz von

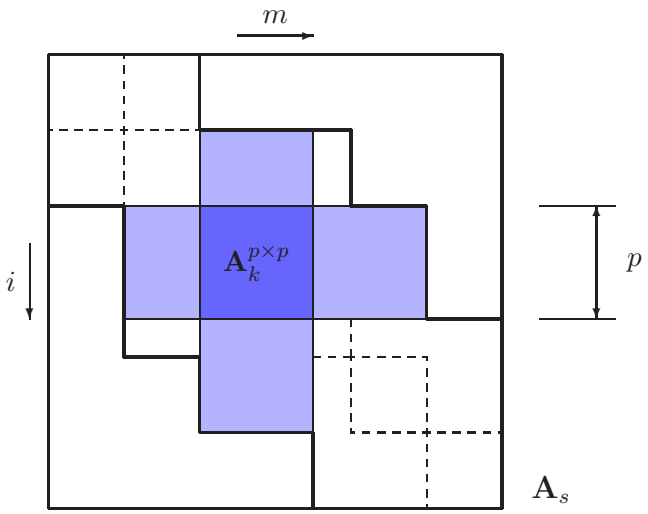


Bild 3.29: Einteilung von $\mathbf{A}_s$ in Blockbereiche mit konstantem Profil

$\mathbf{A}_k^{p\times p}$ mit Nichtdiagonalelementen $|a_{ik}| < (10^{-3}\,\|\mathbf{A}_k^{p\times p}\|_F/p)$ ist bereits ausreichend um die Prekonditionierung effektiv ausführen zu können.

Die Prekonditionierung zerstört nicht das Profil der Matrix (Bild 3.30). Multiplikationen außerhalb des Profils sind aufgrund des konstanten Profils der p Zeilen und Spalten null. Eine Änderung der Reihenfolge der Eigenwertnäherungen auf der Hauptdiagonalen von $\mathbf{A}_s$ wird durch die Prekonditionierung nicht verursacht, aber durch die frühe Diagonaldominanz begünstigt. Eine Umsortierung in eine absteigende Reihenfolge ist nach wenigen QR-Zerlegungen bei fortschreitender Konvergenz wieder hergestellt.

Ergebnisanalyse Der Einsatz der Prekonditionierung beseitigt die Stagnation der Konvergenz in den von den Transformationen gut erfaßten Bereichen vollständig. Bild 3.31 zeigt den Iterationsverlauf mit Einsatz der Spektralverschiebung und der Prekonditionierung. Das Konvergenzverhalten wurde lokal und global verbessert. Die Gesamtanzahl erforderlicher Zyklen ist von 97 auf 80 gesunken.

Bild 3.32 zeigt die von der Prekonditionierung erfaßten Bereiche. Die Diagonalkoeffizienten in den Zeilen und Spalten der schlecht konvergierenden Eigenwerte λ_5, λ_{10}, λ_{28}, λ_{32} und λ_{40} sind dunkel markiert. Infolge ihrer konvexen Profilstruktur ergeben sich für die Beispielmatrix $\mathbf{K}_{47}$ maximale Blockgrößen der Dimension 4.

Der Vergleich von Bild 3.32 mit dem Konvergenzverlauf in Bild 3.31 zeigt, daß die Stagnation der Konvergenz der Eigenwerte λ_5 und λ_{10} durch die Prekonditionierung beseitigt worden ist. Die Diagonalkoeffizienten dieser Zeilen und Spalten profitieren von ihrer zentralen Lage in den prekonditionierten Bereichen. Der Eigenwert λ_5 ist nur vom Eigenwert λ_6 mäßig getrennt, sodaß lediglich der Nichtdiagonalkoeffizient $a_{n-5,n-6}$ als kritisch betrachtet werden muß. Der

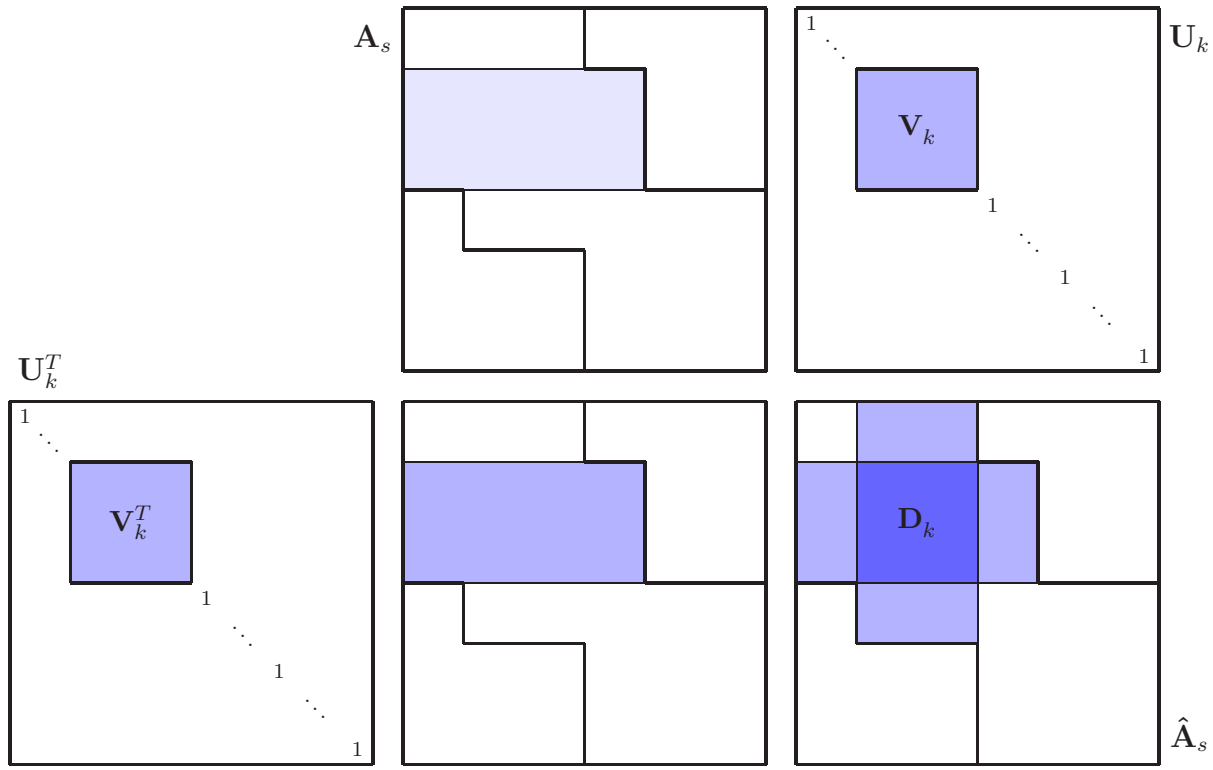


Bild 3.30: Prekonditionierung des k-ten Blocks

Eigenwert λ_{10} ist von den anderen Eigenwerten gut getrennt. Infolge der Spektralverschiebung wurde die natürliche Bestimmungsreihenfolge ($\lambda_k \leq \lambda_{k+1}$) der Eigenwerte $\lambda_9 = 53.65006844$ und $\lambda_{10} = 57.37860945$ vertauscht. Die neue Bestimmungsreihenfolge ist $\lambda_9 = 57.37860945$, $\lambda_{10} = 53.65006844$. Während der Konvergenz von λ_9 sind die Nebendiagonalkoeffizienten in Zeile und Spalte $(n-10)$ infolge des Eigenwertverhältnisses $(\frac{\lambda_k}{\lambda_{k+1}}) > 1$ divergiert und somit groß ($|a_{ik}| \sim |a_{kk}|, k = i-1$). Ohne Prekonditionierung erfordert eine ausreichende Konvergenz des Eigenwertes λ_{10}, 5 Zyklen. Mit Prekonditionierung reduziert sich die Anzahl erforderlicher Zyklen auf 2.

Die stagnierende Konvergenz der Eigenwerte λ_{28}, λ_{32} und λ_{40} ist auf die schlechte Trennung der Eigenwerte in diesen Bereichen zurückzuführen. Bild 3.32 (Seite 68) zeigt die betreffenden Diagonalkoeffizienten rot markiert. Die Eigenwerte λ_{28} und λ_{32} sind jeweils Teil eines Clusters der Dimension 4 schlecht getrennter Eigenwerte (vgl. Tabelle 3.1, Seite 64). Die Prekonditionierung erfaßt nur jeweils zwei der drei grün markierten Problemkoeffizienten in den Zeilen 28 und 32. Der Diagonalkoeffizient in Zeile 40 wird aufgrund des Matrixprofils von der Prekonditionierung nicht erfaßt.

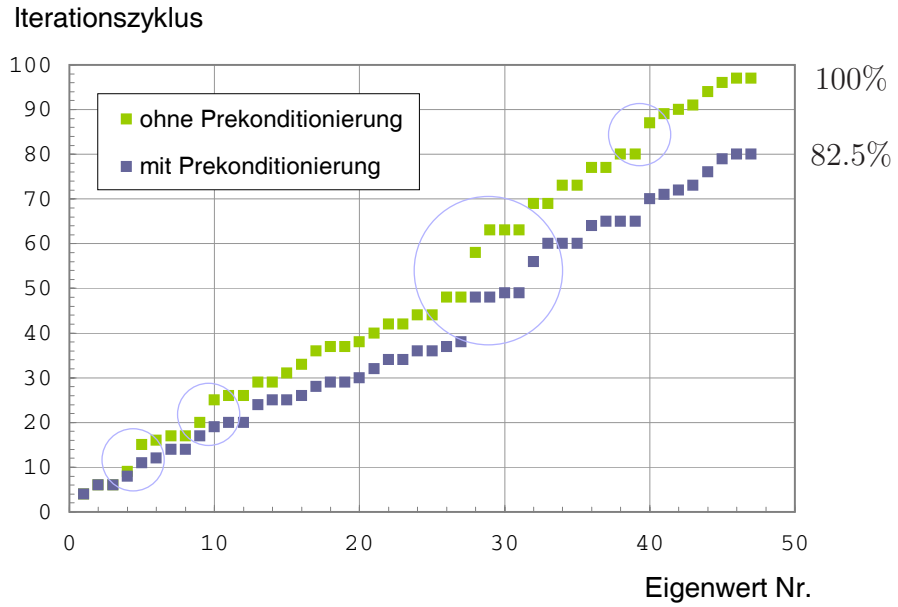


Bild 3.31: Iterationsverlauf der Matrix $\mathbf{K}_{47}$

Bewertung der Prekonditionierung Insgesamt ist festzustellen, daß die Prekonditionierung lokale Konvergenzstörungen teilweise beseitigt und das globale Konvergenzverhalten nachhaltig verbessert. Bild 3.31 zeigt, daß der positive Effekt der Prekonditionierung bereits nach wenigen Zyklen einsetzt und damit insbesondere den Aufwand der numerisch teuren Zerlegungen der Anfangsphase stark reduziert. Eine unzureichende Diagonaldominanz der prekonditionierten Bereiche macht dieses Verfahren jedoch wirkungslos, da die erzeugten Nullelemente durch die nachfolgende QR-Zerlegung mit betragsgroßen Elementen überschrieben werden. Dieser Umstand ist in den ersten Zyklen der Iteration für die gesamte Matrix gegeben und ist bei großen Matrizen auch im fortgeschrittenen Iterationsverlauf für die am schwächsten konvergierenden Bereiche zu beobachten. Eine Prekonditionierung der vollständigen Matrix hat im Vergleich zur ausschließlichen Prekonditionierung der stark konvergierenden Bereiche einen nur sehr geringen Effekt. Zur Reduzierung des numerischen Aufwands werden bei der praktischen Anwendung des Verfahrens lediglich die Zeilen und Spalten im untersten Sechstel der Matrix prekonditioniert. Durch die schrittweise Deflation der berechneten Eigenwerte und der damit verbundenen Verringerung der Matrixdimension wird die Prekonditionierung stufenweise in alle Bereiche der Matrix eingetragen.

Tabelle 3.2 zeigt Vergleichsrechnungen verschiedener Matrizen mit und ohne Prekonditionierung. Spalte 1 enthält die Dimension der berechneten Matrix, Spalte 2 die mittlere Bandbreite und Spalte 3 die Anzahl berechneter Eigenwerte. In Spalte 4 und 5 ist der multiplikative Aufwand der Berechnung ohne und mit Prekonditionierung aufgeführt. Spalte 6 enthält den multiplikativen Aufwand der Prekonditionierung. Spalte 7 zeigt den prozentualen Aufwand der Berechnung mit Prekonditionierung bezogen auf die Berechnung ohne Prekonditionierung. Die

1	2	3	4	5	6	7
N	b	k	FLOP o. Prekond.	FLOP m. Prekond.	FLOP Prekond.	verb. Aufwand
183	23	183	$0.10 \cdot 10^9$	$0.09 \cdot 10^9$	$0.0051 \cdot 10^9$	93.9 %
567	40	567	$0.31 \cdot 10^{10}$	$0.29 \cdot 10^{10}$	$0.0068 \cdot 10^{10}$	91.4 %
943	52	943	$0.14 \cdot 10^{11}$	$0.12 \cdot 10^{11}$	$0.0022 \cdot 10^{11}$	88.8 %
2647	89	2647	$0.31 \cdot 10^{12}$	$0.27 \cdot 10^{12}$	$0.0023 \cdot 10^{12}$	86.7 %
5727	131	5727	$0.31 \cdot 10^{13}$	$0.26 \cdot 10^{13}$	$0.0015 \cdot 10^{13}$	85.8 %
10687	179	10687	$0.20 \cdot 10^{14}$	$0.17 \cdot 10^{14}$	$0.0007 \cdot 10^{14}$	85.5 %

Tabelle 3.2: Vergleich der Iteration mit und ohne Prekonditionierung

Ergebnisse aus Tabelle 3.2 und weitere sind diagrammatisch in Bild 3.41 (Seite 77) dargestellt. Alle Beispiele wurden mit Einsatz der Spektralverschiebung gerechnet. Als Schranke für die Konvergenz der Nichtdiagonalelemente wurde $(\|\mathbf{A}\|_F \cdot \epsilon)$, mit ϵ = Maschinengenauigkeit, gewählt. Die maximale Blockgröße der Prekonditionierung war für alle Beispiele 3.

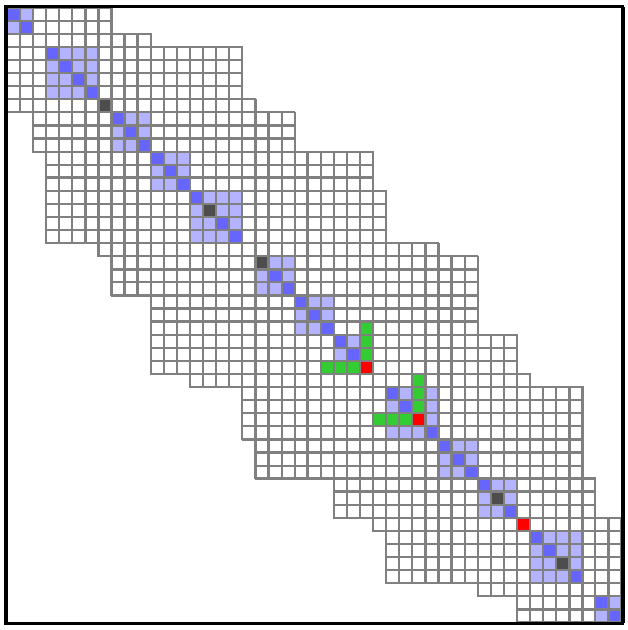


Bild 3.32: Diagonalblöcke $\mathbf{A}_k^{(p\times p)}$ der Beispielmatrix $\mathbf{K}_{47}$

Jacobi-Randkorrektur

Verschiedene Bereiche der Matrix werden von der Prekonditionierung nicht erfaßt. Bedingt durch eine unregelmäßige Profilstruktur existieren Bereiche mit Diagonalblockdimension $p = 1$ (Bild 3.33, links). Eine Prekonditionierung mit Transformationsmatrizen $\mathbf{V}^{p\times p}, p \geq 2$ in diesen Bereichen zerstört die Profilstruktur. Andere Bereiche liegen zwischen dem Profilrand und dem Diagonalblock $\mathbf{A}^{p\times p}$ und profitieren nicht von seinen Eigeninformationen (Bild 3.33, rechts). Diese von der Prekonditionierung nicht bzw. schlecht erfaßten Bereiche verursachen teilweise eine massive Stagnation im Konvergenzverlauf (Bild 3.31). Mit der Konvergenz der Problemkoeffizienten werden meist mehrere Eigenwerte im gleichen Zyklus bestimmt, trotzdem steigt der numerische Aufwand für die Bestimmung dieser Eigenwerte überproportional. Um dieser Stagnation entgegenzuwirken werden die kritischen Bereiche bei Bedarf durch Jacobi-Transformationen behandelt.

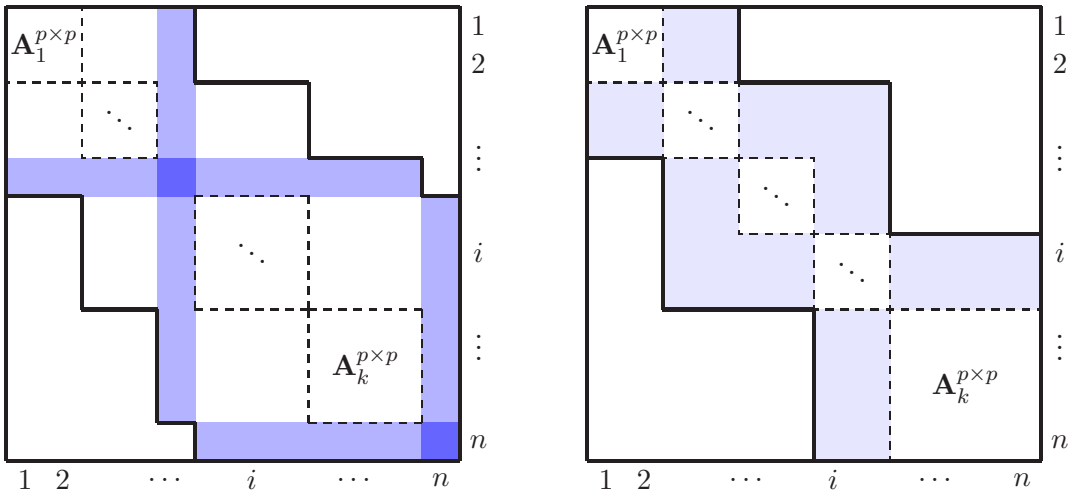


Bild 3.33: Nicht erfaßte bzw. schlecht erfaßte Bereiche der Prekonditionierung

Problemanalyse Die Stagnation im Konvergenzverlauf wird an der Steifigkeitsmatrix $\mathbf{K}_{47}$ aus Beispiel 3 des vorangegangenen Abschnitts analysiert und behandelt.

Beispiel 4 : Konvergenzverlauf ohne Jacobi-Randkorrektur

Im Iterationszyklus 38 sind bereits 27 Eigenwerte bestimmt. Die Dimension der deflationierten Matrix ist 20. Bild 3.31 (Seite 67) zeigt den Iterationsverlauf der Berechnung mit Spektralverschiebung und Prekonditionierung. Bild 3.34 zeigt die

letzen 6 Zeilen und Spalten der Koeffizientenmatrix $\mathbf{A}$ im Zyklus 38. Die Matrix ist um den Parameter $c = 304.34394283$ spektralverschoben. Der Einfluß der Prekonditionierung ist in den Spalten 15 bis 18 deutlich zu erkennen. Die Zeilen und Spalten 19 und 20 werden aufgrund eines Profilwechsels von der Prekonditionierung nicht erfaßt.

[15]	[16]	[17]	[18]	[19]	[20]
50.97304398	−0.00005458	−0.00000085	0.00000000	0.00204458	0.01786924
−0.00005458	50.93965291	0.00000176	−0.00000000	−1.93047146	−0.03446039
−0.00000085	0.00000176	0.05009027	0.00001816	0.00338329	0.00256958
0.00000000	−0.00000000	0.00001816	0.00393727	−0.00276753	0.00324695
0.00204458	−1.93047146	0.00338329	−0.00276753	0.08170009	0.00145979
0.01786924	−0.03446039	0.00256958	0.00324695	0.00145979	0.00000000

Bild 3.34: Submatrix $\tilde{\mathbf{A}}$ im Zyklus 38

Nach zwei weiteren Iterationszyklen haben sich zwei Diagonalblöcke $(15 - 16)$ und $(17 - 20)$ mit stark voneinander abweichender Spur ausgebildet (Bild 3.35). Infolge der schlechten Trennung der Eigenwerte des Diagonalblocks der Zeilen und Spalten 17 bis 20 sind zur Bestimmung des nächsten Eigenwertes acht weitere Zyklen erforderlich.

[15]	[16]	[17]	[18]	[19]	[20]
51.01229295	−0.00001156	0.00000000	0.00000000	0.00000002	0.00000000
−0.00001156	50.97276608	−0.00000000	−0.00000000	−0.00000000	0.00000000
0.00000000	−0.00000000	0.01281178	−0.00000014	0.00087022	0.00001133
0.00000000	−0.00000000	−0.00000014	−0.00029969	−0.00005370	0.00413931
0.00000002	−0.00000000	0.00087022	−0.00005370	0.00407736	0.00005290
0.00000000	0.00000000	0.00001133	0.00413931	0.00005290	0.00000000

Bild 3.35: Submatrix $\tilde{\mathbf{A}}$ im Zyklus 40

In Bild 3.36 ist der Konvergenzfortschritt der letzten Zeile der Submatrix $\tilde{\mathbf{A}}$ in den Zyklen 41 bis 48 dargestellt. Die ε-Schranke der Nichtdiagonalelemente beträgt $2.184e - 13$. Das Diagonalelement $\tilde{a}_{20}$ ist jeweils durch eine Spektralverschiebung in jedem Zyklus auf Null gesetzt. In den Zyklen 41 bis 44 ist eine Stagnation der

Konvergenz an den Koeffizienten der Spalten 18 und 19 zu erkennen. Die meist kubische Konvergenz der Spektralverschiebung geht vollständig verloren und setzt erst wieder vollständig im Zyklus 45 ein. Die insgesamt 11 Zyklen zwischen dem 27. und 28. Eigenwert verschlechtern den sonst monotonen Konvergenzverlauf massiv (Bild 3.31, Seite 67).

	[15]	[16]	[17]	[18]	[19]	[20]
$Zkl41$	-0.00000000	-0.00000000	-0.00000363	-0.00411766	-0.00004944	0.00000000
$Zkl42$	0.00000000	0.00000000	0.00000116	0.00393082	0.00004126	0.00000000
$Zkl43$	0.00000000	0.00000000	-0.00000032	-0.00272677	-0.00001719	0.00000000
$Zkl44$	0.00000000	-0.00000000	0.00000004	0.00043764	0.00000042	0.00000000
$Zkl45$	0.00000000	-0.00000000	$-0.92e-10$	-0.00000123	$-0.29e-12$	0.00000000
$Zkl46$	0.00000000	-0.00000000	0.00000000	$0.30e-13$	$0.33e-12$	0.00000000
$Zkl47$	-0.00000000	0.00000000	-0.00000000	0.00000000	$0.32e-12$	0.00000000
$Zkl48$	-0.00000000	0.00000000	-0.00000000	0.00000000	0.00000000	0.00000000

Bild 3.36: Konvergenzfortschritt der letzten 6 Koeffizienten in Zeile 20 der Submatrix $\tilde{\mathbf{A}}$

Im Iterationszyklus 48 hat die spektralverschobene Matrix $\hat{\mathbf{A}}$ in den letzten beiden Zeilen und Spalten konvergiert. Der doppelte Eigenwert $\lambda_{28/29} = 304.34805799$ ist mit ausreichender Genauigkeit bestimmt. Die nach der Deflation der beiden Nullzeilen und -spalten verbleibende Matrix ist aufgrund der zusätzlichen Zerlegungen stark diagonaldominant (Bild 3.37).

[15]	[16]	[17]	[18]	[19]	[20]
51.00822634	-0.00001415	0.00000000	0.0000000	-0.00000000	-0.00000000
-0.00001415	50.96856110	-0.00000000	-0.0000000	0.00000000	0.00000000
0.00000000	-0.00000000	0.00890548	$-0.21e-09$	0.00000000	-0.00000000
0.00000000	-0.00000000	$-0.21e-09$	-0.0082847	0.00000000	0.00000000
-0.00000000	0.00000000	0.00000000	0.0000000	0.00000000	0.00000000
-0.00000000	0.00000000	-0.00000000	0.0000000	0.00000000	0.00000000

Bild 3.37: Submatrix $\tilde{\mathbf{A}}$ im Zyklus 48

Eine Blockstruktur bildet sich häufig erst in einem fortgeschrittenen Stadium der Iteration aus. Die Nullelemente in den Zeilen und Spalten eines Diagonalblocks sind

im allgemeinen in diesem Iterationsstadium nicht klein genug, um den Block in einer separaten Berechnung zu diagonalisieren. Das Erkennen eines Diagonalblocks während der Iteration ist teuer und stört den Berechnungsablauf. Die Stagnation der Iteration wird deshalb durch den Einsatz einer Jacobi-Randkorrektur auf die letzte Zeile und Spalte der Matrix beseitigt.

Lösungsstrategie Um die Konvergenz der letzen Zeile und Spalte zu verbessern, werden ihre Problemkoeffizienten im Zyklus 40 jeweils durch eine Jacobi-Rotation auf Null reduziert. Die erzeugten Nullelemente werden durch die nachfolgenden Jacobi-Transformationen wieder überschrieben, sind aber im allgemeinen im Betrag wesentlich kleiner als vor der Transformation.

Ist die Konvergenz der Nichtdiagonalelemente am Profilrand nur wenig fortgeschritten, so erzeugt die Jacobi-Transformation Elemente $a_{ik} \geq \varepsilon$ außerhalb des Profils, der von der Transformation betroffenen Zeilen und Spalten. Eine Erweiterung des Matrixprofils in diesem Bereich stört nicht, da die betroffenen Zeilen und Spalten nach Konvergenz des Diagonalelementes nicht weiter Teil der Berechnung sind.

Bei fortgeschrittener Konvergenz ist häufig eine Vielzahl von Koeffizienten in Profilrandnähe kleiner als die gewählte Konvergenzschranke für die Nichtdiagonalelemente. Diese Koeffizienten bleiben von den Jacobi-Transformationen nahezu unbeeinflußt, so daß auf eine Profilerweiterung gänzlich verzichtet werden kann. Für die Anwendung von Jacobi-Rotationen am rechten und unteren Rand der deflationierten Matrix wird die Bezeichnung *Jacobi-Randkorrektur* gewählt.

[15]	[16]	[17]	[18]	[19]	[20]
51.01229295	−0.00001156	0.00000000	0.00000000	0.00000001	0.00000000
−0.00001156	50.97276608	−0.00000000	−0.00000000	−0.00000000	$0.65e-11$
0.00000000	−0.00000000	0.01281178	−0.00000006	0.00087029	0.00000000
0.00000000	−0.00000000	−0.00000006	0.00399216	−0.00000000	$0.48e-11$
0.00000001	−0.00000000	0.00087029	−0.00000000	0.00407804	$0.36e-11$
0.00000000	$0.65e-11$	0.00000000	$0.48e-11$	$0.36e-11$	−0.00429255

Bild 3.38: Submatrix $\tilde{\mathbf{A}}$ im Zyklus 40 nach vollständiger Jacobi-Randkorrektur

Die Eliminierung des Problemkoeffizienten $a_{(20,18)} = 0.00413931$ (Bild 3.35, Seite 70) ist bereits ausreichend um den nächsten Eigenwert $\lambda_{28} = 304.33977327$ in den folgenden zwei QR-Zerlegungen mit ausreichender Genauigkeit zu bestimmen. Aus algorithmischen Gründen wird bei der Jacobi-Randkorrektur jedoch eine zyklische Vorgehensweise gewählt. Um die neu entstandenen betragskleinen Elemente am Profilrand zu vermeiden, werden in einer Jacobi-Randkorrektur ausgehend vom Koeffizienten $a_{n,n-1}$ alle Koeffizienten der letzten Zeile und Spalte durch Jacobi-Rotationen auf Null rotiert. Die gewählte Eliminierungsrichtung, von der Haupt-

diagonalen hin zum Profilrand erzeugt betragskleinere Elemente am Profilrand und entspricht damit dem natürlichen Konvergenzverhalten der QR-Zerlegung.

Durchführung der Jacobi-Randkorrektur Im Iterationszyklus s werden die betragsgroßen Nichtdiagonalkoeffizienten der letzten Zeile und Spalte n der Matrix $\mathbf{A}_s$ mit der Ähnlichkeitstransformation (3.4.31) schrittweise eliminiert. Die Nichtdiagonalkoeffizienten $\hat{a}_{nm} = \hat{a}_{mn}$ der transformierten Matrix $\hat{\mathbf{A}}_s$ sind im allgemeinen nicht null, da die erzeugten Nullelemente durch die schrittweise Berechnung wieder überschrieben werden. Sie sind aber häufig im Betrag wesentlich kleiner als die Koeffizienten $a_{nm} = a_{mn}$ der Matrix $\mathbf{A}_s$.

$$\mathbf{P}^T \mathbf{A}_s \mathbf{P} = \hat{\mathbf{A}}_s \qquad \mathbf{P}^T\mathbf{P} = \mathbf{P}\mathbf{P}^T = \mathbf{I} \tag{3.4.31}$$

Die orthonormale Transformationsmatrix $\mathbf{P}$ aus (3.4.31) wird bei der schrittweisen Durchführung der Transformation aus dem Produkt orthonormaler Matrizen $\mathbf{P}_{nm}$ bestimmt (3.4.32). Als Transformationsmatrix $\mathbf{P}_{nm}$ wird die Rotationsmatrix für eine Drehung θ in der Koordinatenebene (n, m) gewählt.

$$\mathbf{P} = \mathbf{P}_{n,n-1}\,\mathbf{P}_{n,n-2}\,\mathbf{P}_{n,n-3}\ldots \tag{3.4.32}$$

Der Winkel θ der Rotationsmatrizen $\mathbf{P}_{nm}$ wird so gewählt, daß die Nichtdiagonalkoeffizienten $a_{nm} = a_{mn}$ auf Null getrieben werden (s. Bild 3.39) [25],[26].

$$a_{nn} \neq a_{mm} : \quad tan(2\theta) = \frac{2a_{nm}}{a_{nn} - a_{mm}} \qquad mit\ |\theta| < \frac{\pi}{4} \tag{3.4.33}$$

$$a_{nn} = a_{mm} : \quad \theta = \frac{\pi}{4} \tag{3.4.34}$$

Die Eliminierung des Nichtdiagonalkoeffizienten a_{mn} erfolgt nur, falls der Betrag des Elementes größer einer empirisch gewählten Schranke ist. Da die Jacobi-Randkorrektur kein iterativer Prozess ist und der gewünschte Effekt des Verfahrens schon durch die Eliminierung der betragsgrößten Nichtdiagonalkoeffizienten erreicht wird, kann die Schranke großzügig gewählt werden. Im allgemeinen ist die Wahl $\delta = \|\mathbf{A}[n]\|_\infty/10$ ausreichend für eine erfolgreiche Randkorrektur.

Profilerweiterung Werden in der letzten Zeile und Spalte n die k Nichtdiagonalkoeffizienten zwischen dem Profilrand pr und dem Diagonalelement durch Jacobi-Rotationen auf Null getrieben, entstehen Elemente außerhalb des Profils, sofern die letzten $k+1$ Zeilen und Spalten nicht ein konstantes Profil besitzen. Bei ausreichender Konvergenz der letzten $k+1$ Zeilen sind die Nichtdiagonalkoeffizienten zum Profilrand hin häufig sehr klein, so daß zusätzlich erzeugte Koeffizienten außerhalb des Profils im Betrag so klein sind, daß sie unter die Konvergenzschranke der Nichtdiagonalkoeffizienten der Iteration fallen und nicht weiter berücksichtigt werden. Ist dies nicht der Fall, müssen die zusätzlichen Koeffizienten in der nachfolgenden QR-Zerlegung berücksichtigt werden. Diese Berücksichtigung verursacht durch die geforderte Konvexität des

Profils unter Umständen weitere zusätzliche Koeffizienten außerhalb des Profils der Zeilen $n-1$ bis $n-k$.

Bild 3.39 zeigt, daß Jacobi-Transformationen nicht profilerhaltend sind. Im Gegensatz zur QR-Transformation werden die Nullelemente nicht bei der Zerlegung, als Folge einer Linksrotation erzeugt, sondern am Ende der Ähnlichkeitstransformation. Dieser Umstand verhindert eine profilerhaltende Matrixiteration, ermöglicht lokal eingesetzt aber die gewünschte Eliminierung einzelner Elemente am Ende des Berechnungsschritts. Wegen der Symmetrie der Matrix sind die Nichtdiagonalelemente a_{nm} und a_{mn} in den Kreuzungspunkten der modifizierten Zeilen und Spalten am Ende der Ähnlichkeitstransformation null.

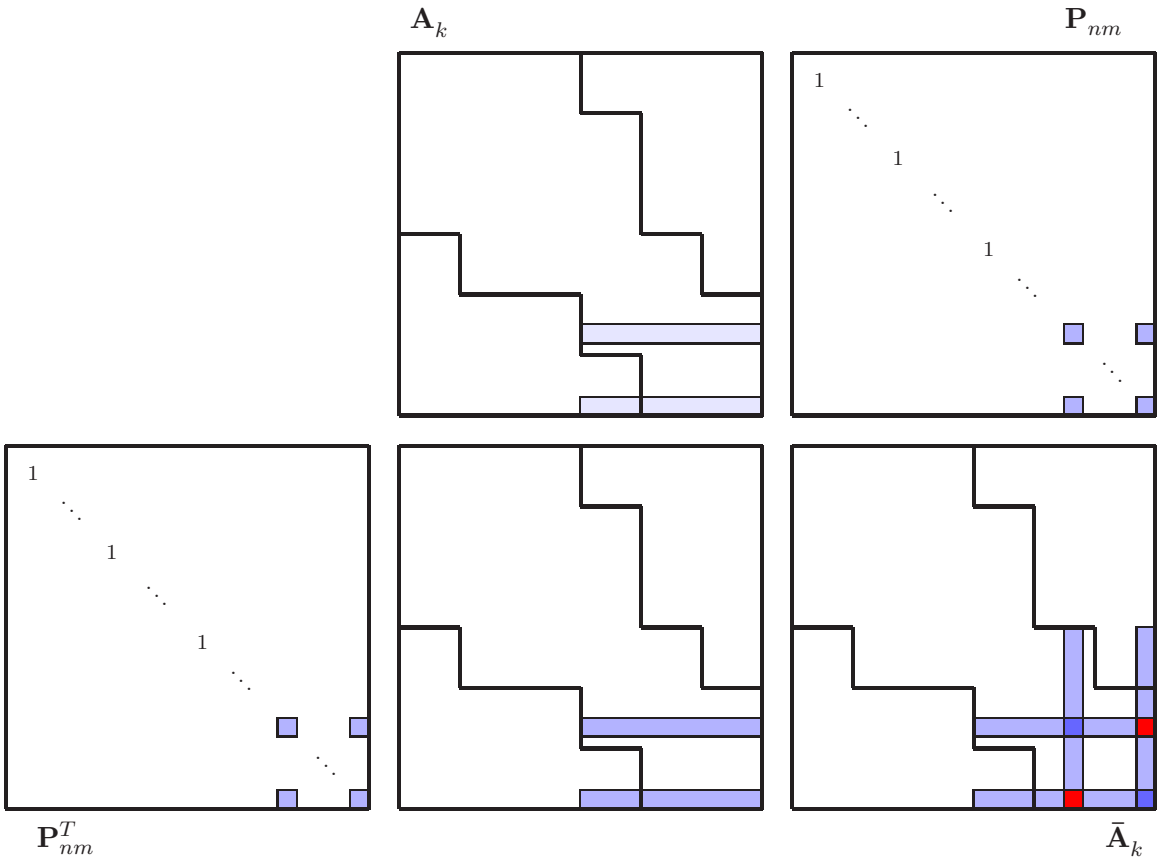


Bild 3.39: Jacobi-Ähnlichkeitstransformation

Bei der praktischen Durchführung der Jacobi-Randkorrektur kann eine Profilerweiterung vermieden werden, indem die Entscheidung über die Eliminierung des Koeffizienten a_{nm} in Abhängigkeit des Profils bzw. des Konvergenzfortschritts der Zeile m getroffen wird. Ist das Profil der Zeile m gleich dem Profil der Zeile n ($pr[m] = pr[n]$), erfolgt eine Transformation. Ist $pr[m] > pr[n]$, erfolgt die Transformation nur, falls die Koeffizienten zwischen $pr[m]$ und $pr[n]$ kleiner der Konvergenzschranke der Nichtdiagonalelemente der Iteration sind.

Ergebnisanalyse Bei Anwendung der Jacobi-Randkorrektur zum frühest geeigneten Iterationszyklus ergibt sich für die vollständige Bestimmung der Eigenwerte der Matrix aus Beispiel 3 der in Bild 3.40 gezeigte Konvergenzverlauf.

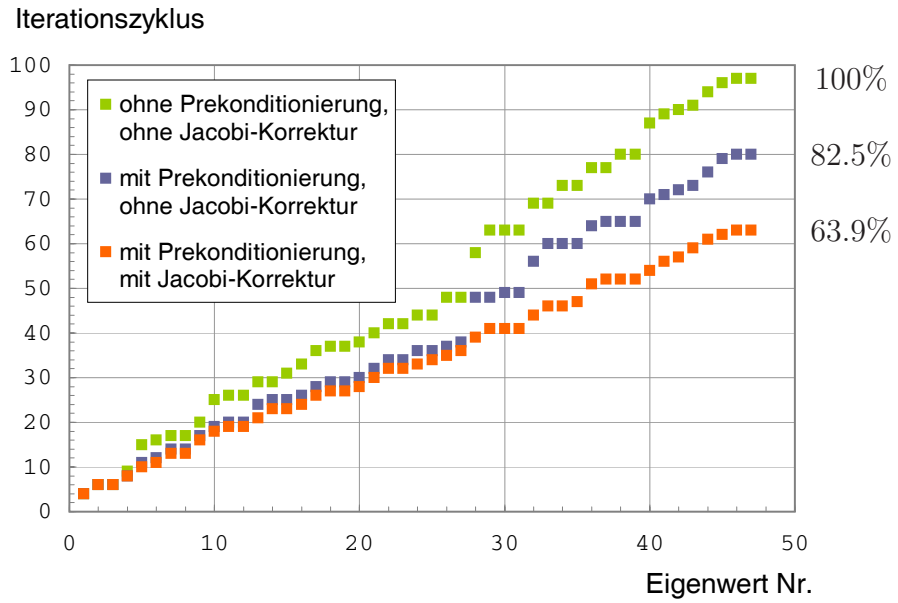


Bild 3.40: Verbesserter Iterationsverlauf der Matrix $\mathbf{K}_{47}$

Bild 3.40 zeigt deutlich eine Verbesserung der Konvergenz der Iteration. Die Eigenwerte des Testbeispiels sind bereits nach 62 Iterationszyklen vollständig bestimmt. Ohne Jacobi-Randkorrektur waren zur vollständigen Bestimmung der Eigenwerte 80 Iterationszyklen erforderlich. Die Jacobi-Randkorrektur wurde ab Iterationszyklus 5 in jedem 2. Zyklus eingesetzt. Die maximal erforderliche Anzahl von Zerlegungen für einen Eigenwert hat sich von 10 auf 5 halbiert. Die durchschnittliche Anzahl von Zerlegungen pro Eigenwert hat sich um über 28% verbessert.

Die durchgeführten Berechnungsbeispiele an großen Profilmatrizen zeigen, daß für die profilrandnahen Koeffizienten der letzten Zeilen und Spalten bei Einsatz der Spektralverschiebung fast ausschließlich kubische Konvergenz vorherrscht. Eine Diagonaldominanz stellt sich rasch ein und wird mit zunehmender Zyklenzahl verstärkt. Die Problemkoeffizienten, die einen Einsatz der Jacobi-Randkorrektur erforderten, befanden sich ausschließlich im letzten Drittel um die Hauptdiagonale.

Tabelle 3.3 zeigt eine Vergleichsrechnung verschiedener Matrizen mit und ohne Jacobi-Randkorrektur. Spalte 1 enthält die Dimension der berechneten Matrix, Spalte 2 die mittlere Bandbreite und Spalte 3, die Anzahl der berechneten Eigenwerte. In Spalte 4 und 5 ist der multiplikative Aufwand der Berechnung ohne und mit Jacobi-Randkorrektur aufgeführt. Spalte 6 enthält den multiplikativen Aufwand der Jacobi-Rotationen. Spalte 7 zeigt den prozentualen

1	2	3	4	5	6	7
N	b	k	FLOP o. Jacobi	FLOP m. Jacobi	FLOP Jacobi	verb. Aufwand
183	23	183	$0.98 \cdot 10^9$	$0.08 \cdot 10^9$	$0.0001436 \cdot 10^9$	84.4%
567	40	567	$0.29 \cdot 10^{10}$	$0.21 \cdot 10^{10}$	$0.0001831 \cdot 10^{10}$	68.7%
943	52	943	$0.13 \cdot 10^{11}$	$0.10 \cdot 10^{11}$	$0.0000350 \cdot 10^{11}$	68.4%
2647	89	2647	$0.27 \cdot 10^{12}$	$0.20 \cdot 10^{12}$	$0.0000123 \cdot 10^{12}$	65.7%
5727	131	5727	$0.26 \cdot 10^{13}$	$0.20 \cdot 10^{13}$	$0.0000030 \cdot 10^{13}$	64.6%
10687	179	10687	$0.17 \cdot 10^{14}$	$0.13 \cdot 10^{14}$	$0.0000007 \cdot 10^{14}$	64.0%

Tabelle 3.3: Vergleich der Iteration mit und ohne Jacobi-Randkorrektur

Aufwand der Berechnung mit Jacobi-Randkorrektur bezogen auf die Berechnung ohne Jacobi-Randkorrektur.

Bewertung der Jacobi-Randkorrektur Die Jacobi-Randkorrektur bietet die Möglichkeit, einzelne Problemkoeffizienten am unteren Matrixrand effektiv zu beseitigen, um einem stagnierenden Konvergenzfortschritt wirksam zu entgegenzuwirken. Die Berechnung erfolgt lokal begrenzt und ist damit numerisch wesentlich günstiger als eine vollständige QR-Zerlegung und im Ergebnis meist wirkungsvoller (Bild 3.41).

Die vorgeschlagene Strategie für den Einsatz des Verfahrens erfordert keine Profilerweiterung der letzten Zeilen und Spalten. Wird die Jacobi-Randkorrektur zu beliebigen Zeitpunkten auf den gesamten unteren Matrixrand eingesetzt, muß teilweise das Profil der letzten Zeilen und Spalten um wenige Koeffizienten erweitert werden. Diese Profilerweiterung bleibt lokal begrenzt und wird durch die Deflation bei Konvergenz in den betroffenen Zeilen und Spalten hinfällig.

Bild 3.41 zeigt die Verbesserung des multiplikativen Aufwands durch die vorgestellten Erweiterungen des Verfahrens. Bestimmt wurde jeweils das vollständige Eigenwertspektrum der Matrix mit der in Abschnitt 3.3.3 vorgeschlagenen Konvergenzschranke. Bereits für geringe Dimensionen fällt der Aufwand deutlich unter 70% des üblichen QR-Algorithmus mit Einsatz von Spektralverschiebung und Deflation. Für alle untersuchten Beispielmatrizen waren rund ein Viertel des Spektrums Eigenwerte mit Multiplizität ≥ 2.

Sowohl für die Prekonditionierung als auch für die Jacobi-Randkorrektur werden nur orthonormale Rotationsmatrizen eingesetzt. Der Fehler der durchgeführten Transformationen wird in Analogie zu den Transformationen des QR-Algorithmus durch die Schranke $(\epsilon \left\|\mathbf{A}\right\|_2)$ begrenzt (siehe 3.3.3). Die Genauigkeit der so berechneten Eigenwertnäherungen wird von den Erweiterungen des Verfahrens nicht beeinträchtigt.

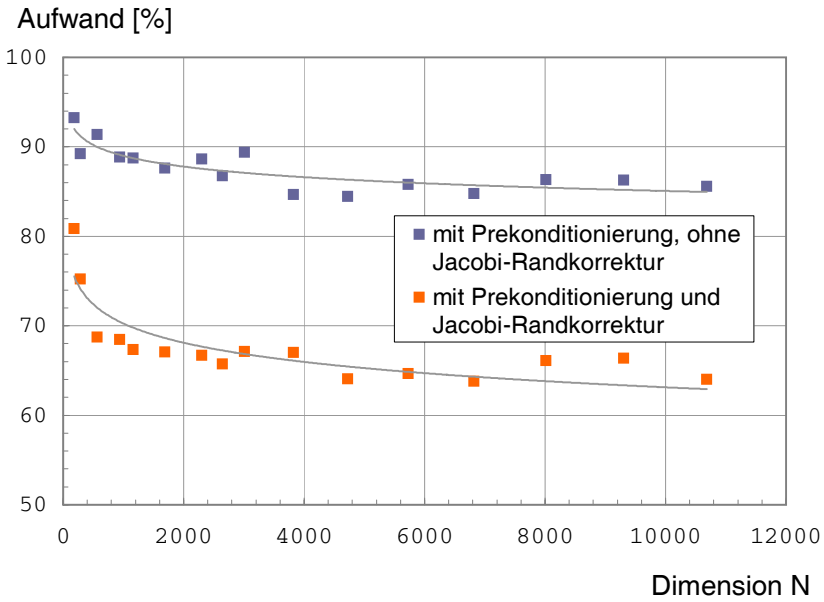


Bild 3.41: Verbesserter Aufwand durch Prekonditionierung und Jacobi-Randkorrektur

Algorithmische Darstellung der Eigenwertberechnung mit der Inversen Matrixiteration

Das QR-Verfahren und die vorgestellten Erweiterungen zur Eigenwertberechnung werden kompakt als Algorithmus zusammengefasst. Eine detaillierte Beschreibung der einzelnen Schritte erfolgt im Kapitel 3.5 (Implementierung) ab Seite 126.

Die Bezeichnung der Matrizen des Algorithmus entspricht der Notation der vorangegangenen Kapitel (Seiten 56 - 74). Die Variable $maxS$ (Zeile 1) kennzeichnet die maximale Zyklenanzahl. Der Verschiebungsparameter c_s der Spektralverschiebung im Schritt s wird implizit durch Summation des letzten Diagonalkoeffizienten a_{nn} in jedem Zyklus berechnet. Das gewünschte Teilspektrum der Iteration wird mit $\psi(\mathbf{A})$ (Zeile 6) bezeichnet.

In den Zeilen 2 bis 13 wird die iterierte Matrix auf Konvergenz in den letzten Zeilen und Spalten untersucht. Falls ein Eigenwert in Zeile n konvergiert hat, wird aus dem Diagonalkoeffizienten a_{nn} und dem Verschiebungsparameter c_s eine Eigenwertnäherung bestimmt. Durch eine Spektralverschiebung um den Diagonalkoeffizienten a_{nn} sind alle Koeffizienten in der Zeile und Spalte n null bzw. im Betrag kleiner einer definierten Schranke für den Wert Null. Die verbleibende Matrix ist damit um den Eigenzustand $(\lambda_n, \mathbf{x}_n)$ deflationiert. Die weitere Berechnung erfolgt an einer um die Dimension eins verringerten Matrix (Zeile 5). Ist das gesuchte Teilspektrum vollständig, wird die Iteration abgebrochen. Solange in Zeile und Spalte n Konvergenz festgestellt wird, werden weitere Eigenwertnäherungen bestimmt und die Matrix schrittweise in ihrer Dimension verringert. Ist das Konvergenzkriterium in Zeile und Spalte n nicht erfüllt, wird der Verschiebungsparameter der Spektralverschiebung mit dem letzten Diagonalkoeffizienten a_{nn} aktualisiert und eine Spektralverschiebung um a_{nn} durchgeführt.

In den Zeilen 14 bis 17 werden die Berechnungsschritte der Ähnlichkeitstransformationen der Iteration ausgeführt. Am Ende der Iteration wird die iterierte Matrix um den Diagonalkoeffizienten a_{nn} spektralverschoben. Zu Beginn jedes Iterationszyklus besitzt damit der Diagonalkoeffizienten a_{nn} den Wert Null. Die Gesamtverschiebung der Eigenwertaufgabe ist in jedem Zyklus der Wert von c_s.

Algorithmus 1 Inverse Matrixiteration - Eigenwertberechnung

1: **for** $s = 1, 2, \ldots, maxS$ **do**
2: **for** $i = n, n-1, \ldots, 1$ **do**
3: **if** *Konvergenz in Zeile n* **then**
4: $\hat{\lambda}_n = a_{nn} + c_s$ { Berechne Eigenwertnäherung zu λ_n }
5: $n = n - 1$ { Deflation : Dekrementiere n um eins }
6: **if** $\psi(\mathbf{A})$ *vollständig* **then**
7: $return$ { Gewünschter Subraum ist komplett }
8: **end if**
9: **else**
10: $c_s = c_s + a_{nn}$ { Berechne Verschiebungsparameter }
11: $\mathbf{A}_s = \mathbf{A}_s - a_{nn}\,\mathbf{I}$ { Spektralverschiebung }
12: **end if**
13: **end for**
14: $\mathbf{A}_s = \mathbf{P}^T\,\mathbf{A}_s\,\mathbf{P}$ { Jacobi-Randkorrektur }
15: $\mathbf{R}_s = \mathbf{Q}_s^T\,\mathbf{A}_s$ { QR-Zerlegung }
16: $\mathbf{A}_{s+1} = \mathbf{R}_s\,\mathbf{Q}_s$ { RQ-Rekombination }
17: $\mathbf{A}_{s+1} = \mathbf{U}^T\,\mathbf{A}_{s+1}\,\mathbf{U}$ { Prekonditionierung }
18: $c_s = c_s + a_{nn}$ { Berechne Verschiebungsparameter }
19: $\mathbf{A}_{s+1} = \mathbf{A}_{s+1} - a_{nn}\,\mathbf{I}$ { Spektralverschiebung }
20: **end for**

3.4.3 Bestimmung der Eigenvektoren

Die Eigenvektoren $\mathbf{x}_i$ der Aufgabe (3.3.1) bestimmen sich nach (3.3.4) aus der Transformationsmatrix $\mathbf{Q}$. In der kanonischen Darstellung (3.3.2) ist $\mathbf{Q}$ die Eigenmatrix $\mathbf{X}$ der speziellen Eigenwertaufgabe (3.3.1) und wird als Grenzwert des Produkts der Transformationsmatrizen $\mathbf{Q}_s$ (3.3.16) bestimmt.

Der numerische Aufwand zur Bestimmung der Näherung $\hat{\mathbf{Q}}^{(s)}$ nach s Iterationen ist von der Ordnung $O[s(bn^2 + nb^2)]$, der Speicherbedarf entspricht einer vollbesetzten $(n \times n)$ Matrix. Zur Reduzierung von Speicherbedarf und numerischem Aufwand werden die Eigenwerte und Eigenvektoren getrennt voneinander bestimmt. Die Eigenmatrix $\mathbf{X}$ wird nicht explizit berechnet. Sind die Eigenwerte bestimmt, werden die zu einem q-fachen Eigenwert $\lambda_i^{(q)}$ zugehörigen q Eigenvektoren $\mathbf{x}_i$ mit der verschobenen Matrix $\mathbf{C} = (\mathbf{A} - \lambda_i^{(q)}\mathbf{I})$ bestimmt. Durch die Spektralverschiebung mit $\lambda_i^{(q)}$ besitzt $\mathbf{C}$ einen q-fachen Eigenwert Null. Die Transformation von $\mathbf{C}$ in eine Rechtsdreiecksmatrix $\mathbf{R}$ mit q Nullzeilen (3.3.8) liefert die Informationen zur Bestimmung der q Eigenvektoren $\mathbf{x}_i$. Die Auswertung des Produkts (3.4.35) von rechts nach links erfordert nur die Berechnung und Speicherung der q letzten Spalten der Eigenmatrix $\mathbf{X}$.

$$\mathbf{X}_q \;=\; \mathbf{P}_{21} \,\ldots\, \mathbf{P}_{pr[1],1}\, \mathbf{P}_{32} \,\ldots\, \mathbf{P}_{ik} \,\ldots\, \mathbf{P}_{n,n-1}\, \mathbf{I}_q \tag{3.4.35}$$

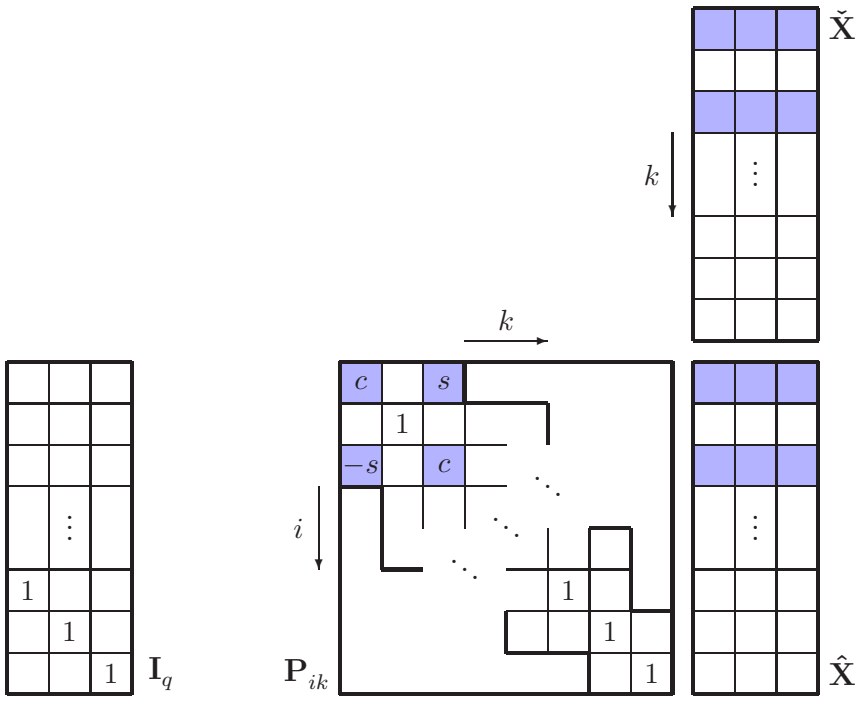


Bild 3.42: Bestimmung der q Eigenvektoren $\mathbf{x}_i$ zum q-fachen Eigenwert $\lambda_i^{(q)}$

Infolge Rundungsfehler ist eine einzelne QR-Zerlegung zur Bestimmung von q Eigenvektoren $\mathbf{x}_{i_m}, (m = 1, \ldots, q)$ zu einem q-fachen Eigenwert $\lambda_i^{(q)}$ teilweise nicht ausreichend. Insbesondere wenn der mehrfache Eigenwert in einem Cluster schlecht getrennter Eigenwerte liegt, wird mit einer QR-Zerlegung nur eine Teilmenge der gesuchten Eigenvektoren mit ausreichender Genauigkeit bestimmt. Im folgenden werden die Ursachen für dieses Verhalten untersucht.

Problemanalyse Als Untersuchungsbeispiel dient die Steifigkeitsmatrix einer Quadratplatte mit 567 Freiheitsgraden.

Beispiel 5 : Einfach gelagerte Quadratplatte mit 567 Freiheitsgraden

Die Platte besitzt eine knotenkonzentrierte Massenbelegung und ist symmetrisch elementiert. Die vier größten Eigenwerte bilden ein Cluster schlecht getrennter Eigenwerte. Der Eigenwert λ_{565} besitzt Multiplizität 2. Tabelle 3.4 zeigt die aus der Eigenwertberechnung bestimmten Näherungslösungen sowie Fehlerschranken für die Eigenwertnäherungen nach Temple-Kato. Die Matrix der speziellen Eigenwertaufgabe wird mit K_{567} bezeichnet.

Eigenwertnr.	Eigenwertnäherung $\hat{\lambda}$	$\lvert\lambda - \hat{\lambda}\rvert \leq \frac{\lVert\mathbf{r}\rVert_2^2}{\delta}$
λ_{564}	74250.477760425550	$2.300e-11$
λ_{565}	$74250.47776043\mathbf{3730}$	$8.290e-10$
λ_{566}	$74250.47776043\mathbf{4350}$	$1.768e-09$
λ_{567}	74250.477760435880	$1.500e-10$

Tabelle 3.4: Betragsgrößte Eigenwerte der Matrix K_{567}

Die Multiplizität 2 des Eigenwerts λ_{565} kann nicht mehr durch Inspektion bestimmt werden. Die Eigenwertnäherungen stimmen nur bis zur 9. Dezimalstelle überein. Die Kenntnis über die Multiplizität q eines Eigenwertes ist für eine korrekte Initialisierung der Spaltenvektoren der Matrix $\mathbf{I}_q$ (Gleichung (3.4.35)) erforderlich. Die Bestimmung aller Eigenvektoren $\mathbf{x}_{564} - \mathbf{x}_{567}$ des Clusters mit einer einzigen Zerlegung der um einen der vier Eigenwerte verschobenen Matrix $\mathbf{C}$ kann nicht mit ausreichender Genauigkeit erfolgen. Der sequentielle Einsatz der Eigenwertnäherungen $\hat{\lambda}_{564} - \hat{\lambda}_{567}$ als Verschiebungsparameter der Zerlegung von $\mathbf{C}$ hingegen liefert für die Eigenwerte λ_{565} und λ_{566} den identischen Eigenvektor $\hat{\mathbf{x}}_{565} = \hat{\mathbf{x}}_{566}$, da für beide Eigenwertnäherungen nur in der letzten Zeile und Spalte Konvergenz mit ausreichender Genauigkeit festgestellt werden kann.

Zur Bestimmung der Eigenvektoren $\mathbf{x}_{565}$ und $\mathbf{x}_{566}$ wird eine der beiden Eigenwertnäherungen $\hat{\lambda}_{565}$ und $\hat{\lambda}_{566}$ als Verschiebungsparameter gewählt. Bild 3.43 zeigt

die letzten vier Zeilen und Spalten der Zerlegung $\mathbf{R}_1 = \mathbf{Q}^T\hat{\mathbf{C}} = \mathbf{Q}^T(\mathbf{A} - \hat{\lambda}^{(2)}_{565}\,\mathbf{I})$.

[564]	[565]	[566]	[567]
−8220.4483456249	624.7918274933	93.3024117347	160.8165495491
0.0000000000	5080.0466107651	−19014.5705784861	20470.2280477072
0.0000000000	0.0000000000	−8442.5645808848	−8442.5645808849
0.0000000000	0.0000000000	0.0000000000	0.0000000966

Bild 3.43: Submatrix $\mathbf{R}_1 = \mathbf{Q}^T(\mathbf{A} - \hat{\lambda}^{(2)}_{565}\,\mathbf{I})$

Einer der beiden Eigenwerte hat in der letzten Zeile der Matrix konvergiert. Eine Näherung zum zweiten Eigenwert konnte in keiner der Zeilen der Matrix K_{567} identifiziert werden.

Während der Zerlegung entstehen jedoch betragskleine Diagonalkoeffizienten. Mit Hilfe der Norm der entsprechenden Zeilen (z.B. $\|\mathbf{r}_{180}\|_2 < 1.e-2$) können diese, gemessen an der durchschnittlichen Zeilennorm der Rechtsdreiecksmatrix $\mathbf{R}_1$ ($\|\mathbf{r}_{\not\emptyset}\|_2 = 18642.46$), als Nullzeilen identifiziert werden. Wegen des großen Fehlers der Diagonalkoeffizienten wird jedoch keine der Zeilen in den nachfolgenden Zerlegungsschritten an das Ende der Matrix transportiert. Die Zeilen werden nach wenigen Schritten mit betragsgroßen Elementen überschrieben.

Die durchgeführte Zerlegung ($\mathbf{Q}^T(\mathbf{A}-\hat{\lambda}^{(2)}_{566}\,\mathbf{I})$ stellt lediglich die Rotationsparameter zur Bestimmung einer der beiden gesuchten Eigenvektornäherungen $\hat{\mathbf{x}}_{565}$ und $\hat{\mathbf{x}}_{566}$ bereit.

Lösungsstrategie Die Auswertung von Gleichung (3.4.35) zur Bestimmung von q Eigenvektoren zu einem q-fachen Eigenwert muß von rechts erfolgen, beginnend mit dem Produkt $\mathbf{P}_{n,n-1}\mathbf{I}_q$. Nur so können die q Spalten von $\mathbf{X}_q$ unabhängig von den restlichen $n-q$ Spalten der Eigenmatrix $\mathbf{X}$ bestimmt werden. Erfordert die Bestimmung der Eigenvektoren zu einem q-fachen Eigenwert $\lambda^{(q)}$ mehr als eine Zerlegung, so ist die Speicherung der Rotationsparameter zusätzlicher Zerlegungen erforderlich (Gleichung (3.4.36)).

$$\mathbf{X}_q = \mathbf{Q}_1\,\mathbf{Q}_2\,\ldots\,\mathbf{Q}_m\,\mathbf{I}_q \tag{3.4.36}$$

$$\mathbf{Q}_i = \mathbf{P}_{21_i}\,\ldots\,\mathbf{P}_{pr[1],1_i}\,\mathbf{P}_{32_i}\,\ldots\,\mathbf{P}_{ik_i}\,\ldots\,\mathbf{P}_{n,n-1_i} \qquad i = 1,\ldots,m \tag{3.4.37}$$

Jede zusätzliche Zerlegung vergrößert damit den Gesamtspeicheraufwand um $\sim b\,N$ Koeffizienten. Alternativ zur Speicherung der zusätzlichen Rotationsparameter, kann jede erforderliche Transformationsmatrix $\mathbf{Q}_{(i)}$ $(i = 1,\ldots,m)$, durch $(i+1)$ Zerlegungen und i Rekombinationen in der erforderlichen Reihenfolge (Gleichung 3.4.36) bestimmt werden. Das Produkt von

m Transformationsmatrizen $\mathbf{Q}_i (i = 1, \ldots, m)$ erfordert damit $(m+1)!$ Zerlegungen und $m!$ Rekombinationen und ist für $m > \sqrt{q}$ unwirtschaftlich.

Stattdessen wird eine lokale Iteration am unteren Matrixrand angestrebt, die eine weitere Zerlegung erübrigt. Der Einsatz dieser Iteration ist vom Matrixprofil und vom Fehler der letzten q Zeilen der Matrix $\hat{\mathbf{R}}$ abhängig. Besitzt die Matrix $\hat{\mathbf{R}}$ der Zerlegung weniger als q Zeilen, die als Nullzeilen zu einem Eigenwert der Multiplizität q identifiziert werden können, wird das Produkt $\hat{\mathbf{C}}_1 = \hat{\mathbf{R}}\hat{\mathbf{Q}}$ gebildet.

Folgende Strategie wird im weiteren verfolgt :

1. In einigen Fällen ist die Konvergenz zum q-fachen Eigenwert $\lambda^{(q)}$ in den letzten q Zeilen und Spalten der Matrix $\hat{\mathbf{C}}_1$ klar erkennbar. Die diagonalnahen Koeffizienten sind aber im Betrag so groß, daß die Informationen der Zerlegung eine nur sehr schlechte Näherung zu den gesuchten Eigenvektoren erwarten lassen. Häufig hat nur einer der q Eigenwerte mit einem akzeptablen Fehler konvergiert. Das Ergebnis der Zerlegung läßt sich durch eine lokale Iteration mit der $(p \times p)$ Submatrix $\hat{\mathbf{C}}_{sub}$ (Bild 3.44) verbessern.

   Für die Submatrix $\hat{\mathbf{C}}_{sub}$ werden die Eigenzustände $(\rho_i, \mathbf{v}_i), (i = 1, \ldots, p)$ mit Hilfe des QR-Verfahrens für vollbesetzte Matrizen bestimmt. Die Eigenvektoren $\mathbf{v}_i$ werden spaltenweise in der Eigenmatrix $\mathbf{V}^{(p \times p)}$ angeordnet, die zur Definition einer orthonormalen Diagonalblockmatrix $\mathbf{U}$ verwendet wird (Bild 3.44).

   Zur Verbesserung der Konvergenz zu den q Eigenvektoren der Matrix $\mathbf{X}_q$, wird Gleichung (3.4.36) um das Produkt mit der orthonormalen Diagonalblockmatrix $\mathbf{U}$ erweitert. $\mathbf{U}$ ist eine Identitätsmatrix mit der Eigenmatrix $\mathbf{V}$ als Diagonalblock in den Zeilen $(N-p)$ bis N (Bild 3.44).

$$\mathbf{X}_q \;=\; \hat{\mathbf{Q}}\,\mathbf{U}\,\mathbf{I}_q \tag{3.4.38}$$

   Die Dimension p der zu diagonalisierenden Blockmatrix $\hat{\mathbf{C}}_{sub}$ wird in Abhängigkeit der Anzahl Elemente in $\hat{\mathbf{C}}_1$ gewählt, die größer einer Schranke η sind und wird auf maximal $0.25b$ begrenzt.

   Gleichung (3.4.39) zeigt die vollständige Transformation von $\hat{\mathbf{C}}_1$ zur Verbesserung der Konvergenz des Eigenwertes $\lambda^{(q)}$, als Voraussetzung für eine verbesserte Eigenvektorberechnung. Gleichung (3.4.39) wird nicht explizit ausgewertet. Das Profil der Zeilen $(N-p, \ldots, N)$ wird vorab um wenige Koeffizienten so erweitert, daß die letzten p Zeilen der Matrix ein konstantes Profil aufweisen und damit den Einsatz der lokalen Iteration profilerhaltend ermöglichen.

$$\mathbf{U}^T\,\hat{\mathbf{Q}}^T\,\hat{\mathbf{C}}_1\,\hat{\mathbf{Q}}\,\mathbf{U} \;=\; \hat{\mathbf{C}}_2 \tag{3.4.39}$$

   Bild 3.44 zeigt, daß die Transformation von $\hat{\mathbf{C}}_1$ mit $\mathbf{U}$ und damit auch das Produkt (3.4.38) in den rot markierten Bereichen einen Fehler verursacht. Nullelemente im markierten Bereich werden wegen $p > q$ mit Elementen $\neq 0$ überschrieben. Die Eigenvektoren

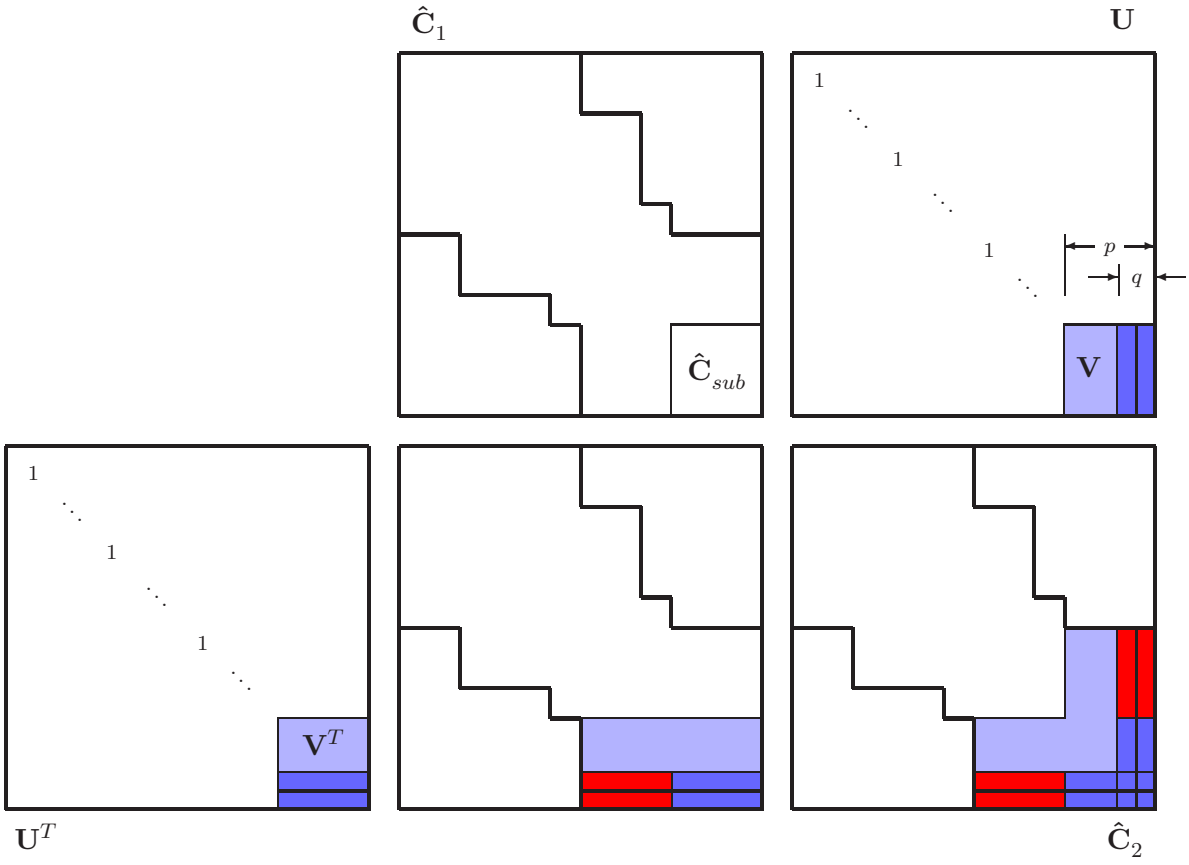


Bild 3.44: Transformation $\mathbf{U}^T\,\hat{\mathbf{C}}_1\,\mathbf{U}$

$\mathbf{v}_i, (i = N-q, \ldots, N)$ sind jedoch infolge der bereits vorangeschrittenen Konvergenz in den letzten q Zeilen und Spalten von $\hat{\mathbf{C}}_1$ Näherungen zu den Einheitsvektoren $\mathbf{e}_{p-q}, \ldots, \mathbf{e}_p$ und besitzen damit an den kritischen, rot markierten Stellen sehr betragskleine Koeffizienten. Die Ordnung der so erzeugten Fehlerkoeffizienten liegt damit im allgemeinen im Bereich des Fehlers der verwendeten Eigenwertnäherung.

Der Aufwand zur Bestimmung von $\mathbf{U}$ ist vernachlässigbar im Vergleich zu einer zusätzlichen Zerlegung. Die Matrix $\mathbf{I}_q$ wird mit den Eigenvektoren $\mathbf{v}_i, (i = N-q, \ldots, N)$ initialisiert.

2. In einigen Fällen läßt sich keine Submatrix $\hat{\mathbf{C}}_{sub}$ von akzeptabler Größe finden, die die erforderlichen Informationen für alle q Eigenvektoren enthält. In diesem Fall ist eine zweite Zerlegung erforderlich. In Abhängigkeit der Aufgabengröße wird entschieden, ob der Mehraufwand durch die Speicherung zusätzlicher Rotationswinkel oder durch einen zusätzlichen operativen Aufwand geleistet wird.

Ergebnisanalyse Die Eigenvektoren aus Beispiel 5 lassen sich nicht durch die lokale Iterationsmethode bestimmen. Hier wird eine zusätzliche Zerlegung durchgeführt. Bild 3.45 zeigt die Koeffizienten der Submatrix $\hat{\mathbf{R}}_2$ nach der zweiten Zerlegung.

	[564]	[565]	[566]	[567]
[564]	9.1512721164	622.8793680852	−1484.3629880310	0.0000000099
[565]	0.0000000000	−0.0000014290	0.0000013740	−0.0000000008
[566]	0.0000000000	0.0000000000	−0.0000022702	0.0000000000
[567]	0.0000000000	0.0000000000	0.0000000000	−0.0000000084

Bild 3.45: Submatrix $\hat{\mathbf{R}}_2$

Die Konvergenz für den doppelten Eigenwert λ_{566} ist nach der zweiten Zerlegung ausreichend für eine sichere Bestimmung beider Eigenvektoren (Zeilen $565/566$). Der Eigenvektor zum vierten Eigenwert λ_{567} des Clusters kann ebenfalls mit akzeptablem Fehler bestimmt werden.

Bild 3.46 zeigt die Submatrix $\hat{\mathbf{R}}_1 = \mathbf{Q}^T(\mathbf{A} - \hat{\lambda}_{500}\,\mathbf{I})$ nach der ersten Zerlegung für den doppelten Eigenwert $\lambda_{500} = \lambda_{501}$. Die Konvergenz der Eigenwerte ist deutlich erkennbar. Die Eigenwertnäherungen sind gut von den restlichen Eigenwerten getrennt und stimmen bis zur 11 Nachkommastelle überein. Der Fehler der Näherungen wird durch die Schranke nach Temple-Kato auf $\leq 7.20e-11$ begrenzt.

Ohne zusätzliche Maßnahme ist jedoch ein vergleichsweise großer Fehler in den Eigenvektoren zu erwarten. Das Ergebnis wird durch eine lokale Iteration verbessert.

	[564]	[565]	[566]	[567]
[564]	5242.9794501483	2011.4617871989	−27175.1950832849	−7122.9497996383
[565]	0.0000000000	3621.8952269670	−7683.2000272067	−29900.8188398782
[566]	0.0000000000	0.0000000000	−0.0001430867	−0.0001400880
[567]	0.0000000000	0.0000000000	0.0000000000	−0.0000290795

Bild 3.46: Submatrix $\hat{\mathbf{R}} = \mathbf{Q}^T(\mathbf{A} - \hat{\lambda}_{500}\,\mathbf{I})$

Die Qualität der Eigenvektorberechnung wird über den relativen Fehler $\delta_{rel} = \frac{\|\mathbf{r}\|_2}{\hat{\lambda}}$ abgeschätzt. Tabelle 3.5 zeigt die relativen Fehler für die Eigenzustände 500 und 501 mit einer Zerlegung (Spalte 2), mit einer zusätzlichen Zerlegung (Spalte 3) und mit einer Zerlegung und lokaler Iteration (Spalte 4).

Die letzen beiden Zeilen von Tabelle 3.5 zeigen den statistischen Mittelwert $\bar{\delta}_{rel}$ der relativen Fehler und die Standardabweichung $\breve{\delta}_{rel}$ aus der Berechnung aller Eigenzustände der Matrix

1	2	3	4
	$\delta_{rel(QR1)}$	$\delta_{rel(QR2)}$	$\delta_{rel(LI)}$
$(\lambda_{500}, \mathbf{x}_{500})$	$8.385e-09$	$1.023e-13$	$1.307e-11$
$(\lambda_{501}, \mathbf{x}_{501})$	$1.218e-09$	$1.027e-13$	$1.133e-11$
$\bar{\delta}_{rel}$	$1.068e-09$	$5.180e-10$	$5.208e-10$
$\breve{\delta}_{rel}$	$5.794e-09$	$3.039e-09$	$3.039e-09$
$(QR2)/(LI)$		$25/0$	$11/14$

Tabelle 3.5: Relativer Fehler der Eigenzustände 500 und 501

K_{567}. Die letzte Zeile aus Tabelle 3.5 zeigt die Anzahl zusätzlicher Zerlegungen (QR2) und die Anzahl lokaler Iterationen (LI) der vollständigen Berechnung.

Durch die lokale Iteration wird die Anzahl zusätzlicher Zerlegungen mehr als halbiert. Der Aufwand der lokalen Iteration ist klein, gemessen am Aufwand einer vollständigen QR-Zerlegung. Die diagonalisierten Submatrizen sind klein in ihrer Dimension und stark diagonaldominant. Wegen der Spektralverschiebung sind die Submatrizen im allgemeinen nicht positiv definit und werden deshalb mit dem numerisch stabilen Jacobi-Verfahren diagonalisiert. Eine große Anzahl mehrfacher Eigenwerte begünstigt den numerischen Aufwand zusätzlich, da mit einer Zerlegung häufig mehrere Eigenvektoren der Multiplizität q bestimmt werden können.

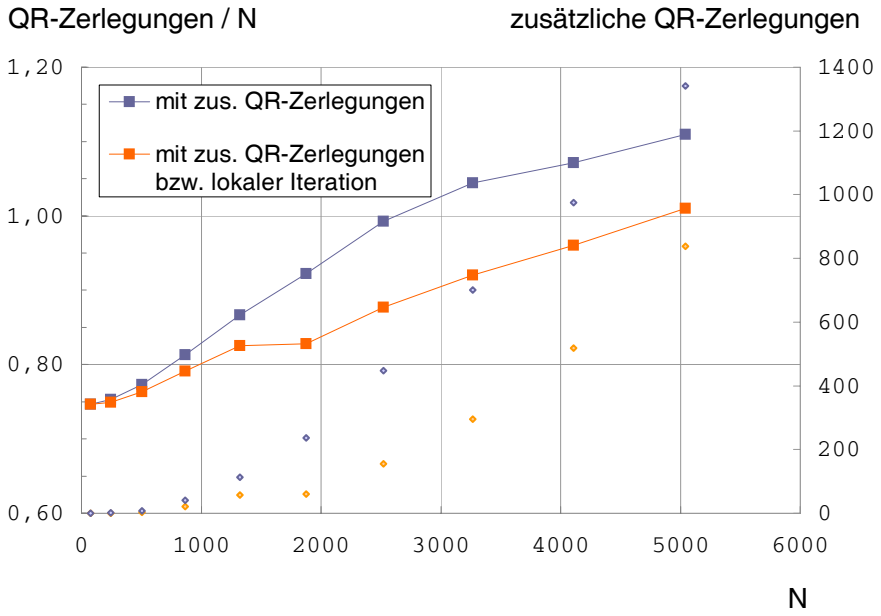


Bild 3.47: Aufwand zur Bestimmung aller Eigenvektoren

Bild 3.47 (Seite 85) zeigt den Aufwand zur Bestimmung aller Eigenvektoren, gemessen an der Anzahl erforderlicher QR-Zerlegungen bezogen auf die Anzahl bestimmter Eigenvektoren. Wegen der großen Anzahl mehrfacher Eigenwerte bleibt das Aufwandsverhältnis auch für größere Dimensionen nahe Eins. In Blau ist der Aufwand mit Zusatzzerlegungen für kritische Eigenzustände gezeigt. In Rot dargestellt ist der Aufwand mit Zusatzzerlegungen und lokalen Iterationen. Die Anzahl der Zusatzzerlegungen der jeweiligen Berechnungsvariante sind als kleine Rauten dargestellt und beziehen sich auf die rechte Ordinate. Die numerisch sehr viel günstigere lokale Iteration hat in allen Fällen die Anzahl zusätzlicher QR-Zerlegungen massiv reduziert.

Der stetige Anstieg des zusätzlichen numerischen Aufwands ist mit der zunehmenden Anzahl schlecht getrennter Eigenwertcluster zu erklären, die insbesondere durch die hohen Frequenzen der berechneten Plattenschwingungen gebildet werden. Für eine Beurteilung ist zu berücksichtigen, daß jeweils das komplette Eigenspektrum der Diskretisierung bestimmt wurde.

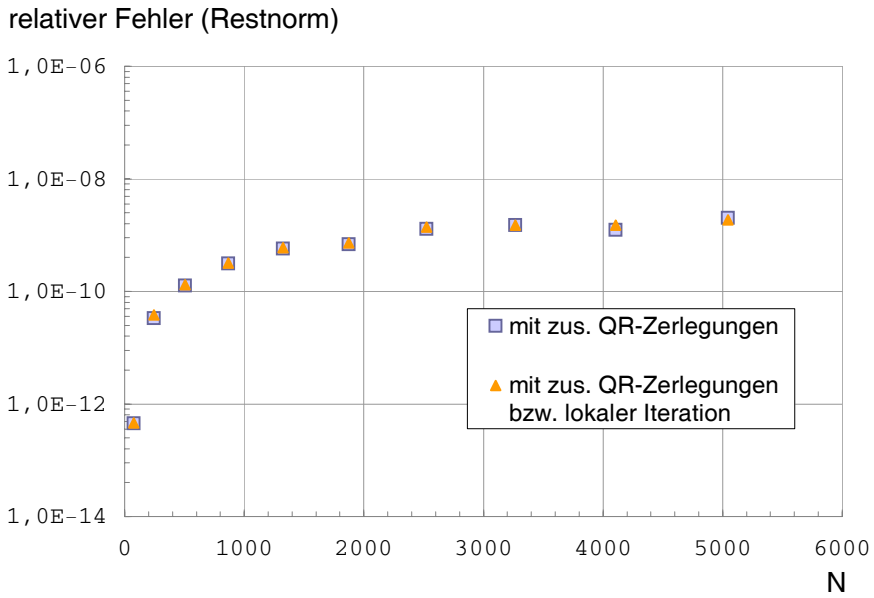


Bild 3.48: A posteriori Fehler

Bild 3.48 zeigt das Mittel der relativen Fehler $\bar{\delta}_{rel}$ für die in Bild 3.47 gezeigten Berechnungen mit und ohne lokaler Iteration. Eine Verschlechterung des Fehlerniveaus infolge der lokalen Iteration ist nicht erkennbar. Für alle Matrizen mit $N > 2000$ wurden jeweils zwischen 4 und 8 Eigenvektoren mit der gewählten Konvergenzschranke nicht mit ausreichender Genauigkeit bestimmt. Diese Eigenvektoren wurden in einer nachfolgenden Berechnung durch eine Zusatzzerlegung zuverlässig berechnet.

Bewertung der lokalen Iteration Der Einsatz einer lokalen Iteration anstelle einer zusätzlichen QR-Zerlegung zur Bestimmung kritischer Eigenzustände verursacht im allgemeinen keine Verschlechterung des Fehlerniveaus der Berechnung. Bild 3.49 zeigt die Berechnungsergebnisse

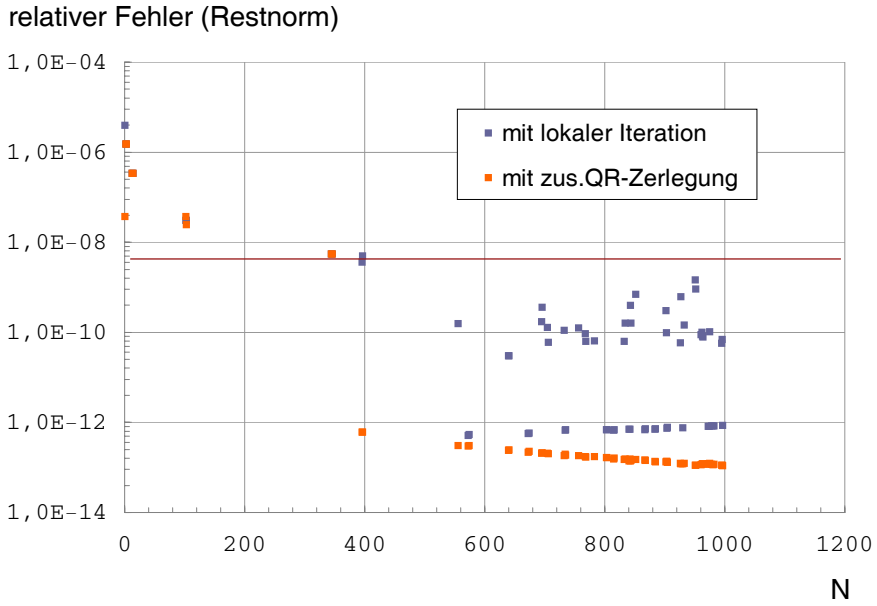


Bild 3.49: A posteriori Fehler für $\sigma(\mathbf{K}_{11163})$

einer Schwingungsaufgabe einer ungelagerten Quadratplatte mit $N = 11163$ Freiheitsgraden. Berechnet wurden die Eigenzustände mit den tausend kleinsten Eigenwerten. Neben den drei Starrkörperbewegungen besitzt ca. ein Viertel der tausend Eigenwerte, Multiplizität zwei. Insgesamt erforderten 67 kritische Eigenzustände eine lokale Iteration bzw. eine Zusatzzerlegung für ausreichende Konvergenz.

Bild 3.49 zeigt nur die relativen Fehler der kritischen Eigenzustände. Der Mittelwert der restlichen Eigenzustände ist durch die dunkelrote Waagerechte gekennzeichnet. Der numerische Mehraufwand für eine zweite Zerlegung spiegelt sich deutlich in einem geringeren Fehler für diese Eigenzustände wider. Bild 3.49 zeigt aber auch, daß der Einsatz lokaler Iterationen zu einem Fehlerniveau deutlich unter dem Mittelwert der restlichen Eigenzustände liegt und sich damit positiv auf den Aufwand und die Genauigkeit der Gesamtberechnung auswirkt.

Algorithmische Darstellung der Eigenvektorberechnung mit der Inversen Matrixiteration

Die Eigenvektorberechnung mit der Inversen Matrixiteration wird kompakt als Algorithmus zusammengefasst. Eine detaillierte Beschreibung der einzelnen Schritte erfolgt im Kapitel 3.5 (Implementierung) ab Seite 136.

Die Bezeichnung der Matrizen des Algorithmus entspricht der Notation des aktuellen Kapitels (Seiten 79 - 83).

In Zeile 2 wird die Matrix der speziellen Eigenwertaufgabe um die gewünschte Eigenwertnäherung λ_i spektralverschoben und anschließend in das Produkt $(\mathbf{Q}_1\,\mathbf{R}_1)$ zerlegt. In Zeile 4 wird überprüft, ob die Anzahl Nullzeilen in $\mathbf{R}_1$ der Multiplizität p des Eigenwerts entspricht. Ist dies nicht der Fall, wird die Ähnlichkeitstransformation der spektralverschobenen Aufgabe durch das Produkt $(\mathbf{R}_1\,\mathbf{Q}_1)$ vervollständigt. In Zeile 6 wird die Eigenvektornäherung $(\mathbf{Q}_1\mathbf{I}^{(p)})$ durch lokale Iteration der letzten Zeilen und Spalten von $\hat{\mathbf{C}}_1$ verbessert. Ist der Fehler in $\hat{\mathbf{C}}_1$ ausreichend klein, erfolgt die Eigenvektorberechnung in Zeile 8, sonst wird eine zusätzliche QR-Zerlegung vorgenommen und die Eigenvektorberechnung nach Zeile 12 ausgeführt.

Ergibt die Überprüfung der Konvergenz in Zeile 4 den Wert *falsch*, so sind die letzten Spalten von $\mathbf{Q}$ gute Näherungen zu den Eigenvektoren des p-fachen Eigenwerts λ_i (Zeile 15).

Algorithmus 2 Inverse Matrixiteration - Eigenvektorberechnung

```
 1: while Subraum nicht vollständig bestimmt do
 2:    C_1 = (A − λ̂_i I)                          { Spektralverschiebung }
 3:    R_1 = Q_1^T C_1                            { QR-Zerlegung }
 4:    if Multiplizität p ≠ Anzahl Nullzeilen in R_1 then
 5:       Ĉ_1 = R_1 Q_1                           { RQ-Rekombination }
 6:       Ĉ_2 = U^T Ĉ_1 U                         { Lokale Iteration }
 7:       if ‖ĉ_{2_n}‖_2 < δ then
 8:          X = U Q_1 I^(p)                      { Eigenvektorberechnung }
 9:       end if
10:       if Multiplizität p ≠ Anzahl Nullzeilen in Ĉ_2 then
11:          R_2 = Q_2^T Ĉ_2                      { RQ-Rekombination }
12:          X = Q_1 Q_2 I^(p)                    { Eigenvektorberechnung }
13:       end if
14:    else
15:       X = Q_1 I^(p)                           { Eigenvektorberechnung }
16:    end if
17: end while
```

3.4.4 Komplexität

Betrachtet wird eine symmetrische Matrix $\mathbf{A}$ mit konvexer Profilstruktur, Bild (3.21). Die Dimension von $\mathbf{A}$ sei N und die mittlere Bandbreite b. Der numerische Aufwand der Iteration wird auf Grundlage der Anzahl Multiplikationen beurteilt. Die Rotation zur Berechnung eines Subdiagonalelementes $a_{ik}, k < i$ während der Zerlegung bzw. während der Rekombination wird als Schritt bezeichnet, die zyklische Durchführung aller Rotationen von Zerlegung und Rekombination, als Zyklus. Der Grundalgorithmus und die vorgestellten Erweiterungen werden im folgenden einzeln betrachtet. Der Gesamtaufwand der Berechnung und Speicherung der Aufgabe ist in Tabelle 3.6 zusammengefaßt.

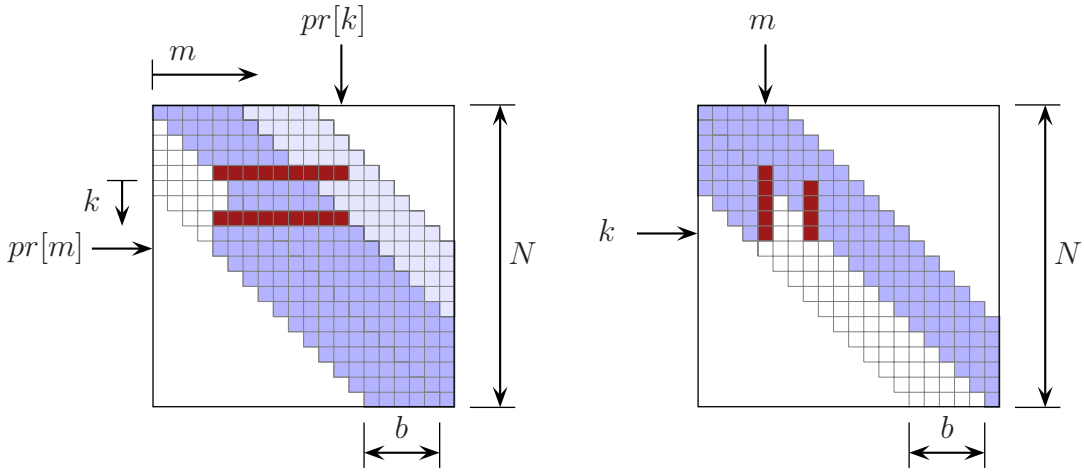


Bild 3.50: Modifizierter Zeilen- und Spaltenbereich der Zerlegung und Rekombination

QR-Algorithmus Die Berechnung der Dreiecksmatrix $\mathbf{R}$ erfordert näherungsweise 4 Multiplikationen in jeder der $1.5b$ Spalten für bN Koeffizienten. Der Faktor 1.5 berücksichtigt dabei die schrittweise temporäre Erweiterung des rechten Matrixprofils in Zeile k bei der schrittweisen Elimination der Subdiagonalelemente $a_{km}, m < k$. Bild 3.50, links, zeigt die Laufbereiche der Schleifenindizes m und k, sowie die letzten Indizes $pr[m]$ und $pr[k]$ der Summen im aktuellen Berechnungsschritt. Die Anzahl Multiplikationen M_{QR} der Zerlegung ergibt sich zu :

$$M_{QR} = \sum_{m=1}^{N-1} \left(\sum_{k=m+1}^{pr[m]} 4\,(pr[k]-m) \right) = \sum_{m=1}^{N-1} \left(\sum_{k=1}^{b} 4\,(b+k+1) \right) \tag{3.4.40}$$

$$= \sum_{m=1}^{N-1} (\,6b\,(b+1)) = (N-1)\,(6b\,(b+1)) \tag{3.4.41}$$

multiplikativer Aufwand QR : $\sim 6\,b^2\,N$

Die Rekombination $\hat{\mathbf{A}} = \mathbf{RQ}$ erfordert näherungsweise 4 Multiplikationen in jeder der $0.5b$ Spalten für bN Koeffizienten. Wegen der Symmetrie von $\hat{\mathbf{A}}$ sind nur die Schritte zur Berechnung der Subdiagonalelemente $\hat{a}_{km}, m < k$ erforderlich. Der Faktor 0.5 berücksichtigt die schrittweise Zerstörung der Koeffizienten $r_{km} = 0, m < k$ der Rechtsdreiecksmatrix $\mathbf{R}$. Bild 3.50, rechts, zeigt die Schleifenindizes k und m des aktuellen Berechnungsschrittes. Die Laufbereiche der Indizes m und k entsprechen den Laufbereichen in Bild 3.50, links. Die Anzahl Multiplikationen M_{RQ} der Zerlegung ergibt sich zu :

$$M_{RQ} = \sum_{m=1}^{N-1} \left(\sum_{k=m+1}^{pr[m]} 2\,(2(k-m)+1) \right) = \sum_{m=1}^{N-1} \left(\sum_{k=1}^{b} (2\,(k+1)+2k) \right) \quad (3.4.42)$$

$$= \sum_{m=1}^{N-1} (\,2b\,(b+2)) = (N-1)\,(2b\,(b+2)) \quad (3.4.43)$$

multiplikativer Aufwand RQ : $\sim\ 2\,b^2\,N$

Der Speicherbedarf der Zerlegung bestimmt den Gesamtspeicherbedarf der Eigenwertberechnung. Die Berechnung der Eigenwerte erfordert die Speicherung von $\sim\ ((2b+1)\cdot N - b^2)$ Koeffizienten. Zusätzlich ist für die Zerlegung die Bereitstellung eines Vektors der Länge b erforderlich. Für die Berechnung des Produktes $\mathbf{RQ}$ wird teilweise die Symmetrie der Koeffizientenmatrix ausgenutzt, für die Zerlegung $\mathbf{QR}$ und den Gesamtspeicherbedarf sind die Koeffizienten des gesamten Bandes erforderlich.

Gesamtspeicherbedarf (double): $(2\,b+1)\,N + O(b)$

Prekonditionierung Die Prekonditionierung wird auf ausgewählte Bereiche der Matrix ausgeführt. Der Berechnungsaufwand wird durch die Dimension p des eingesetzten Diagonalblocks $\mathbf{A}_k^{p\times p}$ bestimmt. Die Prekonditionierung wird in zwei Schritten vollzogen :

Im ersten Schritt wird der Diagonalblock $\mathbf{A}_k^{p\times p}$ mit dem QR-Algorithmus schwach diagonalisiert. Die Komplexität des Algorithmus für vollbesetzte symmetrische Matrizen beträgt für die Zerlegung $\mathbf{QR}$: $(\frac{4}{3}p^3 - \frac{4}{3}p)$ Multiplikationen und für die Rekombination $\mathbf{RQ}$: $\sim (\frac{2}{3}p^3 + 6p^2 - 10p)$ Multiplikationen. Die Berechnung der Eigenmatrix $\mathbf{X}_k^{p\times p}$ erfordert nochmals $(2p^2(p-1))$ Multiplikationen.

multiplikativer Aufwand I : $4\,p^3 + O(p^2)$

Im zweiten Schritt wird die Eigenmatrix $\mathbf{X}_k^{p\times p}$ mit den Zeilen und Spalten k bis $k+p$ der iterierten Matrix $\mathbf{A}_s$ innerhalb des Matrixbandes $2b$ multipliziert.

multiplikativer Aufwand II : $4\,p^2\,b$

Die durchschnittliche Blockdimension p, sowie die Anzahl prekonditionierbarer Blöcke ist abhängig von der Gestalt des Matrixprofils, das durch die Anzahl m der Freiheitsgrade pro Knoten aus der FE-Modellierung beeinflußt ist. Als maximale Blockdimension wird im folgenden $p = m$ angenommen. Wegen der geringen Matrixdimension und dem geringen Grad der Diagonalisierung erfordert die Berechnung wenige Zyklen (i.a. < 10 Zyklen). Die Anzahl prekonditionierter Blöcke wird mit $(\frac{N}{p})$ abgeschätzt. Ein Faktor $\frac{1}{6}$ berücksichtigt, daß die Prekonditionierung nur im unteren Matrixbereich prekonditioniert wird.

multiplikativer Aufwand (gesamt) : $\sim (\frac{N}{6p})\ (40\,p^3\ + 4\,p^2 b + O(p^2))$

Gesamtspeicherbedarf (double): $O(p^2)$

Jacobi-Randkorrektur Die Jacobi-Randkorrektur wird bei Bedarf eingesetzt. Dabei werden nur Nichtdiagonalelemente behandelt, die im Betrag größer einer vorgewählten ε-Schranke sind und deren Behandlung keine Profilerweiterung erfordern. Der Berechungsaufwand beschränkt sich auf die Zeilen und Spalten i und k der ausgewählten Koeffizienten a_{ik} am rechten unteren Matrixrand. Eine Abschätzung des numerischen Aufwands der Jacobi-Randkorrektur und der Prekonditionierung wird empirisch vorgenommen. Verschiedene Beispielrechnungen haben gezeigt, daß der Aufwand proportional zur Anzahl berechneter Eigenwerte k und der Bandbreite b ist.

multiplikativer Aufwand (gesamt) : $\sim k\,b$

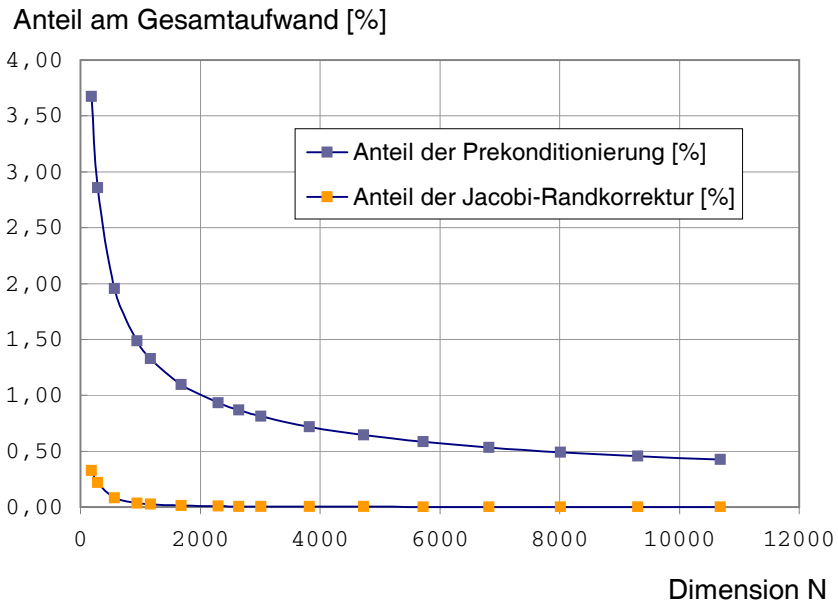


Bild 3.51: Anteil von Prekonditionierung und Jacobi-Randkorrektur am Gesamtaufwand

Bild 3.51 zeigt den prozentualen Anteil der Jacobi-Randkorrektur und der Prekonditionierung am multiplikativen Gesamtaufwand. Die Jacobi-Randkorrektur wurde dabei pro Iterationszyklus dreimal eingesetzt. Für einen operativen Aufwandsvergleich verschiedener Verfahren spielt die Jacobi-Randkorrektur keine Rolle. Der Anteil der Prekonditionierung liegt im allgemeinen bei einmaligem Einsatz pro Iterationszyklus unter 1% des Gesamtaufwands und ist damit ebenfalls für eine Aufwandsvergleich ohne Bedeutung.

Eigenvektoren Die Bestimmung der Eigenvektoren ist mit (3.4.35) unabhängig von der Berechnung der Eigenwerte. Der Aufwand für die Speicherung und Berechnung der Eigenvektoren wird dadurch erheblich reduziert. Die Komplexität zur Berechnung der Eigenvektoren eines q-fachen Eigenwertes λ_i erfordert $6b^2N$ Multiplikationen für die Zerlegung der spektralverschobenen Matrix $\mathbf{C} = (\mathbf{A} - \lambda_i\mathbf{I})$. Für das Produkt der Rotationsmatrizen zur Berechnung der Eigenvektoren eines q-fachen Eigenwertes sind $4q$ Multiplikationen erforderlich.

multiplikativer Aufwand : $\sim N(6b^2 + 4bq)$

Der Aufwand für die lokale Iteration ist wegen der geringen Dimension der Submatrizen und der starken Diagonaldominanz vernachlässigbar.

Operation	Berechnung	multiplikativer Aufwand	Speicherbedarf
1. QR-Zerlegung	$\mathbf{A} = \mathbf{QR}$	$s\,(6\,b^2\,N)$	$2\,(b+1)\,N + O(b)$
2. RQ-Rekombination	$\hat{\mathbf{A}} = \mathbf{RQ}$	$s\,(2\,b^2\,N)$	
3. Prekonditionierung	$\hat{\mathbf{A}} = \mathbf{U}^T\mathbf{A}\,\mathbf{U}$	$\frac{1}{6}s\,N\,(40\,p^2\,+4\,pb)$	p^2
4. Jacobi RK	$\hat{\mathbf{A}} = \mathbf{R}^T\mathbf{A}\,\mathbf{R}$	$s\,(k\,\sqrt{b})$	
Gesamt (Eigenwerte)		$\sim 8\,b^2\,N\,s\,c$	
5. Eigenvektoren	$(\mathbf{A} - \lambda_i\mathbf{I}) = \mathbf{QR}$	$k\,N(6b^2 + 4bq)$	$N\,q$

N	Dimension von $\mathbf{A}$	b	mittlere Bandbreite von $\mathbf{A}$
s	Anzahl der Zyklen	k	Anzahl der berechneten Eigenwerte
p	Dimension der Submatrizen $\mathbf{A}_k^{p\times p}$	q	Multiplizität von λ_i
c	Abminderungsfaktor (s. Seite 93)		

Tabelle 3.6: Komplexität der Inversen Matrixiteration

Abminderungsfaktor Der Gesamtaufwand der Eigenwertberechnung in Tabelle 3.6 ist ein theoretischer Wert, in dem verschiedene Effekte empirisch erfaßt sind.

Durch die stetige Verkleinerung der Aufgabe in der Dimension infolge des Deflationsprozesses verringert sich der multiplikative Aufwand pro Iterationszyklus zunehmend. Werden alle Eigenwerte bestimmt, wird dieser Effekt gut durch den Faktor $c \simeq 0.5$ erfaßt. Bei der Bestimmung einer beliebigen Anzahl von Eigenwerten kann c mit $(1. - \frac{1}{2}\frac{k}{N})$ abgeschätzt werden.

Mit zunehmendem Konvergenzfortschritt nimmt der Betrag der Koeffizienten im Profilrandbereich stetig ab und wird teilweise so klein, daß die Koeffizienten nicht weiter Teil der Iteration sind. Die Bandbreite verringert sich damit ebenfalls stetig. Auch dieser Effekt kann nur vorsichtig abgeschätzt werden, da er stark von der Eigenwertverteilung des Gesamtspektrums abhängig ist. Da dieser Effekt sich erst nach einer größeren Anzahl von Iterationen spürbar auf den Aufwand auswirkt, wird für weniger als $\frac{1}{2}N$ Eigenwerte der Gesamtaufwand der Eigenwertberechnung mit $c = (1. - \frac{1}{2}\frac{k}{N})$ unterschätzt. Ein Faktor $c = (1. - \frac{1}{3}\frac{k}{N})$ spiegelt den tatsächlichen Aufwand für diese Subraumdimensionen besser wider.

Insgesamt ist der abgeschätzte Gesamtaufwand nahe am tatsächlichen Aufwand und wird mit dem Ergebnis aus Tabelle 3.6 gut approximiert.

Der Aufwand für die Eigenvektorberechnung besitzt im allgemeinen keinen iterativen Charakter, da ein Großteil der Eigenvektoren durch eine QR-Zerlegung bestimmt wird. Die Formel aus Tabelle 3.6 zur Abschätzung des Aufwands pro Eigenvektor ist ziemlich exakt am tatsächlichen Aufwand.

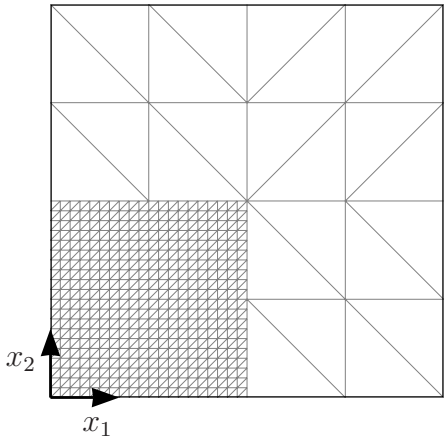
3.4.5 Berechnungsbeispiel mit den Erweiterungen des QR-Verfahrens

Die Tauglichkeit der eingeführten Erweiterungen zur Eigenwert- und Eigenvektorbestimmung an Profilmatrizen großer Dimension wird beispielhaft an einer Modellaufgabe aufgezeigt. Die Verbesserungen des Verfahrens hinsichtlich des Konvergenzverhaltens und der Zuverlässigkeit der Iteration, sowie die erzielte Genauigkeit werden untersucht und bewertet. Die Formeln zur Abschätzung des multiplikativen Aufwands und des Speicherbedarfs werden verifiziert.

Als Berechnungsbeispiel wird eine ungelagerte Quadratplatte gewählt. Damit wird ein in mehrfacher Hinsicht schwieriger Berechnungsfall im Bauingenieurwesen aufgegriffen. Die mittlere Bandbreite b der zu lösenden speziellen Eigenwertaufgabe beträgt ungefähr $1.75\,\sqrt{N}$. Sie liegt damit weit über den üblichen Bandbreiten ($\sqrt{N}$) der Systemmatrizen ebener Tragwerke. Der numerische Aufwand und der Speicherbedarf steigen damit auch bereits für geringere Dimensionen (~ 1000) merkbar an. Die Anzahl mehrfacher Eigenwerte ist wegen der Symmetrie der Geometrie für ebene Tragwerke maximal und damit ein besonders schwieriger Berechnungsfall für viele etablierte Eigenwertverfahren.

Beispiel 6 : Eigenformen einer ungelagerten Quadratplatte

Das vollständige Eigenwertspektrum und zugehörige Eigenformen einer Quadratplatte werden mit dem vorgestellten Verfahren bestimmt. Die Platte besitzt eine knotenkonzentrierte Massenbelegung. Die diagonale Massenmatrix wird durch HRZ-Lumping [7] bestimmt.



Abmessungen	:	$[0,1] \times [0,1]$
Plattendicke	:	0.1
Elastizitätsmodul	:	1.0
Poissonzahl	:	0.0
Rohdichte	:	1.0

Bild 3.52: Elementierung der ungelagerten Quadratplatte

FE-Modell

Das Finite-Element-Netz wurde doppelsymmetrisch mit Dreieckelementen vernetzt. Jeder Quadrant ist mit 20×20 Maschen unterteilt. Das verwendete Finite Element ist

nach der Theorie von Kirchhoff für dünne Platten entwickelt [27],[50]. Als Stützwerte werden an jedem Eckknoten der Dreieckelemente folgende globale Freiheitsgrade geführt :

$$
\begin{array}{lcll}
u_3 & & & \text{transversale Verschiebung in Richtung der } x_3\text{-Achse} \\
\alpha_1 & = & \dfrac{\partial u_3}{\partial x_2} & \text{Rotation um die } x_1\text{-Achse} \\
\alpha_2 & = & -\dfrac{\partial u_3}{\partial x_1} & \text{Rotation um die } x_2\text{-Achse}
\end{array}
$$

Für die Interpolation der Verschiebung u_3 wird ein unvollständiger kubischer Polynomansatz mit 9 freien Parametern gewählt. Eine vollständige Elemententwicklung ist in [27] gezeigt.

Systemgleichungen

Die Dimension des Gleichungssystems beträgt 5043, die mittlere Bandbreite 123. Die Konvexität der Systemmatrix erfordert einen zusätzlichen Speicheraufwand von ca. 1% des Gesamtspeicherbedarfs und liegt damit um fast die Hälfte über dem durchschnittlichen Wert aller durchgeführten Berechnungen. Der Zusatzspeicher beinhaltet zusätzliche Koeffizienten aus einer Profiloptimierung, die auf den Einsatz der Prekonditionierung abgestimmt ist. 1260 Eigenwerte der Aufgabe besitzen Multiplizität > 1.

Dimension N	:	5 043
Mittlere Bandbreite b	:	123
Speicheraufwand (64-Bit Zahl)	:	1 254 609

Bild 3.53: Matrixeigenschaften

Konvergenzverhalten

Eigenwerte : Das vollständige Eigenwertspektrum wurde in 4424 Zyklen bestimmt. Der Aufwand entspricht 0.45 Iterationen der ursprünglichen Matrix (Dimension N) pro Eigenwert. Die Nulleigenwerte der drei Starrkörperbewegungen der Platte wurden gemäß der Theorie (Kap.3.3.2) mit einem Zyklus bestimmt. Mit zunehmender Bandbreite und Dimension der Matrix erfordert die Bestimmung von Nulleigenwerten wegen des ansteigenden Fehlerniveaus häufig zwei Iterationszyklen.

Von 5043 Eigenwerten haben drei Eigenwerte nicht in sortierter Reihenfolge konvergiert. Alle drei Eigenwerte wurden jeweils um eine Stelle in der Sortierung vertauscht.

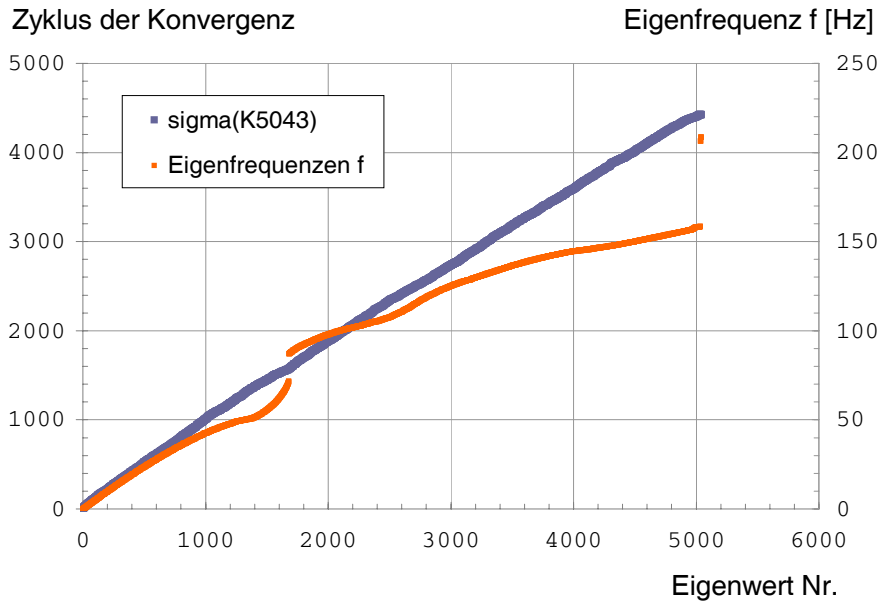


Bild 3.54: Konvergenzverlauf der Iteration für $\sigma(\mathbf{K}_{5043})$

Bild 3.54 zeigt einen linearen Konvergenzverlauf der Berechnung (blau). Dargestellt ist der Iterationszyklus der Konvergenz für die Eigenwerte $\lambda_i, (i = 1, \ldots, 5043)$. In rot dargestellt ist das Frequenzspektrum der ungelagerten Platte. Auf die Eigenwertlücke am Ende des Spektrums ($\sim 208Hz$) folgen zwei Eigenwertcluster der Dimension 4 mit einer Trennung der Eigenwerte von $O(\epsilon\|\mathbf{A}\|_F)$. Mit der vorgestellten Strategie zur Eigenvektorberechnung wurden die Eigenvektoren zuverlässig, mit akzeptablem Fehler $(relativer Fehler :\sim 1.0e-10)$ bestimmt.

Schritt	1. Zyklus	gesamt
QR/RQ	$\mathbf{6.29e+08}$	$\mathbf{1.41e+12}$
	$(6.21e+08)$	$(1.35e+12)$
Prekonditionierung	$\mathbf{8.11e+06}$	$\mathbf{9.84e+09}$
	$(1.54e+06)$	$(6.84e+09)$
Jacobi-Randkorrektur	-	$\mathbf{7.36e+05}$
	-	$(6.20e+05)$

Tabelle 3.7: Multiplikativer Aufwand der Eigenwertberechnung

Tabelle 3.7 zeigt den multiplikativen Aufwand der Eigenwertberechnung für die einzelnen Teilschritte. Betrachtet wird jeweils der Aufwand für einen Zyklus der ursprünglichen Matrix und der Gesamtaufwand der Berechnung. Die Werte in Klammer sind geschätzte Werte nach Abschnitt 3.4.4. Der Gesamtaufwand für den QR/RQ-Schritt wurde mit $c = 0.5$ abgeschätzt. Für $c = 0.53$ entspricht die Abschätzung dem tatsächlichen Aufwand. Die Prekonditionierung kann wegen der nicht a priori bekannten Anzahl von Iterationszyklen zur Diagonalisierung der Submatrix nur schwach abgeschätzt werden. Die Jacobi-Randkorrektur kommt erst mit dem Konvergenzfortschritt am Ende der Matrix zum Einsatz.

Anzahl Zyklen	0	1	2	3	4	5	6	7	8
Anteil bestimmter Eigenwerte [%]	53.2	20.0	17.6	6.0	2.1	0.7	0.2	0.1	0.1

Tabelle 3.8: Aufwandsverteilung der Eigenwertberechnung

Tabelle 3.8 zeigt den multiplikativen Aufwand der Eigenwertberechnung, sowie den Anteil Eigenwerte in Prozent, die in $k(k = 0, \ldots, 8)$ Zyklen bestimmt werden. Über 90% aller Eigenwerte werden innerhalb 2 Zyklen bestimmt. Die Anzahl Eigenwerte, die mehr als 2 Zyklen erfordern ist gleichmäßig über den Iterationsverlauf verteilt. Wegen der kontinuierlichen Reduzierung der Dimension der Aufgabe im Zuge der Deflation, entspricht der Aufwand für das erste Drittel der Eigenwerte näherungsweise dem Aufwand zur Bestimmung der restlichen Eigenwerte.

Eigenvektoren : 5043 Eigenvektoren wurden in 4669 Zerlegungen bestimmt. Für 886 Eigenvektoren war für ausreichende Konvergenz jeweils eine zusätzliche Zerlegung erforderlich. 1685 lokale Iterationen wurden durchgeführt und reduzierten damit den Gesamtaufwand der Eigenvektorbestimmung um ca. 26%.

Bild 3.55 (Seite 98) zeigt den relativen Fehler $(\|\mathbf{r}\|_2/\hat{\lambda})$ für die Restnorm aller Eigenzustände (blau), sowie den relativen Fehler $((\hat{\lambda} - \rho(\mathbf{A}, \hat{\mathbf{x}}))/\hat{\lambda})$ für die Differenz von Eigenwertnäherung und Rayleigh-Quotient (rot). Der Mittelwert für den relativen Fehler der Restnorm beträgt $8.718e - 10$ (grün), für den relativen Fehler im Rayleigh-Quotienten $2.689e - 12$ (grün).

Bild 3.56 (Seite 98) zeigt die Fehlerschranken $sin\theta(\mathbf{x}\hat{\mathbf{x}})$ für die Richtung der Eigenvektornäherungen nach Kapitel 3.3.3. Dargestellt sind die Mittelwerte für Intervalle der Länge 200. Die Schranken aus 3.3.3 sind für betragsgleiche Eigenwerte und zugeordneten Eigenvektoren nicht anwendbar. Intervallweise sind daher nur Eigenvektoren für getrennte Eigenwerte berücksichtigt.

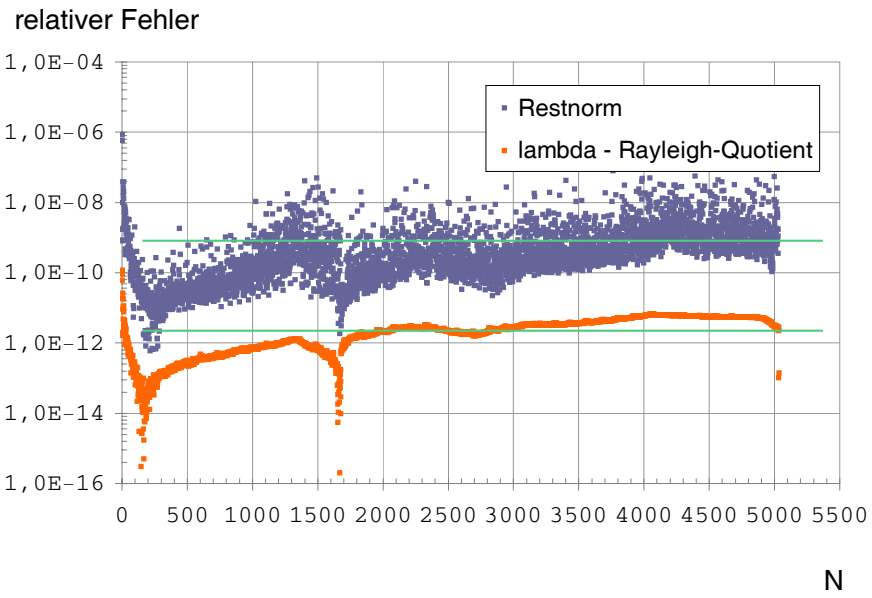


Bild 3.55: A posteriori Fehler für $\sigma(\mathbf{K}_{5043})$

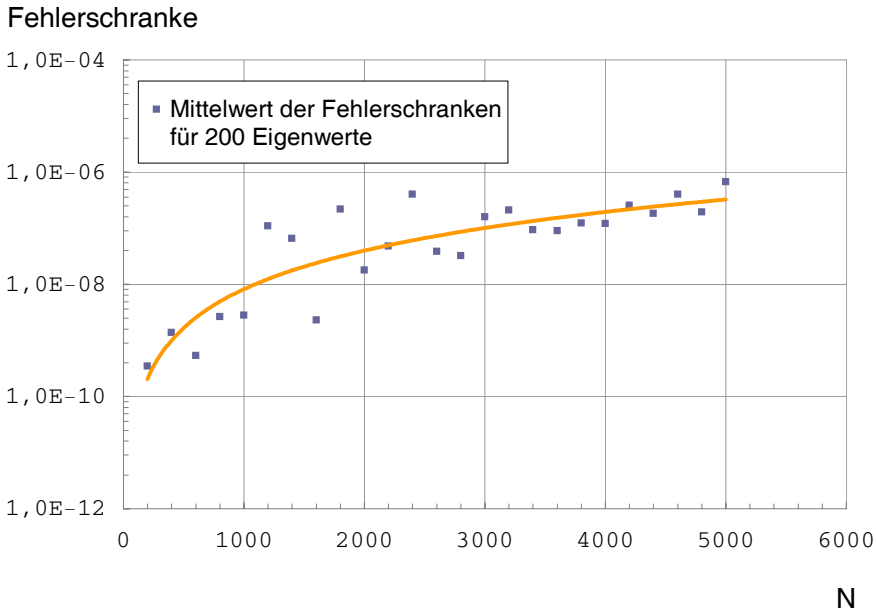


Bild 3.56: Fehlerschranken der Eigenvektoren für $\sigma(\mathbf{K}_{5043})$

Ergebnisinterpretation

Die Bilder 3.57 bis 3.60 (Seite 101) zeigen skalierte Eigenformen der ungelagerten Platte als Isoflächenmodell und Verformungsfigur. Die kleinen Frequenzen $(f_{12}, f_{14}, f_{17}, f_{22})$ zeigen Grundformen der Biege- und Torsionsschwingung. Die höheren Frequenzen dagegen lassen Wiederholungen der Grundformen erkennen. Die Bilder geben einen Eindruck über das tatsächliche Schwingverhalten der ungelagerten Platte, daß sich aus einer linearen Kombination der Eigenformen zusammensetzt.

Die Darstellung der Bilder 3.62 bis 3.64 (Seiten 102, 103) als Isolinienmodell zeigt den hohen Symmetriegrad und den kontinuierlichen Verlauf der Eigenformen. Dargestellt sind die acht ersten Biege- und Torsionsmoden. Die Eigenfrequenzen $f_{7/8}$ und $f_{9/10}$ gehören jeweils zu Eigenwerten mit Multiplizität 2. Die Nulllinie (gelb) der symmetrischen Eigenformen f_4 und f_6 verläuft exakt durch den Mittelpunkt der Platte. Die Eigenfrequenzen f_1 bis f_3 haben den Wert Null. Die zugehörigen Eigenformen repräsentieren die Starrkörperbewegungen der ungelagerten Platte.

i	$\hat{\lambda}_i$	$\rho(\hat{\mathbf{x}})$	$f[Hz]$	$\|\mathbf{r}\|_2/\hat{\lambda}$	$\|\rho-\hat{\lambda}\|/\hat{\lambda}$
2274	417770.010970	417770.010969	102.87	$2.61e-12$	$2.61e-12$
2275	417770.010974	417770.010969	102.87	$1.28e-11$	$1.28e-11$
2276	417770.060202	417770.060206	102.87	$7.58e-12$	$7.56e-12$
2277	417844.735586	417844.735585	102.88	$2.61e-12$	$2.60e-12$

Tabelle 3.9: Frequenzcluster

Einen Eindruck über die Genauigkeit der Berechnung gibt Tabelle 3.9. Gezeigt ist ein enges Frequenzcluster mit den zugehörigen Eigenwertnäherungen in Spalte 2. Die Trennung des doppelten Eigenwerts $\lambda_{2274/2275}$ von den Eigenwerten λ_{2276} und λ_{2277} ist kleiner $(max|\lambda|\cdot 4.0e-08)$ bzw. $(max|\lambda|\cdot 5.0e-05)$ und kann damit als sehr klein betrachtet werden. Spalte 3 zeigt den aus der Eigenvektornäherung ermittelte Rayleigh-Quotient. Die Spalten 4 und 5 enthalten die relativen Fehler der Eigenwertaufgabe bzw. des Rayleigh-Quotienten. Das Orthogonalitätslevel des Clusters liegt bei $4.0e-11$ und ist damit besser als $(max|\lambda|\cdot\epsilon)$. Die Orthogonalität zwischen den Vektoren $\mathbf{x}_{2275}$ und $\mathbf{x}_{2276}$ erfüllt $4.0e-10$, die Orthogonalität zwischen allen anderen Vektoren liegt bei der Maschinengenauigkeit ϵ.

Die Eigenwertnäherungen erfüllen nach Kapitel 3.3.3 mindestens die Schranke $\xi(=\lambda_i + max|\lambda|\,\epsilon\,c)$. Als Schätzwert für den Faktor c wird der Zyklus der Bestimmung der Eigenwerte verwendet. Alle vier Eigenwerte wurden im Zyklus 2043 bestimmt. Die Eigenwerte erfüllen damit alle die Schranke $\lambda_i + 7.8e-07$, was durch

die Fehlerschranke nach Temple-Kato bestätigt wird. Damit sind mindestens sechs Dezimalstellen zuverlässig bestimmt.

Der relative Fehler im Rayleigh-Quotienten ist ein Maß für die Genauigkeit der berechneten Eigenvektornäherung. Die Ergebnisse in Spalte 5 (Tabelle 3.9) lassen auf eine sehr hohe Genauigkeit für die Eigenvektornäherung schließen. Die Bilder 3.65 bis 3.66 (Seiten 104 und 105) zeigen die den gezeigten Eigenfrequenzen zugeordneten Eigenformen als Isolinienmodell. Die Isoliniendarstellung läßt die hohe Symmetrie der Formen gut erkennen und bestätigt den Eindruck über die Genauigkeit in Tabelle 3.9.

Die Eigenformen $\hat{\mathbf{x}}_{2274/75}$ zum doppelten Eigenwert $\lambda_{2274/75}$ sind um $\frac{\pi}{2}$ zueinander gedreht und antisymmetrisch. Der Vergleich der beiden Eigenformen zeigt einen sehr hohen Grad an Übereinstimmung in allen Bereichen. Zur besseren Orientierung sind verschiedene Bereiche blau eingerahmt.

Die Eigenformen $\hat{\mathbf{x}}_{2276}$ und $\hat{\mathbf{x}}_{2277}$ zeigen einen hohen Symmetrie- bzw. Antimetriegrad jeweils um die Mittelsenkrechte und die Diagonalen.

Bild 3.57: $f_{12} = 0.3532Hz$, $f_{14} = 0.5218Hz$

Bild 3.58: $f_{17} = 0.6174Hz$, $f_{22} = 0.7924Hz$

Bild 3.59: $f_{52} = 2.1865Hz$, $f_{76} = 3.4176Hz$

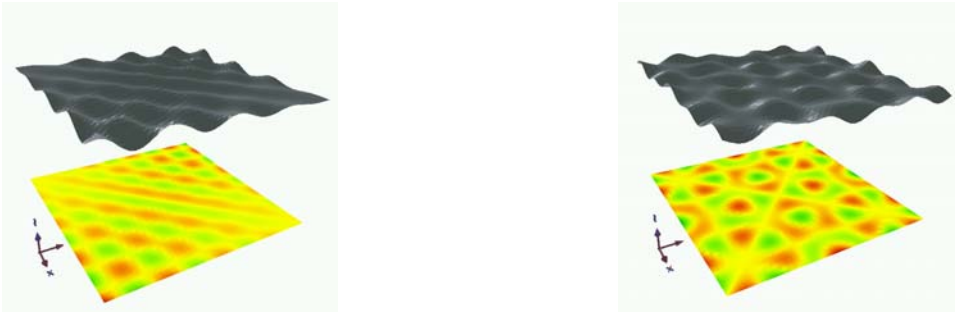


Bild 3.60: $f_{100} = 4.7834Hz$, $f_{102} = 4.7835Hz$

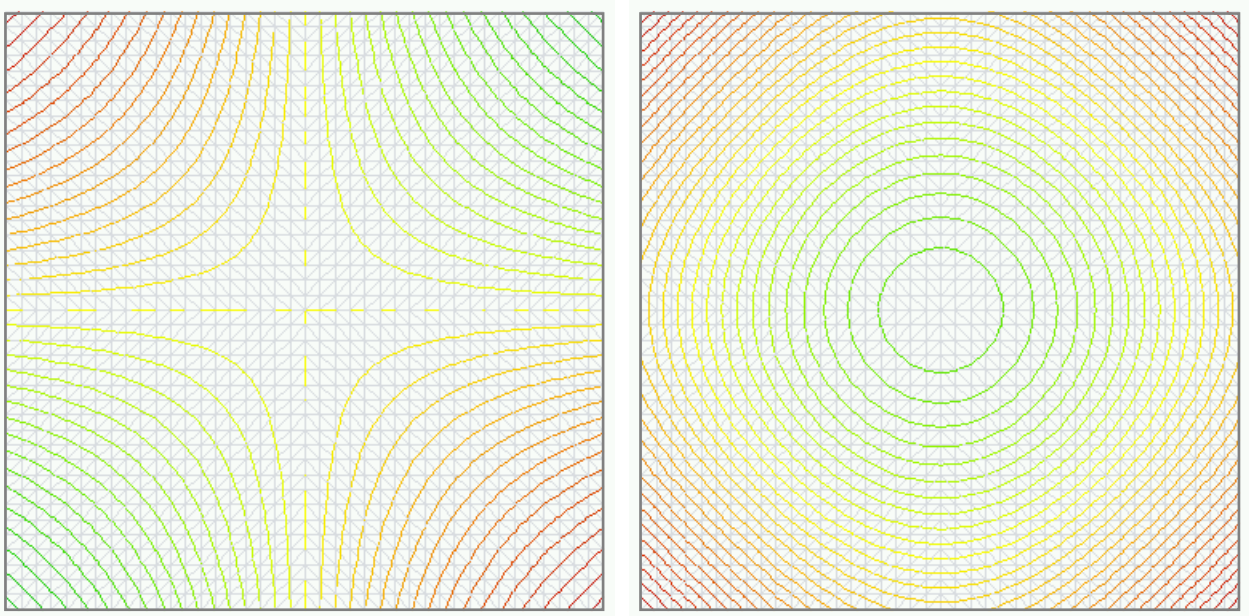


Bild 3.61: Eigenformen zu den Frequenzen $f_4 = 0.0725Hz$ und $f_5 = 0.1023Hz$

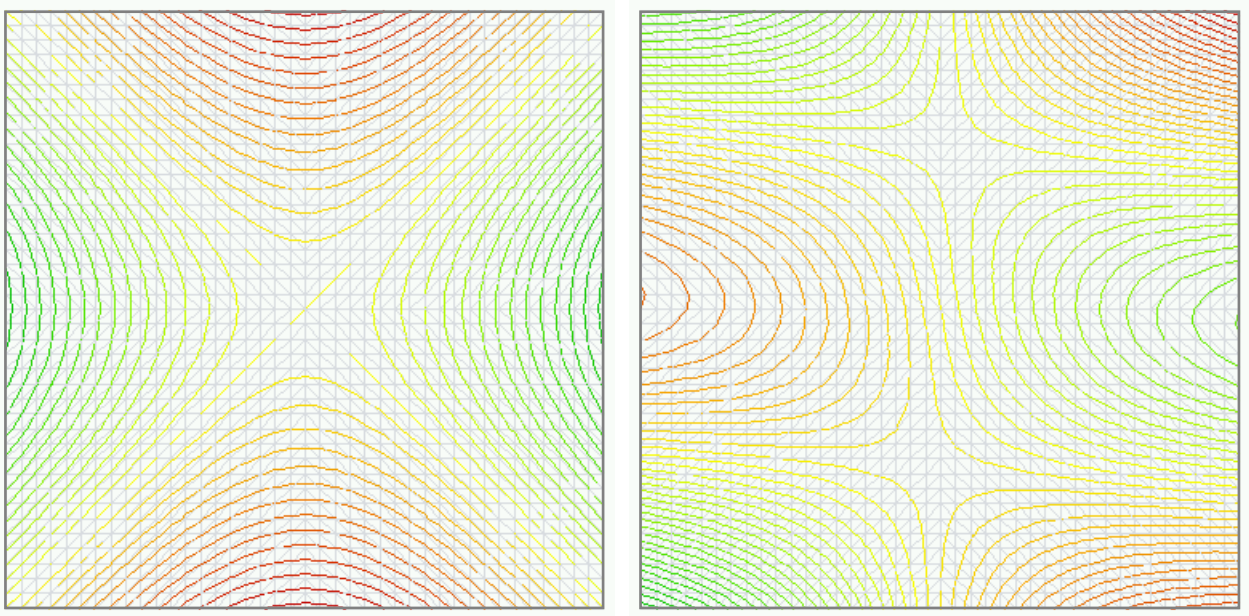


Bild 3.62: Eigenformen zu den Frequenzen $f_6 = 0.1025Hz$ und $f_7 = 0.1771Hz$

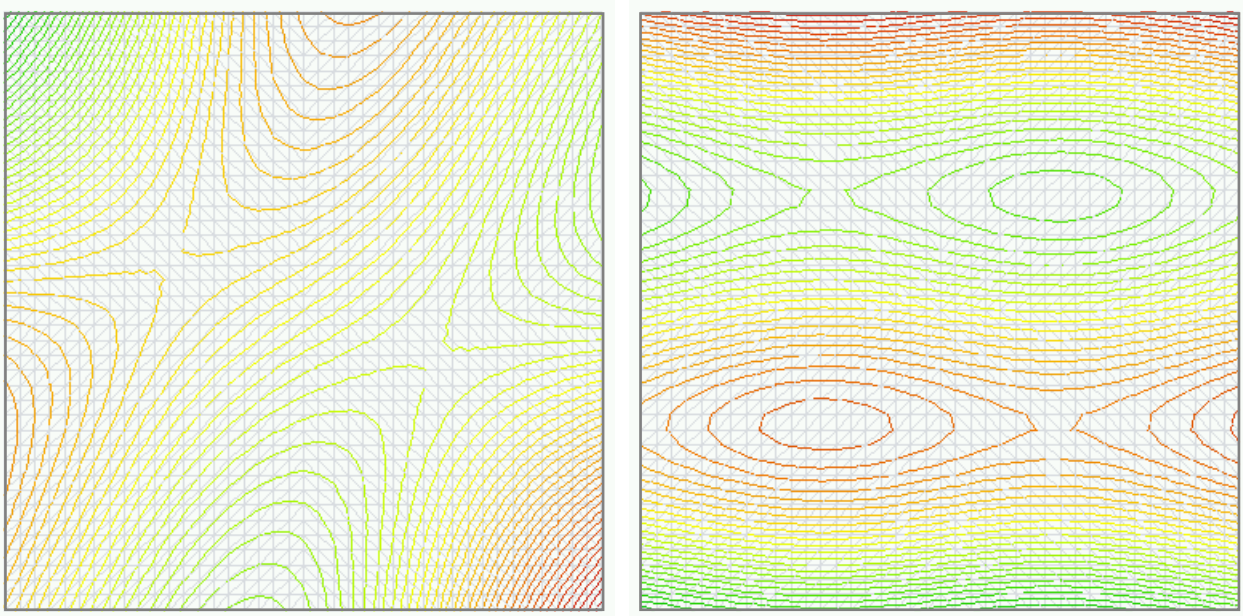


Bild 3.63: Eigenformen zu den Frequenzen $f_8 = 0.1771Hz$ und $f_9 = 0.2821Hz$

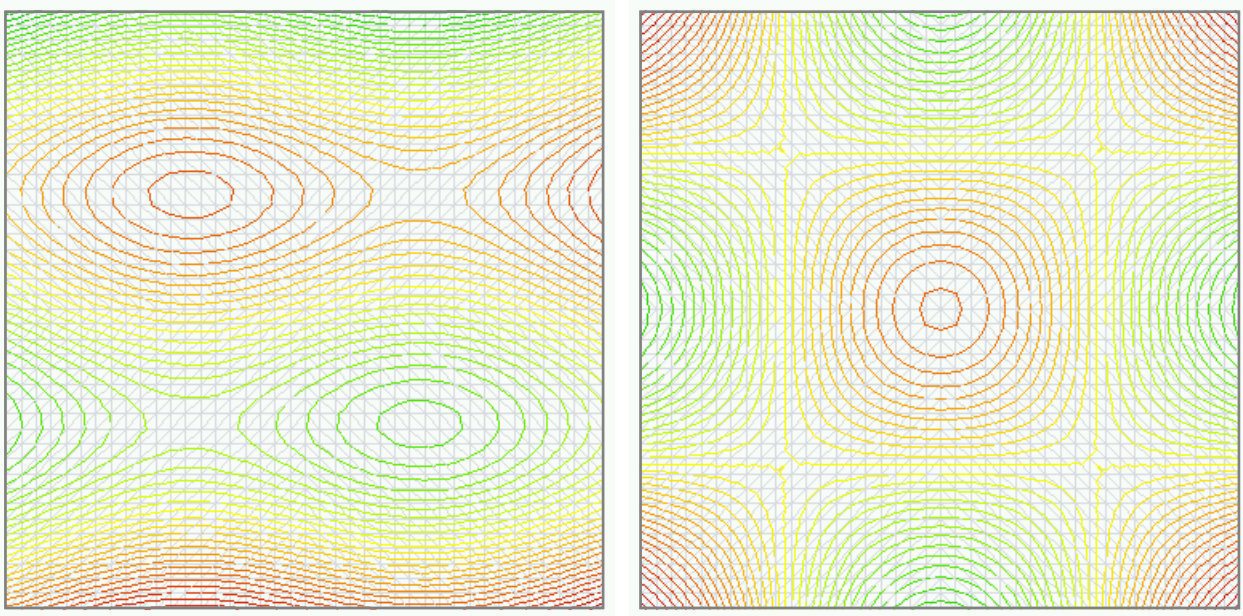


Bild 3.64: Eigenformen zu den Frequenzen $f_{10} = 0.2821Hz$ und $f_{11} = 0.3223Hz$

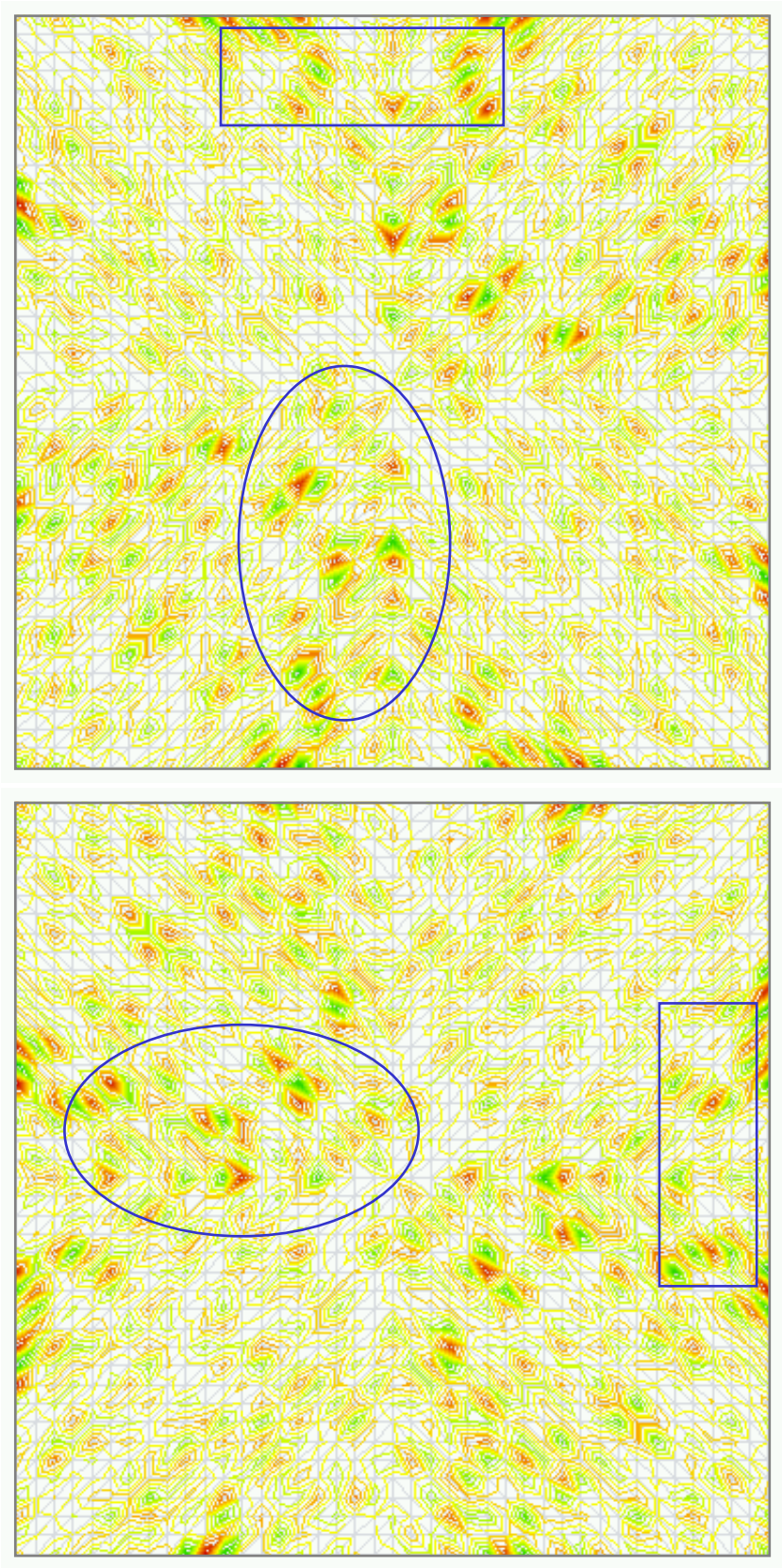


Bild 3.65: Eigenformen zu den Frequenzen f_{2274} und f_{2275}

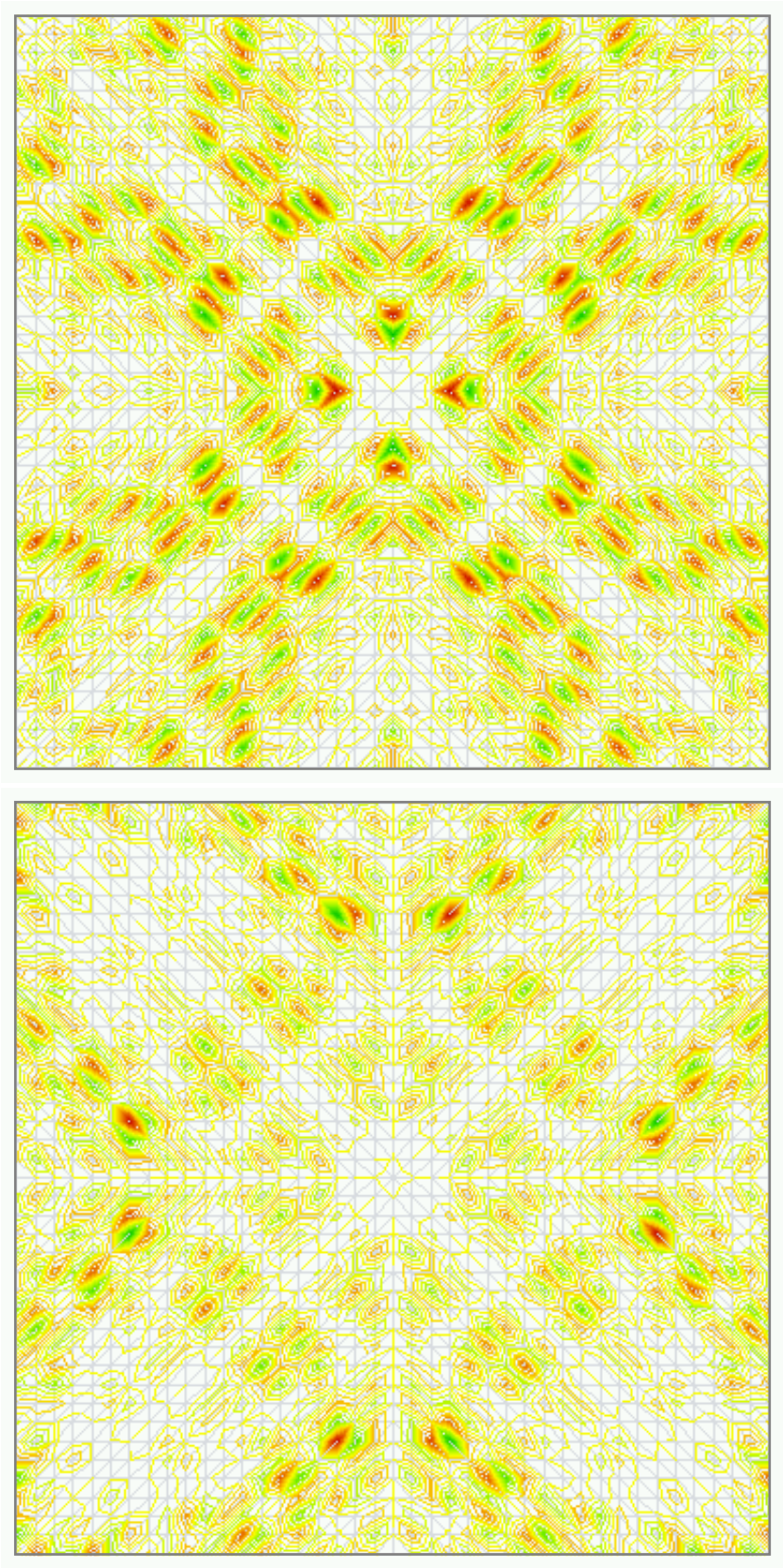


Bild 3.66: Eigenformen zu den Frequenzen f_{2276} und f_{2277}

3.5 Implementierung

3.5.1 Einleitung

Die ingenieurgerechte Bearbeitung von Aufgabenstellungen des Bauingenieurwesens im Rechner erfordert eine Zerlegung der Aufgabe in Modelle, Prozesse und Methoden. Für die Modellierung jeder dieser Teilaufgaben hat sich eine objekt-orientierte Vorgehensweise als zweckmäßig erwiesen und etabliert. Insbesondere bei der softwaretechnischen Umsetzung numerischer Methoden, beispielsweise der Methode der Finiten Elemente hat die objekt-orientierte Formulierung wesentlich zur Effizienzsteigerung beigetragen.

Die traditionelle Modellierung der Ingenieuraufgabe für den Einsatz der Methode der Finiten Elemente umfaßt die Bildung von Objekten zur Beschreibung von Geometrie und physikalischen Verhaltens, sowie zur Analyse der Aufgabenstellung mit Algorithmen und Gleichungssystemen. Die konventionelle Vorgehensweise wie sie in [16] dargestellt ist, stützt sich auf die detaillierte objekt-orientierte Formulierung eines Kernmodells als Datenbasis für die Speicherung des zu analysierenden Bauwerks und seines Verhaltens. Die Untersuchungen und Berechnungen der Aufgabe erfolgen typisch in einem Analyseobjekt, das den Zugriff auf die Datenbasis steuert, die erforderlichen Gleichungssysteme aufbaut und verschiedene Berechnungsalgorithmen anstößt. Die Steuerung und Durchführung einer Analyse dieser Form ist stark eingeschränkt, da sie problemorientiert formuliert ist und keinen Modellierungsspielraum einräumt.

Für wissenschaftliche Untersuchungen sind häufig verschiedene Analyseformen zu verschiedenen Aufgabenstellungen erforderlich, die auch eine teilweise Kopplung von Teilaufgaben erfordern. Vorhandene Softwaresysteme sind für wissenschaftliche Untersuchungen häufig ungeeignet, da sie die beschriebenen Anforderungen nicht erfüllen. Weitere Aspekte erschweren oft die Nutzung der Software :

- Die verwendeten Datenstrukturen des Kernmodells sind nicht ausreichend von der Analyse entkoppelt, erfordern eine hohe Einarbeitungszeit und sind nur bedingt erweiterbar.
- Zwischen den Modellen existieren Objektabhängigkeiten, die eine Austauschbarkeit von Teilmodellen verhindern und die Flexibilität der Implementierung einschränken.
- Die Wiederverwendung von Quellcode ist infolge einer komplexen Datenstruktur und unzureichender Transparenz nicht oder nur unzureichend möglich.
- Die Gesamtimplementierung umfaßt mehrere tausend Zeilen Quellcode, erfordert einen hohen Wartungsaufwand und ist stark fehleranfällig.

Wünschenswert wäre eine Software, die den Zusammenbau einer Untersuchung aus verschiedenen Analysekomponenten ermöglicht, ein hohes Maß an Flexibilität aufweist, einfach erweiterbar ist, Mehrfachimplementierungen vermeidet und über eine starke Transparenz verfügt.

Die softwaretechnische Umsetzung des vorgestellten Verfahrens erfolgt in der objektorientierten Programmiersprache *JAVA*. Als Analyseplattform für die nachfolgenden Untersuchungen und Vergleichsrechnungen wurde im Rahmen dieser Arbeit das Finite-Element-Programmsystem *FELINA (Finite Elements for Linear & Nonlinear Analysis)* entwickelt und implementiert. *FELINA* ist eine objektorientierte Modellierung der Finiten-Elemente-Methode, deren Implementierung folgende Zielsetzungen verfolgt :

- Die Implementierung gewährleistet eine strikte Trennung zwischen dem Kernmodell und dem Analysemodell, sowie zwischen den verschiedenen Teilmodellen der Analyse. Die Kommunikation der einzelnen Modelle erfolgt ausschließlich über die Definition von Schnittstellen. Die Sichtbarkeit auf das Modell und seine Objekte wird auf die wesentlichen Eigenschaften und Fähigkeiten beschränkt. Modell und Objekte bleiben damit vor unerlaubtem Zugriff geschützt. Die verschiedenen Modelle sind durch das Schnittstellenkonzept austauschbar, änderbar und einfach zu erweitern. Dadurch wird ein hohes Maß an Flexibilität bereitgestellt, das sich insbesondere bei der Integration verschiedener Analyseformen (linear, nichtlinear etc.) in das Programmsystem auszeichnet.

- Die Anzahl der Schnittstellenmethoden wird kleinstmöglich gehalten. Alle Schnittstellen und damit ein Großteil der Schnittstellenmethoden werden durch abstrakte Klassen vorimplementiert. Die Implementierung einer Grundfunktionalität, die allen objektorientierten Finite-Elemente-Systeme gemein ist, entfällt für den Entwickler. Der Einsatz der verschiedenen Teilmodelle wird damit stark vereinfacht und gewinnt an Attraktivität. Die Fehleranfälligkeit der Schnittstellenimplementierungen wird reduziert.

- Eine konsequente Strukturierung aller Teilmodelle durch Abstraktion ihres Verhaltens wird durch den Einsatz objektorientierter Prinzipien und Konzepte ermöglicht und unterstützt. Die Übersichtlichkeit der Hierarchie und der Implementierung der Klassen eines Modells wird dadurch maßgeblich gefördert. Mehrfachimplementierungen von Klassen und Methoden verschiedener Teilmodelle werden gänzlich vermieden. Eine Definition der wesentlichen Prinzipien, Konzepte und Begriffe der objektorientierten Methode wie sie im weiteren verwendet werden ist in [15] und [16] vorgenommen.

Im Gegensatz zum Kernmodell der Software besitzen die Algorithmen des Analysemodells teilweise einen stark prozeduralen Charakter. Insbesondere die iterative Lösung nichtlinearer Gleichungssysteme ist mit einem hohen numerischen Aufwand verbunden, der sich vorrangig durch wiederholte Abläufe verschiedener Berechnungsschemata ergibt. Um der effizienten Bearbeitung und Lösung großer Gleichungssysteme Rechnung zu tragen, wird eine detaillierte Modellierung der einzelnen Komponenten der Gleichungssysteme und anderer Größen der Linearen Algebra als nicht zweckmäßig betrachtet. Auf eine Klassenstruktur für Matrizen und Vektoren, die eine Kapselung von Datenstruktur und Algorithmen forciert, wird verzichtet. *JAVA* bietet keinen Mechanismus zum Überladen von Operatoren und bietet damit keinen wesentlichen Vorteil für die Verknüpfung algebraischer Größen durch eine komplexe Klassenstruktur. Stattdessen wird für das zu lösende Gleichungssystem eine programminterne Speicherstruktur gewählt, die die

Effizienz des Lösungsalgorithmus unterstützt und Teil des Analysemodells ist. Grundlegende Matrix-Vektor-Operationen für vollbesetzte $(n \times m)$-Matrizen werden durch eine abstrakte Serviceklasse bereitgestellt.

Die Modellierung der verschiedenen Anforderungen an die Teilmodelle infolge unterschiedlicher Aufgabenstellungen erfolgt durch das Vererbungskonzept. Als Vaterklassen werden abstrakte Klassen definiert, die eine oder mehrere Schnittstellen des Teilmodells implementieren und Methoden für abgeleitete Klassen deklarieren. Das reibungsfreie Zusammenwirken der Teilmodelle wird dadurch gesichert und die Transparenz für den Entwickler stark gefördert.

Die Beziehungen zwischen den Objekten des Analysemodells werden im wesentlichen durch Assoziationen modelliert. Dies ist eine Folge des strukturellen Aufbaus der Analyse und dem Verzicht auf die Modellierung von Matrizen und Vektoren als Objekte der Analyse. Im Gegensatz zum Analysemodell werden die Objektbeziehungen des Kernmodells vorrangig durch Aggregation und Komposition modelliert und durch Mengenbildung strukturiert.

Im folgenden wird ausschließlich die Modellierung und Implementierung des Analysemodells von *FELINA* vorgestellt. Es beinhaltet die Implementierung der Inversen Matrixiteration und weiterer Verfahren zur Eigenwertanalyse, die in nachfolgenden Kapiteln behandelt sind. Darüberhinaus dient das Modell als Basisplattform für Schwingungs- und Stabilitätsanalysen und stellt damit auch Teilmodelle für dynamische und statische, lineare und nichtlineare Analysen zur Verfügung.

Eine vollständige Beschreibung aller Klassen und Schnittstellen von *FELINA* ist in [34] gegeben.

3.5.2 Objektorientierte Modellierung

Modellstruktur

Die Objektstruktur des Analysemodells wird in eigenständigen Teilmodellen modelliert. Die Teilmodelle werden unabhängig voneinander entwickelt und kommunizieren über ihre Schnittstellendefinition. Sie kapseln die Objekte zur Beschreibung ihrer Eigenschaften und ihrer bereitgestellten Funktionalität. Die Strukturierung der Teilmodelle in *JAVA* erfolgt durch die Vereinbarung von Paketen. Sie bilden einen Namensraum für die Teilmodelle und steuern die Sichtbarkeit der Modellobjekte nach außen. Die Schnittstellendefinitionen erfolgen ausschließlich nach dem Grundsatz der *Holeschuld*. Eigenschaften und Fähigkeiten von Objekten anderer Teilmodelle werden über eine Schnittstellenmethode für den paketinternen Gebrauch angefordert. Das Setzen von Eigenschaften, insbesondere das explizite Setzen von Objektreferenzen in einem Initialisierungsschritt entfällt und verkleinert damit wesentlich die Fehleranfälligkeit der Implementierungen.

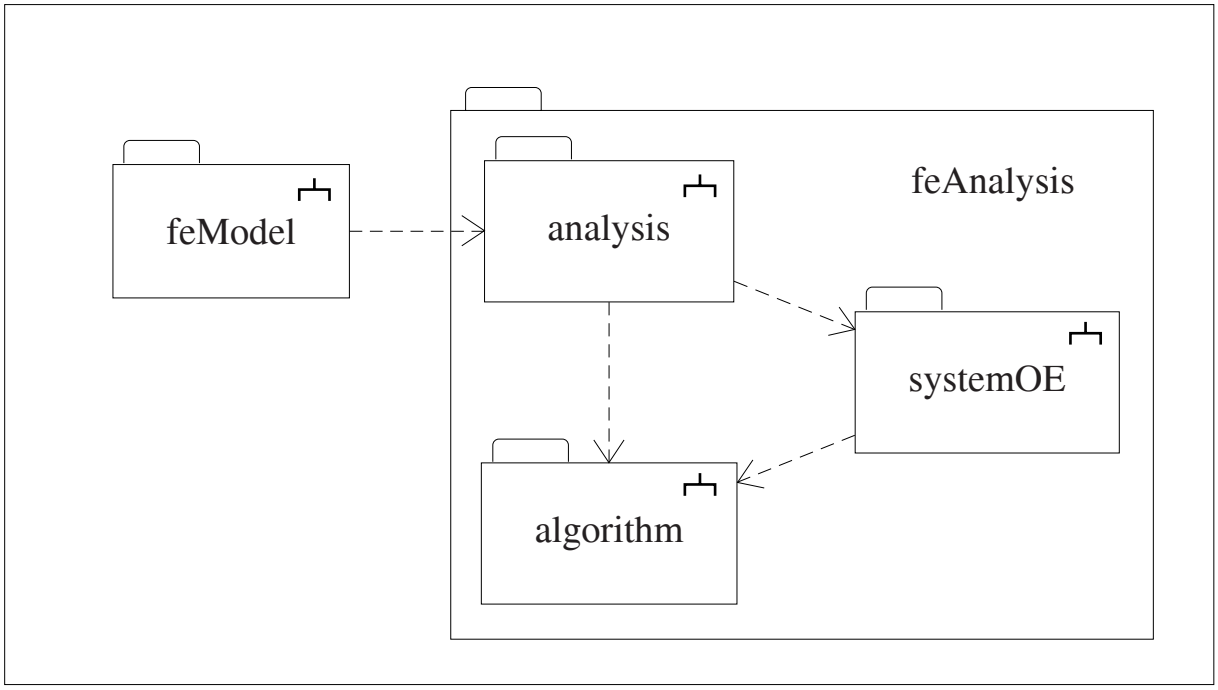


Bild 3.67: Paketstruktur des FE-Kernmodells/Analysemodells

Die zur objektorientierten Modellierung der Aufgabe verwendete Symbolik entspricht dem UML *(Unified Modelling Language)*-Standard [12]. Zur Förderung der Übersichtlichkeit der Modelldiagramme und Implementierungsauszüge werden Sichtbarkeitsmodifikatoren, bis auf Konstruktormethoden, nicht gezeigt.

Bild 3.67 zeigt die Paketstruktur des Kern- und Analysemodells mit seinen Teilmodellen.

Das entwickelte Modell soll insbesondere die einfache Zusammenführung der einzelnen Komponenten der Analyse unterstützen und eine einfache Adaption neuer Analyseschemata ermöglichen. Das vollständige Programmsystem *FELINA* besteht aus folgenden Teilmodellen :

1. **feModel** Das Teilmodell *feModel* definiert und speichert die Objekte zur Beschreibung der Finite-Element Eigenschaften. Die interne Strukturierung folgt den klassischen Regeln und Konzepten objektorientierter Modellierung physikalischer Aufgaben mit der Finiten-Element-Methode, wie beispielsweise in [29] gezeigt.

2. **analysis** Die Analyse wird als Aggregation verschiedener Objekte einer ingenieurgerechten Strukturanalyse modelliert. Das Teilmodell *analysis* steuert die Zusammenführung aller zur Modellierung der Aufgabe erforderlichen Objekte. Insbesondere spezifiziert und steuert dieses Modell die schrittweise Untersuchung der Aufgabe. Die zur Definition der Analyse erforderlichen Objekte resultieren aus der Modellierung der Aufgabe mit der Finiten-Element-Methode. Sie sind Strukturelemente der nachfolgenden Teilmodelle zur Beschreibung geometrischen, topologischen und physikalischen Verhaltens des diskretisierten Lösungsgebietes, zur Beschreibung der algebraischen Bestimmungsgleichungen und zur Lösung des resultierenden Gleichungssystems der Aufgabe.

3. **systemOE** Das Teilmodell *systemOE* beinhaltet Objekte zum Aufbau, zur Speicherung und zur Beschreibung der Systemgleichungen. Die unterschiedlichen Objekttypen dieses Modells unterscheiden sich vorrangig in der Speicherstruktur der Gleichungen.

4. **algorithm** Das Teilmodell *algorithm* stellt Objekte zur Lösung der Systemgleichungen bereit. Die implementierten Algorithmen operieren auf assoziierte Objekte aus 3.

Klassenstruktur

Zur Strukturierung der Objektmengen des Analysemodells und seiner Teilmodelle werden Äquivalenzrelationen formuliert. In *JAVA* werden diese Äquivalenzrelationen als Klassen realisiert. Die Klassen eines Teilmodells werden in einem Paket zusammengefaßt. Die Sichtbarkeit von Objekteigenschaften bleibt auf den paketinternen Gebrauch beschränkt. Beziehungen zwischen Objekten und Objekteigenschaften verschiedener Pakete werden durch die Definition einer oder mehrerer Schnittstellen für jedes Paket hergestellt. Die Spezialisierung der Teilmodelle für eine bestimmte Form der Analyse, beispielsweise einer Eigenwertanalyse, erfolgt durch die Erweiterung der Schnittstellendefinition und ihrer Implementierung.

Die Schnittstellendefinitionen und ihre Implementierungen werden im folgenden vorgestellt. Die Tauglichkeit des Gesamtkonzepts für eine generalisierte Analyseplattform wird beispielhaft gezeigt.

Modell *feModel* : Schnittstellendefinition Das Finite-Element-Objektmodell ist im Paket *feModel* realisiert. Auf eine Beschreibung des paketinternen Aufbaus wird im folgenden verzichtet. Eine ausführliche *JAVA*-Dokumentation aller Schnittstellen, Klassen etc. liegt in [34] vor. Nachfolgend ist die Hauptschnittstelle zu den Objektmengen des Modells gezeigt.

```
1  interface IfaceModel {
2
3    abstract Set getElements();
4    abstract Set getNodes();
5    abstract Set getLoads();
6    abstract Set getSupports();
7    abstract Set getBoundary();
8  } //eoi
```

Listing 3.1: Interface `IfaceModel`

Das Interface `IfaceModel` stellt die Objektmengen des Finite-Element-Objektmodells zur Verfügung. Paketintern werden diese Objekte in Hashmaps verwaltet. Alle Objekte sind durch eindeutige Zeichenketten persistent identifiziert. Mit dem Identifikator als Argument der Hashfunktion wird ein schneller und einfacher Zugriff auf die Objekte ermöglicht. Die Operationen des Analysemodell erfordern keinen selektiven Zugriff auf die Finite-Element-Objekte. Stattdessen wird hier der schnelle Zugriff auf unstrukturierte Objektmengen gefordert. Aus diesem Grund werden die Objekte des Finite-Element-Modells als solche bereitgestellt.

In *JAVA* werden unstrukturierte Mengen durch die Standard-Interfaces `Collection` und `Set` definiert. Das Interface `Set` ist eine Spezialisierung des Interface `Collection` und repräsentiert eine Menge im mathematischen Sinn. Objekte sind in den aufgeführten Mengen nicht mehr als einmal enthalten. Die Methoden des Interfaces stellen Objekte zur Modellierung der Geometrie, der Topologie und des physikalischen Verhaltens bereit.

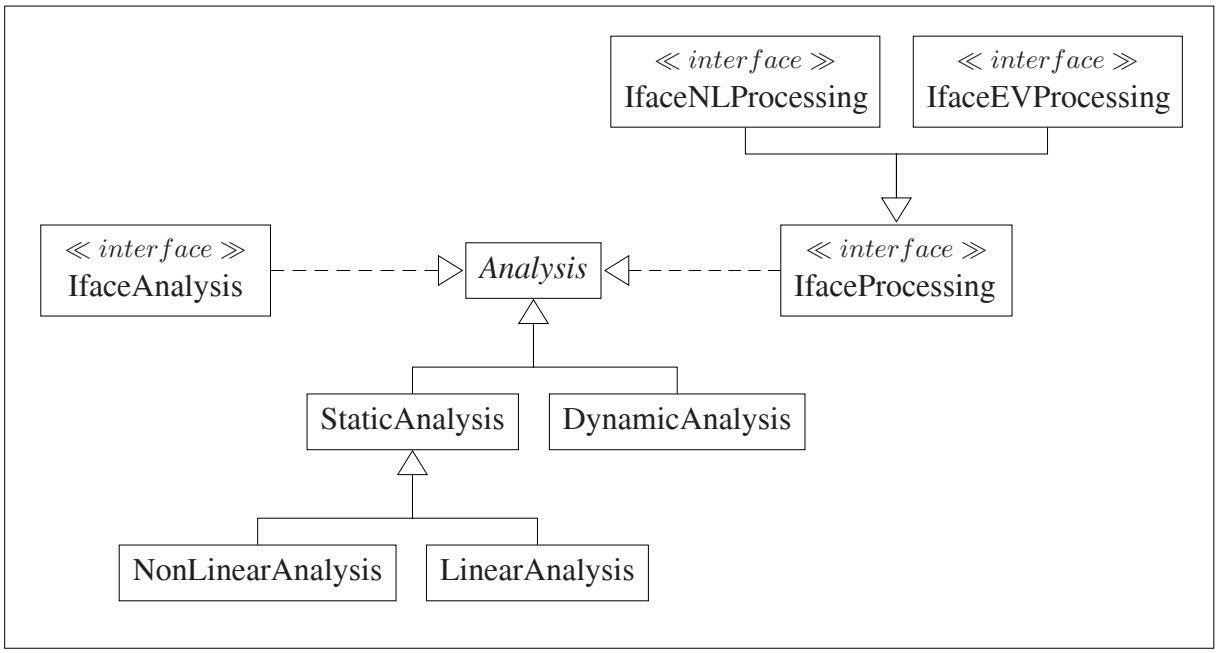


Bild 3.68: Klassenstruktur des Teilmodells *analysis*

Modell *analysis* : Schnittstellendefinition Das Teilmodell *analysis* (Bild 3.68) besitzt zwei Schnittstellendefinitionen. Das Interface `IfaceAnalysis` bildet eine einfache Schnittstelle zur Implementierung der Benutzerschnittstelle der Finite-Element-Applikation.

```
1  interface IfaceAnalysis {
2
3    abstract void perform( IfaceAlgorithm algorithm, IfaceSOE soe );
4    abstract void setResults();
5  } //eoi
```

Listing 3.2: Interface `IfaceAnalysis`

Die Methode `perform(...)` setzt die modellierte Aggregation der Analyse um. Die Schnittstellen der Teilmodelle werden der Analyse über die Parameterliste zugänglich gemacht. Die Methode steuert die schrittweise Untersuchung der Aufgabe. Die Methode `setResults()` speichert die Ergebnisse der Analyse im Finite-Element-Objektmodell des Paketes *feModel*.

Ein weiteres Interface `IfaceProcessing` stellt Methoden für den modellinternen Gebrauch in den Teilmodellen *systemOE* und *algorithm* zur Verfügung. Die Methoden des Interface erlauben die Reaktion auf ein verändertes Finite-Element-Objektmodell im Paket *feModel*. Die Bereitstellung dieser Funktionalität ist zum Beispiel bei inkrementellen Berechnungsschritten einer Untersuchungsmethode erforderlich.

```
1  interface IfaceProcessing {
2
3    abstract void systemStiffness();
4    abstract void statusVector();
5    abstract int  dimension();
6    ...
7  } //eoi
```

Listing 3.3: Interface `IfaceProcessing`

Die Methode `systemStiffness()` bewirkt den teilweisen oder vollständigen Neuaufbau der Steifigkeitsmatrizen des zu lösenden Gleichungssystems. Die Methode `statusVector()` bewirkt den Aufbau eines Vektors zur Identifikation eingeprägter primaler bzw. dualer Größen. Die Methode `int dimension()` liefert die Dimension des Gleichungssystems. Weitere Methoden die beispielsweise zum Update eines Lastvektors führen sind in dieser Schnittstellendefinition denkbar, waren aber in den vorliegenden Implementierungen nicht gefordert.

Für eine Spezialisierung der Analyse, beispielsweise im Rahmen einer Schwingungs- oder Stabilitätsanalyse, sind teilweise Methoden erforderlich, die Zugriff auf analysebedingte Eigenschaften der beteiligten Objekte erfordern. Diese Methoden werden in Subinterfaces definiert (Listing 3.4). Die Algorithmen einer Eigenwertanalyse beispielsweise erfordern die Dimension des zu bestimmenden Subraums, die als Objekteigenschaft der Analyse gespeichert ist.

```
1  interface IfaceEVProcessing extends IfaceProcessing {
2
3    abstract int subspace();
4    ...
5  } //eoi
```

Listing 3.4: Subinterface `IfaceEVProcessing`

Modell *analysis* : Schnittstellenimplementierung Die Schnittstellen des Teilmodells *analysis* werden durch die abstrakte Klasse `Analysis` (Bild 3.68, Seite 112) implementiert. Die Klasse `Analysis` ist Vaterklasse und damit die Generalisierung der verschiedenen Analyseformen. Sie speichert die Referenzen auf die an der Analyse beteiligten Teilmodelle *feModel*, *algorithm* und *systemOE*. Zusätzlich besitzt diese Klasse Objektattribute zur Speicherung der aus dem Paket *feModel* zur Verfügung gestellten Objektmengen. Sämtliche Operationen auf die Objektmengen aus *feModel* erfolgen ausschließlich in der Klasse `Analysis` und davon abgeleiteten Kindklassen. Das zu analysierende FE-Modell wird im Konstruktor der Klasse als Objektattribut gespeichert.

```
1   abstract class Analysis implements IfaceAnalysis, IfaceProcessing {
2
3     int             dimension;       // Anzahl Freiheitsgrade........
4     Set             elements;        // Obj.menge -> Finite Elemente.
5     Set             nodes;           // Obj.menge -> Geometrie.......
6     Set             loads;           // Obj.menge -> Belastung.......
7     ...
8     IfaceSOE        soe;             // Gleichungssystem.............
9     IfaceModel      model;           // FE-Modell....................
10    IfaceAlgorithm  algorithm;       // Loesungsalgorithmus..........
11
12    public Analysis( IfaceModel model ) {
13      this.model     = model;
14      this.elements  = model.getElements();
15      ...
16    } //eom
17    ...
```

Listing 3.5: Klasse `Analysis`

Bis auf die Methode `setResults()` des Interfaces `IfaceAnalysis` werden alle Schnittstellenmethoden durch die Klasse `Analysis` implementiert :

```
17  int  sysDimension()     {...}  // ermittelt Anz. Freiheitsgrade(FG)
18  void systemProfile()    {...}  // bestimmt linkes Matrixprofil.....
19  void systemStiffness()  {...}  // bestimmt Systemsteifigkeit.......
20  void statusVector()     {...}  // identifiziert eingepraegte FG....
21  void perform(...)       {...}  // fuehrt Analyse durch.............
```

Listing 3.6: Klasse `Analysis`

Eine Spezialisierung der Analyse für verschiedene Aufgabenstellungen erfolgt in Kindklassen der Klasse `Analysis`. Die Spezialisierung wird exemplarisch für die Klasse `DynamicAnalysis` gezeigt. Die Klasse implementiert das Subinterface `IfaceEVProcessing` und erweitert damit den externen Zugriff auf Methoden der Eigenwertanalyse.

Die Klasse `DynamicAnalysis` kann als Generalisierung für Untersuchungen eingesetzt werden, die eine Lösung der allgemeinen Eigenwertaufgabe $\mathbf{K}\,\mathbf{u}_i = \lambda_i \mathbf{M}\,\mathbf{u}_i$ erfordern :

```
1   class DynamicAnalysis extends Analysis implements IfaceEVProcessing
2   {
3     int subspace;                // Groesse des zu best. Subraums....
4
5     public DynamicAnalysis( int subspace, IfaceModel model) {
6       super(model);
7       this.subspace = subspace;
8     } //eom
9     ...
```

Listing 3.7: Subklasse `DynamicAnalysis`

Das zu analysierende Finite-Element-Modell und die Anzahl der zu bestimmenden Eigenzustände $(\lambda_i, \mathbf{u}_i)$ werden im Konstruktor der Klasse gesetzt. Als Erweiterung von `Analysis` stellt die Klasse zusätzliche Methoden bereit :

```
9   void systemMass() {...}       // best. Massenbelegung des Systems.
10  void setResults() {...}       // setzt Berechnungsergebnisse......
11  int  subspace()   {...}       // liefert zu best. Subraumgroesse..
12  ...
```

Listing 3.8: Subklasse `DynamicAnalysis`

Die Interfacemethode `perform(...)` (Interface IfaceAnalysis) wird in der Klasse `DynamicAnalysis` überschrieben. Die Methode baut das zu lösende Gleichungssystem auf, stößt den Berechnungsalgorithmus an und speichert die Berechnungsergebnisse im Finite-Element-Objektmodell des Paketes *feModel*. Zusätzlich zum Aufbau der Systemsteifigkeit, wird die Massenbelegung des Systems ermittelt. Der Profilvektor der Systemmatrizen wird unabhängig von der Speicherstruktur der Matrizen bereits in der Methode `perform(...)` der Superklasse aufgebaut. Als Bindeglied zwischen den Element- und Systemgleichungen werden beim Aufbau der Systemgleichungen die Schnittstellenmethoden des Interface `IfaceSOE` aufgerufen. Der Lösungsalgorithmus für ein aufgebautes Gleichungssystem wird durch die Schnittstellenmethoden des Interface `IfaceAlgorithm` angestoßen. Die erforderlichen Objekte der beiden Schnittstellen werden in der Superklasse `Analysis` durch die Methode `perform(...)` als Objektattribute gespeichert.

```
12   void perform( IfaceAlgorithm algorithm, IfaceSOE soe ) {
13     super.perform(algorithm,soe);
14     systemStiffness();          // Aufbau der Systemgleichungen.....
15     systemMass();
16     statusVector();
17     algorithm.solve();          // Durchfuehrung der Berechnung.....
18     algorithm.results();        // Bereitstellen der Ergebnisse.....
19     setResults();               // Speichern der Ergebnisse.........
20   } //eom
```

Listing 3.9: Subklasse `DynamicAnalysis`

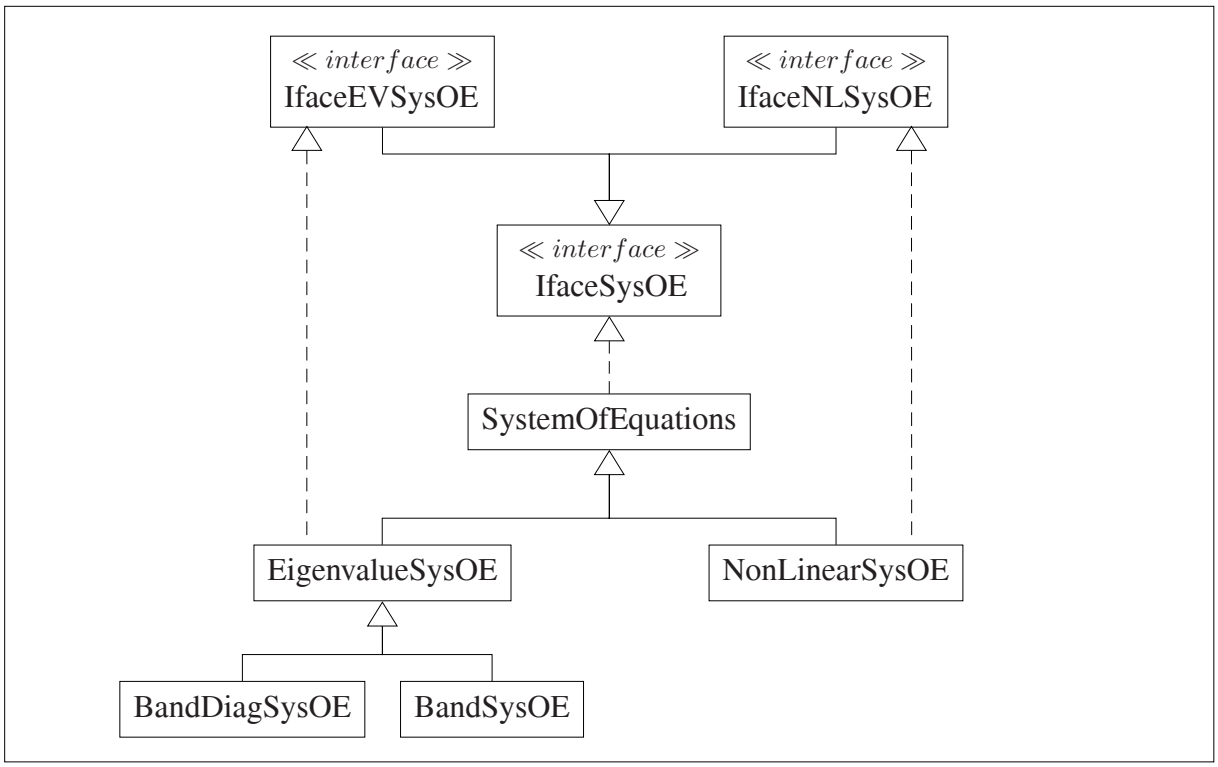


Bild 3.69: Klassenstruktur des Teilmodells *systemOE*

Modell *systemOE* : Schnittstellendefinition Das Paket *systemOE* (Bild 3.69) enthält die Klassen zur Speicherung, zum Aufbau und zur Verwaltung der Systemgleichungen. Für den Zugriff auf die Systemgleichungen stellt das Paket *systemOE* das Interface `IfaceSOE` bereit, das von der Klasse `SystemOfEquations` implementiert wird.

Die algebraische Form der Gleichungssysteme und die Anzahl ihrer Systemgrößen ist durch die Art der Analyse festgelegt. Die interne Speicherstruktur der Systemgrößen dagegen ist unabhängig von der Analyse. Um bei der Speicherung des Gleichungssystems eine weitgehende Unabhängigkeit von der Analyseart zu schaffen, werden die Systemgrößen unabhängig von ihrer Anzahl, von ihrem algebraischen Typ und unabhängig von ihrem Datentyp in einer generalisierten Datenstruktur gespeichert. Die verschiedenen Eigenschaften bezüglich ihrer Speicherstruktur werden in Subklassen der Klasse `SystemOfEquations` modelliert.

Art und Umfang der Speicherallokierung für die Systemmatrizen wird durch die Konstanten `PROFILE`, `BAND` und `FULL` des Interface `IfaceSOE` festgelegt. Die Methode `allocate(...)` bekommt eine der Konstanten als Argument `type` übergeben. Die Methode stellt für alle Systemgrößen des Gleichungssystems den Speicher bereit. Jede Systemgröße wird durch eine eindeutige Zeichenkette identifiziert und in der Klasse `SystemOfEquations` in einer Hashmap abgespeichert. Der Datentyp der Systemgröße wird durch eine Konstante festgelegt, die mit einem primitiven Datentyp assoziiert wird.

```
1   interface IfaceSOE {
2
3     static final int PROFILE  = 1;
4     static final int BAND     = 2;
5     static final int FULL     = 3;
6
7     abstract void   allocate( String id, Class c, int n, int m,
8                                                          int type );
9     abstract void   add( String id, int[] index, double[][] a );
10    abstract void   add( String id, int[] index, double[] a );
11    abstract void   set( String id, Object o );
12    abstract Object get( String id );
13    abstract void   profile( int[] index );
14    abstract void   status( String id, boolean b, int i, double v );
15  }//eoi
```

Listing 3.10: Interface `IfaceSOE`

Für die Speicherallokation in der Methode `allocate(...)` wird der Datentyp zur Laufzeit durch Introspektion aus dem Übergabeparameter `Class c` ermittelt. Diese Vorgehensweise wird durch die in *JAVA* zur Verfügung stehende Reflexion ermöglicht. Die Implementierung derselben Schnittstellenmethode für verschiedene primitive und referenzierte Datentypen wird dadurch auf eine Methode reduziert. Die Konstante vom Datentyp `Class` wird durch die Verknüpfung eines primitiven Datentyps mit dem Schlüsselwort **`class`** erzeugt. Die Parameter `n` und `m` legen die Anzahl Zeilen und Spalten der Systemgröße fest. Das Erzeugen einer Systemmatrix mit Profilstruktur zur Speicherung der Systemsteifigkeit oder das Erzeugen eines Vektors zur Speicherung des Profils beispielsweise wird durch folgende Aufrufe vorgenommen :

```
1   allocate("stiffness", double[].class,dim,dim,IfaceSOE.PROFILE);
2   allocate("profile",   int.class,dim,1,0);
```

Listing 3.11: Erzeugen verschiedener Felder für primitive Datentypen

Mit den Methoden `get()` und `set(...)` kann auf die Matrizen und Vektoren des Gleichungssystems innerhalb und außerhalb des Paketes zugegriffen werden. Mit den Methoden `add(...)` des Interface `IfaceSOE` wird die Verbindung zwischen den Element- und Systemgrößen hergestellt. Die Elementvektoren bzw. Elementmatrizen werden über die in einem Indexvektor gespeicherten Systemindizes in das Gleichungssystem übertragen. Das Erstellen des Matrixprofils mit Hilfe der Systemindexvektoren der Finiten-Element-Objekte erfolgt über die Methode `profile(...)`. Mit der Methode `status(...)` werden eingeprägte primale und duale Freiheitsgrade identifiziert und im Gleichungssystem einsortiert.

Modell *systemOE* : Schnittstellenimplementierung Die Implementierung der Schnittstelle `IfaceSOE` erfolgt durch die Klasse `SystemOfEquations` oder eine davon abgeleitete Subklasse. Objekte von `SystemOfEquations` oder davon abgeleiteten Klassen operieren ausschließlich auf die algebraischen Größen der Analyse. Eine Hashmap `algebra` dient als Objektverzeichnis für alle Größen des zu lösenden Gleichungssystems. Gespeichert werden nur die tatsächlich erforderten Matrizen und Vektoren des Gleichungssystems, das durch die Objekte des Analysemodells spezifiziert und aufgestellt wird.

Die Schnittstellenmethoden werden als Standardimplementierung für Gleichungssysteme mit symmetrischen, zeilenweise gespeicherten Profilmatrizen bereitgestellt. Andere Speichermodelle, die speziell auf Algorithmen zur Eigenwertbestimmung oder auf Algorithmen zur Lösung linearer Gleichungssysteme ausgelegt sind, werden in entsprechenden Subklassen implementiert. Der Implementierungsaufwand beschränkt sich dabei auf die Methoden `add(...)` zum Zusammenbau der Systemmatrizen.

```
1   class SystemOfEquations implements IfaceSOE {
2
3     HashMap algebra = new HashMap(); // Verz. algebr. Systemgroessen
4     int             dimension;       // Anzahl Systemgleichungen....
5     IfaceProcessing analysis;        // Verknuepfung zur Analyse....
6
7     public SystemOfEquations( IfaceProcessing analysis ) {
8       this.analysis  = analysis;
9       this.dimension = analysis.dimension();
10      ...
11    } //eom
12    ...
```

Listing 3.12: Abstrakte Klasse `SystemOfEquations`

Die Allokation des Speichers für die Systemmatrizen erfolgt in den Methoden `set(...)`. Die Methoden ermitteln den Datentyp des zu erzeugenden Feldes, allokieren entsprechend der geometrischen Dimension des Feldes den Speicher und speichern die Feldreferenz im Objektverzeichnis `algebra`.

Nachfolgend ist die Methode zum allokieren des Speichers für zweidimensionale Felder gezeigt (Listing 3.13). Übergabeparameter der Methode sind der Identifikator der algebraischen Größe, der Datentyp des Feldes als Konstante, sowie ein Vektor mit der Anzahl Koeffizienten pro Zeile.

```
12  protected void set( String id, Class c, int[] n ) {
13
14    Object array = Array.newInstance(c,n.length); // Referenzvektor.
15    Object row   = null;
16    Class  component = c.getComponentType();      // best. Datentyp.
17    for( int i=0; i<n.length; i++ ) {
18      row = Array.newInstance(component,n[i]);    // Zeilenreferenz.
19      Array.set(array,i,row);         // speichern der Zeilenreferenz
20    }
21    algebra.put(id,array);            // speichern der Feldreferenz..
22  } //eom
```

Listing 3.13: Methode zur Speicherallokation

Die Klasse `SystemOfEquations` stellt die volle Funktionalität zum Aufbau von Gleichungssystemen einer linearen bzw. nichtlinearen statischen Analyse bereit, die auch vollständig bei einer Eigenwertanalyse erfordert und eingesetzt wird. In einer abgeleiteten Klasse `EigenvalueSOE` werden zusätzliche Methoden bereitgestellt, die das Gleichungssystem für eine Eigenwertanalyse aufbereiten. Desweiteren stellt diese Klasse Methoden bereit, die eine Lösung des Gleichungssystems mit ausgewählten Algorithmen unterstützen.

```
1  int[] lowerProfile(...)    {...}  // bestimme unteres Profil......
2  void  checkConvexity(...)  {...}  // erzeuge konvexes Profil......
3  void  supportedSystem(...) {...}  // eliminiere Starrkoerperbew...
4  void  transformGEVP(...)   {...}  // transformiere Eigenwertaufg..
5  void  transformEvec(...)   {...}  // transformiere Eigenvektor....
```

Listing 3.14: Methoden der Subklasse `EigenvalueSOE`

Die Methode `lowerProfile(...)` ermittelt das untere bzw. im Falle von Symmetrie das rechte Matrixprofil. Die Methode `checkConvexity(...)` erzeugt eine konvexe Profilstruktur. Die Methode `supportedMatrix(...)` eliminiert die Starrkörperbewegungen aus dem Gleichungssystem. Die Methode `transformGEVP(...)` transformiert die allgemeine Eigenwertaufgabe in eine spezielle Eigenwertaufgabe. Die Methode `transformEvec(...)` transformiert einen Eigenvektor der speziellen Eigenwertaufgabe in einen Eigenvektor der allgemeinen Eigenwertaufgabe und erweitert ihn wieder auf die ursprüngliche Anzahl gespeicherter Freiheitsgrade.

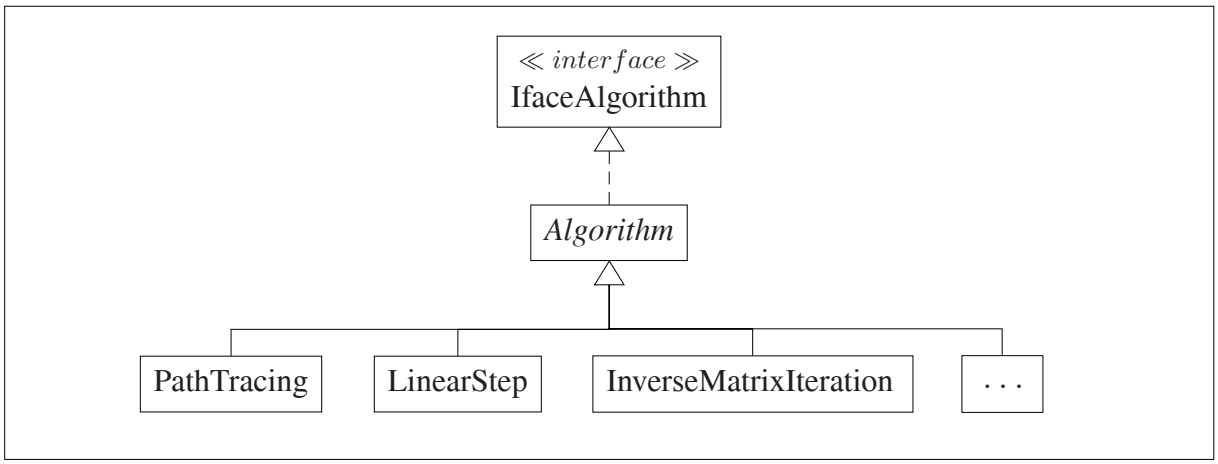


Bild 3.70: Klassenstruktur des Teilmodells *algorithm*

Modell *algorithm* : Schnittstellendefinition Im Paket *algortihm* (Bild 3.70) werden die Algorithmen zur Lösung der verschiedenen Gleichungssysteme bereitgestellt. Die Schnittstelle zu diesem Paket bildet das Interface `IfaceAlgorithm`.

```
1   interface IfaceAlgorithm {
2
3     abstract void    solve();       // Aufruf Loesungsalgorithmus...
4     abstract void    results();     // Speicherung der Ergebnisse...
5     abstract boolean symmetry();    // wahr: Nutzung von Symmetrie..
6     abstract boolean convexity();   // wahr: Nutzung von Konvexitaet
7 } //eoi
```

Listing 3.15: Interface `IfaceAlgorithm`

Die Methode `solve()` stößt den Berechnungsalgorithmus des implementierten Verfahrens an. Der Berechnungsalgorithmus operiert im allgemeinen auf das in der Klasse `SystemOfEquations` gespeicherte Gleichungssystem. Die Berechnungsergebnisse liegen in algebraischer Form, d.h. als Ergebnisvektor und/oder Ergebnismatrix vor. Die Methode `results()` ermöglicht die Speicherung der durch die Berechnung erzeugten Ergebnisvektoren bzw. -matrizen im Objektverzeichnis `algebra` der Klasse `SystemOfEquations`. Mit den Objektmethoden des Analysemodells können die Ergebnisse dann im Finite-Element-Objektmodell des Paketes *feModel* gespeichert werden.

Mit den Methoden `symmetry()` und `convexity()` werden Eigenschaften festgelegt, die für die Nutzung eines bestimmten Lösungsalgorithmus gefordert werden. Die Methode

`symmetry()` liefert den Wert *wahr*, falls der Algorithmus eine Speicherung der Systemmatrizen unter Ausnutzung der Symmetrie unterstützt, mit der Methode `convexity()` wird eine konvexe Profilstruktur für das Gleichungssystem vereinbart.

Modell *algorithm* : Schnittstellenimplementierung Die Schnittstelle des Paketes *algorithm* wird durch die Klasse `Algorithm` implementiert. Die Implementierung der Schnittstellenmethoden muß in Subklassen erfolgen, da der Inhalt bzw. der Wert der Methoden durch den implementierten Algorithmus festgelegt wird. Die Klasse `Algorithm` speichert lediglich die Referenzen auf die Schnittstellen der Pakete *analysis* und *systemOE*.

Die Spezialisierung der Klasse `Algorithm` wird exemplarisch für den vorgestellten Eigenwertlöser gezeigt. Die Klasse `InverseMatrixIteration` erweitert die Klasse `Algorithm` und implementiert die Methoden des Interface `IfaceAlgorithm`.

```
1   class InverseMatrixIteration extends Algorithm {
2
3     EigenvalueSOE      soe;
4     EigenvalueAnalysis analysis;
5     ...
6
7     public InverseMatrixIteration( IfaceProcessing analysis,
8                                    IfaceSOE soe ) {
9       super(analysis,soe);
10      this.soe      = (EigenvalueSOE)soe;
11      this.analysis = (EigenvalueAnalysis)analysis;
12      this.subspace = this.analysis.subspace();
13  } //eom
14
15  public boolean symmetry()  { return false; } //eom
16  public boolean convexity() { return true;  } //eom
17  ...
```

Listing 3.16: Subklasse `InverseMatrixIteration`

Zusätzlich zu den Datentypen der Interfaces der Pakete *analysis* und *systemOE* besitzen die Übergabeparameter des Konstruktors die Datentypen `EigenvalueAnalysis` und `EigenvalueSOE` der Spezialisierung auf Analyseformen der Dynamik. Über die Schnittstellenerweiterung des Paketes *analysis* (Listing 3.7) wird die zu bestimmende Subraumgröße dem Verfahren der Inversen Matrixiteration zugänglich gemacht. Der implementierte Algorithmus erfordert eine konvexe Profilstruktur. Die Symmetrie der Koeffizientenmatrix wird zur Speicherung nicht ausgenutzt.

```
17  public void solve() {
18
19    // (1) ueberfuehre allg. Eigenwertaufg. in spez. Eigenweraufg...
20    // (2) eliminiere Starrkoerperbewegungen.....................
21    this.soe.transformGEVP("stiffness","mass");            // ->(1).
22    this.soe.supportedSystem("stiffness","status");        // ->(2).
23    ...
24    eval        = new Eigenvalue(this);     // Eigenwertberechnung..
25    eigenvalues = eval.exec();
26
27    evec        = new Eigenvector(this);    // Eigenvektorberechnung
28    eigenmatrix = evec.exec();
29
30    // Bestimme Eigenmatrix der urspruengl. Aufgabe.................
31    this.soe.transformEvec(eigenmatrix);
32  } //eom
```

Listing 3.17: Implementierung der Schnittstellenmethode `solve()`

Die Schnittstellenmethode `solve()` stellt dem Algorithmus das aufgestellte Gleichungssystem zur Verfügung. Das vorliegende Gleichungssystem der Methode (Listing 3.17) stammt aus der Modellierung einer Quadratplatte mit konzentrierten Massen. Die allgemeine Eigenwertaufgabe wird in eine spezielle Eigenwertaufgabe überführt. Das Gleichungssystem der gelagerten Platte wird ermittelt und in der internen Datenstruktur des Eigenwertlösers gespeichert. Die Objekte der Klassen `Eigenvalue` und `Eigenvector` ermitteln die Eigenwerte und Eigenvektoren mit dem Verfahren der Inversen Matrixiteration. Die Eigenvektoren der ursprünglichen Aufgabe werden berechnet.

Beispielanwendung

Zur Demonstration der Flexibilität und Extensibilität des implementierten Konzepts sind im folgenden die Schritte gezeigt, die zum Aufbau des Kern- und Analysemodells erforderlich sind.

Als erstes Aufgabenbeispiel dient eine Eigenwertanalyse einer Quadratplatte. Die Abmessungen der Platte und die Feinheit der Maschendichte werden beim Erzeugen des Finite-Element-Objektmodells dem Konstruktor übergeben. Die Analyse (`DynamicAnalysis`) umfaßt eine Eigenwertanalyse für das diskretisierte Finite-Element-Modell (`SquarePlate`). Mit dem Objekt `soe` vom Typ `EigenvalueSysOE` wird ein Gleichungssystem für eine allgemeine Eigenwertaufgabe bereitgestellt. Als Algorithmus zur Berechnung des Gleichungssystems wird das Verfahren der Inversen Matrixiteration gewählt. Mit den Methoden `perform(...)` und `setResults()` des Interface `IfaceAnalysis` werden die Berechnungsschritte der Analyse angestoßen. Die Ergebnisse werden wahlweise im Objektmodell gespeichert oder stehen zur weiteren Verwendung innerhalb des Paketes `feAnalysis` zur Verfügung.

```
1  int subspace = 20;        // Anzahl zu bestimmender Eigenzustaende
2  // Quadratplatte 12.m/12.m, 24/24 Maschen.....................
3  IfaceModel        model     = new SquarePlate(12.,12.,24,24);
4  IfaceEVProcessing analysis  = new DynamicAnalysis(subspace,model);
5  IfaceEVSysOE      soe       = new EigenvalueSysOE(analysis);
6  IfaceAlgorithm    algorithm = new InverseMatrixIteration(analysis,
7                                                           soe);
8  analysis.perform(algorithm,soe);
9  analysis.setResults();
```

Listing 3.18: Eigenwertanalyse einer Quadratplatte

Das zweite Beispiel zeigt eine geometrisch-nichtlineare Berechnung eines Zweibocks unter Einzellast zur Simulation des Durchschlagverhaltens. Das nichtlineare Analysemodell (`NonLinearAnalysis`) stellt die Methoden für inkrementelle Berechnungsschritte bereit. Als Lösungsalgorithmus (`PathTracing`) wird ein Pfadverfolgungsalgorithmus nach *Crivelli* eingesetzt.

```
1  IfaceModel        model     = new ToggleSystem();
2  IfaceNLProcessing analysis  = new NonLinearAnalysis(model,0.2);
3  IfaceSysOE        soe       = new SystemOfEquations(analysis);
4  IfaceAlgorithm    algorithm = new PathTracing(analysis,soe);
5
6  analysis.perform(algorithm,soe);
7  analysis.setResults();
```

Listing 3.19: Geometrisch nichtlineare Analyse eines Zweibocks

Soll anstelle einer nichtlinearen Berechnung eine lineare Berechnung erfolgen, werden die Zeilen 2 und 4 durch nachfolgende Zeilen ersetzt.

```
1   IfaceProcessing analysis  = new StaticAnalysis(model);
2   IfaceAlgorithm  algorithm = new LinearStep(analysis,soe);
```

Listing 3.20: Lineare Analyse eines Zweibocks

3.5.3 Implementierung der Inversen Matrixiteration

Die Implementierung der Inversen Matrixiteration erfolgt in einem eigenen Paket *invMIteration*. Das Paket beinhaltet die in Bild 3.71 dargestellten Klassen. Das Paket ist Bestandteil des Algorithmenpakets *algorithm*.

Paket *invMIteration*

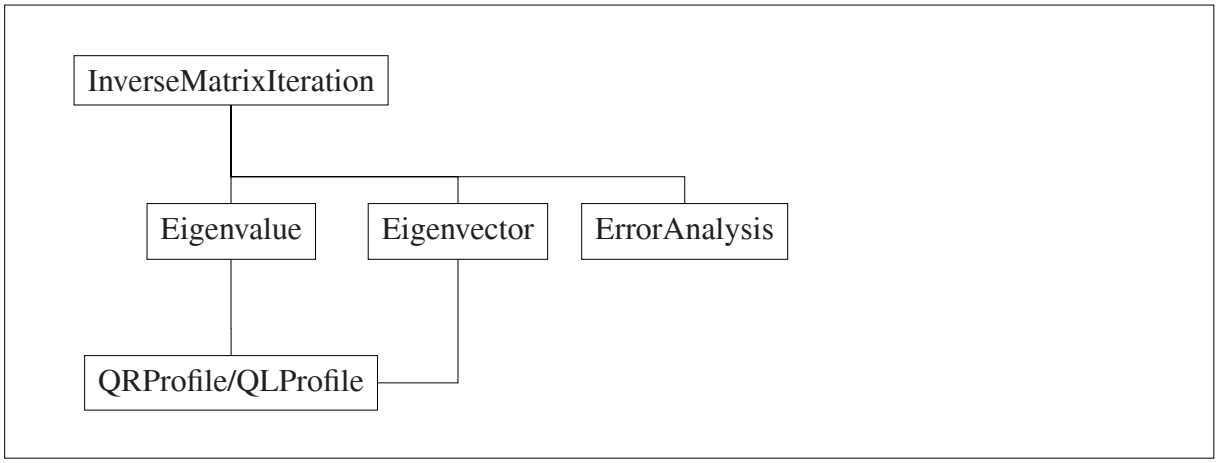


Bild 3.71: Klassenstruktur im Paket *invMIteration*

Die Algorithmen zur Bestimmung einer beliebigen Anzahl von Eigenwerten und Eigenvektoren, sowie zur Überprüfung der Qualität der berechneten Näherungslösungen wird in den Klassen `Eigenvalue`, `Eigenvector` und `ErrorAnalysis` modelliert. Der Algorithmus für die QR-Zerlegung und RQ-Rekombination der Profilmatrix wird sowohl von der Klasse `Eigenvalue`, als auch von der Klasse `Eigenvector` genutzt und wird deshalb in einer eigenen Klasse `QRProfile` implementiert.

QR-Verfahren vs. QL-Verfahren Eine Faktorisierung der Koeffizientenmatrix $\mathbf{A}$ in die Matrizen $\mathbf{Q}_R$ und $\mathbf{R}$ erfolgt durch eine spaltenweise Eliminierung der Subdiagonalelemente von $\mathbf{A}$, links beginnend mit dem Koeffizienten a_{21} (siehe 3.4.1). Wird die Berechnungsreihenfolge umgekehrt, so bedeutet dies eine spaltenweise Eliminierung der Superdiagonalelemente von $\mathbf{A}$, beginnend von rechts mit dem Koeffizienten $a_{(n-1,n)}$. Das resultierende Zerlegungsprodukt ergibt eine Linksdreiecksmatrix $\mathbf{L}$ und eine Orthonormalmatrix $\mathbf{Q}_L$. Die Indizes R und L der Orthonormalmatrix $\mathbf{Q}$ kennzeichnen ihren Berechnungsursprung. In [30] ist gezeigt, daß die beiden Zerlegungen mathematisch äquivalent sind und für die theoretische Behandlung des Verfahrens nicht unterschieden werden müssen.

Im Gegensatz zum QR-Algorithmus konvergieren beim QL-Algorithmus die betragskleinsten Eigenwerte von der ersten Zeile beginnend schrittweise und sortiert bis zur letzten Zeile und Spalte

der Matrix. Für gestufte Matrizen kann dies von Vorteil sein, falls sich betragskleine Koeffizienten am Anfang und betragsgroße Koeffizienten am Ende der Matrix häufen. Diese Eigenschaft der iterierten Matrix ist für die vorliegende Aufgabenstellung nicht gegeben.

Bei der Implementierung der beiden Verfahren hat sich jedoch gezeigt, daß der QL-Algorithmus infolge der Berechnungsreihenfolge der Koeffizienten insgesamt eleganter codierbar ist. Insbesondere die Festlegung der Laufbereiche von Schleifen ist in *JAVA*, mit Hilfe des Attributs `length` zur Abfrage der Feldgröße, ohne Kenntnis des rechten Matrixprofils möglich.

Trotz einiger Vorteile hinsichtlich der Codierung des QL-Algorithmus wird im folgenden zur Förderung der Übersichtlichkeit der in Abstimmung mit der Theorie der vorangegangenen Kapitel entwickelte QR-Algorithmus vorgestellt.

Speicherstruktur

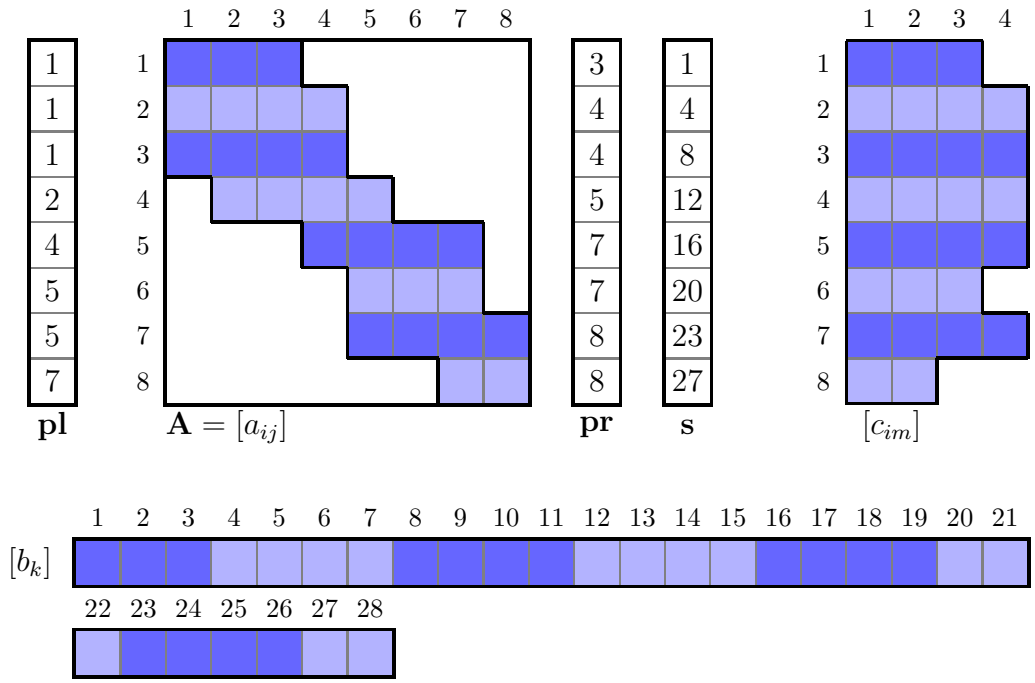


Bild 3.72: Speicherstruktur von $\mathbf{A}$

Bild 3.72 zeigt die Speicherstruktur für die Koeffizientenmatrix $\mathbf{A}$ der speziellen Eigenwertaufgabe. Der multiplikative Hauptaufwand des vorgestellten Verfahrens ($\sim 75\%$) bildet die Faktorisierung von $\mathbf{A}$ (Gleichung (3.3.5)). Dieser Berechnungsschritt modifiziert zeilenweise die Matrix. Die Matrix wird deshalb zeilenweise abgespeichert. Im Rahmen dieser Arbeit wurden zwei verschiedene Speichermodelle untersucht und implementiert.

Das erste Speichermodell speichert zeilenweise die Koeffizienten zwischen dem linken und rechten Matrixprofil in einem zweidimensionalen Feld $c[\,][\,]$ vom Datentyp `double`. Der Zugriff auf die Koeffizienten erfolgt über den Referenzvektor $c[\,]$ mit den Zeilenreferenzen, dem Spaltenindex j und dem Profilvektor $\mathbf{pl}$. Damit wird jeder Koeffizient `a[i][j] = c[i][j-pl[i]]` dreifach indiziert.

Das zweite Speichermodell speichert zeilenweise die Koeffizienten zwischen dem linken und rechten Matrixprofil in einem eindimensionalen Feld $b[\,]$ vom Datentyp `double`. Der Index des ersten Koeffizienten einer Zeile in der Datenstruktur $b[\,]$ wird analog zum Profilvektor $\mathbf{pl}$, in einem Vektor $\mathbf{s}$ vom Datentyp `int` vermerkt. Der Zugriff auf die Koeffizienten erfolgt über eine Indexrechnung mit dem Vektoren $\mathbf{pl}$ und $\mathbf{s}$, und dem Spaltenindex j. Jeder Koeffizient `a[i][j] = b[s[i]+j-pl[i]]` wird damit ebenfalls dreifach indiziert. Die Differenz `s[i]-pl[i]` wird vorab berechnet und im Vektor `s[ ]` gespeichert. Der Koeffizientenzugriff vereinfacht sich damit zu `a[i][j] = b[s[i]+j]`. Die Anzahl der Indizierungen reduziert sich auf zwei. Bei Schleifenbildung über die Zeilen der Matrix ist ein Koeffizientenzugriff mit einer Indizierung möglich.

Eine wesentliche Design-Philosophie von *JAVA* ergibt sich aus dem Anspruch nach einem sicheren und robusten Sprachkonzept [47]. Realisiert wird dieser Sicherheitsgedanke durch das *JAVA* eigene Exceptionhandling, das eine Überprüfung des Bytecodes zur Laufzeit durch den *JAVA*-Interpreter vornimmt. Ein Aspekt dieser Überprüfung beinhaltet die Überwachung jedes indizierten Zugriffs auf Felder. Diese Überwachung der Feldgrenzen ist teuer und kann das Laufzeitverhalten einer Applikation dramatisch verschlechtern. Die Reduktion indizierter Feldzugriffe wirkt sich damit positiv auf das Laufzeitverhalten der Implementierung aus. Die im folgenden vorgestellten Algorithmen operieren auf die Datenstruktur $[b_k]$.

Algorithmus zur Bestimmung der Eigenwerte

Die im folgenden dargestellten Algorithmen werden anhand von Implementierungsauszügen erläutert. Sie spiegeln die wesentlichen Berechnungsschritte in teilweise vereinfachter Form wieder. Auf die Definition lokaler Variablen wird verzichtet, soweit die Definition für das Verständnis nicht erforderlich ist. Auf Hilfsvariablen, die zur Indexrechnung und damit zur Effizienzsteigerung des Algorithmus üblicherweise eingesetzt werden, wird zu Gunsten der Übersichtlichkeit gänzlich verzichtet.

Der Algorithmus zur Bestimmung der Eigenwerte ist in der Klasse `Eigenvalue` implementiert. Bis auf die iterierte Matrix `a[]` und das Attribut `dim` zur Speicherung der aktuellen Dimension nach Deflation im Zyklus s, werden nur Objektattribute vereinbart, deren Wert sich im Laufe der Iteration nicht ändert.

Zur Initialisierung der vereinbarten Objektattribute wird im Konstruktor der Klasse ein Objekt vom Typ `InverseMatrixIteration` übergeben. Die Klasse `InverseMatrixIteration` verwaltet die Referenzen auf die interne Datenstruktur des Gleichungssystems und wird für die verschiedenen Algorithmen des Paketes *invMIteration* als *Repository* für das Gleichungssystem verwendet. Die Klasse stellt damit auch Instanzen der unreduzierten Matrix, beispielsweise für die Eigenvektorberechnung zu Verfügung.

```
1   public class Eigenvalue {
2
3     final static double ZERO=2.22e-16;  // EPSILON-KONSTANTE........
4     double[]  eigenvalue;               // Eigenwertnaeherungen.....
5     double[]  a;                        // Koeffizientenmatrix A....
6     int[]     pl;                       // Profilvektor pl..........
7     int[]     s;                        // Indexvektor (s-pl).......
8     int       dim;                      // Systemgroesse im Zyklus s
9     int       sub;                      // zu best. Subraumdimension
10    QRProfile qr;                       // QR-Zerlegung.............
11    InverseMatrixIteration iteration    // Repository...............
12    ...
13
14  public Eigenvalue( InverseMatrixIteration iteration ) {
15    this.iteration = iteration;
16    this.sub       = iteration.subspace();
17    this.pl        = iteration.upperProfile();
18    this.s         = iteration.sMinusP();
19    this.a         = iteration.matrix();
20    this.dim       = pl.length;
21    this.qr        = new QRProfile(a,pl,s);
22  } //eom
23  ...
```

Listing 3.21: Klasse `Eigenvalue`

Die Objektmethode `iterate(...)` steuert den schrittweisen Berechnungsablauf. Der Methode wird ein Anfangsshift für die Spektralverschiebung übergeben. Ist der Übergabeparameter ungleich 0.0, so werden mehrere QR-Schritte mit konstantem Shift der Iterationsschleife vorgeschalten, um den gewünschten Wertebereich der verschobenen Matrix zu festigen. Innerhalb der Iterationsschleife wird zunächst der Konvergenzfortschritt überprüft. Dabei wird berücksichtigt, daß das Verfahren häufig mehrere Eigenwerte im selben Zyklus bestimmt. Mit jedem gefundenen Eigenwert, wird in einem Deflationsschritt die Dimension der Matrix um eins verringert und das Verschiebungsinkrement `ann` neu bestimmt.

```
23  void iterate( double shift ) {
24    int p    = 0;                      // Eigenwertzaehler..........
25    int n    = dim-1;                  // letzter Feldindex.........
26    int last = dim-sub;                // letzter Eigenwertindex.....
27    ...
28    for( cycle=0; cycle<maxCycle; cycle++ ) {
29      ann = 0.;
30      for( i=n; i>=last; i-- ) {
31        if( convergence() ) {              // Konvergenz in Zeile n.
32          eigenvalue[p++] = a[s[n]+n]+shift;
33          if( p==sub ) return;             // alle Eigenwerte best..
34          dim--; n--;                      // Deflation.............
35          ann += a[s[n]+n];                // shift update.........
36        }
37        else {
38          shift += ann;                    // Spektralverschiebung..
39          for( m=0; m<dim; m++ ) a[s[m]+m] -= ann;
40          break;
41        }
42      }
43      jacobi();                          // Jacobi-Randkorrektur.......
44      qr.decompose(dim);                 // Zerlegung     A(s)   = Q R
45      qr.recombine(dim);                 // Rekombination A(s+1) = R Q
46      preCond();                         // Prekonditionierung.........
47      ...
48      shift += ann;                      // Spektralverschiebung.......
49      for( m=0; m<dim; m++ ) a[s[m]+m] -= ann;
50    }
51  } //eom
52  ...
```

Listing 3.22: Methode `iterate(...)`

Die Methode `convergence()` überprüft die Konvergenz der Nichtdiagonalkoeffizienten der letzten Zeile und Spalte. Die Konvergenzschranke δ wird nach Kapitel 3.3.3 bestimmt.

$$\delta \;=\; \varepsilon \cdot \|\mathbf{A}\|_F \cdot cycle/N \tag{3.5.1}$$

Für praxisnahe Berechnungen ist für eine Lösung der Ingenieuraufgabe oft eine geringere Genauigkeit der gesuchten Eigenwertnäherungen ausreichend. Eine Steuerung der Genauigkeit kann über das vorgeschlagene Konvergenzkriterium erfolgen. Die Genauigkeit der Eigenwertnäherungen ist proportional zur Schranke δ und liefert bis zu einer Wahl von $\varepsilon \cong 1.e-6$ zuverlässige Berechnungsergebnisse. In den vorliegenden Berechnungen wird für ε die Maschinengenauigkeit ϵ gewählt.

Die Änderung der Diagonalkoeffizienten in aufeinanderfolgenden Schritten wird als Konvergenzkriterium nicht betrachtet, da es sich bereits bei geringer Clusterbildung als nicht zuverlässig erweist. Eine Überprüfung der Nichtdiagonalelemente ist ohnehin erforderlich, damit Konvergenzverzögerungen zuverlässig erkannt werden.

Der Zeitpunkt für den Einsatz der Jacobi-Randkorrektur und der Prekonditionierung entscheidet maßgeblich über den Erfolg der Verfahren. Der numerische Aufwand für die Jacobi-Randkorrektur jedoch ist vernachlässigbar im Vergleich zu einer vollständigen QR-Zerlegung. Die Jacobi-Randkorrektur kann deshalb bedenkenlos mehrfach in einem Iterationszyklus eingesetzt werden.

```
52  void jacobi() {
53    int i = dim-1;
54    ...
55    for( m=dim-2; m>=pl[i]; m-- ) {
56      ...                                  // bestimme Problemkoeff....
57      if( pl[m]!=pl[i] ) {                 // ueberpruefe Profil und ..
58        save = 0.;                         // Diagonalisierungsgrad....
59        for( int k=pl[m],j=1; k<pl[i]; k++,j++ )
60          save += Math.abs(a[s[m]+k]);
61        save /= j;
62        if( save>epsilon ) continue;       // entscheide ueber Jacobi -
63      }                                    // Randkorrektur............
64      ...                                  // bestimme Parameter der ..
65      cos = 1./Math.sqrt(1.+tan*tan);      // Rotation.................
66      sin = tan*cos;
67      ...                                  // transformiere P(t)*A*P...
68    }
69  } //eom
70  ...
```

Listing 3.23: Objektmethode `jacobi()`

Für die Prekonditionierung hat sich gezeigt, daß ein Einsatz direkt nach einem QR-Schritt am effektivsten ist. Sie behandelt effektiv Problemkoeffizienten um die Hauptdiagonale und diagonalisiert durch die Iteration entkoppelte Submatrizen. Die Blockdimension p jedes Diagonalblocks wird durch den Profilvektor $\mathbf{pl}$ bestimmt. Die Diagonalisierung der Blockmatrix $\mathbf{A}^{p\times p}$ erfolgt in einem separaten Schritt mit dem QR-Verfahren für vollbesetzte Matrizen. Die Matrix $\mathbf{A}^{p\times p}$ wird

dafür mit der *native*-Methode `arraycopy` der Klasse `System` in ein Feld `m[][]` kopiert. Die Methode `arraycopy` kopiert zusammenhängende Feldbereiche beliebiger Dimension. Sie liegt in Maschinencode vor und ist damit schneller als interpretierter Bytecode.

```
70   void preCond() {
71
72     int size = 0;                          // Blockdimension p.........
73     ...
74     for( int n=dim-2; n>=0; n-- ) {
75       ...                                  // bestimme Blockdimension p
76       double[][] m = new double[size][size];
77       for( i=0; i<size; i++ )              // bestimme Blockmatrix A_k.
78         System.arraycopy(a,s[i+n],m[i],0,size);
79       double[][] v = QR.exec(m);           // bestimme Matrix V(pxp)...
80       ...                                  // V(t)*A*V.................
81     }
82   } //eom
83   ...
84 } //eoc
```

Listing 3.24: Objektmethode `preCond()`

Der eigentliche QR-Schritt erfolgt durch die Objektmethoden `decompose(...)` und `recombine(...)` der Klasse `QRProfile`. Den Methoden wird die aktuelle Systemdimension übergeben.

Schließlich wird in der Methode `iterate(...)` nach jedem Berechnungszyklus der Shift dem aktuellen Berechnungsfortschritt angepasst. Um die Sortierung der Eigenwerte nicht vollständig zu verlieren wird die Spektralverschiebung am Ende eines Iterationszyklus mit dem betragskleinsten Diagonalkoeffizienten durchgeführt. Zur Verringerung des Aufwands zur Bestimmung des betragskleinsten Diagonalkoeffizienten wird die Suche auf die letzten 20 Diagonalkoeffizienten begrenzt.

Die Klasse `QRProfile` stellt mit den Objektmethoden `decompose(...)` und `recombine(...)` eine Implementierung des profilerhaltenden QR-Verfahrens zur Verfügung. Im Konstruktor der Klasse wird vorab die Differenz $(\mathbf{s} - \mathbf{u})$ bestimmt, die den Zugriff auf die Koeffizienten der internen Datenstruktur ermöglicht. Für die Zerlegung wird ein temporärer Vektor `h[]` zur Speicherung der Pivotzeile bereitgestellt.

```
1   public class QRProfile {
2     double[] h;                          // Hilfsvektor der Zerlegung
3     double[] a;                          // Matrix A.................
4     int[]    pl;                         // linkes Matrixprofil......
5     int[]    s;                          // Index Zeilenanfang.......
6     int      dim;                        // Systemdimension..........
7     ...
8     public QRProfile( double a[], int[] pl, int[] s ) {
9       ...                                // speichere Attribute......
10      for( int i=0; i<dim; i++ )         // berechne s-pl vorab......
11        s[i] -= pl[i];
12    } //eom
13    ...
```

Listing 3.25: Klasse `QRProfile`

Die Objektmethode `decompose(...)` läuft über alle Subdiagonalkoeffizienten $a_{mn}, (m > n)$ innerhalb des Profils und eliminiert diese schrittweise durch elementare Rotationsmatrizen $(\mathbf{R}_{mn}^T \mathbf{A}_s)$. Für die Reduktion der Koeffizienten in Spalte `n` wird auf den temporär mit Zeile `n` belegten Vektor `h[]` operiert. Eine aufwendige Erweiterung der Profil- und Speicherstruktur der Pivotzeile `n` wird dadurch vermieden.

```
13  void decompose( int dim ) {
14    ...                                        // bestimme Feldgrenzen
15    for( n=0; n<(dim-1); n++ ) {               // n -> Spaltenindex...
16      h = new double[s.length];                // belege temporaerer .
17      System.arraycopy(a,s[n],h,pl[n],len);    // Speicher h.........
18      for( m=n+1; m<max; m++ ) {               // m -> Zeilenindex....
19        ...                                    // bestimme sin/cos....
20        leftRotation();                        // Linksrotation R(t)*A
21        ...                                    // speichere sin/cos...
22      }
23    }
24  } //eom
25  ...
```

Listing 3.26: Objektmethode `decompose(...)`

Die Methode `rotationParameter()` bestimmt die Parameter $sin(\theta)$ und $cos(\theta)$ der Rotationsmatrix $\mathbf{R}$ ohne explizite Berechnung des Rotationswinkels θ. Aus der Forderung $a_{mn} = 0.$ folgt mit Gleichung (3.3.12) der Tangens des Rotationswinkels :

$$tan(\theta) \;=\; -\frac{a_{mn}}{a_{nn}} \tag{3.5.2}$$

Mit Gleichung (3.5.2) wird folgende numerisch stabile Bestimmung der Parameter $sin(\theta)$ und $cos(\theta)$ vorgenommen :

```
1   ...
2   if( Math.abs(amn)<ZERO ) {                  // amn = 0. => keine ..
3     a[s[m]+n] = 0.;  continue;                // Transformation......
4   }
5   else if( Math.abs(amn)>Math.abs(ann) ) {    // Rotation entgegen ..
6     tan = -ann/amn;                           // dem Uhrzeigersinn...
7     sin = 1./Math.sqrt(1.+(tan*tan));
8     cos = sin*tan;
9   }
10  else {                                      // Rotation mit dem ...
11    tan = -amn/ann;                           // Uhrzeigersinn.......
12    cos = 1./Math.sqrt(1.+(tan*tan));
13    sin = cos*tan;
14  }
15  ...
```

Listing 3.27: Bestimmung der Rotationsparameter

Zur Speicherung der Rotationsparameter steht die Speicherstelle des eliminierten Koeffizienten a_{mn} zur Verfügung. Wegen seiner Mehrdeutigkeit ist der Tangens als Rotationsinformation zur Speicherung nicht geeignet. Stattdessen wird folgendes Schema nach [13] verwendet :

```
1   ...
2   if( Math.abs(cos)==0. )                     // tausche Zeilen n&m..
3     a[s[m]+n] = 1.;
4   else if( Math.abs(sin)<Math.abs(cos) ) {    // waehle min(sin,cos)
5     sgn = (cos<0.)?-1:1;                      // zur Speicherung, da
6     a[s[m]+n] = sgn*sin/2.;                   // numerisch stabiler .
7   }                                           // bei Rekonstruktion .
8   else {                                      // von cos und sin.....
9     sgn = (sin<0.)?-1:1;
10    a[s[m]+n] = 2.*sgn/cos;
11  }
12  ...
```

Listing 3.28: Speicherung der Rotationsparameter

Die Objektmethode `recombine(...)` bestimmt schrittweise die iterierte Matrix $\mathbf{A}_{s+1}$. Analog zur Bestimmungsreihenfolge der Methode `decompose(...)` werden die Rotationen auf die Rechtsdreiecksmatrix $\mathbf{R}_s$ ausgeführt. Mit dem gewählten Speicherschema für die faktorisierte Orthonormalmatrix $\mathbf{Q}$ lassen sich die Rotationsmatrizen $\mathbf{R}_{mn}$ rekonstruieren :

```
1   void recombine( int dim ) {
2     ...
3     for( n=0; n<(dim-1); n++ ) {
4       ...                                    // bestimme Feldgrenzen
5       for( m=n+1; m<max; m++ ) {
6         ...
7         rho = a[s[m]+n];                     // hole sin/cos........
8         a[s[m]+n] = 0.;
9         if( rho==1. ) {                      // tausche Spalten n&m.
10          cos = 0.;
11          sin = 1.;
12        }
13        else if( Math.abs(rho)<1. ) {
14          sin = 2.*rho;
15          cos = Math.sqrt(1.-sin*sin);
16        }
17        else {
18          cos = 2./rho;
19          sin = Math.sqrt(1.-(cos*cos));
20        }
21        rightRotation();                     // Rechtsrotation A*R..
22      }
23      ...
24    }
25  } //eom
```

Listing 3.29: Objektmethode `recombine(...)`

Algorithmus zur Bestimmung der Eigenvektoren

Die Bestimmung der Eigenvektoren erfolgt in der Klasse `Eigenvector`. Die Referenzen auf die Speicherstruktur der Systemgleichungen werden analog zur Klasse `Eigenvalue` durch ein Objekt vom Typ `InverseMatrixIteration` initialisiert. Mit den zuvor bestimmten Eigenwertnäherungen werden die zugehörigen Eigenvektornäherungen beliebig bestimmt. Der Algorithmus zur Berechnung von m Eigenvektoren ist in der Objektmethode `double[][] eigenstate()` implementiert.

```
1  double[][] eigenstate() {
2     int         sub  = last-first+1;     // Subraumgroesse.....
3     int         multi;                   // algebr. Multiplizitaet...
4     double[][] x = new double[sub][];    // Eigenmatrix[sub][dim]....
5     double[][] evec = null;              // Eigenvektor[multipl][dim]
6     int[]       evi;                     // Zeilenindex d. Konvergenz
7     int         es = first;              // Eigenwertindex.....
8     ...
9     do {
10      multi = multiplicity(eigenvalue,es);   // best. Multiplizitaet
11      shift = eigenvalue[es];                // bestimme Shift......
12      shiftA(a,shift);                       // (A-pI).............
13      qr  = new QRProfile(a,pl,s);           // (A-pI) = QR.........
14      qr.decompose(dim);
15      evi = multiplicity(a,multi);           // pruefe Multipl......
16
17      if( evi==null ) {                      // Verfeinerungsschritt
18        qr.recombine(dim);
19        evec = refineLI(a,es,shift);         // lokale Iteration....
20        if( evec==null )
21          evec = refineQR(a,es,shift);       // Zusatzzerlegung.....
22      }
23      else
24        evec = initializeEvec(evi);          // Initialisiere EV....
25
26      eigenvector(a,evec);                   // bestimme EV.........
27      ...                                    // speichere EV........
28      es++;                                  // naechster EV........
29      ...                                    // pruefe auf Abbruch..
30    } while(es<last);
31    return x;
32  } //eom
33  ...
```

Listing 3.30: Objektmethode `eigenstate()`

Für eine korrekte Bestimmung der Eigenvektoren ist die Kenntnis über die Multiplizität des dazugehörigen Eigenwerts erforderlich. Die Multiplizität q des Eigenwerts λ_k wird zunächst mit Hilfe

der berechneten Eigenwertnäherungen bestimmt. Nach der Zerlegung $((\mathbf{A} - \lambda_m \mathbf{I}) = \mathbf{QR})$ der spektralverschobenen Matrix, wird die Multiplizität an der Matrix $\mathbf{R}$ nochmals verifiziert. Dabei wird berücksichtigt, daß infolge numerischer Fehler in der Eigenwertnäherung und der Zerlegung die q gesuchten Nullzeilen nicht zwangsläufig den q letzten Zeilen von $\mathbf{R}$ entsprechen. Die Systemindizes der Nullzeilen werden in einem Indexfeld `evi` vermerkt und zur Initialisierung der Identitätsmatrix $\mathbf{I}_q^{N\times q}$ verwendet. Eine Zeile aus $\mathbf{R}$ wird als Nullzeile akzeptiert, falls ihre Euklidische Norm die Fehlerschranke δ_{QR} erfüllt.

$$\delta_{QR} \;=\; (\|\mathbf{A}\|_E \, \varepsilon/N)^{\frac{2}{3}} \tag{3.5.3}$$

Ergibt die Überprüfung der Multiplizität weniger als q Nullzeilen, so ist eine Zerlegung für die zuverlässige Eigenvektorbestimmung zum Eigenwert λ_k nicht ausreichend. In einem Verfeinerungsschritt werden die Eigenwertnäherung und die Eigenvektornäherungen durch zusätzliche Maßnahmen verbessert bzw. bestimmt.

Zunächst wird überprüft, ob eine numerisch günstige, lokale Iteration mit akzeptablem Fehler erfolgen kann. Die Überprüfung erfolgt mit den Koeffizienten der schlecht konvergenten Zeilen der Matrix $\mathbf{R}$. Ist eine lokale Iteration möglich, so wird ein Diagonalblock moderater Größe mit dem Jacobi-Verfahren diagonalisiert. Die Eigenvektornäherungen zu den betragskleinsten Eigenwerten des Diagonalblocks werden zur Initialisierung der Matrix $\mathbf{I}_q^{N\times q}$ eingesetzt.

Ist eine lokale Iteration nicht möglich, wird in einem zweiten Schritt mit einer zusätzlichen Zerlegung die erforderliche Konvergenz in der Matrix $\mathbf{R}$ herbeigeführt.

Mit den bestimmten Rotationsparameter der Zerlegung wird in der Methode `eigenvector(...)` das Produkt (3.4.35) berechnet.

Für die berechneten Näherungslösungen wird der relative Fehler der Restnorm bestimmt. Erfüllen die berechneten Eigenzustände nicht die gewünschten Genauigkeitsanforderungen, wird durch eine Nachiteration das Ergebnis im Eigenwert und Eigenvektor verbessert.

Kapitel 4

Vergleichsverfahren

4.1 Einleitung

Das Ziel des folgenden Kapitels ist, eine Einschätzung über die Wirtschaftlichkeit und Gebrauchstauglichkeit des in Kapitel 3 entwickelten Verfahrens für große Profilmatrizen zu erzielen. Die Untersuchungsergebnisse verschiedener Beispiele werden den Ergebnissen etablierter Methoden zur Lösung der speziellen Eigenwertaufgabe gegenübergestellt und bewertet. Als Vergleichsverfahren werden das Givens-QRI- und IRLES-Lanczos-Verfahren vorgestellt und implementiert.

Als Bewertungsgrundlage für die Wirtschaftlichkeit des Verfahrens dient ein operativer Aufwandsvergleich unter der Voraussetzung vergleichbarer Berechnungsergebnisse. Eine Leistungsbeurteilung auf der Grundlage von Zeitmessungen wird wegen der sich ständig verändernden Randbedingungen im Hard- und Softwarebereich nur qualitativ vorgenommen. Zusätzlich zum operativen Vergleich wird der erforderliche Speicherbedarf der Verfahren betrachtet.

Bei der Beurteilung der Gebrauchstauglichkeit kommen folgende Kriterien zum Einsatz :

1. Die Güte der berechneten Näherungslösungen wird anhand von Fehlerschranken beurteilt.

   (a) Die Restnorm $\|\mathbf{A}\hat{\mathbf{x}}_i - \hat{\lambda}_i\hat{\mathbf{x}}_i\|$ ermöglicht eine Gesamteinschätzung der berechneten Näherungslösung $(\hat{\lambda}_i, \hat{\mathbf{x}}_i)$. Sie verifiziert vorrangig die Richtigkeit der Lösung anhand des Gleichgewichtsfehlers. Ist das Eigenwertverfahren stabil, so ist $(\hat{\lambda}_i, \hat{\mathbf{x}}_i)$ die Lösung der Aufgabe $(\mathbf{A} + \mathbf{E})\mathbf{x}_i = \lambda_i\mathbf{x}_i$ mit $\|\mathbf{E}\|/\|\mathbf{A}\| \leq \gamma\epsilon$. Die Restnorm $\|\mathbf{A}\hat{\mathbf{x}}_i - \hat{\lambda}_i\hat{\mathbf{x}}_i\|$ befriedigt damit $O(\gamma\epsilon\|\mathbf{A}\|)$ mit (γ : kleine Konstante).

   (b) Der Fehler in der Differenz $(\rho(\hat{\mathbf{x}}_i) - \hat{\lambda}_i)$ gibt eine Einschätzung für den Fehler im Rayleigh-Quotienten und damit auch für die Güte der Eigenvektornäherung $\hat{\mathbf{x}}$. Die Fehlerordnung der Eigenwertnäherungen $\hat{\lambda}_i$ beträgt $O(\epsilon\|\mathbf{A}\|)$ und entspricht näherungsweise dem Fehler für die Berechnung des Rayleigh-Quotienten. Ein Fehler der Ordnung $O(\epsilon\|\mathbf{A}\|)$ in der Differenz $(\rho(\hat{\mathbf{x}}_i) - \hat{\lambda}_i)$ wird deshalb als ausreichend klein betrachtet.

(c) Die Fehlerschranken nach Temple-Kato für Eigenwerte und Eigenvektoren sind strenge Fehlerschranken. Sie ermöglichen eine Trennung der Eigenwerte eines engen Clusters und geben Aufschluß über die Richtungsabweichung zwischen genäherter und exakter Eigenvektorlösung.

2. Die Zuverlässigkeit der Lösung wird durch die Berechnung von Eigenzuständen in verschiedenen Bereichen des Eigenwertspektrums $\sigma(\mathbf{A})$ untersucht. Die ermittelte Lösung wird über Sturmsche Ketten [3] auf ihre Vollständigkeit überprüft. Das Konvergenzverhalten bei der Ermittlung der Eigenzustände mit betragskleinsten Eigenwerten, mit betragsgrößten Eigenwerten und mit Eigenwerten in einem vorgegebenen Intervall wird untersucht.

Testmatrizen Die Testmatrizen der nachfolgenden Untersuchungen stammen aus einer Finite-Elemente-Formulierung der Schwingungsaufgabe ebener, einfach gelagerter Rechteckplatten. Die Plattenmodelle sind in ihrer Geometrie und ihrer Diskretisierung einfach skalierbar und ermöglichen damit die Konstruktion von Matrizen mit den gewünschten Eigenschaften, hinsichtlich der Anzahl Freiheitsgrade N, der mittleren Bandbreite b und der Anzahl mehrfacher Eigenwerte s.

Für eine bessere Vergleichbarkeit der Berechnungsergebnisse wird die Matrixkennzahl w (4.1.1) definiert. Die Matrixkennzahl erfaßt die Speicherintensität der Matrix und ermöglicht damit auch Rückschlüsse auf den numerischen Aufwand der Berechnung.

$$w := \left(\frac{b^2}{N}\right)^{\frac{1}{d^2}} \qquad \text{mit } d = \text{geometrische Dimension der Aufgabe} \qquad (4.1.1)$$

Der Wert von w liegt für typische Aufgabenstellungen aus dem Bauwesen zwischen 1.0 und 1.5. Der Aufwand der Analyse einer Schwingungsaufgabe steigt mit zunehmendem Wert von w.

Tabelle 4.1 zeigt die Eigenschaften der Testmatrizen. Die Art der Elementierung ist in Bild 3.52 in Kapitel 3 gezeigt. Spalte 1 in Tabelle 4.1 enthält den Identifikator der zu berechnenden Matrix, Spalte 2 zeigt die Maschenanzahl der Elementierung. Spalte 3 enthält die Matrixdimension, Spalte 4 die mittlere Bandbreite und Spalte 5 die Matrixkennzahl w. Die Spalten 6 und 7 geben Aufschluß über die Intervallgröße des Eigenwertspektrums. Spalte 8 enthält die Frobeniusnorm $\|\mathbf{A}\|_F$ der Matrix.

Alle Untersuchungen wurden mit einer Vielzahl Matrizen unterschiedlicher Eigenschaften durchgeführt. Die Testmatrizen aus Tabelle 4.1 wurden als repräsentativ für diese Untersuchungen gewählt.

Hard- und Softwareumgebung Die Berechnungen aller Untersuchungen erfolgten mit einem *Intel Pentium 4 / CPU 2.40 GHz* Prozessor. Der Arbeitsspeicher beträgt $512MB$, der Level 2 Cache $512KB$.

1	2	3	4	5	6	7	8
Matrix	$m \times n$	N	b	w	$min\lambda$	$max\lambda$	$\|\mathbf{A}\|_F$
K4727	40×40	4727	119	1.32	1.6186	$4.9480e+06$	$7.27e+14$
K5043	40×40	5043	123	1.32	0.0000	$5.7795e+07$	$2.94e+15$
K7803	50×50	7803	153	1.32	0.0000	$1.0923e+08$	$4.55e+15$
K10687	60×60	10687	179	1.32	1.6209	$2.5050e+07$	$7.86e+16$
K10827	48×75	10827	144	1.18	106.7974	$2.6020e+09$	$1.64e+18$
K10073	34×100	10073	102	1.00	471.7970	$4.7390e+10$	$5.05e+19$

Tabelle 4.1: Testmatrizen

Die Bytecode-Interpretation erfolgte mit der Standard-*JAVA(TM)2*-Laufzeitumgebung (Sun), Version 1.4.1_02 unter dem Betriebssystem *Linux*. Die Laufzeitumgebung verfügt über einen *HotSpot*-Compiler, der Bytecode in Maschinencode übersetzt und überwacht und damit die Ausführungsgeschwindigkeit im Vergleich zum Bytecode-Interpreter und im Vergleich zum *Just-In-Time*-Compiler wesentlich verbessert.

Zur Einschätzung der Performance der verwendeten Hard- und Softwareumgebung wurde ein Benchmark-Test des Linearen Algebra Softwarepaketes *Linpack* durchgeführt. Der Benchmark-Test löst ein dichtbesetztes lineares Gleichungssystem der Dimension 500 mit dem Gauss-Algorithmus und ermittelt die Anzahl Fließkommazahloperationen pro Sekunde [51]. Der Test ergibt für die eingesetzte Arbeitsumgebung $68.67 Mflop/s$.

Vorgehensweise Im folgenden wird die Theorie der implementierten Vergleichsverfahren in kompakter Form vorgestellt. Der numerische Aufwand der Verfahren wird durch eine Komplexitätsanalyse ermittelt. Die Konvergenzeigenschaften und das numerische Verhalten der Verfahren wird untersucht. Ein multiplikativer Vergleich und ein Vergleich des Speicherbedarfs wird durchgeführt und den Ergebnissen der Inversen Matrixiteration gegenübergestellt. Der Zeitbedarf zur Berechnung großer Matrizen wird qualitativ ermittelt. Alle Untersuchungsergebnisse werden bewertet.

4.2 Vergleichsmethode I : Verfahren von Lanczos

Zu den wichtigsten Verfahren zur Eigenwertbestimmung großer Profilmatrizen zählen die Subraummethoden. Insbesondere Methoden, die die spezielle Eigenwertaufgabe auf Krylov-Subräume der Form (4.2.1) abbilden erweisen sich für Matrizen großer Dimension als sehr leistungsfähig, wenn nur eine geringe Anzahl von Eigenzuständen zu bestimmen ist.

$$\mathcal{K}^m(\mathbf{A};\mathbf{v}) \;:=\; span\,\{\mathbf{v},\,\mathbf{A}\mathbf{v},\,\mathbf{A}^2\mathbf{v},\,\ldots,\,\mathbf{A}^{m-1}\mathbf{v}\} \tag{4.2.1}$$

Krylov-Subräume werden durch einen einzigen Vektor, den Startvektor der Krylov-Methode, bestimmt und schrittweise aufgebaut bzw. erweitert. Sie sind invariant bezüglich Skalierung und Spektralverschiebung.

$$\mathcal{K}^m(\alpha\mathbf{A}+\beta\mathbf{I};\mathbf{v}) \;=\; \mathcal{K}^m(\mathbf{A};\mathbf{v}) \tag{4.2.2}$$

Zu den erfolgreichsten und am besten entwickelten Krylov-Subraummethoden zählt das Verfahren von Lanczos. Dieses Verfahren geht zurück auf eine Entwicklung von *C. Lanczos* (1950), zur Bestimmung weniger Eigenvektoren symmetrischer Matrizen [30]. Das Gesamtpotential der Methode wurde jedoch erst 1972 durch die Untersuchungen von *Paige* ersichtlich, der die Gründe für die numerische Instabilität der Methode entdeckte.

4.2.1 Einfaches Lanczos-Verfahren

Das Lanczos-Verfahren zur Bestimmung einer Teilmenge $\psi(\mathbf{A}) \in \sigma(\mathbf{A})$ basiert auf dem schrittweisen Aufbau von Krylov-Subräumen $\mathcal{K}^m(\mathbf{A};\mathbf{v})$, mit deren Basisvektoren $\mathbf{q}_k$ die Eigenwertaufgabe (4.2.3) auf die Eigenwertaufgabe (4.2.4) wesentlich kleinerer Dimension abgebildet wird.

$$\mathbf{A}\,\mathbf{x}_i \;=\; \lambda_i\mathbf{x}_i \qquad\qquad \mathbf{A}\in\mathbb{R}^{(N\times N)} \tag{4.2.3}$$

$$\mathbf{T}\,\mathbf{y}_i \;=\; \rho_i\mathbf{y}_i \qquad\qquad \mathbf{T}\in\mathbb{R}^{(m\times m)}, m\ll N \tag{4.2.4}$$

Als Transformationsbasis sind die Krylov-Vektoren $\mathbf{v}, \mathbf{A}\mathbf{v}, \ldots, \mathbf{A}^{m-1}\mathbf{v}$ nur wenig geeignet, da diese mit wachsendem m immer mehr in Richtung des dominanten Eigenvektors zeigen. Stattdessen wird kontinuierlich eine orthonormale Basis $\mathbf{Q}_m = [\mathbf{q}_1 \ldots \mathbf{q}_m]$ zum Subraum $\mathcal{K}^m$ aufgebaut, die durch die Berechnung des Matrix-Vektorprodukts $\mathbf{q}_{i+1} = \mathbf{A}\mathbf{q}_i, (i = 1,\ldots,m)$ und anschließender Orthonormalisierung bezüglich der Basisvektoren $\mathbf{q}_1,\ldots,\mathbf{q}_i$ schrittweise erweitert wird. $\mathbf{Q}_m$ wird als Lanczos-Basis bezeichnet [30].

Das Ergebnis der Transformation mit der Lanczos-Basis $\mathbf{Q}_m$ wird mit folgendem Satz beschrieben.

Satz 4.2.1 *Gegeben sei der Krylov-Subraum $\mathcal{K}^m(\mathbf{A};\mathbf{v})$, definiert durch die symmetrische Matrix $\mathbf{A} \in \mathbb{R}^{(N\times N)}$ und den Vektor $\mathbf{v} \in \mathbb{R}^{(N\times 1)}$. $\mathbf{K}_m$ sei die Krylov-Basis zu $\mathcal{K}^m$ mit der QR-Zerlegung $\mathbf{K}_m = \mathbf{Q}_m\,\mathbf{R}_m$. Dann ist das Ergebnis der Transformation $\mathbf{Q}_m^T\mathbf{A}\mathbf{Q}_m$ eine symmetrische Tridiagonalmatrix $\mathbf{T}_m \in \mathbb{R}^{(m\times m)}$. $\mathbf{T}_m$ und $\mathbf{Q}_m$ sind eindeutig durch $\mathbf{A}$ und $\mathbf{v}$ bestimmt.*

Beweis : siehe [30] und [42]

Wegen des steigenden Aufwands zur Beseitigung numerischer Instabilität bei nicht exakter Arithmetik im Rechner ist das Lanczos-Verfahren als Methode zur vollständigen Tridiagonalisierung von $\mathbf{A}$ nicht attraktiv. Die Stärke des Verfahrens liegt in der Bestimmung einer kleinen Anzahl von extremalen Eigenzuständen großer Matrizen mit Profilstruktur.

Im folgenden wird der Lanczos Algorithmus in seiner einfachsten Form hergeleitet und beschrieben. Dabei wird zunächst exakte Arithmetik zugrunde gelegt. Anschließend wird die Erweiterung des Verfahrens für den praktischen Rechnereinsatz mit dem IRLES-Verfahren dokumentiert. Die Problematik endlicher Zahlendarstellung im Rechner und die daraus resultierende numerische Instabilität des Verfahrens wird im weiteren aufgegriffen, ein geeignetes Reorthogonalisierungsverfahren wird vorgestellt und implementiert.

Konstruktion einer Subraumbasis

Zur Berechnung von p Eigenwerten und Eigenvektoren $(p \ll N)$ der Aufgabe (4.2.3) wird die Tridiagonalisierung nach einer geeigneten Anzahl von m Zyklen $(m > p)$ mit der Teilbasis $\mathbf{Q}_m$ abgebrochen. Die Eigenwerte der Tridiagonalmatrix $\mathbf{T}^{(m\times m)}$ werden als Näherungslösungen zu den Eigenwerten von $\mathbf{A}$ bestimmt. Die Entwicklung der ersten m Spalten liefert :

$$\mathbf{A}\,\mathbf{Q}_m \;=\; \mathbf{Q}_m\,\mathbf{T}_m + \mathbf{r}_m\,\mathbf{e}_m^T \qquad\qquad \mathbf{Q}_m^T\mathbf{Q}_m \;=\; \mathbf{I}^{(m\times m)} \tag{4.2.5}$$

$\mathbf{e}_m$ m-ter Einheitsvektor $(m \times 1)$

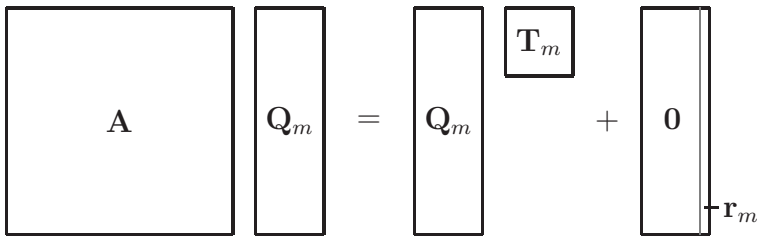


Bild 4.1: Lanczos-Zerlegung im Schritt m

Die m Basisvektoren $\mathbf{q}_i, (i = 1, \ldots, m)$ werden als Lanczos-Vektoren bezeichnet, der Vektor $\mathbf{r}_m$ als Restvektor der Aufgabe (4.2.5) im Schritt m. Er folgt unmittelbar aus dem Orthogonalisierungsprozess des iterierten Vektors $\hat{\mathbf{q}}_{m+1} = \mathbf{A}\mathbf{q}_m$ zur Teilbasis $\mathbf{Q}_m$.

$$\mathbf{r}_m \;=\; \beta_m\,\mathbf{q}_{m+1} \;=\; \mathbf{A}\,\mathbf{q}_m - \alpha_m\mathbf{q}_m - \beta_{m-1}\mathbf{q}_{m-1} \tag{4.2.6}$$

Mit (4.2.6) in (4.2.7) folgt die Lanczos-Zerlegung von $\mathbf{A}$ im Schritt m.

$$\mathbf{A}\,\mathbf{Q}_m \;=\; \mathbf{Q}_m\,\mathbf{T}_m + \beta_m\,\mathbf{q}_{m+1}\,\mathbf{e}_m^T \tag{4.2.7}$$

α_1	β_1		
β_1	α_2	$\ddots$	
	$\ddots$	$\ddots$	β_{m-1}
		β_{m-1}	α_m

Bild 4.2: Tridiagonalmatrix $\mathbf{T}_m$

Wegen der Symmetrie von $\mathbf{A}$ und der daraus resultierenden Tridiagonalform von $\mathbf{T}_m$ bedarf es im Schritt m lediglich einer Orthogonalisierung von $\hat{\mathbf{q}}_{m+1}$ bezüglich der Lanczos-Vektoren $\mathbf{q}_m$ und $\mathbf{q}_{m-1}$. Die Koeffizienten $\alpha_m = \mathbf{q}_m^T(\mathbf{A}\mathbf{q}_m)$ und $\beta_m = \|\mathbf{r}_m\|_2$ bauen schrittweise die Tridiagonalmatrix $\mathbf{T}_m$ (Bild 4.2) der Lanczos-Zerlegung (4.2.5) auf.

Nach m Schritten ist die spezielle Eigenwertaufgabe (4.2.3) mit Hilfe der orthonormalen Lanczos-Vektoren $\mathbf{q}_i, (i = 1, \ldots, m)$ auf die wesentlich kleinere Aufgabe (4.2.4) des Krylov-Subraums $\mathcal{K}^m$ abgebildet.

Die Abbildung der Aufgabe auf einen Subraum $\mathcal{K}^m$ und die anschließende Berechnung der Eigenzustände $(\rho_i, \mathbf{y}_i)$ wird als Rayleigh-Ritz-Verfahren bezeichnet. In [30] und [35] wird mit Hilfe des Minimax-Theorems gezeigt, daß die Ritzwerte ρ_i die bestmöglichen Näherungen aus dem Subraum $\mathcal{K}^m$ zu den Eigenwerten $\lambda_i(\mathbf{A})$ darstellen.

$$\rho_i \;=\; \max_{S}\;\min_{\mathbf{0}\neq\mathbf{y}_i\in S^m}\;\frac{\mathbf{y}_i^T\mathbf{T}\,\mathbf{y}_i}{\mathbf{y}_i^T\mathbf{y}_i} \tag{4.2.8}$$

$$\;=\; \max_{S}\;\min_{\mathbf{0}\neq\mathbf{y}_i\in S^m}\;\frac{\mathbf{y}_i^T\,\mathbf{Q}_m^T\,\mathbf{A}\,\mathbf{Q}_m\,\mathbf{y}_i}{\mathbf{y}_i^T\,\mathbf{Q}_m^T\,\mathbf{Q}_m\,\mathbf{y}_i} \tag{4.2.9}$$

$$\;=\; \max_{S}\;\min_{\mathbf{0}\neq\mathbf{y}_i\in S^m}\;\frac{\tilde{\mathbf{x}}_i^T\,\mathbf{A}\,\tilde{\mathbf{x}}_i}{\tilde{\mathbf{x}}_i^T\,\tilde{\mathbf{x}}_i} \qquad i = 1, 2, \ldots, m \tag{4.2.10}$$

Die Vektoren $\tilde{\mathbf{x}}_i = \mathbf{Q}_m^T \mathbf{y}_i$ werden als Ritz-Vektoren bezeichnet und sind die bestmöglichen Näherungen zu den Eigenvektoren $\mathbf{x}_i$ der ursprünglichen Aufgabe (4.2.3).

Welche Eigenwerte durch den Subraum $\mathcal{K}^m$ angenähert werden, wird entscheidend von der Wahl des Startvektors $\mathbf{q}$ beeinflußt. Die Anordnung der Eigenwerte $\rho_1, \ldots, \rho_n$ von $\mathbf{T}$ relativ zu den Eigenwerten λ_i von $\mathbf{A}$ ist a priori nicht bekannt. Die Bestimmung der p kleinsten Eigenwerte von $\mathbf{A}$ in Krylov-Subräumen $\mathcal{K}^p$ bis $\mathcal{K}^m, (p < m)$ erfordert damit eine Untersuchung der Trennungseigenschaften des bestimmten Teilspektrums $\psi(\mathbf{A})$.

Der folgende Algorithmus beschreibt die einzelnen Schritte der Rekursionsformel (4.2.6) zur Bestimmung einer m-Schritt-Lanczos-Zerlegung.

Algorithmus 3 Einfaches Lanczos Verfahren

Startwerte : $\mathbf{r}_0 \neq \mathbf{0}, \mathbf{q}_0 = \mathbf{0}, \beta_0 = \|\mathbf{r}_0\|$

1: **for** $i = 1, 2, \ldots, m$ **do**
2: $\quad \mathbf{q}_i = \mathbf{r}_{i-1}/\beta_{i-1}$ { Normalisierung des i-ten Lanczos-Vektors}
3: $\quad \mathbf{u}_i = \mathbf{A}\,\mathbf{q}_i$ { Berechnung des $i+1$-ten Krylov-Vektors }
4: $\quad \mathbf{r}_i = \mathbf{u}_i - \beta_{i-1}\,\mathbf{q}_{i-1}$ { Gram-Schmidt-Orthogonalisierung des }
5: $\quad \alpha_i = \mathbf{q}_i^T\,\mathbf{u}_i$ { $i+1$-ten Krylov-Vektors zur Teilbasis $\mathbf{Q}_i$ }
6: $\quad \mathbf{r}_i = \mathbf{r}_i - \alpha_i\,\mathbf{q}_i$
7: $\quad \beta_i = \|\mathbf{r}_i\|$
8: **end for**

Beurteilung der Approximation nach m Schritten

Zur Abschätzung der Genauigkeit der berechneten Ritzwerte und -vektoren $(\rho_i, \mathbf{y}_i)$ als Näherungslösung zum Eigenzustand $(\lambda_i, \mathbf{x}_i)$, wird die Restnorm $\|\mathbf{A}\tilde{\mathbf{x}}_i - \rho_i\tilde{\mathbf{x}}_i\|$ betrachtet. Mit der Eigenmatrix $\mathbf{Y}_m$ (4.2.11) multipliziert zu Gleichung (4.2.7) reduziert sich die Bestimmung dieser Norm auf die Berechnung der Norm des Restvektors $\|\mathbf{r}_m\|$:

$$\mathbf{Y}_m = [\mathbf{y}_1, \ldots, \mathbf{y}_m] = \mathbf{Q}_m^T\,\tilde{\mathbf{X}}_m \tag{4.2.11}$$

$$\mathbf{A}\,\mathbf{Q}_m\,\mathbf{Y}_m = \mathbf{Q}_m\,\mathbf{T}_m\,\mathbf{Y}_m + \beta_m\,\mathbf{q}_{m+1}\,\mathbf{e}_m^T\,\mathbf{Y}_m \tag{4.2.12}$$

$$\mathbf{A}\,\tilde{\mathbf{X}}_m = \tilde{\mathbf{X}}_m\mathbf{\Lambda}_m + \beta_m\,\mathbf{q}_{m+1}\,\mathbf{e}_m^T\,\mathbf{Y}_m \tag{4.2.13}$$

mit

$$\mathbf{\Lambda}_m = \mathbf{Y}_m^T\,\mathbf{T}_m\,\mathbf{Y}_m = diag[\rho_1, \ldots, \rho_m] \tag{4.2.14}$$

Für den Eigenzustand i folgt :

$$\mathbf{A}\,\tilde{\mathbf{x}}_i = \rho_i\,\tilde{\mathbf{x}}_i + \beta_m\,\mathbf{q}_{m+1}\,(\mathbf{e}_m^T\,\mathbf{Y}_m\mathbf{e}_i) \tag{4.2.15}$$

Mit der Norm von (4.2.15) und der Eigenschaft $\|\mathbf{q}_{m+1}\| = 1.0$ ergibt sich eine geeignete Fehlerschranke für die Ritzwerte ρ_i (4.2.17) und damit für die Eigenwertnäherungen zu $\mathbf{A}$ [13].

$$\|\mathbf{A}\,\tilde{\mathbf{x}}_i - \rho_i\,\tilde{\mathbf{x}}_i\| \quad = \quad |\beta_m|\,|y_{mi}| \tag{4.2.16}$$

$$\min_{\lambda_i \in \lambda(\mathbf{A})} |\lambda_i - \rho_i| \quad \leq \quad |\beta_m|\,|y_{mi}| \qquad\qquad i = 1, \ldots, m \tag{4.2.17}$$

Reorthogonalisierung

In exakter Arithmetik bilden die m Vektoren $\mathbf{q}_i, (i = 1, \ldots, m)$ eine orthonormale Basis zum Krylov-Subraum $\mathcal{K}^m$. Mit der endlichen Arithmetik im Rechner geht die Orthogonalität zwischen diesen Vektoren verloren, sobald ein Eigenwert λ_i konvergiert [30], [35]. Mit fortschreitender Berechnung nimmt die lineare Abhängigkeit der Lanczos-Vektoren $\mathbf{q}_j, (j = i + 1, \ldots)$ mehr und mehr zu. Die neuen Lanczos-Vektoren weisen dabei starke Komponenten in Richtung des konvergierten Ritz-Vektors $\hat{\mathbf{x}}_i$ auf, mit der Folge, daß das Verfahren Kopien zu den Basisvektoren des bestimmten invarianten Subraums $\mathcal{K}^m$ produziert. Die daraus resultierenden Eigenzustände, sogenannte *spurious modes*, gehören nicht zum Lösungsraum der Aufgabe.

Eine Vielzahl von Entwicklungen zur Bekämpfung des Orthogonalitätsverlusts spiegelt den großen Forschungsaufwand von über zwei Jahrzehnten wieder. In [30] und [35] sind die wichtigsten Entwicklungen beschrieben. Die sicherste Art und Weise zur Vermeidung von *spurious modes* ist vollständige Reorthogonalisierung der Lanzcos-Vektoren. Der numerische Aufwand und der Speicherbedarf steigt dabei kontinuierlich mit zunehmendem Berechnungsfortschritt und ist nur akzeptabel für eine feste, vorgegebene Subraumgröße.

Als Alternative zur vollständigen Reorthogonalisierung werden teilweise partielle oder selektive Reorthogonalisierungstrategien eingesetzt. Beide Strategien bewirken eine Semiorthogonalität der Lanczos-Vektoren. Das Skalarprodukt $(\mathbf{q}_i^T \mathbf{q}_j), (i \neq j)$ erfüllt dabei im allgemeinen eine Schranke $\delta \leq \sqrt{\epsilon}$. In [40] ist das Konzept der Semiorthogonalität erläutert und es wird gezeigt, daß die Semiorthogonalität ausreichend sein kann, um akzeptable Eigenwertnäherungen zu berechnen.

Bei einer partiellen Reorthogonalisierung wird der neu berechnete Lanczos-Vektor $\mathbf{q}_{i+1}$ nur gegen die Lanczos-Vektoren orthogonalisiert, zu denen die Semiorthogonalität verloren geht. Das Orthogonalitätslevel der Lanczos-Vektoren im Schritt m wird dabei durch die Skalare $\omega_{i,m+1} = \mathbf{q}_i^T \mathbf{q}_{m+1}, (i = 1, \ldots, m-1)$ überwacht. Für $\omega_{i,m+1} \leq \delta$ wird eine Reorthogonalisierung durchgeführt [40].

Bei der selektiven Orthogonalisierung dient als Entscheidungskriterium für eine Reorthogonalisierung nicht die Orthogonalität zwischen den Lanczos-Vektoren, sondern die Änderung der Orthogonalität zwischen den Lanczos-Vektoren $\mathbf{q}_i$ und den Ritz-Vektoren $\hat{\mathbf{x}}_i$. Dafür wird in regelmäßigen Intervallen die Restnorm $\|\hat{\mathbf{r}}_2\|$ der Ritz-Vektoren mit (4.2.16) ermittelt. Für $\|\hat{\mathbf{r}}_2\| \leq \sqrt{\varepsilon}\|\mathbf{A}\|_E$ werden die Ritz-Vektoren als gute Näherung betrachtet, berechnet und orthonormalisiert. In der nachfolgenden Berechnung wird jeder neu berechnete Lanczos-Vektor gegen die *guten* Ritz-Vektoren orthogonalisiert.

Beide Methoden der Semiorthogonalisierung sind numerisch günstiger als die vollständige Reorthogonalisierung, besitzen jedoch andere mögliche Schwachstellen, die in [35] und [30] dokumentiert sind.

Ein vollständiger Verzicht auf Reorthogonalisierung ist möglich, bedarf jedoch einer ständigen Überwachung des Eigenwertspektrums und verschiedener Kriterien zur Identifizierung *echter* Eigenwertnäherungen von einer Vielzahl von *spurious modes*.

4.2.2 IRLES - Implicitly Restarted Lanczos with Exact Shifts

Wegen der Abhängigkeit zwischen dem Startvektor des Lanczos-Verfahrens und den daraus berechneten Eigenzuständen wird die in 4.2.1 vorgestellte einfachste Form des Lanczos-Verfahrens sehr schnell unwirtschaftlich, wenn eine feste Anzahl von Eigenzuständen lückenlos bestimmt werden soll. Mehrfache Eigenwerte werden nur zufällig aufgrund von Rundungsfehlern approximiert. Die erforderliche Dimension des Krylov-Subraums zur vollständigen Bestimmung der gesuchten Eigenzustände ist a priori nicht bekannt. Der Aufwand der Iteration steigt zunehmend, mit jedem neu bestimmten Basisvektor.

Zur Reduzierung des erforderlichen Speicherbedarfs und des numerischen Aufwands zur Beseitigung des Orthogonalitätsverlusts zwischen den Lanczos-Vektoren wird das Verfahren ab einer bestimmten Subraumgröße abgebrochen und mit modifiziertem Startvektor neu gestartet.

Die im folgenden vorgestellte Variante des Lanczos-Verfahrens begrenzt den Speicherbedarf und den numerischen Aufwand durch eine feste Subraumgröße. Enthalten die Basisvektoren des aufgebauten Subraums nicht alle Informationen zur Bestimmung der gewünschten Eigenzustände, wird das Verfahren mit einem modifizierten Startvektor neu gestartet. Die Modifikation des Startvektors erfolgt über einen Polynomfilter, der den Startvektor in den gewünschten invarianten Subraum zwingt.

IRL - Implicitly Restarted Lanczos - Verfahren

Mit dem Lanczos-Verfahren (4.2.1) wird ein Subraum der Dimension $k+p$ aufgebaut. Die k besten Eigenwertnäherungen des Subraums zu den gesuchten Eigenzuständen werden bestimmt. Der Subraum wird anschließend wieder so auf Dimension k verkleinert, daß er nur die k besten Eigenwert- und Eigenvektornäherungen enthält.

Es sei k die Anzahl der gesuchten Eigenzustände einer symmetrischen, reellen Matrix $\mathbf{A}$. Die Anzahl p zusätzlich erzeugter Lanczos-Vektoren zur Bestimmung der gesuchten Eigenzustände sei nur unwesentlich größer als k. Im Schritt $k+p$ des Lanczos-Verfahrens ist eine Basis zum Subraum $\mathcal{K}^{(k+p)}$ aufgebaut.

$$\mathbf{A}\,\mathbf{V}_{k+p} = \mathbf{V}_{k+p}\,\mathbf{T}_{k+p} + \mathbf{r}_{k+p}\,\mathbf{e}_{k+p}^T \tag{4.2.18}$$

$$= \mathbf{V}_m\,\mathbf{T}_m + \beta_m\,\mathbf{v}_{m+1}\,\mathbf{e}_m^T \qquad m = k+p \tag{4.2.19}$$

Nach $k+p$ Lanczos-Schritten enthält der Krylov-Subraum $\mathcal{K}^{(k+p)}$ bereits einige angemessene Näherungen zu gewünschten Eigenwerten, aber auch schlechte Näherungen und Näherungen zu anderen, unerwünschten Eigenwerten. Die p schlechtesten Eigenvektornäherungen werden durch die Anwendung der QR-Zerlegung mit Spektralverschiebung auf die tridiagonale Matrix $\mathbf{T}_{k+p}$ aus den ersten k Gleichungen aus (4.2.18) entfernt. Als Verschiebungsparameter werden die p unerwünschten Eigenwertnäherungen eingesetzt. Für die erste QR-Zerlegung folgt :

$$(\mathbf{T}_m - \rho_1\,\mathbf{I}) := \mathbf{Q}\,\mathbf{R} \qquad \mathbf{Q}^T\mathbf{Q} = \mathbf{I} \quad \mathbf{Q},\mathbf{R} \in \mathbb{R}^{m\times m} \tag{4.2.20}$$

Substitution von (4.2.20) in (4.2.18) und Multiplikation mit $\mathbf{Q}$ ergibt :

$$(\mathbf{A} - \rho_1 \mathbf{I})\,\mathbf{V}_m - \mathbf{V}_m\,(\mathbf{T}_m - \rho_1 \mathbf{I}) = \mathbf{r}_m\,\mathbf{e}_m^T \tag{4.2.21}$$

$$(\mathbf{A} - \rho_1 \mathbf{I})\,\mathbf{V}_m - \mathbf{V}_m\,\mathbf{Q}\,\mathbf{R} = \mathbf{r}_m\,\mathbf{e}_m^T \tag{4.2.22}$$

$$(\mathbf{A} - \rho_1 \mathbf{I})\,\mathbf{V}_m\,\mathbf{Q} - \mathbf{V}_m\,\mathbf{Q}\,\mathbf{R}\,\mathbf{Q} = \mathbf{r}_m\,\mathbf{e}_m^T\,\mathbf{Q} \tag{4.2.23}$$

$$\mathbf{A}\,\mathbf{V}_m\,\mathbf{Q} - (\mathbf{V}_m\,\mathbf{Q})\,(\mathbf{R}\,\mathbf{Q} + \rho_1 \mathbf{I}) = \mathbf{r}_m\,\mathbf{e}_m^T\,\mathbf{Q} \tag{4.2.24}$$

Mit den Definitionen (4.2.25) und (4.2.26) folgt für (4.2.24) :

$$\hat{\mathbf{V}} := \mathbf{V}_m\,\mathbf{Q} \tag{4.2.25}$$

$$\hat{\mathbf{T}} := \mathbf{R}\,\mathbf{Q} + \rho_1 \mathbf{I} = \mathbf{Q}^T\,\mathbf{T}\,\mathbf{Q} \tag{4.2.26}$$

$$\mathbf{A}\,\hat{\mathbf{V}} - \hat{\mathbf{V}}\,\hat{\mathbf{T}} = \mathbf{r}_m\,\mathbf{e}_m^T\,\mathbf{Q} \tag{4.2.27}$$

Durch die Ähnlichkeitstransformation (4.2.26) bleibt $\hat{\mathbf{T}}$ symmetrisch und tridiagonal. Die orthonormale Matrix $\mathbf{Q}$ besitzt wegen der Tridiagonalform von $\hat{\mathbf{T}}$ obere Hessenbergform. Durch Entwicklung der ersten Spalte in Gleichung (4.2.22) folgt der Zusammenhang zwischen dem Startvektor $\mathbf{v}_1$ und dem modifizierten Startvektor $\hat{\mathbf{v}}_1$ für das nachfolgende p-Schritt Lanczos-Verfahren.

$$(\mathbf{A} - \rho_1 \mathbf{I})\,\mathbf{V}_m\,\mathbf{e}_1 - \mathbf{V}_m\,\mathbf{Q}\,\mathbf{R}\,\mathbf{e}_1 = \mathbf{r}_m\,\mathbf{e}_m^T\,\mathbf{e}_1 \tag{4.2.28}$$

$$(\mathbf{A} - \rho_1 \mathbf{I})\,\mathbf{v}_1 = \hat{\mathbf{V}}\,\mathbf{R}\,\mathbf{e}_1 \tag{4.2.29}$$

$$(\mathbf{A} - \rho_1 \mathbf{I})\,\mathbf{v}_1 = \hat{\mathbf{v}}_1\,r_{11} \tag{4.2.30}$$

mit

$$\hat{\mathbf{v}}_1 = \hat{\mathbf{V}}\,\mathbf{e}_1 \tag{4.2.31}$$

$$r_{11} = \mathbf{e}_1^T \mathbf{R}\,\mathbf{e}_1 \tag{4.2.32}$$

Die Anwendung aller p unerwünschten Eigenwertnäherungen ρ_i führt auf folgende modifizierte Lanczos-Faktorisierung :

$$\mathbf{A}\,\hat{\mathbf{V}}_{k+p} = \hat{\mathbf{V}}_{k+p}\,\hat{\mathbf{T}}_{k+p} + \mathbf{r}_{k+p}\,\mathbf{e}_{k+p}^T\,\tilde{\mathbf{Q}} \tag{4.2.33}$$

$$\hat{\mathbf{V}}_{k+p} = \mathbf{V}_{k+p}\,\tilde{\mathbf{Q}} \tag{4.2.34}$$

$$\hat{\mathbf{T}}_{k+p} = \tilde{\mathbf{Q}}^T\,\mathbf{T}_{k+p}\,\tilde{\mathbf{Q}} \tag{4.2.35}$$

$$\tilde{\mathbf{Q}} = \mathbf{Q}_1\,\mathbf{Q}_2\,\ldots\,\mathbf{Q}_p \qquad \tilde{\mathbf{Q}}^T\,\tilde{\mathbf{Q}} = \mathbf{I} \tag{4.2.36}$$

Wegen der oberen Hessenbergform der Matrizen $\mathbf{Q}_i, i = 1, \ldots, p$ sind die ersten $k-1$ Koeffizienten der letzten Zeile m von $\tilde{\mathbf{Q}}$ gleich null. Der letzte Term in (4.2.33) ist damit ein Vektor mit $(k-1)$ Nullelementen :

$$\mathbf{r}_{k+p}\, \mathbf{e}_{k+p}^T\, \tilde{\mathbf{Q}} = \mathbf{v}_{k+p+1}\, \beta_{k+p}\, \mathbf{e}_{k+p}^T\, \tilde{\mathbf{Q}} \tag{4.2.37}$$

$$= \mathbf{v}_{k+p+1} \, [\underbrace{0 \ \ldots \ 0 \,|\, (\beta_m\, \hat{q}_{m,k})}_{k} \,|\, \underbrace{(\beta_m\, \hat{q}_{m,j})}_{p}] \qquad j = k+1, \ldots, m \tag{4.2.38}$$

Zur Reduktion der modifizierten $(k+p)$-Schritt Lanczos-Faktorisierung auf Dimension k wird der Vektor (4.2.37) in Gleichung (4.2.33) substituiert. Die Matrizen und Vektoren in (4.2.33) werden nach den Dimensionen k und p partitioniert :

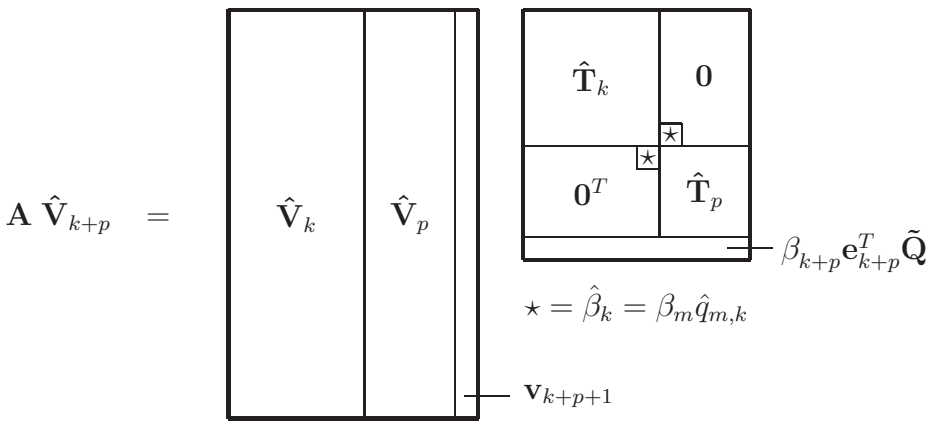


Bild 4.3: Modifizierte $(k+p)$-Schritt Lanczos-Faktorisierung

Wegen der Orthogonalität der Vektoren $\hat{\mathbf{q}}_i$ in (4.2.36) sind auch die Vektoren in $\mathbf{V}_{k+p+1}$ orthogonal. Gleichung (4.2.33) ist damit analog zu (4.2.18) eine Lanczos-Zerlegung der Matrix $\mathbf{A}$. Die Entwicklung der ersten k Spalten von Gleichung (4.2.33) baut schließlich wieder eine orthonormale Basis zum Subraum $\mathcal{K}^k$ auf. Der Startvektor dieser Lanczos-Zerlegung ist darstellbar als Linearkombination der Eigenvektornäherungen zu den k akzeptierten Eigenwertnäherungen aus (4.2.18).

Der neue Startvektor $\mathbf{v}_{k+1}$ ist durch einen Polynomfilter gefiltert, der als Nullstellen die verwendeten p Verschiebungsparameter $\rho_j \in \{\rho_1, \ldots, \rho_{k+p}\}$ besitzt. Ausgehend von der k-Schritt Lanczos-Zerlegung wird mit dem modifizierten Startvektor das Lanczos-Verfahren erneut gestartet. Dieser Prozess wird solange wiederholt, bis die gewünschten Eigenzustände im Subraum $\mathcal{K}^{(k+p)}$ konvergiert haben.

Zum Erhalt der Orthogonalität zwischen den Lanczos-Vektoren wird eine Reorthogonalisierung vorgenommen. Eine klassische bzw. modifizierte Gram-Schmidt-Orthogonalisierung reicht häufig nicht aus um Orthogonalität mit einem Orthogonalitätslevel ϵ zu gewährleisten. Insbesondere wenn die Norm des orthogonalisierten Vektors $\mathbf{v}_{i+1}$ sich nur sehr geringfügig ändert, ist die Gram-Schmidt-Orthogonalisierung numerisch nicht vollständig stabil [44]. Stattdessen wird eine Methode nach *Daniel, Gragg, Kaufman* und *Stewart* (DGKS-Methode) eingesetzt, die sich durch hervorragende numerische Stabilität und Genauigkeit auszeichnet. Dabei werden folgende Berechnungsschritte im Schritt j ausgeführt :

$$(1) \quad \mathbf{s} = \mathbf{V}_j^T \mathbf{r}_j \tag{4.2.39}$$

$$(2) \quad \mathbf{r}_j = \mathbf{r}_j - \mathbf{V}_j \mathbf{s} \tag{4.2.40}$$

$$(3) \quad \alpha_j = \alpha_j + s_j \tag{4.2.41}$$

Die DGKS-Methode wird nur eingesetzt, falls $\|(\mathbf{A}\,\mathbf{v}_j)\|_2 \leq \tau \|\mathbf{r}_j\|_2$. In [19] wird $\tau = 0.717$ vorgeschlagen. In einem iterativen Verfeinerungsschritt wird die Methode wiederholt, falls $(\mathbf{r}_j^T \mathbf{v}_i) > \epsilon, (i = 1, \ldots, j-1)$.

Mehrfache Eigenwerte

In der vorliegenden Form kann der Lanczos-Algorithmus nicht dazu verwendet werden, Eigenwerte mit Multiplizität > 1 zu bestimmen. Die Lanczos-Vektoren $\mathbf{q}_i, (i = 1, \ldots, m)$ bilden eine orthonormale Basis zum Krylov-Subraum $\mathcal{K}^m(\mathbf{A}, \mathbf{q}_1) = \{\mathbf{q}_1, \mathbf{A}\mathbf{q}_1, \ldots, \mathbf{A}^{m-1}\mathbf{q}_1\}$, indem die berechneten Eigenvektornäherungen $\tilde{\mathbf{x}}_i$ liegen. Damit ist das Verfahren nicht in der Lage Eigenvektoren zu bestimmen, die orthogonal zum Startvektor $\mathbf{q}_1$ und damit orthogonal zum Krylov-Subraum $\mathcal{K}^m$ sind. Aus dem p-dimensionalen Subraum $\mathcal{S}^p$ zu einem mehrfachen Eigenwert $\lambda_i^{(p)}$ kann deshalb nur einer der p Eigenvektoren $\mathbf{x}_k, (k = 1, \ldots, p)$ aus dem Startvektor $\mathbf{q}_1$ bestimmt werden.

Aus theoretischer Sicht stellt dieser Umstand einen erheblichen Mangel des Lanczos-Verfahrens dar, dessen Bedeutung im praktischen Rechnereinsatz jedoch teilweise aufgehoben wird. Die Konvergenz von α_i zu einem Eigenwert λ_i ist durch ein betragskleines Subdiagonalelement β_i in der Matrix $\mathbf{T}^{(m \times m)}$ gekennzeichnet. In Analogie zu Kapitel 3.24 setzt mit der Konvergenz von β_i gegen Null ein Deflationsprozess ein, der die Eigenwertnäherung α_i vom Rest der Matrix $\mathbf{T}^{(m \times m)}$ entkoppelt. Zusätzlich werden in den nachfolgenden Berechnungsschritten durch Rundungsfehler neue Eigenvektorkomponenten eingetragen, die eine Bestimmung mehrfacher Eigenwerte ermöglichen.

Dieser Prozess der Selbstdeflation stellt jedoch keine zuverlässige Strategie zur Bestimmung mehrfacher Eigenwerte dar. Im vorliegenden Verfahren wird eine Deflation dadurch ausgeführt, daß Subdiagonalelemente bei Erfüllen der Bedingung (4.2.42) explizit auf Null gesetzt werden.

$$|\beta_i| \leq \varepsilon \left(|\alpha_i| + |\alpha_{i+1}|\right) \tag{4.2.42}$$

Zusätzlich wurde eine stabile Deflationsmethode nach [38] implementiert, die durch eine Ähnlichkeitstransformation der Gleichung (4.2.18) mit einer Orthonormalmatrix $\mathbf{U}$ eine Deflation ausgewählter Eigenzustände mit nutzerspezifizierten Fehlerschranken erlaubt. Zwei Verfahren werden dabei zur Deflation angewendet. Mit einem *locking*-Verfahren werden gewünschte, bereits konvergierte Eigenzustände in der Subraumbasis festgesetzt. Das entsprechende Subdiagonalelement β_i wird dabei auf den Wert Null transformiert. Mit einem *purging*-Verfahren werden unerwünschte, bereits konvergierte Eigenzustände aus der Subraumbasis entfernt. Die unerwünschten Eigenzustände werden dabei in die letzten Zeilen und Spalten der Gleichung (4.2.18) transformiert und im Zuge des impliziten Neustarts aus der Berechnung entfernt.

Die Entwicklung geeigneter Orthonormalmatrizen für den *locking* bzw. *purging* Prozess ist in [38] gezeigt. Beide Prozesse ermöglichen eine stabile Beeinflussung der zu bestimmenden Eigenzustände bereits während eines frühen Stadiums der Berechnung. Verschiedene Testläufe haben jedoch gezeigt, daß der numerische Aufwand im allgemeinen dadurch nicht bzw. nur sehr geringfügig verbessert wird.

4.2.3 Komplexität des IRLES-Verfahren

Der Speicherbedarf des IRLES-Verfahrens wird maßgeblich von der Subraumgröße m zur Bestimmung von k Eigenzuständen $(\lambda_i, \mathbf{x}_i)$ bestimmt. Die Systemmatrix $\mathbf{A}$ wird unter Ausnutzung von Symmetrie gespeichert.

Gesamtspeicherbedarf (double): $N\,(b + m + 3) + 2\,m^2 + O(m)$

Eine zuverlässige Abschätzung der Komplexität des IRLES-Verfahren ist schwierig. Das Konvergenzverhalten des Verfahrens wird durch verschiedene Faktoren maßgeblich beeinflußt. Die Anzahl p zusätzlich generierter Lanczos-Vektoren während der Restartphase läßt sich nicht eindeutig a priori festlegen. Ein optimaler Wert für p kann nur durch Variation in verschiedenen Testläufen angenähert werden. Im allgemeinen liefert das Verfahren gute Konvergenz mit $p \geq k$ und erfordert damit mindestens eine Verdopplung der gesuchten Subraumgröße.

Weitere Faktoren, die eine Abschätzung der Komplexität beeinflußen, sind die Verteilung der Eigenwerte im Gesamtspektrum und die Wahl des Startvektors. Das Verfahren kann nur Eigenformen bestimmen, die im Startvektor enthalten sind. Fehlende Richtungen werden durch Rundungsfehler im Rechner während der Iteration in die Berechnung eingetragen, verhindern jedoch teilweise eine rasche Konvergenz.

Tabelle 4.2 gibt eine Übersicht über die Komplexität der einzelnen Berechnungsphasen des IRLES-Verfahrens. Eine vollständige Reorthogonalisierung in jedem Iterationszyklus wird angenommen. Die Terme $[\cdot]$ in Klammer berücksichtigen den Zusatzaufwand für die Faktorisierung der Matrix $\mathbf{A}$, falls im *shift-and-invert* Modus gearbeitet wird (s. Kapitel 4.4).

Operation	Berechnung	multiplikativer Aufwand	Speicherbedarf
1	$\mathbf{v}_i,\ \mathbf{r}_i, \alpha_i, \beta_i,$	$N\,(k+p)\,(2\,b+5)$	$N\,(b+1)+2\,N$
1.1	$[\,\mathbf{A}=\mathbf{L}\,\mathbf{D}\,\mathbf{L}^T\,]$	$[\,\frac{1}{2}\,N\,b^2+\frac{3}{2}\,N\,b\,]$	
1.2	$(\ i=1,\ldots,(k+p)\ )$	$(k+p)^2\,N$	$N\,(k+p)+(k+p)$
2			
2.1	$\mathbf{T}=\mathbf{X}\,\mathbf{D}\,\mathbf{X}^T$	$s\ 5\,(k+p)^3$	$(k+p)^2+2\,(k+p)$
2.2	$s\ 2\,N\,k\,(k+p)$	$(k+p)^2$	
2.3		$s\ N\,p\,(2\,b+5)$	
2.4	$(\ i=k,\ldots,(k+p)\ )$	$s\ N\,p\,(2\,k+p)$	
3		$\sim N\,(k+p)(2\,b+5+k+p)+$ $s\,N\,p\,(2b+4k+p+5)+$ $s\,N\,2k^2+s\,O(k+p)^3$ $+\,[\,\frac{1}{2}\,N\,b^2+\frac{3}{2}\,N\,b\,]$	$N\,(b+k+p+3)+$ $2\,(k+p)^2+O(k+p)$
4	$\tilde{\mathbf{X}}_{(N\times k)}=\mathbf{V}\,\mathbf{X}$	$N\,k^2$	

1 $(k+p)$-Lanczos Schritte
1.1 [Shift-And-Invert Mode]
1.2 Reorthogonalisierung von $\mathbf{V}_{(N\times i)}$
2 Iteration $s\ =\ 1,2,3\ldots$
2.1 $(k+p)$ Eigenzustände $(\rho_i,\mathbf{x}_i)$ von $\mathbf{T}$
2.2 p-fache Faktorisierung $(\mathbf{T}-\rho_i\mathbf{I})=\mathbf{Q}\,\mathbf{R}$
2.3 p-Lanczos Schritte
2.4 Reorthogonalisierung von $\mathbf{V}_{(N\times i)}$
3 Gesamtaufwand (Eigenwerte)
4 Bestimmung von k Ritz-Vektoren $\tilde{\mathbf{x}}_j$

N Dimension von $\mathbf{A}$
b mittlere Bandbreite von $\mathbf{A}$
k Anzahl berechneter Eigenwerte = Dimension des gewünschten Subraums
p Anzahl zusätzlicher Lanczos-Vektoren
s Anzahl impliziter Neustarts

Tabelle 4.2: Komplexität des *Implicitly Restarted Lanczos with Exact Shifts*-Verfahren

4.3 Vergleichsmethode II : Givens-QRI-Verfahren

Die vollständige Lösung der Eigenwertaufgabe mit dem *Givens-QRI-Verfahren (GQRI)* erfolgt in drei Teilschritten :

1. *Tridiagonalisierung symmetrischer Matrizen :* Nach einem Algorithmus von *Schwarz* wird die symmetrische Profilmatrix $\mathbf{A}$ durch eine Folge von Ähnlichkeitstransformationen in eine tridiagonale Form $\mathbf{T}$ überführt. Als orthonormale Transformationsmatrizen werden ebene Rotationsmatrizen eingesetzt die zur Eliminierung einzelner Koeffizienten verwendet werden.

   Eine Verwendung von Spiegelmatrizen nach der Methode von Householder eignet sich für Matrizen mit ausgeprägter Profilstruktur nur bedingt, da die Profilstruktur im unreduzierten Teil der Matrix durch die Transformationen schnell zerstört wird. Verschiedene Entwicklungen haben Strategien hervorgebracht, die eine vollständige Wiederherstellung des Matrixprofils gewährleisten bzw. mit einer teilweisen Wiederherstellung des Matrixprofils einen erhöhten Speicherbedarf akzeptieren. Beide Strategien sind mit einem hohen numerischen Aufwand verbunden und für große Matrizen mit geringer Bandbreite nur bedingt geeignet [18].

2. *QR-Verfahren für tridiagonale Matrizen mit impliziter Shifttechnik :* Diese Spezialisierung der QR-Methode hat sich bei der Bestimmung aller Eigenwerte tridiagonaler Matrizen als besonders effektiv und stabil erwiesen. Der geringe Speicherbedarf $(2n)$ und die schnelle Konvergenz der Methode minimieren den numerischen Aufwand dieses Berechnungsschritts.

3. *Inverse Vektoriteration zur Eigenvektorbestimmung :* Mit den bestimmten Eigenwertnäherungen aus 2. werden ausgewählte Eigenvektoren durch die Inverse Vektoriteration mit Spektralverschiebung als Näherung bestimmt. Die klassische *Householder QRI-Methode* für vollbesetzte Matrizen bestimmt die Eigenvektoren der Tridiagonalmatrix $\mathbf{T}$ und transformiert diese anschließend mit der Transformationsbasis aus 1. in die Eigenvektoren der ursprünglichen Aufgabe. Diese Vorgehensweise ist für Bandmatrizen großer Dimension unzweckmäßig, da sie die Speicherung des akkumulierten Transformationsprodukts erfordert. Stattdessen werden die Eigenvektoren mit den Eigenwertnäherungen und der inversen Vektoriteration direkt an der Koeffizientenmatrix $\mathbf{A}$ der ursprünglichen Aufgabe bestimmt.

Für die Speicherung der Eigenwertaufgabe wird die Symmetrie der Koeffizientenmatrix in allen drei Teilprozessen genutzt.

Lösungsansatz

Die spezielle Eigenwertaufgabe (4.3.1) wird durch die Ähnlichkeitstransformation (4.3.3) in die Tridiagonalform (4.3.2) überführt. Die Eigenwerte der Aufgabe (4.3.2) sind die Eigenwerte der ursprünglichen Aufgabe (4.3.1). Die Eigenvektoren $\mathbf{x}_i$ werden mit der orthonormalen Transformationsmatrix $\mathbf{Q}$ aus den Eigenvektoren $\mathbf{y}_i$ bestimmt.

$$\mathbf{A}\,\mathbf{x}_i = \lambda_i\,\mathbf{x}_i \tag{4.3.1}$$

$$\mathbf{T}\,\mathbf{y}_i = \lambda_i\,\mathbf{y}_i \tag{4.3.2}$$

$$\mathbf{T} := \mathbf{Q}^T\mathbf{A}\,\mathbf{Q} \qquad \mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\,\mathbf{Q}^T = \mathbf{I} \tag{4.3.3}$$

$$\mathbf{y}_i := \mathbf{Q}^T\mathbf{x}_i \tag{4.3.4}$$

Die orthonormale Transformationsmatrix $\mathbf{Q}$ wird schrittweise bei der Transformation von $\mathbf{A}$ auf Tridiagonalform $\mathbf{T}$ aus dem Produkt elementarer Transformationsmatrizen $\mathbf{R}_m$ aufgebaut.

$$\mathbf{Q} = \prod_m \mathbf{R}_m \tag{4.3.5}$$

Die Tridiagonalmatrix $\mathbf{T}$ wird mit dem QR-Verfahren in die Diagonalform (4.3.6) überführt. Die Eigenwerte der Aufgabe (4.3.6) sind die Diagonalkoeffizienten d_{ii}. Sie sind die Eigenwerte der ursprünglichen Aufgabe (4.3.1). Die Eigenvektoren von (4.3.6) sind die Einsvektoren $\mathbf{e}_i$. Die Eigenvektoren $\mathbf{x}_i$ sind mit den orthonormalen Transformationsmatrizen $\mathbf{Q}$ und $\mathbf{V}$ bekannt (4.3.9).

$$\mathbf{D}\,\mathbf{e}_i = \lambda\,\mathbf{e}_i \tag{4.3.6}$$

$$\mathbf{D} := \mathbf{V}^T\mathbf{T}\,\mathbf{V} \qquad \mathbf{V}^T\mathbf{V} = \mathbf{V}\,\mathbf{V}^T = \mathbf{I} \tag{4.3.7}$$

$$\mathbf{e}_i := \mathbf{V}^T\mathbf{y}_i \tag{4.3.8}$$

$$\mathbf{x}_i := \mathbf{Q}\,\mathbf{V}\,\mathbf{e}_i \tag{4.3.9}$$

Die Transformationsmatrix $\mathbf{V}$ wird mit der Diagonalisierung von $\mathbf{T}$ aus dem Produkt elementarer Transformationsmatrizen $\mathbf{P}_j$ gebildet. Im Gegensatz zur Matrix $\mathbf{Q}$, ist $\mathbf{V}$ nicht nach einem endlichen Prozess vollständig bestimmt, sondern ist das Ergebnis einer iterativen Berechnung von Näherungen zu den Eigenwerten von $\mathbf{A}$.

$$\mathbf{V} = \lim_{s\to\infty} \mathbf{V}_s = \prod_{j=1}^{s-1} \mathbf{V}_s \cdot \mathbf{P}_j \tag{4.3.10}$$

Die vollständige Bestimmung der Eigenvektoren $\mathbf{x}_i$ erfordert die explizite Berechnung des Produkts $\mathbf{Q}\,\mathbf{V}$. Der multiplikative Aufwand dieser Berechnung ist proportional $O(N^3)$. Der

Speicherbedarf beträgt N^2. Für große Matrizen $(N > 50)$ ist diese Vorgehensweise unwirtschaftlich und nur bedingt durchführbar. Für die Bestimmung einzelner Eigenvektoren werden häufig mit der Inversen Vektoriteration die gewünschten Eigenvektoren $\mathbf{y}_i$ der Tridiagonalform (4.3.2) bestimmt und mit Gleichung (4.3.4) in die Eigenvektoren $\mathbf{x}_i$ der ursprünglichen Aufgabe transformiert. Diese Vorgehensweise erfordert wegen der Kenntnis von $\mathbf{Q}$ ebenfalls einen Speicherbedarf von $O(N^2)$. Der multiplikative Aufwand ist ebenfalls proportional $O(N^3)$.

Für die Bestimmung einer beliebigen Anzahl von Eigenvektoren $\mathbf{x}_i$ bei Kenntnis der Eigenwerte λ_i wird im folgenden die Inverse Vektoriteration direkt auf die symmetrische Profilmatrix $\mathbf{A}$ angewandt.

4.3.1 Transformation auf Tridiagonalform

Die Profilmatrix $\mathbf{A}$ mit Profilvektor $\mathbf{pr}$ zur Speicherung des rechten Matrixprofils wird mit dem Algorithmus von Schwarz auf Tridiagonalform $\mathbf{T}$ transformiert. Als Transformationsmatrizen werden Matrizen $\mathbf{R}(i,k)$ verwendet, die sukzessive die Koeffizienten $a_{ik}(k = pr_i, \ldots, i+2)$ durch Rotationen in der i,k-Ebene eliminieren. Diese Matrizen werden allgemein als Givens-Matrizen bezeichnet. Sowohl der Algorithmus von Schwarz als auch die klassische Methode nach Householder für dichtbesetzte Matrizen erzeugen dabei zusätzliche Elemente außerhalb des Matrixprofils. Die nachfolgenden Rotationen werden so gewählt, daß die Profilstruktur schrittweise wieder hergestellt wird.

Wegen des ausgeprägten Bandcharakters der in dieser Arbeit untersuchten Matrizen wird für die Givens-QRI-Methode (GQRI), eine aus algorithmischer Sicht vorteilhaftere Bandmatrix-Speicherstruktur gewählt. Die Profilmatrix $\mathbf{A}$ mit maximaler Bandbreite b wird in $b+1$ Diagonalen abgespeichert. Die Vektoren zur Speicherung der Hauptdiagonalen und der Subdiagonalen (■) von $\mathbf{A}$ besitzen Dimension N und werden mit der Haupt- und Subdiagonalen der Tridiagonalmatrix $\mathbf{T}$ überschrieben. Ein zusätzlicher Vektor der Dimension $N-b$ (□) wird zur temporären Speicherung der Zusatzelemente bereitgestellt.

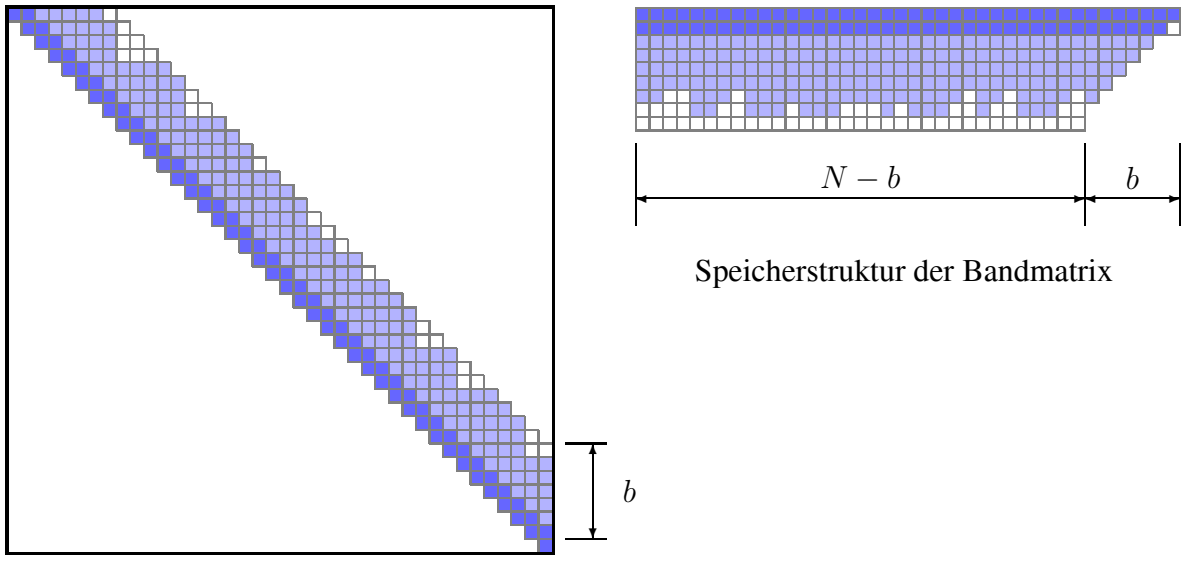


Bild 4.4: Symmetrische Profilmatrix als Bandmatrix gespeichert

Reduktionsprozess

Der Tridiagonalisierungsprozess erfolgt zeilenweise, beginnend mit der obersten Matrixzeile. Mit der Transformation (4.3.11) werden die Koeffizienten a_{ik} der Zeile i vom Profilrand bis

zur ersten Subdiagonalen schrittweise eliminiert. Die Indizes i und k der Rotationsmatrix $\mathbf{R}_{ik}$ beziehen sich auf die Position des eliminierten Koeffizienten a_{ik}, nicht aber auf die modifizierten Zeilen und Spalten der Ähnlichkeitstransformation (4.3.11). Im Gegensatz zur QR-Methode aus Kapitel 3.2, belegen die Rotationsparameter $c(= cos(\phi))$ und $s(= sin(\phi))$ in $\mathbf{R}_{ik}$ nicht die Positionen $(i,i),(k,k),(i,k)$ und (k,i), sondern gruppieren sich um die Hauptdiagonale an den Positionen $(k,k),(k-1,k-1),(k,k-1)$ und $(k-1,k)$. Die Indizes der modifizierten Zeilen und Spalten unterscheiden sich damit stets um den Wert eins.

$$\begin{aligned} \hat{\mathbf{A}} &= \mathbf{R}_{ik}^T\,\mathbf{A}\,\mathbf{R}_{ik} & i &= 1,\ldots,N-2 \\ & & k &= i+b,\ldots,i+2 \end{aligned} \tag{4.3.11}$$

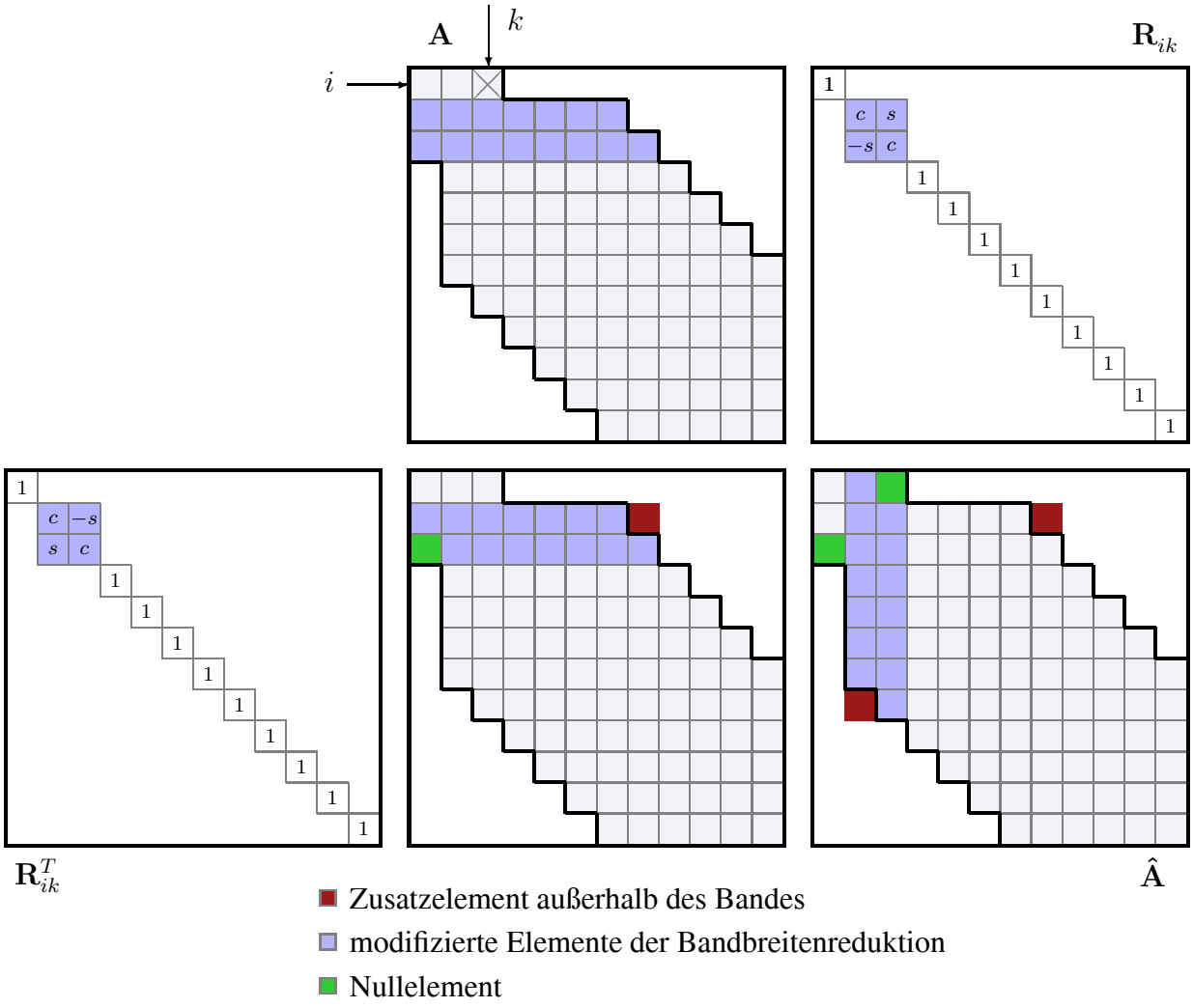


Bild 4.5: Bandbreitenreduktion : Transformation $\mathbf{R}_{ik}^T\,\mathbf{A}\,\mathbf{R}_{ik}$

Bild 4.5 zeigt die vollständige Ähnlichkeitstransformation (4.3.11). Zur besseren Veranschaulichung wurde auf die Ausnutzung der Symmetrie verzichtet.

Jede Transformation mit (4.3.11) erzeugt ein Zusatzelement außerhalb des Bandes an Position (p, q) (4.3.12). Zur Wiederherstellung der Bandstruktur wird das Zusatzelement durch nachfolgende Transformationen schrittweise nach unten aus der Matrix herausgetrieben (Bild 4.6). Die Transformationen zur Wiederherstellung des Profils sind von der Form (4.3.12).

$$\check{\mathbf{A}} = \mathbf{R}_{pq}^T \hat{\mathbf{A}}\, \mathbf{R}_{pq} \qquad \begin{aligned} p &= k-1 \\ q &= k+b \end{aligned} \tag{4.3.12}$$

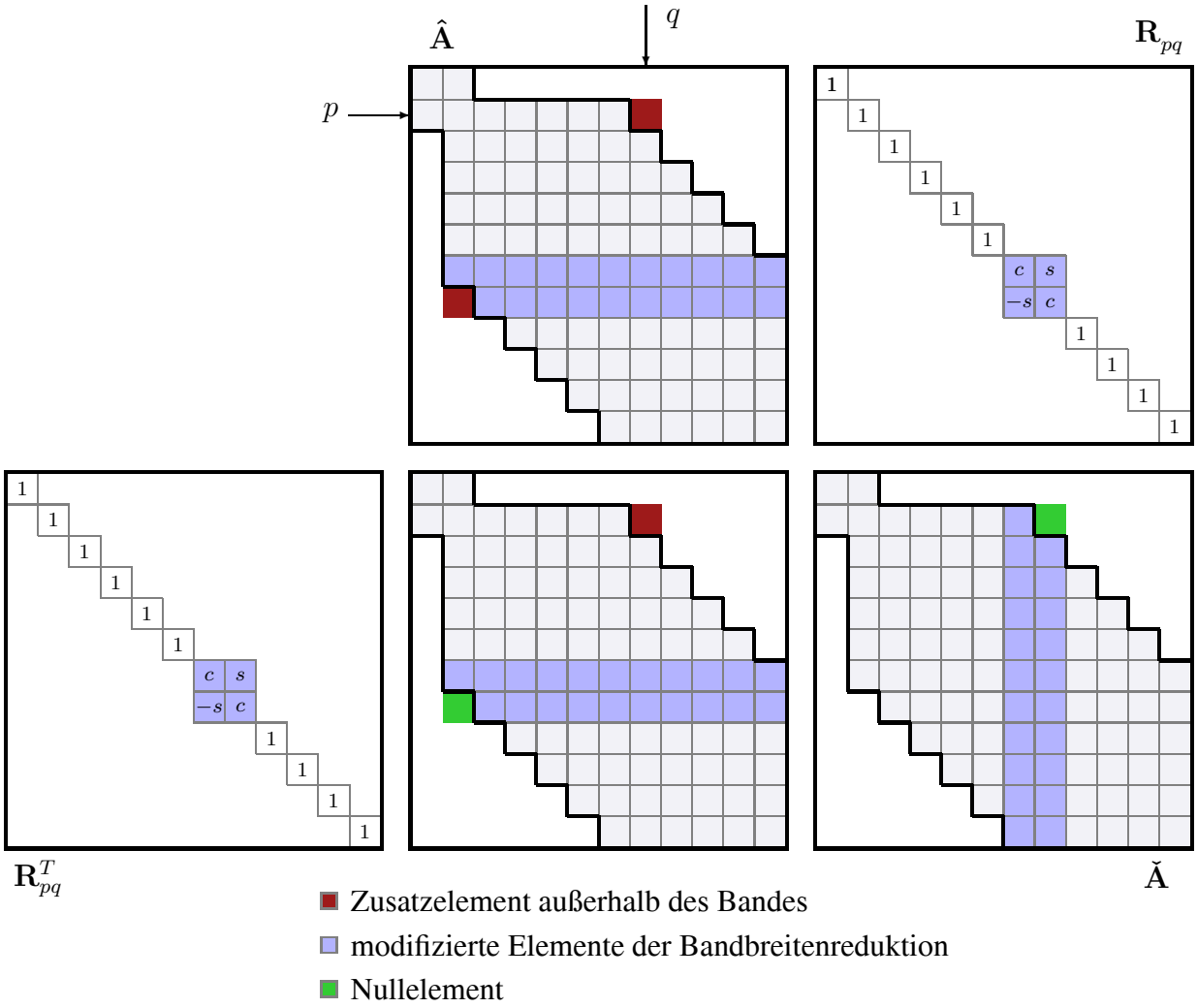


Bild 4.6: Bandbreitenreduktion : Transformation $\mathbf{R}_{pq}^T \hat{\mathbf{A}}\, \mathbf{R}_{pq}$

Im allgemeinen verursacht die Transformation (4.3.12) weitere Elemente außerhalb des Matrixprofils an der Position $(\hat{p}, \hat{q})$, analog zur Transformation (4.3.11), die mit (4.3.13) eliminiert

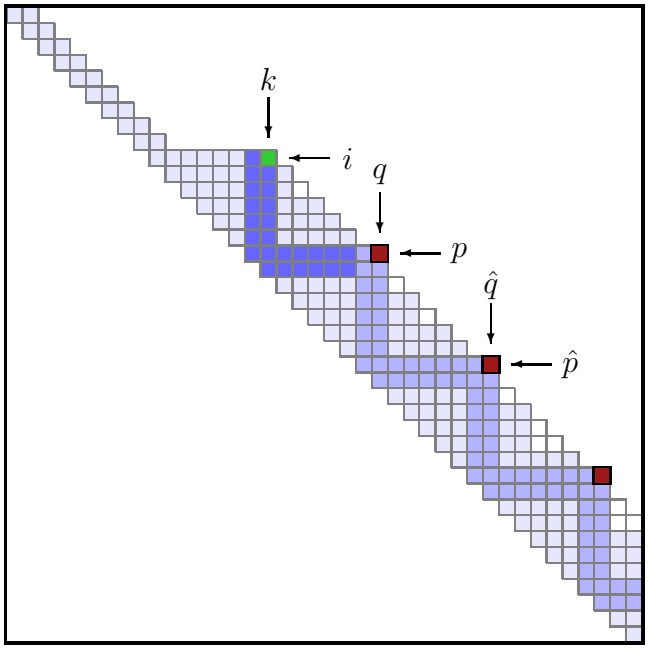
werden. Dieser, unter der Bezeichnung *bulge chasing* bekannte Prozess wird solange mit modifizierter Transformationsmatrix $\mathbf{R}_{\hat{p}\hat{q}}$ fortgesetzt, bis die Indizes $\hat{p}, \hat{q}$ größer der Matrixdimension sind und die Bandstruktur wiederhergestellt ist.

$$\bar{\mathbf{A}} = \mathbf{R}_{\hat{p}\hat{q}}^T \check{\mathbf{A}}\, \mathbf{R}_{\hat{p}\hat{q}} \qquad \begin{aligned} \hat{p} &= p + b \\ \hat{q} &= q + b \end{aligned} \tag{4.3.13}$$

Die Rotationswinkel werden aus der Bedingung (4.3.14) bestimmt.

$$sin(\phi) \cdot a_{m,n-1} + cos(\phi) \cdot a_{m,n} = 0. \qquad \begin{aligned} m &\in \{i, p, \hat{p}\} \wedge \\ n &\in \{k, q, \hat{q}\} \end{aligned} \tag{4.3.14}$$

Bild 4.7 zeigt die Elimination des Koeffizienten a_{ik} (grün) und den anschließenden *bulge chasing*-Prozess der Koeffizienten a_{pq}, $a_{\hat{p}\hat{q}}$, ...



- Zusatzelement außerhalb des Bandes
- modifizierte Elemente der Bandbreitenreduktion
- modifizierte Elemente des *bulge chasing*-Prozess
- Nullelement

Bild 4.7: Bandbreitenreduktion

4.3.2 QR-Verfahren für tridiagonale Matrizen mit impliziter Shifttechnik

Die Mehrzahl der Verfahren, die Eigenwerte durch eine Folge von Ähnlichkeitstransformationen bestimmen, setzen eine Tridiagonalform der Matrix voraus. Die Leistungsfähigkeit der Verfahren wird dadurch stark verbessert. Als effektivste Entwicklung zur vollständigen Bestimmung der Eigenwerte tridiagonaler Matrizen hat sich der QR-Algorithmus für tridiagonale Matrizen mit impliziter Shifttechnik etabliert.

Im Gegensatz zum klassischen QR-Verfahren für vollbesetzte Matrizen erfolgt die Transformation auf Diagonalform im tridiagonalen Fall nicht durch wiederholte explizite Zerlegung und Rekombination der iterierten Matrix, sondern durch einen zyklischen Prozess, wie er bereits im vorangegangenen Abschnitt zur Tridiagonalisierung von $\mathbf{A}$ beschrieben wurde.

Im Schritt s der Iteration wird die spektralverschobene Matrix $(\mathbf{T}_s - c_s\,\mathbf{I})$ analog zu Kapitel 3.2 in das Produkt $\mathbf{Q}_s\mathbf{R}_s$ zerlegt (4.3.15) und anschließend durch Multiplikation der Zerlegungsfaktoren in umgekehrter Reihenfolge neu gebildet (4.3.16). Anders als in Kapitel 3.2, wird im tridiagonalen Fall die Assoziativität des Produkts (4.3.19) ausgenutzt.

$$\mathbf{T}_s - c_s\,\mathbf{I} = \mathbf{Q}_s\,\mathbf{R}_s \tag{4.3.15}$$

$$\mathbf{T}_{s+1} = \mathbf{R}_s\,\mathbf{Q}_s + c_s\,\mathbf{I} \tag{4.3.16}$$

$$= \mathbf{Q}_s^T\,(\mathbf{T}_s - c_s\,\mathbf{I})\,\mathbf{Q}_s + c_s\,\mathbf{I} \tag{4.3.17}$$

$$= \mathbf{Q}_s^T\,\mathbf{T}_s\,\mathbf{Q}_s \tag{4.3.18}$$

$$= \ldots\,(\mathbf{P}_{54}^T\,(\mathbf{P}_{43}^T\,(\mathbf{P}_{32}^T\,(\mathbf{P}_{21}^T\;\mathbf{T}_s\;\mathbf{P}_{21})\;\mathbf{P}_{32})\;\mathbf{P}_{43})\;\mathbf{P}_{54})\;\ldots \tag{4.3.19}$$

$$\mathbf{Q}_s = \mathbf{P}_{21}\;\mathbf{P}_{32}\;\mathbf{P}_{43}\;\mathbf{P}_{54}\;\ldots\;\mathbf{P}_{n,n-1} \tag{4.3.20}$$

Wegen (4.3.19) erzeugt die Transformation mit $\mathbf{P}_{21}$ ein Zusatzelement außerhalb der Tridiagonalform. Dieses Element wird analog zum *bulge chasing*-Prozess in 4.3.1 durch geeignete Ähnlichkeitstransformationen mit Matrizen $\hat{\mathbf{P}}_{m,m-2}$ (4.3.21) schrittweise aus der Matrix ausmultipliziert (Bild 4.8) [13], [43]. Die Indizes der Rotationsmatrizen beziehen sich auf die Position des zu eliminierenden Elements. Die Rotationsmatrizen $\hat{\mathbf{P}}_{m,m-2}$ sind analog zu Abschnitt 4.3.1 belegt.

$$\hat{\mathbf{T}}_{s+1} = \hat{\mathbf{P}}_{n,n-2}^T\,\ldots\,\hat{\mathbf{P}}_{42}^T\,\hat{\mathbf{P}}_{31}^T\;\mathbf{P}_{21}^T\,\mathbf{T}_s\,\mathbf{P}_{21}\;\hat{\mathbf{P}}_{31}\,\hat{\mathbf{P}}_{42}\,\ldots\,\mathbf{P}_{n,n-2} \tag{4.3.21}$$

$$= \hat{\mathbf{Q}}_s^T\,\mathbf{T}_s\;\hat{\mathbf{Q}}_s \tag{4.3.22}$$

$$\hat{\mathbf{Q}}_s = \mathbf{P}_{21}\;\hat{\mathbf{P}}_{31}\,\hat{\mathbf{P}}_{42}\,\ldots\,\mathbf{P}_{n,n-2} \tag{4.3.23}$$

Wegen der Tridiagonalform von $(\mathbf{T}_s - c_s\,\mathbf{I})$ und der damit verbundenen Struktur der Rotationsmatrizen $\mathbf{P}_{i+1,i}$ sind die Vektoren $(\mathbf{Q}_s\,\mathbf{e}_1)$ und $(\mathbf{P}_{21}\,\mathbf{e}_1)$ identisch und damit nur abhängig von der

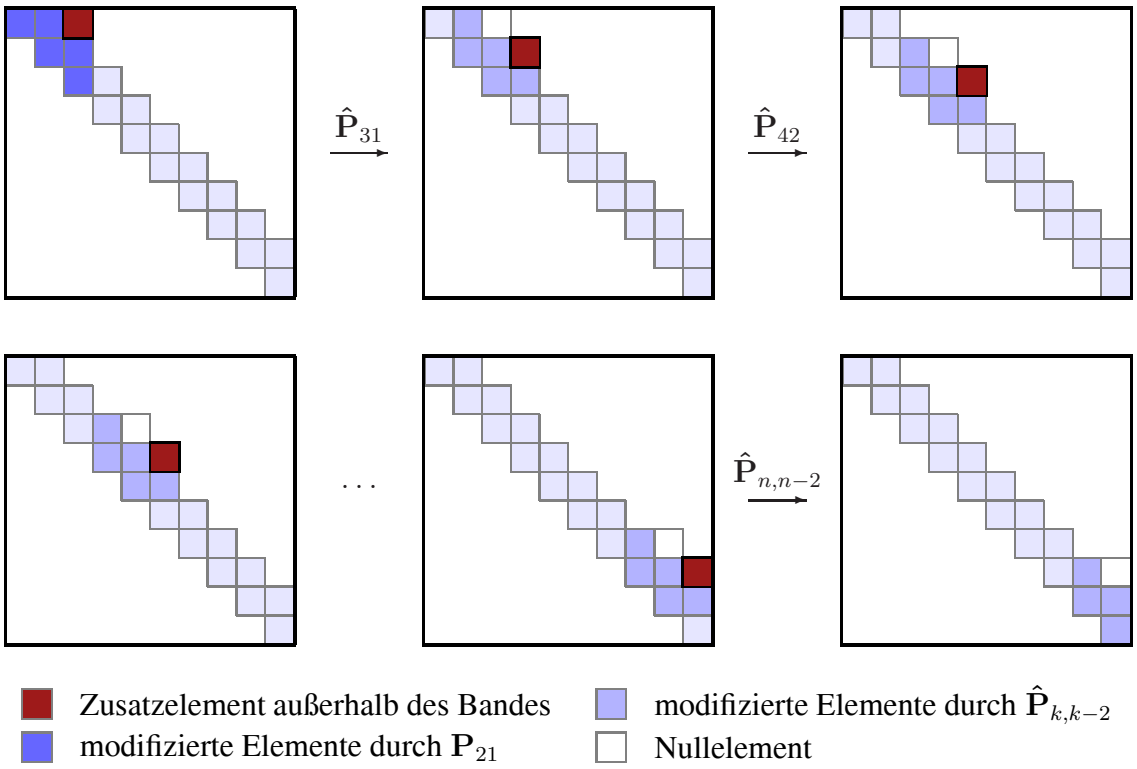


Bild 4.8: *Bulge chasing*-Prozess der symmetrischen Tridiagonalmatrix im Schritt s

ersten Spalte der iterierten Matrix $(\mathbf{T}_s - c_s\,\mathbf{I})$. Damit sind auch die Vektoren $(\mathbf{Q}_s\,\mathbf{e}_1)$ und $(\hat{\mathbf{Q}}_s\,\mathbf{e}_1)$ identisch. In [13] und [43] ist gezeigt, daß die Faktoren $\mathbf{Q}_s$ und $\mathbf{T}_{s+1}$ in Gleichung (4.3.18) im wesentlichen durch die erste Spalte der Matrix $\mathbf{Q}_s$ festgelegt sind und damit im wesentlichen $\hat{\mathbf{Q}}_s$ und $\hat{\mathbf{T}}_{s+1}$ entsprechen.

Mit Satz 3.3.1 ist die QR-Zerlegung der Matrix $\mathbf{T}_s$ wegen einer Phasenmatrix $\mathbf{E} = diag[\pm 1]$ nicht eindeutig. Die orthonormalen Transformationsmatrizen $\mathbf{Q}_s$ und $\hat{\mathbf{Q}}_s$ unterscheiden sich durch eine spaltenweise Skalierung mit den Koeffizienten von $\mathbf{E}$, wobei $e_{11} = 1$. Für die iterierten Matrizen $\mathbf{T}_{s+1}$ und $\hat{\mathbf{T}}_{s+1}$ folgt, daß sie im wesentlichen gleich sind. Der Parameter c_s der Spektralverschiebung wird lediglich zur Bestimmung von $\mathbf{P}_{21}$ benötigt und damit implizit in die Iteration eingetragen.

In [30] ist gezeigt, daß das QR-Verfahren für tridiagonale Matrizen mit impliziter Shifttechnik stets konvergiert. Die Konvergenz der Subdiagonalelemente ist dabei häufig kubisch oder besser [30], [48].

4.3.3 Eigenvektorbestimmung durch Inverse Vektoriteration

Mit den in Abschnitt 4.3.2 bestimmten Eigenwertnäherungen werden die dazugehörigen Eigenvektornäherungen durch inverse Vektoriteration gewonnen. Die Berechnung erfolgt an der Matrix $\mathbf{A}$ der ursprünglichen Eigenwertaufgabe (4.3.1) unter Ausnutzung von Symmetrie und Profilstruktur.

Besitzt $\mathbf{A}$ die Eigendarstellung (4.3.24) und ist regulär, so existiert eine Inverse mit der Eigendarstellung (4.3.25). Die Eigenvektoren von $\mathbf{A}^{-1}$ entsprechen den Eigenvektoren von $\mathbf{A}$. Die Eigenwerte von $\mathbf{A}^{-1}$ sind die reziproken Eigenwerte von $\mathbf{A}$. Damit sind die betragskleinsten Eigenwerte von $\mathbf{A}$, die betragsgrößten Eigenwerte von $\mathbf{A}^{-1}$ und werden durch die Vektoriteration bestimmt. Analog zu Abschnitt 3.4.2 werden beliebige Eigenwerte λ_i von $\mathbf{A}$ durch eine Spektralverschiebung mit einer geeigneten Näherung μ_i zum betragskleinsten Eigenwert der Aufgabe (4.3.26).

$$\mathbf{A} = \mathbf{X}\,\mathbf{\Lambda}\,\mathbf{X}^T \qquad mit\ \ \mathbf{\Lambda} = diag[\lambda_i], \quad |\lambda_1| < \ldots < |\lambda_n| \tag{4.3.24}$$

$$\mathbf{A}^{-1} = \mathbf{X}\,\mathbf{\Lambda}^{-1}\,\mathbf{X}^T \qquad mit\ \ \mathbf{\Lambda}^{-1} = diag[1./\lambda_i] \tag{4.3.25}$$

$$(\mathbf{A} - \mu_i\,\mathbf{I})\,\mathbf{x}_i = \rho_i\,\mathbf{x}_i \qquad mit\ \ \rho_i = \lambda_i - \mu_i \tag{4.3.26}$$

Die Vektoriteration erzeugt ausgehend von einem Startvektor $\mathbf{u}_0 \neq \mathbf{0}$ eine Reihe von Vektoren der Form $(\mathbf{A}^k\mathbf{u}_0)$. In [43], [35], [13] wird die Konvergenz dieser Vektorfolge zum Eigenvektor des betragsgrößten Eigenwerts $max\,|\lambda|$ bewiesen. Die Bestimmung des betragskleinsten Eigenwerts $min\,|\lambda|$ erfolgt mit $\mathbf{A}^{-1}$ und wird deshalb als *Inverse Vektoriteration* bezeichnet.

Im Schritt s der Iteration wird der iterierte Vektor $\mathbf{u}_s$ durch Lösen des Gleichungssystems (4.3.27) bestimmt. Der iterierte Vektor $\mathbf{u}_s$ wird auf Einheitslänge normiert. Eine Näherung im Schritt s zum betragskleinsten Eigenwert der Aufgabe (4.3.26) wird mit dem Rayleigh-Quotient $\rho(\mathbf{A}, \mathbf{u}_s)$ bestimmt.

$$\mathbf{A}\,\mathbf{u}_s = \mathbf{u}_{s-1} \tag{4.3.27}$$

$$\lim_{s\to\infty} \mathbf{u}_s = \mathbf{x}_i \qquad mit\ \ \|\mathbf{x}_i\|_2 = 1.0 \tag{4.3.28}$$

$$\lim_{s\to\infty} \rho_s = \rho_i \tag{4.3.29}$$

Die Eigenwertnäherung μ_i sei mit einem Fehler $\delta(\approx \varepsilon\|\mathbf{A}\|_2)$ behaftet, so daß $\mu_i + \delta = \lambda_i$. Der Startvektor $\mathbf{u}_0$ der Iteration sei beliebig mit $\|\mathbf{u}_0\|_2 = 1.0$. Die iterierten Vektoren $\mathbf{u}_s$ werden in jedem Schritt durch einen Faktor m_s so normiert, daß $max(u_j) = 1.0\ (j = 1, \ldots, n)$. Bei ausreichend kleinem Fehler δ ist der iterierte Vektor $\mathbf{u}_1$ bereits nach dem ersten Iterationsschritt eine akzeptable Näherung zum gesuchten Eigenvektor $\mathbf{x}_i$. Der Startvektor wird als Linearkombination der Eigenvektoren der Aufgabenstellung (4.3.1) dargestellt.

$$\mathbf{u}_0 = \sum_{k=1}^{N} c_k\,\mathbf{x}_k \tag{4.3.30}$$

$$\tag{4.3.31}$$

Für den ersten Iterationsschritt folgt mit (4.3.30) :

$$\mathbf{u}_1 = m_1 \left(\mathbf{A} - \mu_i \mathbf{I}\right)^{-1} \mathbf{u}_0 \tag{4.3.32}$$

$$= m_1 \left(\mathbf{A} - \mu_i \mathbf{I}\right)^{-1} \sum_{k=1}^{N} c_k \mathbf{x}_k \tag{4.3.33}$$

$$= m_1 \left(\frac{c_1}{\lambda_i - \mu_i} \mathbf{x}_i + \sum_{\substack{k=1 \\ k \neq i}}^{N} \frac{c_k}{\lambda_k - \mu_i} \mathbf{x}_k \right) \tag{4.3.34}$$

$$= \frac{m_1 c_i}{\lambda_i - \mu_i} \left(\mathbf{x}_i + \sum_{\substack{k=1 \\ k \neq i}}^{N} \frac{c_k}{c_i} \frac{\lambda_i - \mu_i}{\lambda_k - \mu_i} \mathbf{x}_k \right) \tag{4.3.35}$$

Ist μ_i eine gute Näherung zu λ_i, dann folgt mit Gleichung (4.3.35), daß der iterierte Vektor $\mathbf{u}_1$ eine gute Näherung zum gesuchten Eigenvektor $\mathbf{x}_i$ ist. Wegen $\mu_i \approx \lambda_i$ ist Gleichung (4.3.32) jedoch fast singulär. Die Lösung $\mathbf{u}_1$ ist deswegen anfällig gegen kleine Störungen der Matrix $(\mathbf{A} - \mu \mathbf{I})$. In [48] und [30] wird die schlechte Konditionierung des Gleichungssystems (4.3.32) diskutiert und gezeigt, daß im Falle getrennter Eigenwerte ein Fehler infolge endlicher Zahlendarstellung im Rechner fast vollständig in der Richtung des gesuchten Eigenvektors $\mathbf{x}_i$ liegt, falls λ_i von den restlichen Eigenwerten $\lambda_k, (k \neq i)$ gut getrennt ist . Für schlecht getrennte Eigenwertcluster bleibt das Problem der Sensitivität des Gleichungssystems erhalten und wird häufig durch eine geringfügige Störung der Eigenwertnäherung μ_i mit $\delta \approx \varepsilon$ verbessert, aber nicht in jedem Fall gelöst [10].

Bei der Bestimmung einer größeren Anzahl von Eigenvektoren ist der Orthogonalitätsverlust unter den Eigenvektornäherungen die maßgebliche Schwierigkeit, die es zu beheben gilt. Trotz einer teilweise sehr kleinen Restnorm für die mit den Näherungen $\hat{\lambda}_k$ bestimmten Eigenvektornäherungen $\hat{\mathbf{x}}_k$ kann, insbesondere im Fall schlecht getrennter Eigenwerte, der Orthogonalitätsverlust zwischen den Vektoren erheblich sein. Die zuverlässige Bestimmung einer orthonormalen Subraumbasis bedarf dann einer Orthogonalisierung sowohl des Startvektors, als auch des iterierten Vektors zu den bereits bestimmten Eigenvektoren. Häufig reicht dies jedoch nicht aus, um ein Orthogonalitätslevel im Bereich der Darstellungsgenauigkeit ϵ des Rechners zu garantieren. Eine systematische Untersuchung des Orthogonalitätsverlusts wird in [42] vorgenommen. Für die Bestimmung einer größeren Anzahl von Eigenzuständen bedeutet die Wiederherstellung der Orthogonalität zwischen den berechneten Eigenvektornäherungen einen erheblichen numerischen Mehraufwand, der sich maßgeblich auf die Effizienz des Verfahrens auswirkt.

4.3.4 Komplexität des Givens-QRI-Verfahren

Der Gesamtspeicheraufwand des vorgestellten Verfahrens ist konstant und wird im wesentlichen durch die Bandbreite b und die Anzahl m berechneter Eigenvektornäherungen bestimmt.

Gesamtspeicherbedarf (double): $N\,(b+1+m)$

Der multiplikative Aufwand des *GQRI*-Verfahrens wird für die drei Teilschritte des Verfahrens bestimmt.

***Givens* Reduktion** Die Anzahl zu eliminierender Elemente innerhalb des Matrixbandes beträgt $(b-1)\,(N-\frac{1}{2}b-1)$. Nullelemente infolge einer Profilstruktur, wie sie in Bild 4.4 gezeigt sind, werden im allgemeinen während der schrittweisen Reduktion auf Tridiagonalform überschrieben und werden damit Teil des nachfolgenden Reduktionsprozesses.

Das im schrittweisen Reduktionsprozess jeweils generierte Zusatzelement bewegt sich pro Transformation des *bulge chasing*-Prozess um b Positionen nach unten. Pro Zeile sind $(b-1)$ Elemente zur Bandbreitenreduktion zu eliminieren. Die Anzahl Rotationen N_R wird damit durch die Summe (4.3.36) festgelegt :

$$N_R \;=\; \sum_{i=b}^{N} \frac{(b-1)}{b}\,i \;=\; \frac{1}{2}\,\frac{(b-1)}{b}\,(N\,(N+1)-b\,(b+1)) \tag{4.3.36}$$

Jede Rotation erfordert unter Ausnutzung der Symmetrie 8 Multiplikationen im Bereich $(b+1)$.

multiplikativer Aufwand $(Q^T A\,Q = T)$ *:* $\sim 4\,\frac{(b^2-1)}{b}\;(N\,(N+1)-b\,(b+1))$

QR Tri-Verfahren Ein Iterationszyklus des impliziten QR-Verfahrens erfordert ca. $10\,N$ Multiplikationen. Pro Eigenwert sind im allgemeinen nicht mehr als 2 Iterationszyklen erforderlich. Wegen der schrittweisen Deflation der Matrix verringert sich der Aufwand nochmals um den Faktor $\frac{1}{2}$. Eine vollständige Bestimmung der Eigenwerte erfordert im allgemeinen ca. $9\,N^2$ Multiplikationen [3],[30].

multiplikativer Aufwand $(U^T T\,U = D)$ *:* $\sim 9\;N^2$

Inverse Vektoriteration Jede Eigenvektornäherung erfordert eine Zerlegung der spektralverschobenen Profilmatrix $(\mathbf{A}-\hat{\lambda}_i\mathbf{I})$ mit $(\frac{1}{2}\,N\,b^2)$ Multiplikationen. Im Schnitt werden für ausreichende Konvergenz $2-3$ Iterationsschritte erfordert. In jedem Iterationsschritt erfordert die Lösung des Gleichungssystems (4.3.27) $(N\,b)$ Multiplikationen. Der Orthogonalisierungsaufwand für m Vektoren beträgt ungefähr $(2\,m^2\,N)$ Multiplikationen.

multiplikativer Aufwand : $\sim 2\,N\,(m^2+m\,(\frac{1}{4}\,b^2+b))$

Operation	Berechnung	multiplikativer Aufwand	Speicherbedarf
1	$\mathbf{T}_{(N\times N)} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$	$4\,\frac{(b^2-1)}{b}\;(N\,(N+1) - b\,(b+1))$	$N\,(b+1)$
2	$\mathbf{T}_{(N\times N)} = \mathbf{X}\mathbf{D}\mathbf{X}^T$	$\sim 9\,N^2$	
3		$\sim N^2\,(4\,b+9) + O(bN)$	
4	$\mathbf{u}_i = (\mathbf{A} - \mu_k\,\mathbf{I})^{-1}\,\mathbf{u}_{i-1}$	$\sim 2\,N\,(m^2 + m\,(\frac{1}{4}\,b^2 + b))$	$N\,(b+m)$

1 Givens-Tridiagonalisierung von $\mathbf{A}$
2 QR-Verfahren mit impliziter Shifttechnik
3 Gesamtaufwand (Eigenwerte)
4 Vektoriteration für m Eigenvektoren

N Dimension von $\mathbf{A}$
b mittlere Bandbreite von $\mathbf{A}$
m Anzahl Eigenvektoren

Tabelle 4.3: Komplexität des *Givens-QRI*-Verfahren

4.4 Vergleich der Verfahren

4.4.1 Konvergenzeigenschaften

Inverse Matrixiteration Das Konvergenzverhalten der Inversen Matrixiteration wurde in Kapitel 3 behandelt.

Zu den wesentlichen Eigenschaften des Verfahrens zählt die schrittweise Konvergenz der Eigenwerte, beginnend mit dem Betragskleinsten. Diese Eigenschaft ermöglicht den Einsatz einer Spektralverschiebung zur Konvergenzbeschleunigung, aber auch zur selektiven Bestimmung einzelner Eigenwerte oder zur Bestimmung der Eigenwerte eines vorgegebenen Intervalls. Für die Bestimmung der Eigenwerte eines vorgegebenen Intervalls hat es sich als zweckmäßig erwiesen, als initialen Verschiebungsparameter die Intervallmitte zu wählen. Eine streng sortierte Bestimmung der Eigenwerte ist damit nicht mehr erforderlich, da die Konvergenz von Eigenwerten zu beiden Intervallgrenzen hin gewünscht ist. Damit wird auch eine kontinuierliche Anpassung des Verschiebungsparameters, sowie der Einsatz der Prekonditionierung unproblematisch. Sind die Eigenwerte zu einer der Intervallgrenzen hin vollständig bestimmt, wird die Iteration mit einer Verschiebung in die Mitte des verbleibenden, noch nicht bestimmten Teilintervalls fortgeführt. Diese Strategie wird wiederholt, bis alle Eigenwerte des Intervalls bestimmt sind. Abschließend werden die Eigenwerte des Intervalls auf Vollständigkeit überprüft. Im allgemeinen bestimmt das Verfahren k Eigenwerte eines Intervalls in $\leq 2k$ Zyklen.

1	2	3	4
$\hat{\lambda}$	$\|\mathbf{A}\hat{\mathbf{x}} - \hat{\lambda}\hat{\mathbf{x}}\|_2$	$\|\hat{\lambda} - \rho\|$	Zyklus
509623.9719396392	$2.9633e-06$	$1.8568e-08$	8
509623.9719396470	$2.6323e-06$	$1.8859e-08$	8
518093.3160390588	$1.4386e-05$	$2.4331e-08$	8
518093.3160390608	$2.3044e-06$	$1.5134e-08$	8

Tabelle 4.4: Aufwand zur Bestimmung einer beliebigen Teilmenge aus $\sigma(\mathbf{A})$

Tabelle 4.4 zeigt Berechnungsergebnisse für die Matrix K10073. Berechnet wurden die Eigenwerte im Intervall $[5.0e+05, 5.2e+05]$. Die Anzahl der Eigenwerte im Intervall wurde vorab mit der Sturmschen Kette ermittelt. Spalte 1 zeigt die Eigenwertnäherungen des Intervalls. Spalte 2 gibt anhand des Gleichgewichtsfehlers Aufschluß über die Genauigkeit der Lösung. Spalte 3 enthält die Differenz der Eigenwertnäherung und des Rayleigh-Quotienten und Spalte 4 enthält den Zyklus der Konvergenz des Eigenwerts. Alle vier Eigenwerte des Intervalls wurden im selben Zyklus bestimmt. Das Intervall enthält zwei Eigenwerte mit Multiplizität 2. Die Eigenwerte stimmen auf 6 Nachkommastellen überein. Die Restnorm ist um den Faktor $1e-04$ kleiner als $(\epsilon\|\mathbf{A}\|)$.

Die Bestimmung einer kleinen Anzahl der betragsgrößten Eigenwerte von $\mathbf{A}$ ist durch die Spektralverschiebung grundsätzlich möglich, aber häufig mit hohem numerischem Aufwand verbunden. Als Verschiebungsparameter zu Beginn der Iteration wird die Spur $sp(\mathbf{A})$ gewählt. Als eine der Invarianten der Ähnlichkeitstransformation ist sie eine sichere Größe zur Verschiebung des Eigenwertspektrums $\sigma(\mathbf{A})$, so daß der betragsgrößte Eigenwert von $\mathbf{A}$ zum betragskleinsten Eigenwert der verschobenen Matrix wird. Eine vorsichtige Anpassung des Verschiebungsparameters kann nach wenigen Zyklen erfolgen. Im allgemeinen bestimmt das Verfahren eine kleine Anzahl k betragsgrößter Eigenwerte in $(< 5k)$ Zyklen. Teilweise hat sich die Bestimmung der betragsgrößten Eigenwerte aber als nicht vollständig zuverlässig erwiesen. Das gesuchte Teilspektrum wurde in einigen Fällen nicht vollständig bzw. mit einem nicht akzeptablem Aufwand bestimmt. Ein wesentlich besseres Konvergenzverhalten ist zu erwarten, wenn die Intervallgrenzen für das gesuchte Teilspektrum am oberen Rand des Eigenwertspektrums als Näherung bekannt sind.

Implicitly Restarted Lanczos Die Stärke der Lanczos-Verfahren liegt in der Bestimmung einer kleinen Anzahl von Eigenzuständen großer Profilmatrizen.

Das IRL-Verfahren bestimmt sowohl die betragsgrößten als auch die betragskleinsten Eigenwerte an der ursprünglichen Matrix $\mathbf{A}$ der speziellen Eigenwertaufgabe. Durch den Neustart, verbunden mit der impliziten Shift-Technik, werden nicht gewünschte Eigenwerte schrittweise durch einen *purging*-Prozess aus der Iteration entfernt. Es zeigt sich aber, daß bei der Bestimmung der betragskleinen Eigenwerte die Dominanz der betragsgroßen Eigenwerte das Konvergenzverhalten wesentlich mitbestimmt. Der *purging*-Prozess muß teilweise für eine große Anzahl betragsgroßer Eigenwerte mehrfach erfolgen, da sie wiederholt im Eigenwertspektrum der Iteration auftauchen. Eine Verifizierung der Berechnungsergebnisse und des Konvergenzverhaltens der vorliegenden Implementierung wurde stichprobenhaft mit der Originalsoftware (ARPACK [19]) der Entwickler dieses Verfahrens vorgenommen. Eine Reihe von Berechnungen an großen Matrizen haben gezeigt, daß die Bestimmung der betragskleinsten Eigenwerte mit der Matrix $\mathbf{A}$ als Iterationsmatrix in vielen Fällen nicht vollständig möglich ist bzw. die Anzahl der Restarts eine akzeptable Grenze (~ 300) deutlich überschreitet. Deshalb wurde in allen Berechnungen die nachfolgend erläuterte Strategie eingesetzt.

Konvergenzuntersuchungen verschiedener Autoren zeigen, daß das Verfahren im allgemeinen sehr schnell zu den Eigenzuständen mit extremalen Eigenwerten konvergiert, wogegen Eigenwerte im Innern des Gesamtspektrums nur mäßig und teilweise gar nicht approximiert werden. [17],[37]. Dieser Umstand kann durch eine Berechnung im *shift & invert mode* behoben werden. Anstelle von $\mathbf{A}$ operiert die Iteration im *shift & invert mode* auf $(\mathbf{A} - \rho\mathbf{I})^{-1}, \rho \in \mathbb{R}$ und bestimmt die Näherungen $\frac{1}{\lambda-\rho}, \lambda \in \sigma(\mathbf{A})$. Diese Berechnungsstrategie wird häufig zur Konvergenzverbesserung der Iteration eingesetzt. Die betragskleinsten Eigenwerte werden in diesem Berechnungsmodus mit $\rho = 0$, mit deutlich besserem Konvergenzverhalten approximiert.

Die Konvergenz des IRL-Verfahrens wird neben der Wahl des Startvektors maßgeblich von der eingesetzten Subraumgröße $(k + p)$ beeinflußt. Das Verfahren liefert gute Ergebnisse für $p \geq k$, falls eine nur kleine Anzahl von Eigenzuständen gesucht wird. Mit $p \sim k$ ist das Verfahren

weitgehend insensitiv hinsichtlich der Wahl von p. Mit ansteigender Größe des gesuchten Teilspektrums, kann die strenge Wahl von $p \sim k$ stark entspannt werden.

Da für keine der Berechnungen geeignete a priori-Informationen vorlagen, wurde der Startvektor der Iteration mit Zufallszahlen aus dem Intervall $[-1, +1]$ belegt und anschließend auf die Länge 1.0 normiert.

Givens-QRI-Verfahren Das vorgestellte Givens-QRI-Verfahren bestimmt alle Eigenwerte und ermöglicht eine unabhängige, selektive Eigenvektorbestimmung. Ist die Anzahl der Eigenzustände a priori bekannt und klein, so werden die Eigenwerte an der Tridiagonalmatrix mit einem numerisch günstigeren Verfahren, z.B. dem Bisektionsverfahren, bestimmt. Die vollständige Bestimmung der Eigenwerte mit dem QR-Verfahren ist infolge der impliziten Shifttechnik sehr effizient. Die Konvergenz dieser Iteration ist am allgemeinen kubisch und besser [13],[30].

Die Konvergenz der inversen Vektoriteration zur Bestimmung der Eigenvektornäherungen ist mit einer guten Näherung, zu einem vom restlichen Spektrum gut getrennten Eigenwert, ebenfalls sehr gut. Im allgemeinen genügen zwei Iterationen für eine ausreichende Genauigkeit der Lösung. Allerdings treten Fälle auf, die auch mit einer Vielzahl von Iterationen zu keiner zufriedenstellenden Eigenvektornäherung führen. Die etablierten Software-Pakete *EISPACK* und *LAPACK* für Aufgabenstellungen der Linearen Algebra in Forschung und Wissenschaft bieten derzeit keine Implementierungen, die uneingeschränkte Sicherheit bei der Eigenvektorberechnung garantieren.

In [10] ist gezeigt, daß eine falsche Zuordnung der Eigenwertnäherung zum exakten Eigenwert zu Eigenvektornäherungen mit großer Restnorm führen und damit nicht brauchbar sind. Liegt mehr als ein Eigenwert nahe des Verschiebungsparameters $\hat{\lambda}_m$ zur Berechnung der Eigenvektornäherung $\hat{\mathbf{x}}_m$, so ist das zu lösende Gleichungssystem stark sensitiv gegenüber Störungen in $\hat{\lambda}_m$ und wird deshalb meist künstlich gestört. Eine zu große Störung jedoch verschlechtert die Genauigkeit der Lösung erheblich.

Werden mehrere Eigenvektoren mit der inversen Vektoriteration bestimmt, besteht die Gefahr des Orthogonalitätsverlustes unter den Eigenvektornäherungen. Dies ist insbesondere im Fall schlecht getrennter Eigenwerte zu beobachten. In [42] ist gezeigt, daß eine Reorthogonalisierung der iterierten Vektoren gegen alle bereits bestimmten Eigenvektornäherungen nicht immer erfolgreich durchgeführt werden kann. Häufig ist im Falle schlecht getrennter Eigenwerte der zu orthogonalisierende Vektor fast parallel zu bereits vorhandenen Vektoren. Der Reorthogonalisierungsprozess verursacht eine Auslöschung signifikanter Richtungen im iterierten Vektor und damit auch ein Scheitern der Orthogonalisierung. Das Ergebnis der Reorthogonalisierung ist ein nahezu zufälliger Vektor, trotz einer kleinen Restnorm.

Eine ausführliche Analyse verschiedener Ursachen die zum Scheitern der Eigenvektorbestimmung mit der Inversen Vektoriteration führen können, wird in [42], [11] und [10] vorgenommen.

Die Reorthogonalisierung erfolgt im allgemeinen durch eine Gram-Schmidt-Orthogonalisierung. Für einen praxistauglichen Einsatz erfordert die Software einen stabilen Reorthogonalisierungsmechanismus und eine stete Überwachung des Orthogonalitätslevels der Eigenvektoren. Für eine

größere Anzahl von Eigenvektoren bedeutet dies einen erheblichen numerischen Mehraufwand. Weit mehr wiegt aber der Umstand, daß alle Eigenvektornäherungen für eine Reorthogonalisierung im Arbeitsspeicher verfügbar sein müssen. Für die vollständige Eigenvektorbestimmung bedeutet dies ein zusätzlicher Speicheraufwand der Ordnung $O(N^2)$.

In [42], [11] und [30] wird eine blockweise Reorthogonalisierung für schlecht getrennte Eigenwertcluster vorgeschlagen. Die Eigenwertcluster werden durch eine empirisch gewählte Schranke identifiziert. Ist eine neue Eigenvektornäherung zu einem Eigenwert des Clusters bestimmt, wird dieser zu allen bereits berechneten Eigenvektornäherungen dieses Clusters reorthogonalisiert. Der numerisch teure Orthogonalisierungsprozess wird so auf im allgemeinen wenige Vektoren eines Clusters beschränkt. Die Wahl einer geeigneten Schranke zur Identifikation des Clusters birgt jedoch weitere Gefahren :

Ist das Kriterium zur Identifikation eines engen Eigenwertclusters zu großzügig, steigt der Orthogonalisierungsaufwand erheblich an. Die Anzahl Eigenvektornäherungen, die im Arbeitsspeicher vorgehalten werden müssen führen dann infolge eines Speicherüberlaufs möglicherweise zum Programmabbruch. Wird das Kriterium für die Eigenwertcluster hingegen zu streng gewählt, so liegen aufeinander folgende Cluster so eng beieinander, daß die Orthogonalität zwischen den Clustern verloren geht.

Der effektive Einsatz der Inversen Vektoriteration beschränkt sich auf eine kleine bis moderate Anzahl von Eigenvektornäherungen.

Bewertung der Konvergenzeigenschaften

Die Bestimmung der betragskleinsten Eigenwerte ist in allen drei Implementierungen sehr zuverlässig. Dies ist ein wesentlicher Berechnungsfall für die Aufgabenstellungen im Bauwesen. Die Bestimmung von Eigenwerten an der spektralverschobenen Matrix $(\mathbf{A} - \lambda\mathbf{I})$ ist sowohl mit der Inversen Matrixiteration, als auch mit dem Lanczos-Verfahren zuverlässig möglich. In beiden Fällen steigt jedoch der Aufwand, da wiederholt unerwünschte Eigenwertnäherungen der Ränder des Gesamtspektrums in den Berechnungen auftauchen. Der Einfluß dieser dominanten Eigenzustände ist in beiden Fällen nur schwer zu dämpfen.

Mehrfache Eigenwerte sind für die Inverse Matrixiteration unproblematisch. Durch die vorgestellten Erweiterungen des Verfahrens werden mehrfache Eigenwerte und schlecht getrennte Eigenwerte häufig im selben Zyklus bestimmt. Die Konvergenz des Verfahrens ist stetig und verbessert sich mit zunehmender Anzahl bestimmter Eigenwerte.

Das vorgestellte Lanczos-Verfahren verfügt über einen Deflationsmechanismus der mehrfache Eigenwerte zuverlässig ermittelt. Für eine kleine Anzahl von Eigenzuständen wird das Konvergenzverhalten sehr stark von einer Reihe von ad-hoc-Entscheidungen geprägt. Die Wahl des Startvektors, die Wahl der Subraumgröße und die Wahl der Konvergenzschranken sind einige Größen, die eine zuverlässige Prognose über die Anzahl der Iterationen verhindert. Ein Teil dieser a priori Informationen verliert mit zunehmender Subraumgröße, d.h. mit ansteigendem Informationsgehalt der Projektion der ursprünglichen Aufgabe auf einen Subraum, an Bedeutung. Zur

Festlegung geeigneter Eingangsgrößen in die Berechnung ist jedoch häufig eine Wiederholung der Berechnung mit variierten Einganggrößen erforderlich. Zur Verifizierung der Berechnungsergebnisse sind zusätzliche Maßnahmen in jedem Fall erforderlich.

Das Givens-QR-Verfahren bestimmt zuverlässig das vollständige Eigenwertspektrum. Eine sukzessive Bestimmung aufeinanderfolgender Eigenwerte ist jedoch mit dem QR-Verfahren für tridiagonale Matrizen nicht zuverlässig möglich. Die Konvergenzreihenfolge ist zufällig. Die in der Literatur [39], [49] vorgeschlagenen Abbruchschranken der Iteration haben sich als nur bedingt zuverlässig erwiesen. Eine Anpassung der Konvergenzschranken an die berechneten Tridiagonalmatrizen war teilweise erforderlich um akzeptable Berechnungsergebnisse bei akzeptablem Aufwand zu garantieren.

Die Bestimmung der Eigenvektoren mit der inversen Vektoriteration ist nicht uneingeschränkt zuverlässig. Schlechte Eigenwertnäherungen und enge Eigenwertcluster verhindern teilweise eine Eigenvektorbestimmung mit ausreichender Genauigkeit bzw. eine zuverlässige Näherung zum gesuchten Eigenvektor. Das Orthogonalitätslevel der Eigenvektoren muß explizit sichergestellt werden. Die inverse Matrixiteration reagiert sehr ähnlich auf enge Eigenwertcluster, ist jedoch insgesamt stabiler und zuverlässiger in der Eigenvektorbestimmung. Fehler infolge einer falschen Zuordnung der Eigenwertnäherung zum exakten Eigenwert sind im Vergleich zur inversen Vektoriteration unproblematisch, da sehr engliegende Cluster vollständig mit einer Zerlegung oder falls erforderlich durch eine zusätzliche Zerlegung bestimmt werden und damit alle problematischen Eigenpaare gemeinsam erfassen. Eine Störung von Eigenwertnäherungen zur Vermeidung gleicher Eigenvektoren ist im allgemeinen nicht erforderlich, da mit Hilfe der stabilen QR-Zerlegung und der in 3.4.3 vorgestellten stabilen lokalen Iteration die Multiplizität von Eigenwertnäherungen bis zum kleinsten erreichbaren Fehlerniveau sicher bestimmt werden kann. Die Eigenvektorbestimmung kann selektiv über beliebig große Subräume erfolgen, da die berechneten Eigenvektornäherungen nicht weiter im Arbeitsspeicher vorgehalten werden müssen.

4.4.2 Speicherbedarf

Inverse Matrixiteration Der Speicherbedarf zur Bestimmung der Eigenwerte ist über den Iterationsverlauf konstant. Das Verfahren kann die Symmetrie und eine dünne Koeffizientenbesetzung der Matrix nicht ausnutzen. Durch die Ausnutzung eines konvexen Matrixprofils bleibt jedoch der Speicherbedarf für die üblichen Bandbreiten der Aufgabenstellungen im Bauwesen auch für große Matrizen gut. Der zusätzlich erforderliche Speicher zur Erzeugung eines konvexen Matrixprofils ist vernachlässigbar. Eine zweckmäßige Elementierung des Lösungsgebietes der Aufgabenstellung vorausgesetzt, bleibt der zusätzlich erforderliche Speicher im allgemeinen $\leq 0.5\%$.

Da die Bestimmung der Eigenwerte unabhängig von den Eigenvektoren erfolgt, wird die Anzahl der gesuchten Eigenzustände nicht durch den Speicherbedarf begrenzt. Für die selektive Berechnung der Eigenvektoren ist eine Kopie der ursprünglichen Matrix erforderlich, die infolge der kompakten Speicherungsstruktur auch für große Matrizen im Arbeitsspeicher gehalten werden kann. Entscheidend für den Erfolg der Methode ist jedoch, daß im Vergleich zu den Subraum-Verfahren keine speicherintensive Transformationsbasis in den Berechnungsablauf eingeht.

Implicitly Restarted Lanczos Der Speicherbedarf des IRL-Verfahrens wird von der zur Iteration verwendeten Subraumgröße $(k+p)$ dominiert. Die Matrix der Aufgabe kann selbst im *shift & invert mode* durch Ausnutzung von Symmetrie und einer allgemeinen Profilstruktur kompakt gespeichert werden. Allerdings ist die ursprüngliche Matrix für nachfolgende Berechnungsschritte durch die Zerlegung zur Lösung des Gleichungssystems zerstört und bedarf einer zusätzlichen Kopie.

Mit zunehmender Dimension der Aufgabe entscheidet die Subraumgröße $(k + p)$ über den erfolgreichen Einsatz der Methode. Mit näherungsweise $(k + p) = b, (b =$ mittlere Bandbreite$)$ ist der Speicherbedarf des IRL-Verfahrens und der Inversen Matrixiteration zur Bestimmung der Eigenwerte äquivalent. Das IRL-Verfahrens bestimmt jedoch nur ca. $50\% - 70\%$ der Eigenwertnäherungen des bestimmten Subraums mit ausreichender Genauigkeit zur Lösung der Eigenwertaufgabe.

Givens-QRI-Verfahren Der Algorithmus zur Reduktion der Profilmatrix auf Tridiagonalgestalt modifiziert die Zeilen und Spalten der Matrix gleichermaßen. Wegen des *bulge-chasing* Prozesses hat sich die gezeigte Diagonal-Speicherform als zweckmäßig erwiesen. Die Ausnutzung einer allgemeinen Profilstruktur ist nicht sinnvoll, da sie die Kontinuität im Algorithmus stört und ohnehin nur für wenige Schritte am Berechnungsanfang genutzt werden kann. Das Verfahren nutzt die Symmetrie der Matrix. Der Speicheraufwand wird durch die Bandbreite der Matrix bestimmt. Die ursprüngliche Matrix wird durch die Tridiagonalmatrix des zweiten Berechnungsschrittes überschrieben und bedarf einer Kopie für die nachfolgende Eigenvektorberechnung.

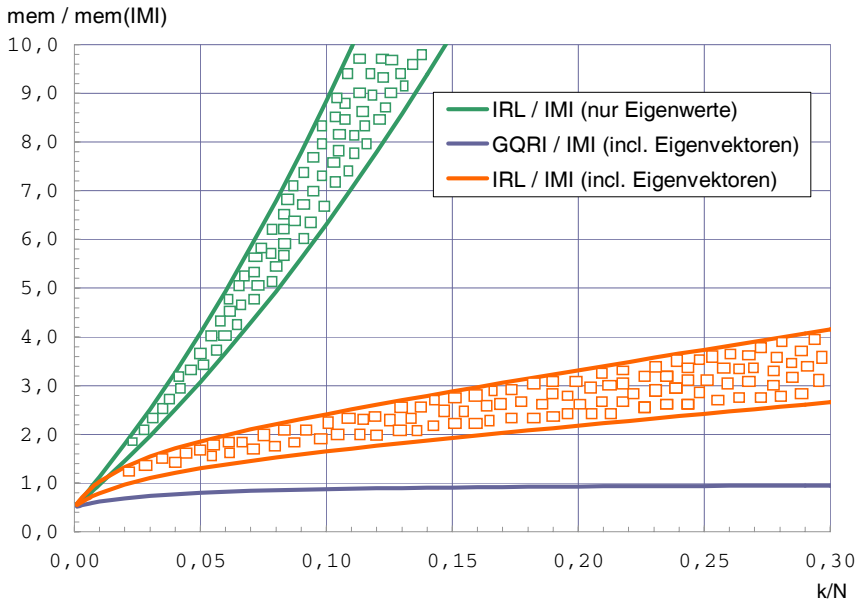


Bild 4.9: Speicherbedarf

Bewertung des Speicherbedarfs

Bild 4.9 zeigt den Speicherbedarf der Vergleichsverfahren (mem) bezogen auf den Speicherbedarf der Inversen Matrixiteration (mem(IMI)) für die Matrizen K10687, K10827 und K10073. Vergleichsrechnungen mit Matrizen unterschiedlicher Dimension und Bandbreite zeigen, daß die gewählten Matrizen repräsentativ für eine Bewertung sind. Die Abszisse zeigt die bestimmte Subraumgröße k bezogen auf die Dimension N der Aufgabe.

Der Speicherbedarf des Lanczos-Verfahrens streut infolge der unterschiedlichen Wahl einer geeigneten Subraumgröße. Die verwendeten Subraumgrößen variieren zwischen $1.5\,k$ und $2.0\,k$ (k = Anzahl gesuchter Eigenzustände). In grün ist das Verhältnis des Speicherbedarfs zwischen der Inversen Matrixiteration und dem Lanczos-Verfahren unter Vernachlässigung des erforderlichen Speichers der Eigenvektorberechnung gezeigt. In rot dargestellt ist das Speicherverhältnis unter Berücksichtigung der Eigenvektorberechnung. Dieser Fall ist maßgeblich, da beim Lanczos-Verfahren die Eigenvektoren implizit Teil der Eigenwertbestimmung sind. Das Lanczos-Verfahren bestimmt die Eigenvektoren aus der invarianten Subraumbasis der Iteration. Der Speicherbedarf für die Eigenwert- und die Eigenvektorbestimmung ist damit für eine bestimmte Subraumgröße konstant. Der Speicherbedarf der Inversen Matrixiteration steigt linear mit der Anzahl gesuchter Eigenvektoren. In beiden Fällen ist zu erkennen, daß das IRL-Verfahren mit zunehmender Subraumgröße schnell an eine Wirtschaftlichkeitsgrenze stößt. Bereits bei der Berechnung von 10% der Eigenzustände hat sich der Speicherbedarf gegenüber der Inversen Matrixiteration verdoppelt.

Für die Berechnungen aller Untersuchungen standen 512MB Arbeitsspeicher zur Verfügung. Da die Eigenwertanalyse im allgemeinen nur eine Teilaufgabe im Rahmen einer Strukturuntersuchung ist, wurde neben der Speicherung der erforderlichen Systemmatrizen und Systemvektoren auch das Finite-Element-Objektmodell zur Weiterbearbeitung im Arbeitsspeicher vorgehalten. Dieses Vorgehen schränkt den für die Eigenwertberechnung zur Verfügung stehenden Arbeitsspeicher erheblich ein, ist jedoch üblich und zweckmäßig für Ingenieuranwendungen. Die Eigenwertberechnung mit dem IRL-Verfahren war damit für große Matrizen $(N \geq 10000)$ auf ca. 20% der Eigenwerte begrenzt.

Der Speicherbedarf des Givens-QRI-Verfahrens ist durch die Nutzung der Symmetrie der Matrix günstiger als für die Inverse Matrixiteration. Für die Eigenwertbestimmung halbiert sich der Speicherbedarf. Mit der Bestimmung der Eigenvektoren dominieren diese schnell den Speicherbedarf. Bild 4.9 zeigt, daß bei der Bestimmung von ca. 15% der Eigenvektoren der Speicherbedarf der beiden Verfahren äquivalent ist (blau). Beide Verfahren eigenen sich damit auch für die wirtschaftliche Berechnung einer großen Anzahl Eigenzustände von Matrizen großer Dimension.

4.4.3 Operativer Vergleich

Der operative Vergleich wird über den multiplikativen Aufwand der Verfahren geführt.

Inverse Matrixiteration Der Aufwand der Inversen Matrixiteration wird maßgeblich von der mittleren Bandbreite b der Profilmatrix bestimmt. Sowohl in der Zerlegung (QR), als auch in der Rekombination (RQ) ist b im Quadrat enthalten. Das Verfahren kann eine konvexe Profilstruktur der Matrix, nicht aber eine dünne Besetzung innerhalb des Profils vorteilhaft nutzen. Die Bandbreite b entscheidet damit über einen akzeptablen numerischen Aufwand bei der Berechnung der Eigenwertaufgabe großer Dimension. Typische Aufgabenstellungen aus dem Bauingenieurwesen besitzen häufig Bandbreiten, die eine effektive Anwendung des Verfahrens gewährleisten.

Der Hauptaufwand der Iteration liegt mit ca. 75% in der QR-Zerlegung von $\mathbf{A}$. Die schrittweise Deflation von $\mathbf{A}$ reduziert den Aufwand der einzelnen Zerlegungen linear. Alle anderen vorgestellten Erweiterungen reduzieren die Anzahl der Zerlegungen insgesamt. Die zusätzlichen Operationen der Prekonditionierung und der Jacobi-Randkorrektur sind vernachlässigbar gemessen am Aufwand einer Zerlegung. Damit wird der Aufwand des gesamten Iterationsprozesses von der erforderlichen Anzahl an Zerlegungen dominiert.

Bild 4.10 zeigt den multiplikativen Aufwand der vollständigen Eigenwertberechnung der Matrix $K10687$. Die blau dargestellten Quadrate beziehen sich auf die linke Ordinate und erfassen den Gesamtaufwand der Berechnung jeweils nach 100 neu konvergierten Eigenwertnäherungen. Die rot dargestellten Quadrate beziehen sich auf die rechte Ordinate und erfassen den Aufwand für die fortlaufende Berechnung von jeweils 100 Eigenwerten. Beide Aufwandsdarstellungen zeigen klar die Stärken und Schwächen des Verfahrens hinsichtlich des numerischen Aufwands.

Der Aufwand für das erste Drittel des Eigenwertspektrums entspricht näherungsweise dem Aufwand der restlichen zwei Drittel. Als Folge der langsamen Anfangskonvergenz streut der Auf-

wand im ersten Drittel und glättet sich mit zunehmendem Berechnungsfortschritt. Dieses Verhalten ist zu erwarten, da sich zu Beginn der Iteration eine Sortierung der Eigenwertnäherungen lediglich im untersten Drittel der Matrix ausbildet und nur in den letzten Diagonalpositionen, unterstützt durch die Prekonditionierung und die Jacobi-Randkorrektur, festigt. Die Sortierung der Eigenwerte auf der Diagonalen erfordert einen häufigen Zeilentausch im restlichen Matrixbereich. Dadurch stagniert teilweise der globale Konvergenzfortschritt und verursacht den unterschiedlichen Aufwand der Iterationen.

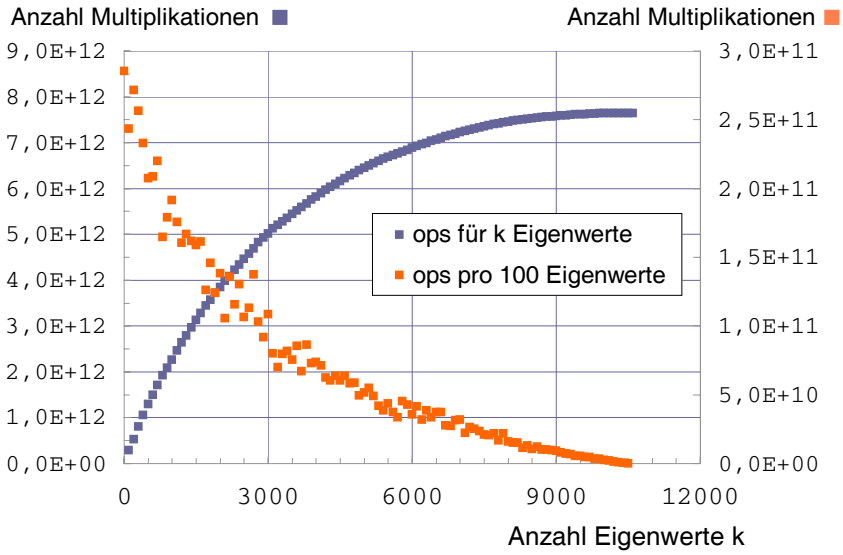


Bild 4.10: Aufwandsverteilung K_{10687}

Bild 4.11 zeigt die Anzahl Eigenwerte in Prozent, die mit k Iterationszyklen bestimmt wurden. Die Ergebnisse der drei Testmatrizen variieren trotz unterschiedlicher Matrixkennzahl nur geringfügig. Annähernd 60% der Eigenwerte werden durch die Berechnung der restlichen Eigenwerte ohne Zusatzaufwand mitbestimmt. Der Knick in den Kurven ebenso wie der starke Abfall der Kurven ist auf die Verbesserung des Verfahrens durch die vorgestellten Erweiterungen zurückzuführen. Die sehr geringen Prozentzahlen für drei und mehr Iterationen bestätigen die Ergebnisse aus Kapitel 3 und deuten auf einen stetigen Konvergenzfortschritt.

Der Aufwand der Zerlegung mit $(6\,b^2\,N)$ ist die maßgebliche Größe bei der Bestimmung der Eigenvektoren. Die Eigenvektorbestimmung ist jedoch unabhängig von der Eigenwertbestimmung. Die Auswahl der zu bestimmenden Eigenvektoren kann selektiv erfolgen. Die vorgestellte Strategie der Eigenvektorbestimmung verringert die Anzahl der Mehrfachzerlegungen und erhöht die Zuverlässigkeit der Ergebnisse.

Implicitly Restarted Lanczos Der numerische Aufwand der Lanczos-Verfahren wird im wesentlichen durch zwei Operationen geprägt.

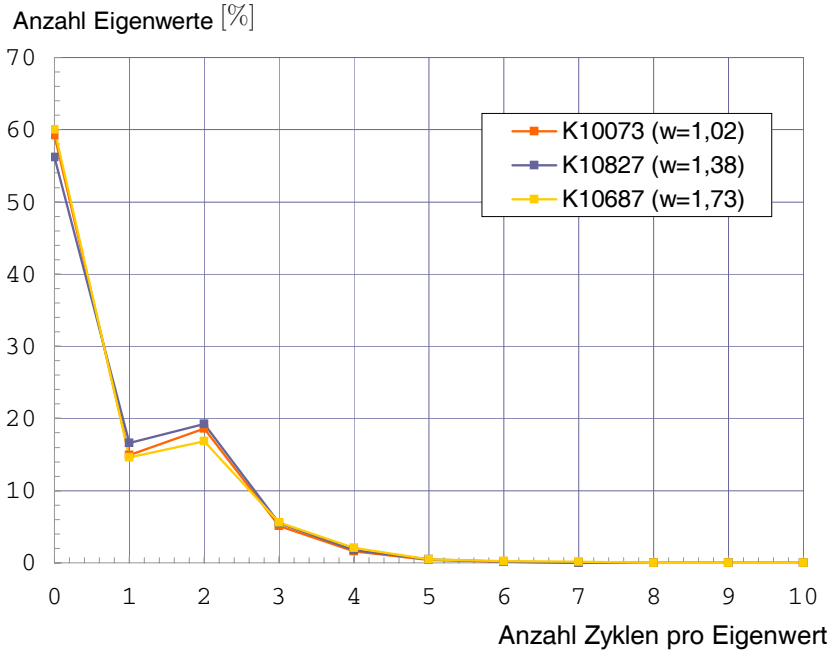


Bild 4.11: Aufwandsverteilung

Als typischer Vertreter der Krylov-Subraummethoden ist der maßgebliche Berechnungsschritt zum schrittweisen Aufbau des Subraums eine Matrix-Vektor-Multiplikation. Das Verfahren ist dabei in der Lage eine allgemeine Profilstruktur, eine dünne Koeffizientenbesetzung und die Symmetrie der Matrix voll auszunutzen. Dieser Umstand reduziert den multiplikativen Aufwand erheblich im Vergleich zur Faktorisierung der Matrix mit QR-basierten Verfahren. Darüber hinaus wird die ursprüngliche Matrix nicht durch Transformationen überschrieben und bleibt für nachfolgende Berechnungsschritte erhalten.

Der zweite wesentliche Berechnungsschritt des Lanczos-Verfahrens bildet die Orthogonalisierung der zum Aufbau des Subraums erzeugten Vektoren. Dieser Vorgang erfolgt durch eine klassische Gram-Schmidt-Orthogonalisierung mit drei Vektoren in jedem Berechnungsschritt. Der maßgebliche Aufwand liegt jedoch weniger in der Orthogonalisierung der iterierten Vektoren, vielmehr ist es der Orthogonalitätserhalt der ständig anwachsenden Subraumbasis. Die erforderliche Reorthogonalisierung der Basisvektoren operiert auf vollbesetzte $(n \times m)$ Matrizen und ist numerisch teuer. Durch den Neustart der Methode wird die Subraumgröße begrenzt, und der numerische Aufwand verringert. Die wirtschaftliche Anwendung des Verfahrens bleibt aber stets auf die Bestimmung einer kleinen bis moderaten Anzahl von Eigenzuständen begrenzt.

Die Anwendung des IRL-Verfahrens auf $(\mathbf{A} - \rho\mathbf{I})^{-1}$ mit $\rho = 0$ zur Berechnung der betragskleinsten Eigenwerte verbessert den Berechnungsaufwand erheblich. Leider verliert das Verfahren damit einige Vorteile hinsichtlich des numerischen Aufwands und des Speicherbedarfs. Die Iteration erfordert eine Zerlegung der Matrix $\mathbf{A}$. Die Matrix-Vektor-Multiplikation des Verfahrens wird durch eine Vorwärts- und Rückwärtslösung des resultierenden Gleichungssystems ersetzt.

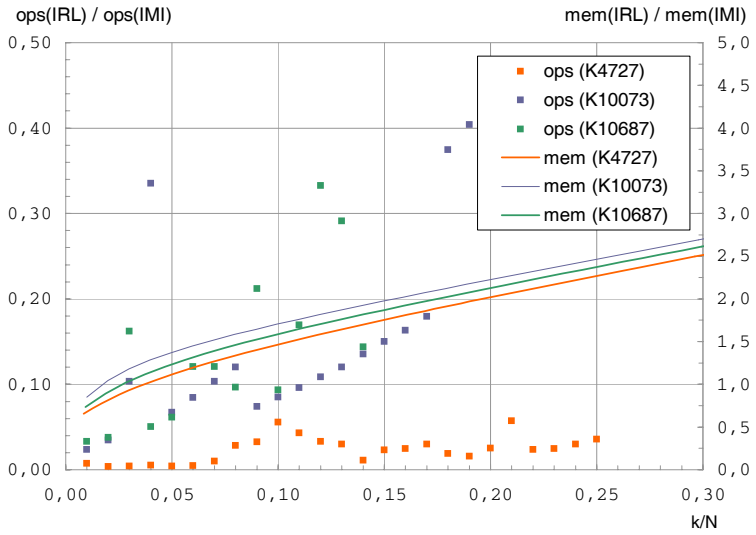


Bild 4.12: Vergleich IRL vs. Inverse Matrixiteration, $(k+p) = 1.5k$

Beides ist mit einem erhöhten numerischen Aufwand verbunden. Die ursprüngliche Matrix der Aufgabe bleibt nicht erhalten und wird durch das Zerlegungsprodukt ersetzt. Die dünne Koeffizientenbesetzung innerhalb des Profils bleibt im allgemeinen nicht erhalten.

Für die berechneten Testmatrizen zeigt sich jedoch, daß der zusätzliche multiplikative Aufwand gerechtfertigt ist im Vergleich zur Verbesserung des Konvergenzverhaltens zu den gesuchten betragskleinsten Eigenwerten.

Um den Einfluß der vorgewählten Subraumgröße auf das numerische Verhalten des Verfahrens aufzuzeigen, wurden alle Berechnungen für k Eigenzustände mit den Subraumgrößen $(k+p) = 1.5k$, $(k+p) = 1.8k$ und $(k+p) = 2.0k$ durchgeführt. Mit zunehmendem k nimmt der Einfluß dieser Wahl ab. Für eine kleine Anzahl zu bestimmender Eigenwerte $(< 2\%)$ wird in [38] und [37] die Wahl von $p \geq k$ vorgeschlagen. Wegen des ansteigenden Speicherbedarfs mit der Subraumbasis ist dies jedoch nicht zweckmäßig und für $k > 0.01N$ auch nicht erforderlich. Die Anzahl erforderlicher Restarts der Lanczos-Iteration sinkt kontinuierlich mit anwachsendem Subraum, da dieser bereits einen Großteil der erforderlichen Eigeninformation enthält.

In den Bildern 4.12 bis 4.14 ist ein operativer Vergleich des IRL-Verfahrens mit der Inversen Matrixiteration für die unterschiedlichen Subraumgrößen gezeigt. Die rechte Ordinate zeigt das Verhältnis des multiplikativen Aufwands von IRL-Verfahren und Inverser Matrixiteration. Die linke Ordinate bezieht sich auf das Verhältnis des Speicherbedarfs der beiden Verfahren. Auf der Abszisse ist der berechnete Anteil des Gesamtspektrums (k/N) aufgetragen.

Der Einfluß der Subraumgröße ist direkt an der abnehmenden Streuung des Aufwands ablesbar. Während bei einer Subraumgröße von $1.5k$ noch mehrere Restarts zur vollständigen Bestimmung des gesuchten Teilspektrums erforderlich sind, werden die Eigenzustände bei einer Subraumgröße von $1.8k$ bzw. $2.0k$ mit fast konstanter Anzahl Restarts bestimmt.

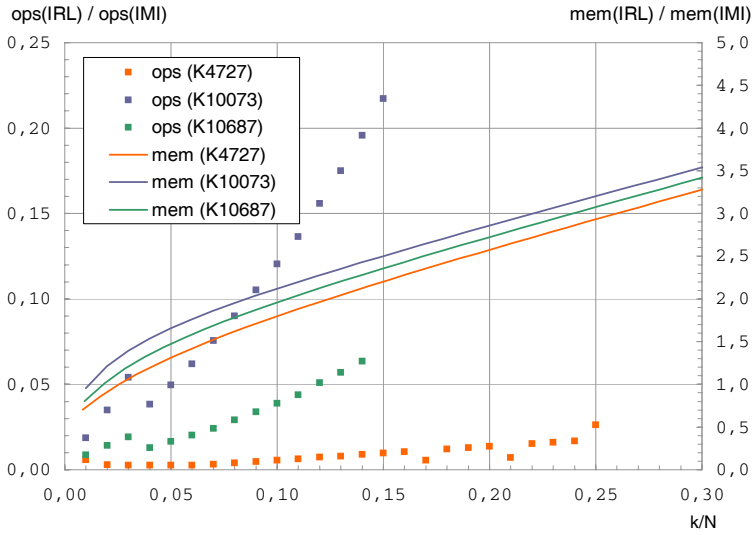


Bild 4.13: Vergleich IRL vs. Inverse Matrixiteration, $(k+p) = 1.8k$

Der multiplikative Aufwand des IRL-Verfahrens ist in allen drei Fällen besser als für die Inverse Matrixiteration. Dies ist für eine kleine bis moderate Anzahl von Eigenzuständen zu erwarten, da der Iteration des Lanczos-Verfahrens die numerisch teuren Anfangsiterationen der Inversen Matrixiteration gegenüberstehen. Eine größere Anzahl von Eigenzuständen scheitert am Speicheraufwand des Verfahrens.

Die Eigenvektornäherungen liefert das Lanczos-Verfahren praktisch mit den Eigenwertnäherungen zu einem sehr geringen Preis. Durch die stetige Reorthogonalisierung der Lanczos-Basis erfüllen die Eigenvektornäherungen ein Orthogonalitätslevel von mindestens $O(N\epsilon)$.

Der numerische Gesamtaufwand des Lanczos-Verfahrens wird im Vergleich zur Inversen Matrixiteration nochmals verbessert. Die Wirtschaftlichkeitsgrenze des Verfahrens ist aber infolge des Speicheraufwands bereits für eine moderate Anzahl von Eigenzuständen erreicht und unterstreicht die besondere Eignung des Verfahrens wenn nur eine kleine Anzahl von Eigenzuständen gesucht wird.

Givens-QRI-Verfahren Der Aufwand zur Bestimmung der Eigenwerte durch Transformation auf Tridiagonalgestalt und anschließender Lösung mit dem impliziten QR-Verfahren ist proportional zum Quadrat der Dimension $O(N^2)$ der Aufgabe. Näherungsweise 95% des Aufwands der Berechnung wird durch die Transformation auf Tridiagonalgestalt verursacht. Der Aufwand zur vollständigen Bestimmung der Eigenwerte mit der Inversen Matrixiteration ist proportional zu $O(b^2\,N^2)$ und damit deutlich höher bei gleichwertiger Güte der Eigenwertnäherungen. Mit dem Aufwand des Givens-QR-Verfahrens bestimmt die Inverse Matrixiteration im allgemeinen weniger als 1% des Gesamtspektrums.

Tabelle 4.5 zeigt den operativen Vergleich der Verfahren. Der Vorteil der Inversen Matrixiteration

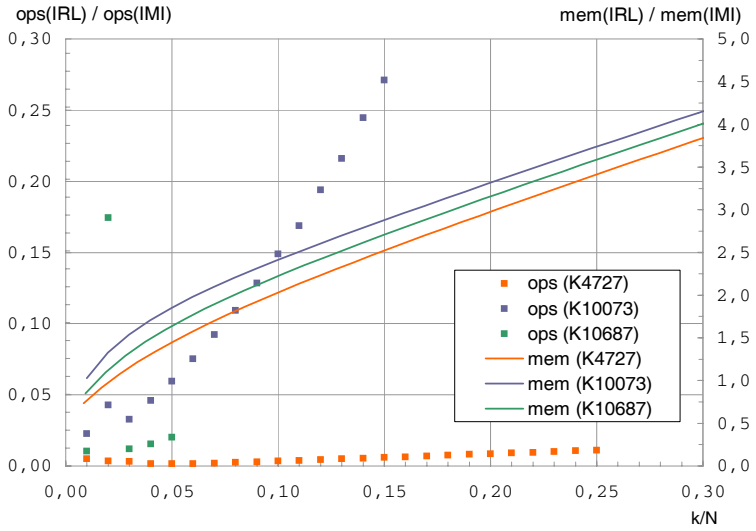


Bild 4.14: Vergleich IRL vs. Inverse Matrixiteration, $(k+p) = 2.0k$

der schrittweisen Bestimmung beginnend mit dem betragskleinsten Eigenwert kann in diesem Vergleich nur für eine kleine Anzahl von Eigenwerten genutzt werden.

1	2	3	4
Matrix	ops_{GQR}	ops_{InvMI}	ops_{GQR}/ops_{InvMI}
K5043	1.31 e+10	9.45 e+11	0.01
K10073	4.43 e+10	2.18 e+12	0.02
K10827	7.05 e+10	4.73 e+12	0.01
K10687	8.46 e+10	7.65 e+12	0.01

Tabelle 4.5: Aufwandsvergleich : Givens-QR vs. Inverse Matrixiteration

Die Eigenvektorbestimmung mit der inversen Vektoriteration erfordert die Zerlegung eines fast-singulären linearen Gleichungssystems. Eine Cholesky-Zerlegung scheidet hierfür aus. in [42] wird stattdessen eine Gauss-Elimination vorgeschlagen, die eine geeignete profilerhaltende Pivotstrategie verfolgt und ausreichende Stabilität garantiert. Der Speicherbedarf für diese Form der Zerlegung wird dabei ungefähr vervierfacht. In der vorliegenden Implementierung wird anstelle einer klassischen Gauss-Zerlegung eine LDL-Zerlegung durchgeführt. Die Zerlegung erhält die Profilstruktur der ursprünglichen Matrix, verfügt aber nicht über die numerische Stabilität der Gauss-Zerlegung für die fast-singulären Matrizen $(\mathbf{A} - \nu\mathbf{I}), \nu \neq \lambda$ [30]. In allen durchgeführten Berechnungen sind keine numerischen Instabilitäten aufgetreten.

4.4.4 Zeitvergleich

Der Zeitvergleich wird über die Systemzeit von *JAVA* qualitativ ermittelt. Die Zeitmessung erfaßt nicht die tatsächliche CPU-Zeit für die Ausführung der Algorithmen, gibt aber einen ausreichenden Eindruck über das Laufzeitverhalten der Implementierungen.

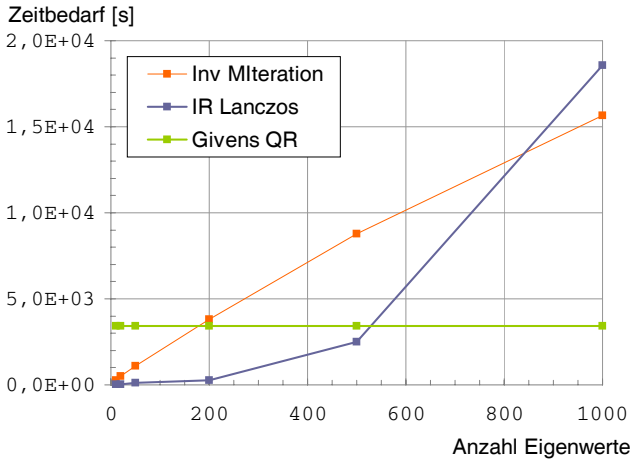


Bild 4.15: Zeitvergleich

Bild 4.15 zeigt einen Zeitvergleich der Verfahren für Berechnungen mit der Matrix K10687. Auf der Ordinate ist der Zeitbedarf der Berechnung in Sekunden aufgetragen, auf der Abszisse die Anzahl bestimmter Eigenwerte. Mit dem IRL-Verfahren und der Inversen Matrixiteration wurden jeweils 10, 20, 50, 200, 500 und 1000 Eigenwerte bestimmt. Das Givens-QR-Verfahren löst in der vorliegenden Implementierung die vollständige Eigenwertaufgabe.

Der Zeitvergleich bestätigt den operativen Aufwand. Der Zeitbedarf für das Givens-QR-Verfahren ist konstant und gering, falls mehr als $500 - 600$ $(\hat{=} \sim 5\%$ der Eigenwerte$)$ berechnet werden. Das Verfahren bestimmt alle Eigenwerte und ist damit für weniger als 4% der Eigenwerte uneffektiv. Andere Implementierungen des Verfahrens ermöglichen teilweise eine gezieltere Bestimmung der Eigenwerte oder die Berechnung der Eigenwerte in einem vorgegebenen Intervall. Zu berücksichtigen bleibt jedoch, daß der Hauptberechnungsaufwand sowohl numerisch als auch zeitlich $(\geq 95\%)$ in der Tridiagonalisierung der Matrix liegt. Ein besserer Algorithmus zur Bestimmung der Eigenwerte der Tridiagonalmatrix bringt damit keine wesentliche Verbesserung des Gesamtberechnungsaufwands.

Für eine kleine bis moderate Anzahl von Eigenwerten liefert das IRL-Verfahren einen sehr günstigen Zeitbedarf. Der Zeitverlauf ist unterlinear und wird erst ab einer größeren Anzahl von Eigenwerten vom Reorthogonalisierungsprozeß der Lanczos-Basis dominiert. Eine wesentliche Schwäche des Verfahrens ist jedoch, daß bei unzureichender Subraumgröße teilweise keine Konvergenz in den ersten $\sim$ 300 Restarts erfolgt und die Berechnung mit vergrößerter Subraumbasis wiederholt werden muß. Die Zahl 300 wurde als maximale Restartanzahl festgesetzt und kennzeichnet in mehrerer Hinsicht eine Wirtschaftlichkeitsschranke. In den durchgeführten

Berechnungen wurde die maximale Subraumgröße $(k+p) = 1.7\,k$ gewählt. Der Berechnungslauf für 50 Eigenwerte hat nicht innerhalb der Restartschranke konvergiert und wurde mit einer maximalen Subraumgröße von $(k+p) = 2\,k$ wiederholt.

Die Inverse Matrixiteration zeigt einen linearen Zeitverlauf der verschiedenen Berechnungen. Für eine kleine Anzahl von Eigenwerten (≤ 50) ist das Verfahren dem IRL-Verfahren unterlegen, der Zeitbedarf liegt aber noch in einem akzeptablen Bereich, falls nur die Eigenwerte von Interesse sind. Für diesen Bereich ($\sim 0.5 - 1.0\%$ der Eigenwerte) ist das Verfahren auch wirtschaftlich, falls nicht die betragskleinsten Eigenwerte, sondern Eigenwerte in einem vorgegebenen Intervall bestimmt werden. Die Beurteilung der Wirtschaftlichkeit des Verfahrens kann nicht ausschließlich am Aufwand der Berechnung vorgenommen werden. Die Zuverlässigkeit der Lösung und die Qualität der Eigenwertnäherungen sind weitere wesentliche Aspekte, die zur Bewertung des Verfahrens berücksichtigt werden. Für eine größere Anzahl von Eigenwerten (< 500) unterliegt das Verfahren dem Givens-QR-Verfahren.

Die Bestimmung der Eigenvektoren erfolgt mit dem IRL-Verfahren implizit durch die vorhandene Lanczos-Basis. Ein zusätzlicher Zeitbedarf entfällt. Eine selektive Eigenvektorbestimmung für ein bekanntes Eigenwertspektrum ist damit jedoch nicht möglich. Dies ist als entscheidender Nachteil zu werten, falls Eigenvektoren stark verteilt über das Gesamt-Eigenwertspektrum bestimmt werden müssen. Die selektive Eigenvektorbestimmung für die Matrix K10687 mit der Inversen Vektoriteration erfordert einen mittleren Zeitaufwand von $\sim 10s$ pro Eigenvektor. Der Zeitbedarf pro Eigenvektor mit der Inversen Matrixiteration liegt bei $\sim 40s$ und ist deutlich höher. Werden jedoch mehrere Eigenvektoren bestimmt, erfordert die Inverse Vektoriteration mehrfach eine Reorthogonalisierung des iterierten Vektors zu den bereits bestimmten Eigenvektoren, insbesondere dann, wenn Eigenwerte schlecht getrennt sind. Für die Inverse Matrixiteration entfällt dieser Aufwand. Im folgenden gezeigte Untersuchungen zur Güte der berechneten Eigenvektornäherungen rechtfertigen aber den teilweise höheren Zeitbedarf der Inversen Matrixiteration.

Aufwand zur Bestimmung eines Teilspektrums

Tabelle 4.6 zeigt einen Aufwandsvergleich für die Bestimmung eines Teilspektrums in einem vorgegebenen Intervall. In Spalte 1 ist das berechnete Intervall gezeigt. Die Spalten 2 und 3 enthalten das Verhältnis des multiplikativen Aufwands der Vergleichsverfahren zur Inversen Matrixiteration. Für das Givens-QR-Verfahren wird nur der Aufwand zur Reduktion der Matrix auf Tridiagonalgestalt betrachtet. Ein geeignetes Nachfolgeverfahren zur Bestimmung innerer Eigenwerte ist in der vorliegenden Implementierung nicht vorhanden. Durch die Hauptlast des Aufwands in der Tridiagonalisierung ($> 95\%$) ist dieser Umstand für den Vergleich nicht relevant. Spalte 4 enthält die Anzahl der Eigenwerte im Intervall.

Für das IRL-Verfahren konnte mit $(k+p) = 2k$ teilweise keine Konvergenz innerhalb von 300 Restarts erreicht werden. Die maximale Subraumgröße wurde deshalb in allen Fällen auf $(k+p) = 3k$ vergrößert.

1	2	3	4	5
Matrix	Intervall	ops_{Giv}/ops_{InvMI}	ops_{IRL}/ops_{InvMI}	k
K5043	$[1.0e+05, 1.5e+05]$	0.48	0.11	23
K5043	$[2.0e+05, 3.0e+05]$	0.30	0.02	35
K7803	$[5.0e+05, 5.2e+05]$	2.78	0.06	4
K10073	$[6.6e+07, 6.8e+07]$	217.98	0.02	2

Tabelle 4.6: Aufwand zur Bestimmung einer beliebigen Teilmenge aus $\sigma(\mathbf{A})$

Das Givens-QRI-Verfahren ist für Subraumdimensionen dieser Größe nur teilweise konkurrenzfähig und liegt im Aufwand häufig außerhalb eines akzeptablen Bereichs. Wegen der kleinen Anzahl von Eigenwerten erfordert das IRL-Verfahren deutlich mehr Restarts als in den Berechnungen der Bilder 4.12 bis 4.14. Trotzdem ist der Aufwand der Berechnung deutlich besser als der Aufwand der Inversen Matrixiteration. Tabelle 4.7 zeigt aber, daß sich die Genauigkeit der Lösung des IRL-Verfahrens im Vergleich zur Inversen Matrixiteration verschlechtert hat. Gezeigt ist jeweils der Mittelwert des relativen Fehlers der Restnorm $\bar{\delta}_{rel}$ für die Berechnung mit dem IRL-Verfahren und der Inversen Matrixiteration. Das genauere Ergebnis der Inversen Matrixiteration rechtfertigt in Anbetracht der geringen Subraumdimension den numerischen Aufwand der Berechnung.

1	2	3	4
Matrix	Intervall	$\bar{\delta}_{rel}$ (InvMI)	$\bar{\delta}_{rel}$ (IRL)
K5043	$[1.0e+05, 1.5e+05]$	$1.0388e-11$	$9.0331e-10$
K5043	$[2.0e+05, 3.0e+05]$	$3.0931e-12$	$6.1757e-09$
K7803	$[5.0e+05, 5.2e+05]$	$1.0798e-11$	$1.5360e-09$
K10073	$[6.6e+07, 6.8e+07]$	$8.7291e-12$	$3.9454e-10$

Tabelle 4.7: Relativer Fehler der Berechnungsergebnisse

Bewertung des Aufwands

Der operative Vergleich der Verfahren zeigt eine klare Überlegenheit des Givens-QR-Verfahrens für die vollständige Eigenwertbestimmung. Für eine kleine Anzahl von Eigenzuständen großer Matrizen (≤ 20), erweist sich die Lanczos-Implementierung als leistungsfähig und zuverlässig.

Für Matrizen großer Dimension (> 5000) ist der numerische Aufwand der Inversen Matrixiteration für die ersten 5% in einem vergleichbaren Bereich, wenn man berücksichtigt, daß die Steuerung der Eigenwertqualität durch das Abbruchkriterium wesentlich einfacher und unmittelbarer

erfolgt, als bei den Vergleichsverfahren. Eine für praktische Anwendungen im Bauingenieurwesen übliche Genauigkeit von 3 bis 4 Dezimalstellen reduziert den Aufwand in einen vergleichbaren Bereich. Beim Lanczos-Verfahren sind nutzergesteuerte Abbruchschranken nur durch zusätzliche Orthogonaltransformationen realisierbar, wie sie in [38] gezeigt und untersucht werden. Das Konvergenzverhalten wird dabei nicht bzw. nur wenig verbessert. Das Givens-QR-Verfahren kann hinsichtlich nutzerdefinierter Abbruchschranken nur unwesentlich beeinflußt werden. Die Reduktion auf Tridiagonalgestalt bildet den Hauptaufwand $(\geq 95\%)$ der Berechnung, ist ein nicht iterativer Prozess und besitzt damit kein Abbruchkriterium.

Der Aufwand für die Eigenvektoren des Lanczos-Verfahrens ist vernachlässigbar, da ihre Bestimmung bereits Teil der Eigenwertbestimmung ist. Der Aufwand für die Eigenvektorbestimmung mit der inversen Vektoriteration ist angesichts der Gauss-Zerlegung für jeden Eigenvektor und dem nicht zuverlässig kalkulierbaren Orthogonalisierungs- und Reorthogonalisierungsaufwands gleichwertig zum Aufwand der Inversen Matrixiteration. Die Zuverlässigkeit hinsichtlich des Berechnungsablaufs und der berechneten Lösung ist für die Inverse Matrixiteration wegen der stabilen QR-Zerlegung höher einzuschätzen.

4.4.5 Qualität der Lösung

Die Qualität der Lösung wird mit den Fehlerschranken aus Kapitel 3 an den Berechnungsergebnissen der Matrix $K10073$ bewertet.

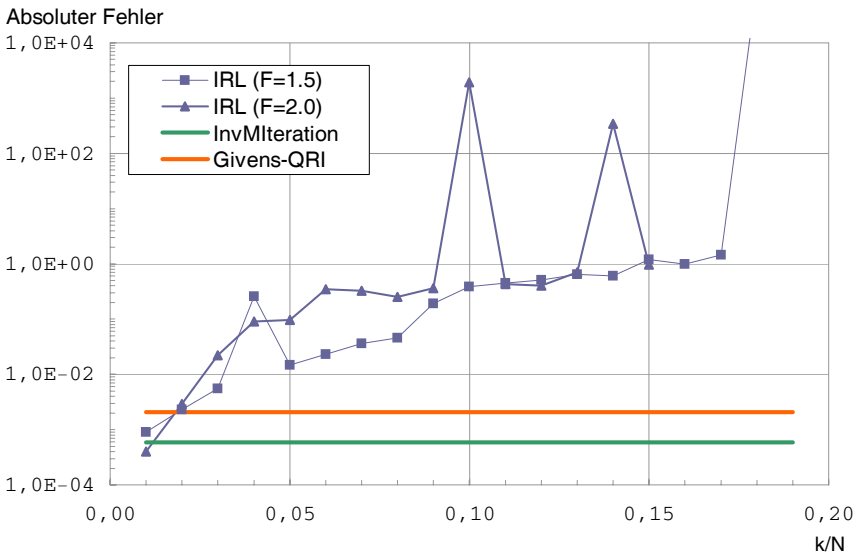


Bild 4.16: Fehler der Restnorm : $\|\mathbf{A}\hat{\mathbf{x}}_i - \hat{\lambda}_i\hat{\mathbf{x}}_i\|$

Bild 4.16 zeigt den Fehler der Restnorm verschiedener Berechnungen. Das IRL-Verfahren bestimmt in jeder Berechnung einen festen Subraum der Dimension $(k + p)$. Die Konvergenz und

das Berechnungsergebnis werden durch die Wahl von $(k+p)$ beeinflußt. Die roten und blauen Quadrate markieren jeweils den statistischen Mittelwert einer Berechnung für die Subraumgrößen $(k+p) = 1.5k$ und $(k+p) = 2.0k$. Das IRL-Verfahrens verliert mit zunehmender Subraumgröße an Genauigkeit und besitzt Ausreißergrößen. Diese Verschlechterung der Restnorm deutet auf Fehler in der erzeugten Subraumbasis hin, die während der Iteration eingetragen werden und im Zusammenhang mit einem Orthogonalitätsverlust stehen.

Die Inverse Matrixiteration wird nicht durch eine a priori Wahl einer Eingangsgröße beeinflußt und bestimmt das gewünschte Teilspektrum schrittweise in einer Berechnung. Das statistische Mittel der berechneten Eigenzustände mit der Inversen Matrixiteration ist in grün dargestellt.

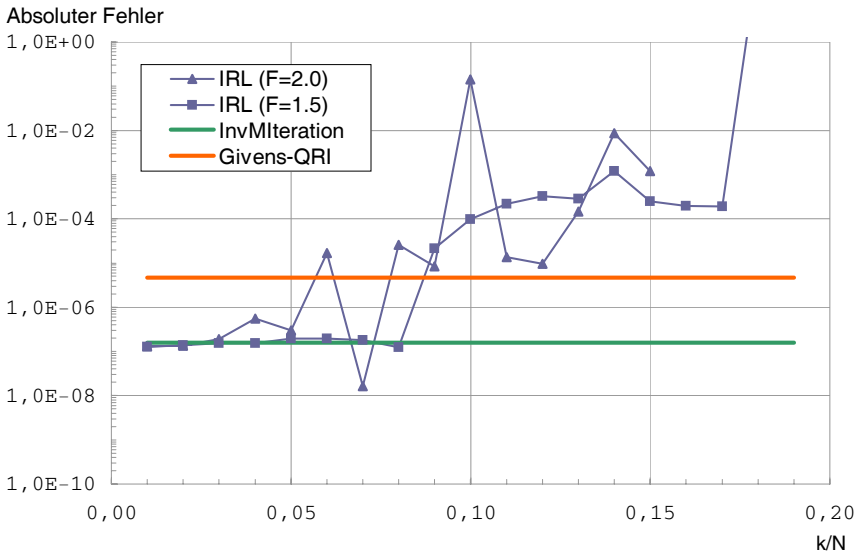


Bild 4.17: $|\rho_i - \hat{\lambda}_i|$

Bild 4.17 zeigt die Differenz zwischen Rayleigh-Quotienten und berechneter Eigenwertnäherung. Auch hier besitzt das IRL-Verfahren eine gute Genauigkeit für eine kleine bis moderate Anzahl von Eigenzuständen, verschlechtert sich aber mit zunehmender Subraumgröße. Die Gründe für die Abnahme der Genauigkeit liegen in Analogie zum Fehler der Restnorm im Orthogonalitätsverlust der erzeugten Lanczos-Basis.

Die Differenz des Givens-QR-Verfahren und der Inverse Matrixiteration liegt konstant im Bereich $O(\epsilon|\mathbf{A}\|_2)$ und erfüllt damit die geforderte Genauigkeit sehr gut. Die Eigenvektornäherungen können damit als sehr gute Näherung angesehen werden. Dies bestätigt auch die Fehlerschranke nach Temple-Kato, die die Richtungsabweichung der Eigenvektornäherung zum exakten Eigenvektor begrenzt. Für das vorliegende Beispiel ($K10073$) erfüllen die Eigenvektoren im Mittelwert $4.70e-08$ für das Givens-QR-Verfahren und $1.56e-09$ für die Inverse Matrixiteration. Für die Lanczos-Implementierung variiert der Mittelwert der Schranke mit zunehmender Subraumgröße, analog zur Restnorm, zwischen $\sim 1.0e-11$ und $1.0e-06$.

Kapitel 5

Anwendungsbeispiel

5.1 Schwingung einer Deckenplatte - Aufgabenstellung

Das vorgestellte Verfahren wird zur Analyse schwingungsgefährdeter Bauwerke eingesetzt. Der numerische Hauptaufwand der Analyse liegt in der Lösung der allgemeinen Eigenwertaufgabe zur Ermittlung des Frequenzgehalts des Bauwerks und der zugehörigen Schwingformen. Die Kenntnis des vollständigen Eigenschwingverhaltens ermöglicht die Berechnung einer guten Näherungslösung für das Verformungsverhalten des Bauwerks unter dynamischer Beanspruchung, zu jedem Zeitpunkt t. Der Einfluß der Bauwerkschwingung auf das Tragverhalten und die Belastung von Personen durch Erschütterungen sind damit berechenbar.

Exemplarisch wird das Schwingverhalten einer Deckenplatte untersucht. Die Platte wird durch dynamische Maschinenlasten an zwei Positionen belastet. Für die Aufstellung stehen betriebsbedingt drei Plattenbereiche zur Verfügung (Bild 5.3). Der Einfluß der verschiedenen Maschinenpositionen auf das Schwingverhalten der Platte wird systematisch untersucht. Die Beteiligung einzelner Eigenschwingformen am Gesamtverformungsverhalten wird ermittelt und bewertet. Die Auswirkungen der Belastung auf die konstruktive Durchbildung der Platte und auf die im Gebäude befindlichen Personen werden ermittelt, mit den zulässigen, normativen Größen verglichen und bewertet.

Das vollständige Frequenzspektrum der diskretisierten Platte, sowie die für die Untersuchungen relevanten Eigenschwingformen werden mit dem Verfahren der Inversen Matrixiteration bestimmt. Für die erforderlichen Untersuchungen zum Schwingverhalten der Platte werden die Bewegungsgleichungen der Aufgabe mit dem in Kapitel 2.3 vorgestellten Verfahren der Modalanalyse entkoppelt.

5.1.1 Modellierung

Geometrie und Materialkennwerte

Bild 5.3 (Seite 190) zeigt eine Grundrißskizze im Maßstab 1:250. Die Bauwerkslänge beträgt 40m, die maximale Breite 13m. Die Deckenplatte ist an den rot markierten Wänden und Stützen gelagert. Die Deckenplatte ist als Stb.-Massivplatte mit folgenden Kennwerten ausgeführt :

Elastizitätsmodul	$3.0e+07$	kN/m^2
Poissonzahl	0.2	
Plattendicke	0.2	m

Finite-Element-Modell

Die Platte wurde mit 4200 Dreieckelementen und 2260 Knoten elementiert. An jedem Knoten sind die globalen Freiheitsgrade u_z, α_x und α_y spezifiziert (siehe 3.4.5).

Die Elementmasse wurde durch ein HRZ-Lumping [7] an den Elementknoten konzentriert. Die Massenmatrix besitzt Diagonalform und ist positiv definit. Die Dämpfung des Systems wird durch modale Dämpfung nach Gleichung (2.44) erfaßt. Die Eigenwertanalyse erfolgt damit am ungedämpften System (Gleichung (2.25)) [2]. Durch die massen- und steifigkeitsproportionale Dämpfung entsprechen die Eigenformen der gedämpften Platte den Eigenformen der ungedämpften Platte.

Die gedämpften Eigenkreisfrequenzen ω_{i_D} ergeben sich zu :

$$\omega_{i_D} = \omega_i \sqrt{1-\xi_i^2} \qquad (5.1.1)$$

ξ_i : Dämpfungsrate

ω_i : i-te Eigenkreisfrequenz des ungedämpften Systems

Vereinfachend wird für alle Eigenschwingungen der Platte eine Dämpfungsrate von 2% gewählt.

Systemgleichungen

Dimension N	:	$6\,438$
Mittlere Bandbreite b	:	77
Speicheraufwand (64-Bit Zahl)	:	$1\,008\,064$

Bild 5.1: Matrixeigenschaften

Die Dimension des Gleichungssystems (2.21) beträgt 6438 Gleichungen. Die mittlere Bandbreite der Steifigkeitsmatrix beträgt 77 Koeffizienten, die maximale Bandbreite 97 Koeffizienten.

Masseannahmen

Für die Schwingungsberechnung werden Lastannahmen aus Eigengewicht und Verkehrslast getroffen. Die Gesamtmasse wird aus folgenden Lastanteilen ermittelt :

Stb. Massivdecke	$d = 0.20m \quad \gamma = 25\,kN/m^3$	5.0	kN/m^2
Deckenaufbau		0.8	kN/m^2
$\sum$		5.8	kN/m^2
Gesamtmasse	$5\,800N/m^2\ /\ (0.20\,m\ 9.81\,m/s^2)$	$\sim 2\,960.0$	kg/m^3

Der Einfluß der Verkehrslast auf die Massenverteilung wird als sehr gering eingestuft und deshalb vernachlässigt.

Dynamische Belastung der Deckenplatte

Die Deckenplatte wird durch unwuchtig rotierende Maschinen mit folgenden Kenngrößen belastet :

Erregerfrequenz f	34	Hz
Erregerkreisfrequenz $\Omega\ =\ 2\,\pi\,f$	213.63	rad/s
Wuchtgüte $Q\ (=e\,\Omega\ =\ e\,2\,\pi\,f)$ e : Exzentrizität	5.2	mm/s
Rotationsmasse m_r	280	kg

Die Maschinen sind direkt auf der Deckenplatte aufgelagert. Ihre Geometrie ist in Bild 5.2 dargestellt. Die Gesamtmasse eines Maschinenblocks beträgt $2\,500\,kg$.

Die Radialbeschleunigung $\ddot{u}_r$ der rotierenden Masse m_r bewirkt eine Radialkraft f_r. Die Projektion von f_r in das kartesische Koordinatensystem der Deckenplatte ergibt einen Kraftanteil $f_z(t)$ senkrecht und einen Kraftanteil $f_{x/y}(t)$ parallel zur Plattenmittelfläche, sowie ein Moment um die y- bzw. x-Achse.

Nach DIN 4024,T1 wird für den Betriebszustand die Wuchtgüte um eine Gütestufe ungünstiger angesetzt, als es der Maschinengruppe nach der VDI-Richtlinie 2060 entspricht. Soll das An- und Abfahren der Rotoren erfaßt werden, ist der 6-fache Wert des Betriebszustandes anzusetzen [31]. Für den Betriebszustand ergeben sich mit der nächsthöher liegenden Wuchtgüteklasse $Q16$ folgende Lasten :

$$f_{x/y}(t) \quad = \quad 0.957\,sin(\Omega t)\ [kN] \tag{5.1.2}$$

$$f_z(t) \quad = \quad 0.957\,cos(\Omega t)\ [kN] \tag{5.1.3}$$

$$m_{y/x}(t) \quad = \quad 0.383\,sin(\Omega t)\ [kNm] \tag{5.1.4}$$

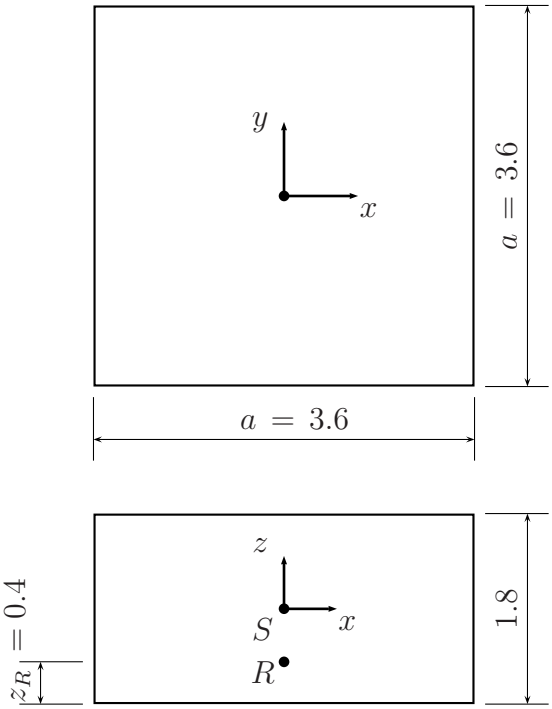


Bild 5.2: Maschinengeometrie

z_R bezeichnet den Abstand von Lagerfuge zu Rotationsmittelpunkt.

Für den *Störfallunwucht*-Zustand beim An- und Abfahren der Rotoren ergeben sich mit dem 6-fachen Wert von $Q16$ folgende Lasten :

$$f_{x/y}(t) = 5.742\, sin(\Omega t)\ [kN] \tag{5.1.5}$$

$$f_z(t) = 5.742\, cos(\Omega t)\ [kN] \tag{5.1.6}$$

$$m_{y/x}(t) = 2.297\, sin(\Omega t)\ [kNm] \tag{5.1.7}$$

In den folgenden Untersuchungen werden nur die vertikalen Schwingungen der Deckenplatte betrachtet. Als Belastung der Deckenplatte werden die Vertikallast $f_z(t)$ und das Moment $m_{y/x}(t)$ angesetzt. Beide Lasten werden auf die Knoten der skizzierten Lastbereiche (Bild 5.3) verteilt. Die einzelnen Lastanteile aus $f_z(t)$ und $m_{x/y}(t)$ werden im Systemlastvektor $\mathbf{p}(t)$ der Bewegungsgleichung (2.21) (Kapitel 2.3) angeordnet.

Belastungsvarianten

Als Gesamtbelastung der Deckenplatten wird die dynamische Belastung aus zwei Maschinenblöcken betrachtet. Beide Maschinenblöcke stehen zwischen den Achsen B und C (siehe Bild

5.3, Seite 190). Betriebsbedingt ergeben sich drei Aufstellungsvarianten :

1	2	3	4	5
Variante	Maschine 1, Achsen	Rotationsachse	Maschine 2, Achsen	Rotationsachse
1	I / II	y	II / III	y
2	I / II	y	II / III	x
3	I / II	y	IV / V	y

Tabelle 5.1: Aufstellungsvarianten der Maschinenblöcke

Die Spalten 2 und 4 der Tabelle 5.1 enthalten die Achsen zwischen denen die Maschinenblöcke positioniert sind. Die Wirkungsrichtung der Momentenbelastung ist in den Spalten 3 und 5 durch die Achsrichtung der Rotationswelle gekennzeichnet (vergleiche Darstellung in Bild 5.3).

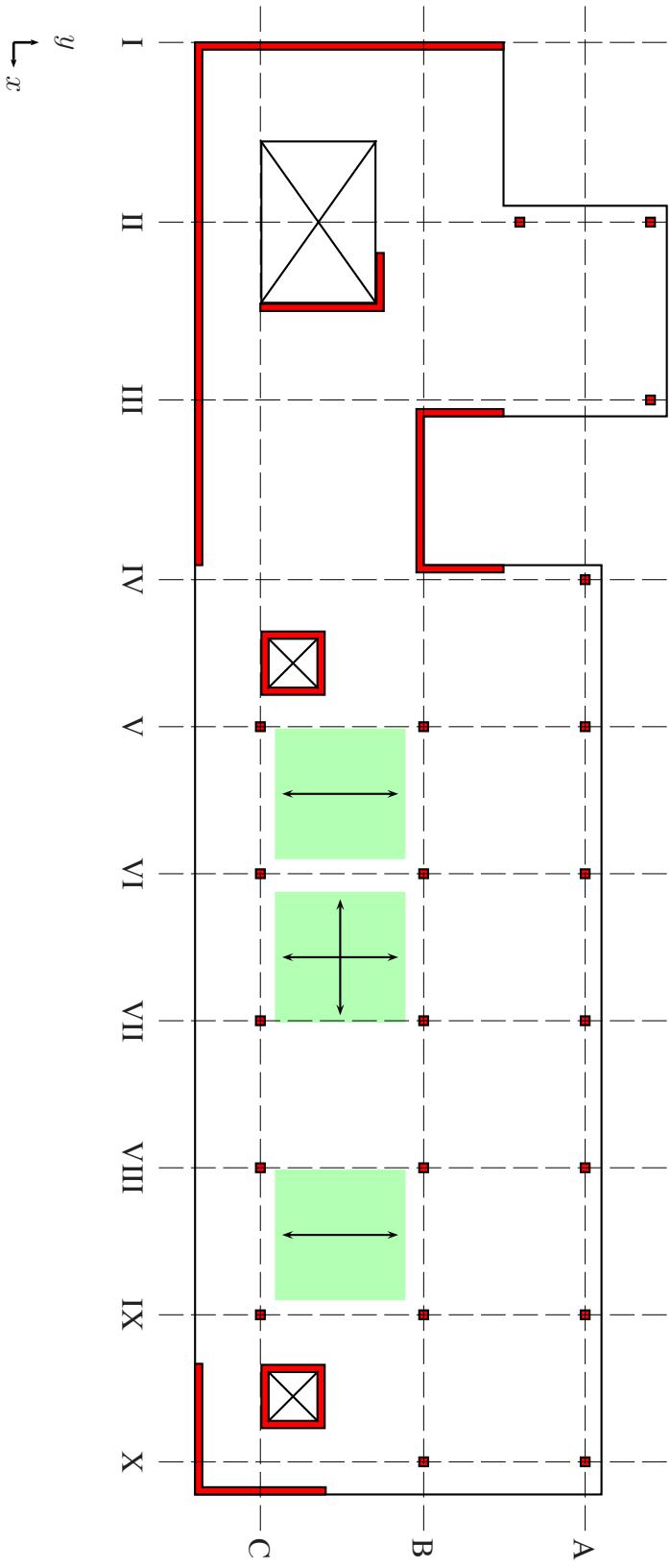


Bild 5.3: Skizze eines Gebäudegrundrisses

5.1.2 Analyse

Eigenschwingverhalten der Platte

Durch die zusätzliche Masse des Maschinenblocks an unterschiedlichen Positionen ergeben sich theoretisch unterschiedliche Frequenzspektren. Die Berechnung der Frequenzspektren hat jedoch gezeigt, daß der Einfluß der Zusatzmasse zu gering ist, um erkennbare Unterschiede in den Einzelfrequenzen festzustellen. Bild 5.4 zeigt 6396 Eigenfrequenzen des gedämpften Systems. Die 42 betragsgrößten Eigenfrequenzen bilden ein enges Cluster bei ungefähr $10\,600\,Hz$ und sind zur besseren Übersicht des Hauptfrequenzbereichs im Diagramm nicht dargestellt.

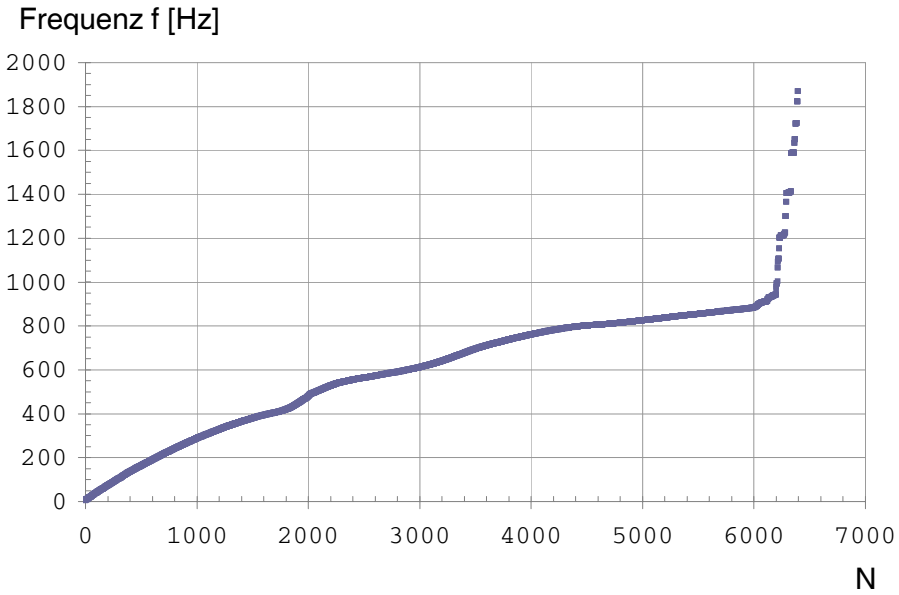


Bild 5.4: Eigenfrequenzen $[Hz]$

Über 96% der Eigenfrequenzen der Platte liegen unter $1\,000\,Hz$. Die Frequenzen liegen sehr eng beieinander und bilden teilweise sehr schlecht getrennte Cluster. Bild 5.5 zeigt das Frequenzband der Platte bis $80\,Hz$. Das Frequenzband umfaßt 221 Eigenfrequenzen. Die Platte besitzt keine mehrfachen Eigenfrequenzen, jedoch viele kleine Frequenzcluster von zwei bis zehn Eigenfrequenzen. In Bild 5.5 sind solche Cluster gut bei $10\,Hz$, $18\,Hz$ oder $30\,Hz$ identifizierbar.

Die Werte der ersten 25 Eigenfrequenzen des Frequenzbandes sind in Tabelle 5.2 (Seite 195) aufgeführt. Tabelle 5.3 (Seite 196) enthält einen Ausschnitt aus dem Eigenfrequenzspektrum, das im Bereich der Erregerfrequenz der Maschinen liegt, sowie die Eigenfrequenzen mit betragsgroßen Beteiligungsfaktoren aus der Momentenbelastung. Die Tabellen enthalten außerdem den berechneten Eigenwert, die Eigenperiode, sowie den Beteiligungsfaktor der i-ten Eigenschwingung am Gesamtschwingverhalten der Platte infolge der dynamischen Belastung, jeweils für die Aufstellungsvarianten 1/2, und die Variante 3. Die gezeigten Beteiligungsfaktoren sind so skaliert, daß

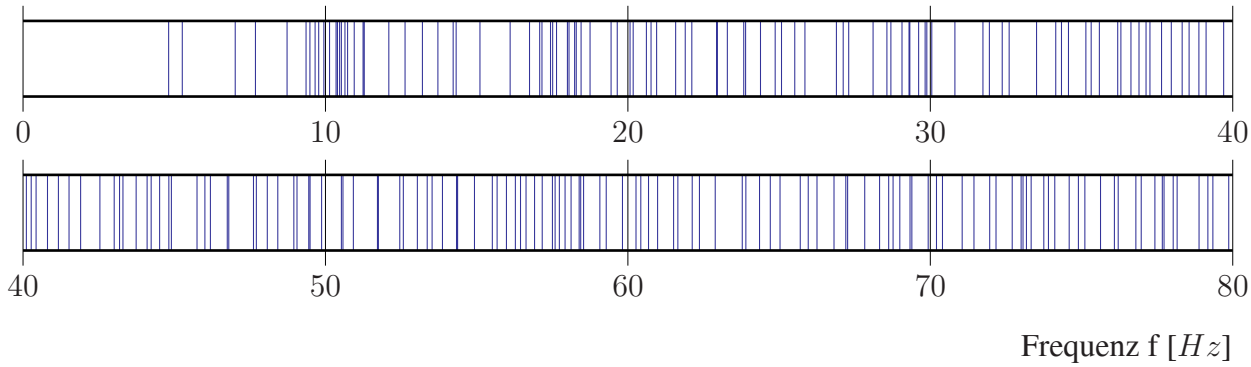


Bild 5.5: Frequenzband bis $80Hz$

der betragsgrößte Wert nach Gleichung (2.37) den Wert 100 besitzt. Die Eigenschwingungen der grau unterlegten Frequenzen sind in den Bildern 5.7 bis 5.10 dargestellt. Der Einwirkungsort der dynamischen Belastungen ist jeweils als blaues Quadrat gezeigt.

Bild 5.6 zeigt die Beträge der skalierten Beteiligungsfaktoren der Eigenschwingungen für die Aufstellungsvarianten 1 und 2. Dargestellt sind nur die Beteiligungsfaktoren mit $b_i > 0.1$ im Frequenzbereich bis $1000Hz$. Um den Ursprung der angeregten Eigenschwingung besser zu erfassen, sind die Beteiligungsfaktoren, die aus der vertikalen Belastung $f_z(t)$ stammen in rot, die aus $m_{x/y}(t)$ stammen in blau markiert.

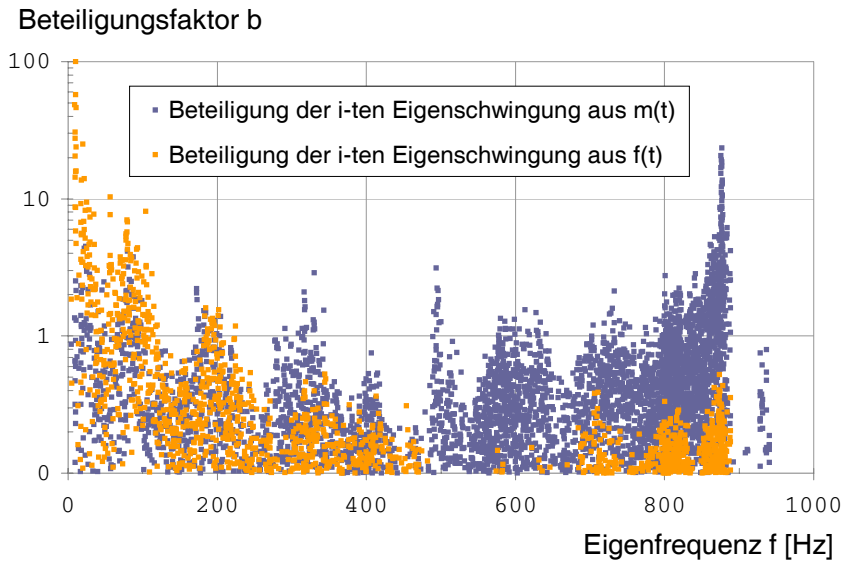


Bild 5.6: Beteiligungsfaktoren $|b_i|$

Die Anzahl Eigenschwingungen, die zu signifikanten Verformungen der dynamisch belasteten Deckenplatte führen ist erwartungsgemäß gering. Diese Eigenschwingungen konzentrieren sich

stark im unteren Frequenzbereich und stammen fast ausschließlich aus der vertikalen Schwingungsbelastung $f_z(t)$. Die großen Beteiligungsfaktoren aus $m_{x/y}(t)$ im oberen Frequenzbereich sind für das Schwingverhalten der Platte nicht maßgeblich.

Eine Approximation des Verformungsverhaltens mit den Eigenschwingungen die einen Beteiligungsfaktor $b_i > 10.$ besitzen, erfordert die Untersuchung der ersten 300 Eigenfrequenzen. Wird $b_i > 5.$ für eine Approximation verwendet, ist die Kenntnis des halben Gesamtspektrums der Aufgabe erforderlich.

Analyse des Eigenschwingverhaltens Die Frequenzen f_5, f_{13} und f_{14} (Seiten 197, 198) zeigen starke Verformungen des linken, freien Plattenrandes. Alle vier gezeigten Frequenzen der Bilder 5.7 und 5.8 (Seiten 197, 198) führen als Folge der engen punktförmigen Deckenlagerung zu starken Verschiebungen in der Auflagerfläche der Maschinenblöcke. Lediglich der Maschinenblock in der Achse IV/V bleibt in der Auflagerfläche für die Frequenz f_{13} verschiebungsfrei. Dieser Umstand kann aber nicht als vorteilhaft gegenüber den anderen Standorten betrachtet werden, da die Frequenz f_{18} hier eine große Verformung aufweist und einen Beteiligungsfaktor besitzt, der die Frequenz für beide Aufstellungsvarianten als maßgebend kennzeichnet. Für die Aufstellungsvarianten 1 und 2 liefert die Frequenz f_{13} den maximalen Beteiligungsfaktor, für die Aufstellungsvariante 3, die Frequenz f_{18}. Beide Schwingformen besitzen mittig unter jeweils einem der Maschinenblöcke die maximale Durchbiegung. Dieser Umstand ist auch am großen Beteiligungsfaktor der Frequenz f_{14} für die Aufstellungsvariante 3 klar erkennbar. Die Maschinenblöcke haben durch ihre Abmessungen und ihre Positionierung in den kleinen Deckenfeldern keine Möglichkeit zur Schwingung als starrer Körper. Die gesamte Auflagerfläche wird von den maßgeblichen Frequenzen durch Biegeverformungen belastet. Die wesentlichen Verformungen der Deckenplatte sind im Bereich zwischen den Achsen V und X zu erwarten. Die kleineren Frequenzen verursachen hier starke Querschwingungen (y-Richtung). Ab der zehnten Eigenfrequenz gehen die Schwingungen in Längs- bzw. Wechselschwingungen (x- und y-Richtung) über.

Eine Richtungsänderung der Momentenbelastung führt zu einer Änderung in den Beteiligungsfaktoren, sowohl im Index der beteiligten Eigenschwingungen, aber auch in der Stärke der Beteiligung an der Plattenschwingung. Die Verformungen der Deckenplatte infolge der dynamischen Erregung resultieren maßgeblich aus der vertikalen Schwingungserregung. Die Momentenbelastung spricht erst Eigenschwingungen oberhalb $800Hz$ an, beispielsweise f_{5920} (Bild 5.10, Seite 200). Diese Frequenzen können jedoch nicht als wesentlich für das Gesamtschwingverhalten angesehen werden. Der Eigenfrequenzbereich liegt nicht oder nur unbedeutend im Frequenzspektrum möglicher Erregerkräfte. In [3] ist außerdem gezeigt, daß für die hohen Eigenwerte, hinsichtlich dem exakten Eigenschwingverhalten der Tragstruktur nur wenig oder gar keine Genauigkeit erwartet werden kann.

Die Eigenfrequenzen f_{80}, f_{81} und f_{82} (Bilder 5.9 und 5.10) liegen am nächsten zur Erregerfrequenz $f = 34Hz$ der Maschinen und sind bei Dauerbetrieb stark resonanzgefährdet.

Die Bilder 5.11 und 5.12 zeigen weitere Eigenschwingformen. Bild 5.11 (Seite 201) zeigt die ersten drei Eigenschwingformen. Die Beteiligungsfaktoren der Eigenschwingungen sind sehr

gering. Sie werden durch die dynamische Last der Maschinen nicht angeregt, trotzdem sind sie wesentlich für eine Beurteilung der Deckenschwingung, da sie unterhalb $10Hz$ liegen. Damit liegen sie in einem von Menschen induzierten Frequenzbereich. Eine Fourieranalyse periodischer Anregung durch Gehen, Laufen oder Springen ergibt, daß ca. 95% der darin enthaltenen Frequenzen unter $10Hz$ liegen. Die maßgeblichen Frequenzen für Gehen liegen zwischen $1.65Hz$ und $2.35Hz$, für Laufen und Springen zwischen $2.0Hz$ und $3.0Hz$. Höhere Erregerfrequenzen $(> 5.0Hz)$ werden beispielsweise in Büros, Tanzsälen oder Sporthallen erreicht [21].

Die Eigenfrequenzen f_{25} und f_{493} (Bild 5.12) sind Beispiele für fast isolierte Schwingungen, die noch im Bereich möglicher Erregerfrequenzen liegen. Im Frequenzbereich zwischen $160 - 170Hz$ liegen weitere isolierte Eigenschwingungen der Ecken zwischen den Achsen I bis V. Die Schwingung der Platte ist für diese Frequenzen nahezu null. Lediglich die Deckenecken schwingen mit großer Amplitude. Diese Schwingungen werden keine Auswirkungen auf das Tragverhalten haben. Durch ihre geringen Beteiligungsfaktoren $(b_i \sim 1)$ bleiben sie für Schwingungsanalysen unentdeckt. Eine lokale Schädigung der Bausubstanz ist aber bei längerer Einwirkungsdauer nicht auszuschließen. Für die betroffenen Bereichen können diese Schwingungen durchaus maßgebliche Bemessungsgrößen verursachen. Für eine sichere Entdeckung dieser Schwingungen ist die Kenntnis eines großen Frequenzbereichs erforderlich. Sie werden nach folgender Prozedur numerisch günstig bestimmt :

Die bestimmten Eigenvektornäherungen werden so skaliert, daß ihr betragsgrößter Koeffizient den Wert Eins besitzt. Für die so skalierten Eigenvektoren wird das Skalarprodukt auf sich selbst gebildet (Gleichung (5.1.8)). Aus der kleinen Teilmenge von Eigenvektoren mit den kleinsten Skalaren η_i werden diejenigen als isolierte Schwingung betrachtet, die weniger als $(c\,N)$ Freiheitsgrade mit $x_{k_i}^2 < \varepsilon$ besitzen. Die Schranken c und ε werden empirisch gewählt, beispielsweise $c = 0.2$ und $\varepsilon = 0.0001$.

$$\eta_i \;=\; \mathbf{x}_i^T \mathbf{x}_i \;=\; \sum_{k=1}^{N} x_{k_i}^2 \tag{5.1.8}$$

1	2	3	4	5	6
i	λ_i	f [Hz]	T [s]	$\|b_i\|$ Variante (1/2)	$\|b_i\|$ Variante (3)
1	915.090980	4.81	0.2077	2.649457	0.578600
2	1094.660321	5.27	0.1899	0.452292	0.340438
3	1943.355003	7.02	0.1425	0.061290	0.040694
4	2331.453374	7.68	0.1301	0.084160	0.051785
5	3009.055113	8.73	0.1145	48.687091	27.751214
6	3457.402493	9.36	0.1068	29.153761	2.550831
7	3552.387619	9.49	0.1054	19.480332	8.928968
8	3681.924687	9.66	0.1035	26.638471	1.537093
9	3773.093694	9.78	0.1022	9.115484	6.522584
10	3905.990528	9.95	0.1005	13.916686	0.865071
11	4055.256953	10.14	0.0986	5.876636	3.811556
12	4229.376846	10.35	0.0966	15.625538	0.298102
13	4267.087082	10.40	0.0961	100.000000	18.271440
14	4325.592251	10.47	0.0955	8.973085	79.077149
15	4374.148678	10.53	0.0950	54.858337	15.016031
16	4473.063580	10.64	0.0939	21.898366	13.976491
17	4546.589153	10.73	0.0931	15.612872	7.143221
18	4734.355389	10.95	0.0913	42.750640	100.000000
19	4990.584250	11.24	0.0889	4.596485	10.868332
20	5025.734671	11.28	0.0886	0.078012	0.069487
21	5775.219800	12.09	0.0826	1.154742	1.343189
22	6299.328191	12.63	0.0791	0.017156	0.068319
23	6884.738989	13.21	0.0757	0.209355	3.480916
24	7428.932679	13.72	0.0728	0.136318	0.140148
25	7985.627411	14.22	0.0703	0.707656	0.608917

Tabelle 5.2: Modalanalyse : unteres Frequenzspektrum

1	2	3	4	5	6
i	λ_i	f [Hz]	T [s]	$\|b_i\|$ Variante (1/2)	$\|b_i\|$ Variante (3)
75	39772.317770	31.74	0,0315	1.077814	2.866316
76	40307.380926	31.95	0,0313	0.105855	1.354199
77	41386.046018	32.38	0,0309	2.056236	3.180080
78	41974.126493	32.61	0,0307	0.213608	0.384398
79	44339.064800	33.51	0,0298	0.387484	0.073456
80	46033.723045	34.15	0,0293	2.137145	1.751220
81	46546.994182	34.34	0,0291	0.113853	0.965508
82	47175.786913	34.57	0,0289	7.296166	0.668529
83	48759.063692	35.14	0,0285	2.374800	2.607406
84	49246.733468	35.32	0,0283	5.302228	7.690829
85	50004.765763	35.59	0,0281	0.527290	2.082999
86	51723.692761	36.20	0,0276	1.395471	0.552005
5912	30379230.705324	877.25	0.0011	9.550243	7.457275
5913	30381495.080968	877.32	0.0011	17.860145	3.959193
5914	30386604.289921	877.34	0.0011	0.029199	4.905411
5915	30387958.917737	877.36	0.0011	12.653853	4.235571
5916	30389446.975163	877.37	0.0011	10.048811	4.286032
5917	30390188.777689	877.40	0.0011	1.630571	6.557470
5918	30392134.136911	877.48	0.0011	4.622855	2.909792
5919	30397315.615227	877.49	0.0011	13.376848	2.011602
5920	30398235.979855	877.59	0.0011	6.899737	6.410782
5921	30405321.566554	877.65	0.0011	5.399698	0.301399
5922	30409050.029479	877.70	0.0011	10.234258	9.476333
5923	30412704.593378	877.74	0.0011	3.349224	2.038953
5924	30415512.462397	877.82	0.0011	3.641007	2.262959

Tabelle 5.3: Modalanalyse : mittleres und oberes Frequenzspektrum

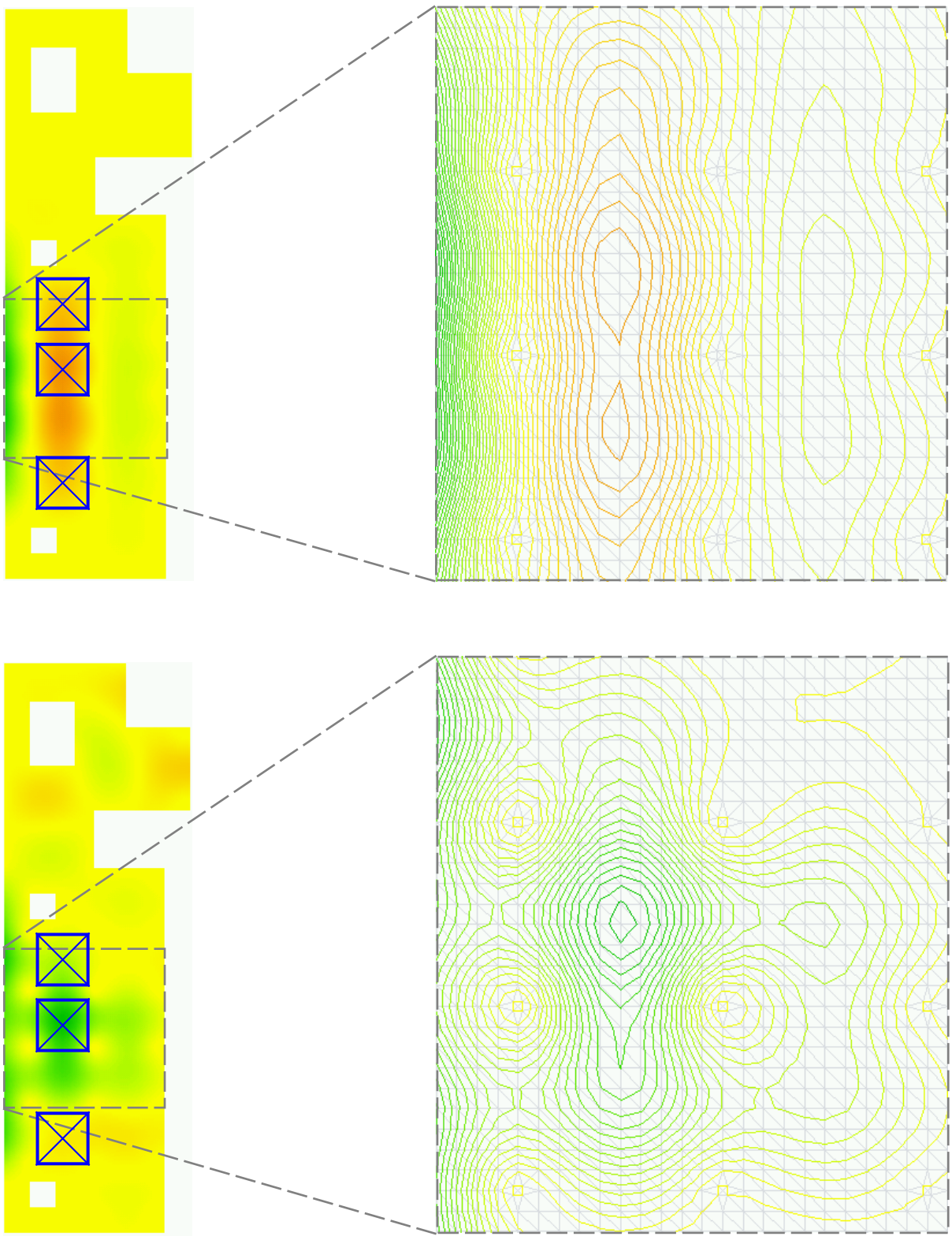


Bild 5.7: Schwingformen der Frequenzen $f_5 = 8.73Hz$ ($\uparrow$) und $f_{13} = 10.40Hz$ ($\downarrow$)

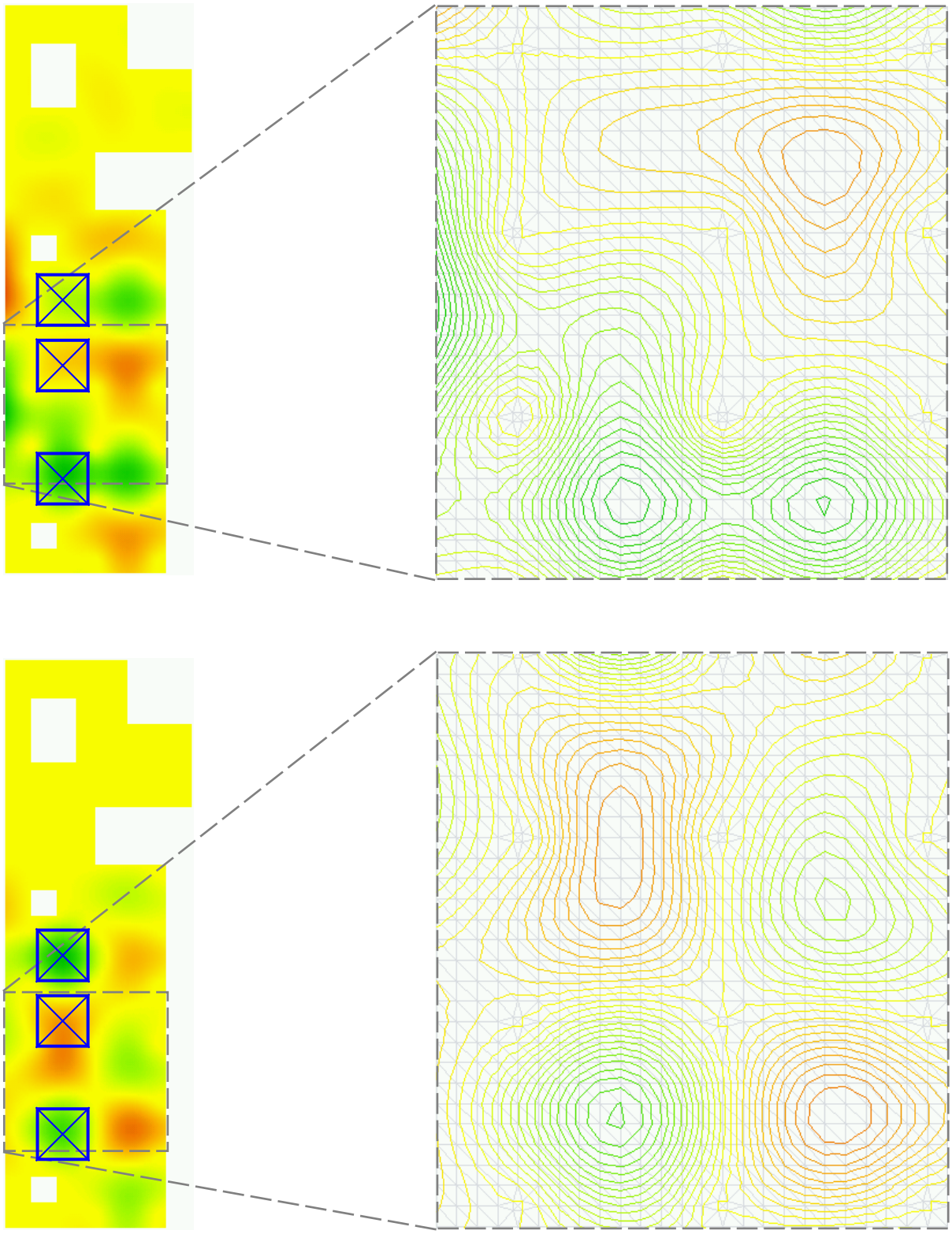


Bild 5.8: Schwingformen der Frequenzen $f_{14} = 10.47Hz$ (↑) und $f_{18} = 10.95Hz$ (↓)

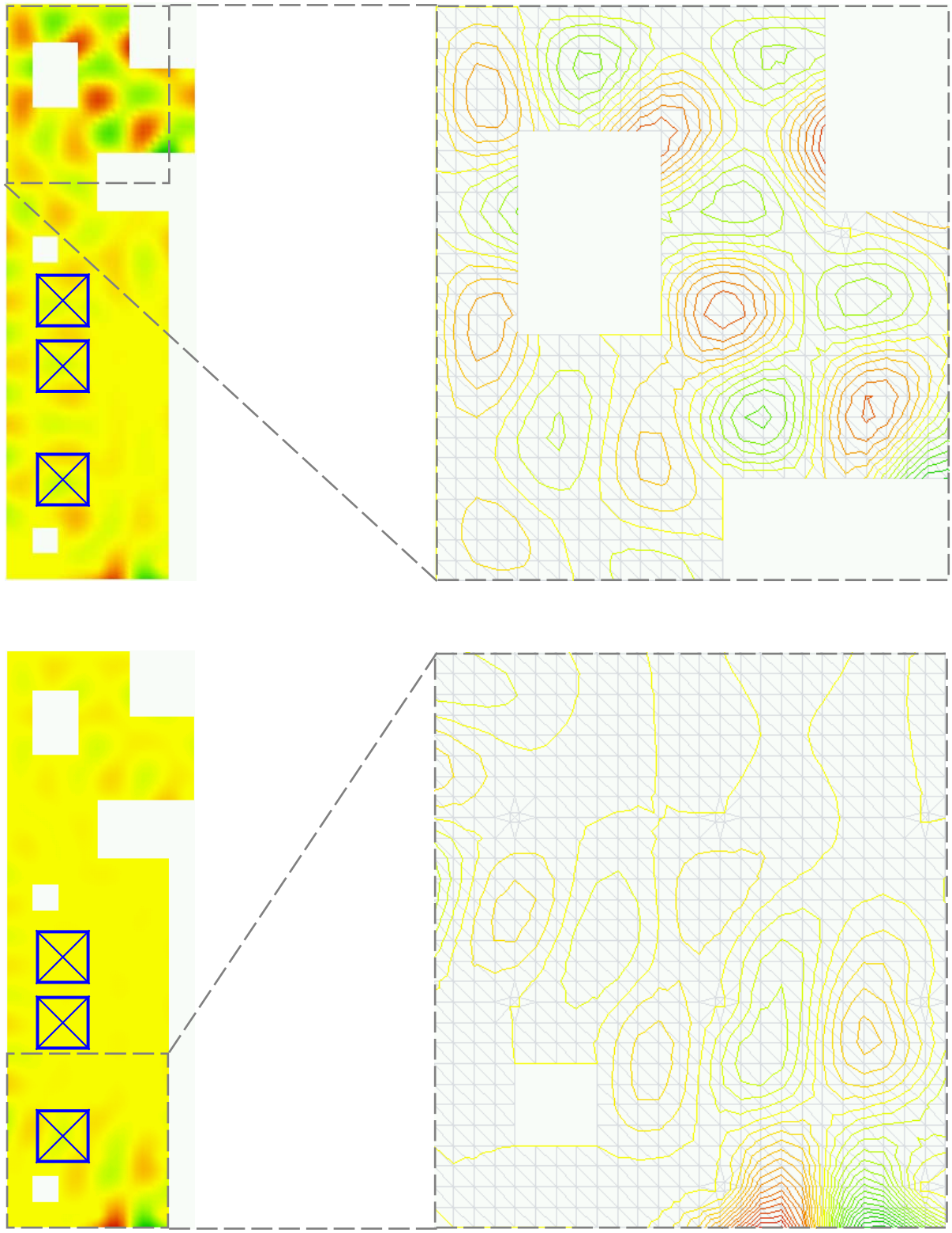


Bild 5.9: Schwingformen der Frequenzen $f_{80} = 34.15Hz$ (↑) und $f_{81} = 34.34Hz$ (↓)

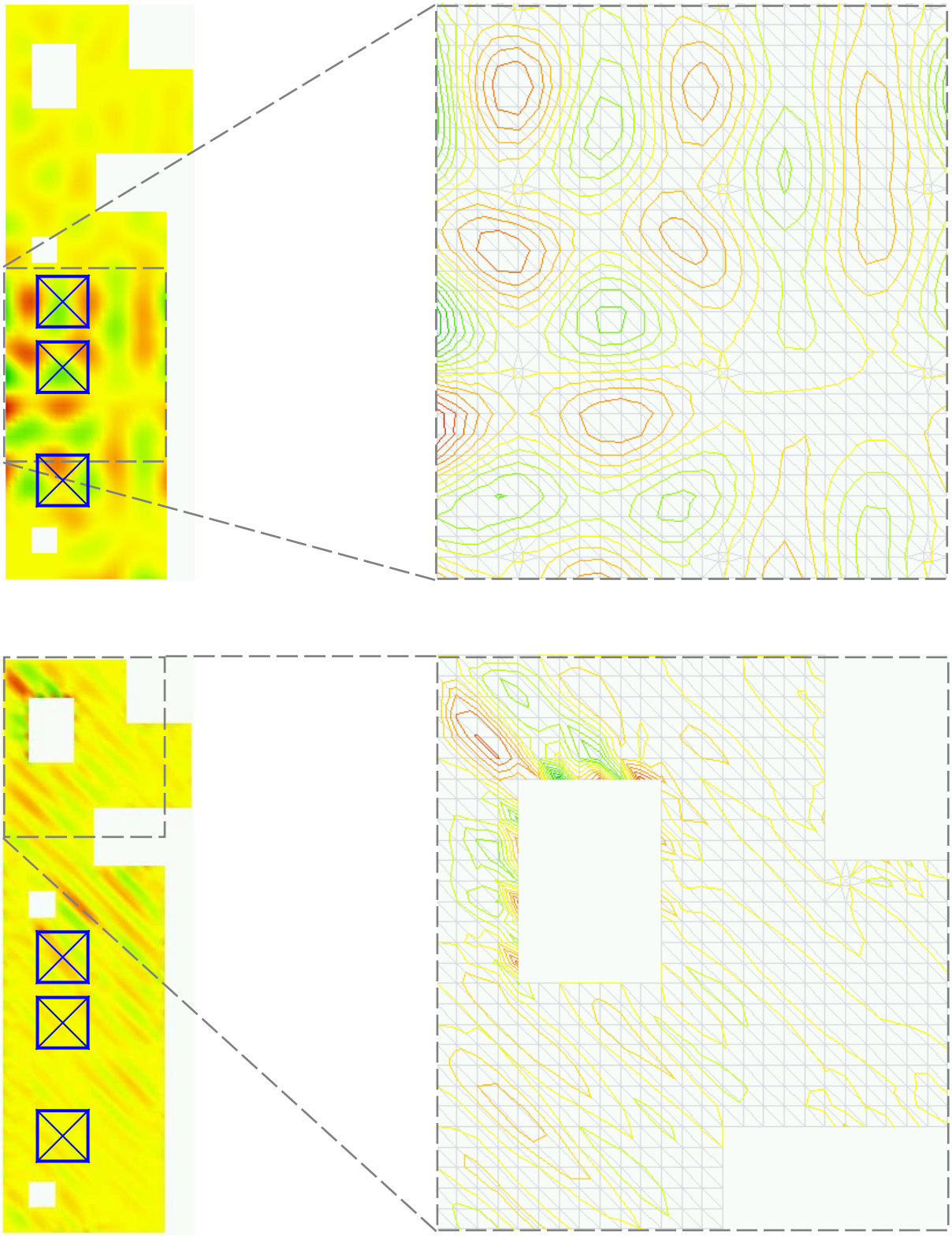


Bild 5.10: Schwingformen der Frequenzen $f_{82} = 34.57Hz$ (↑) und $f_{5920} = 877.59Hz$ (↓)

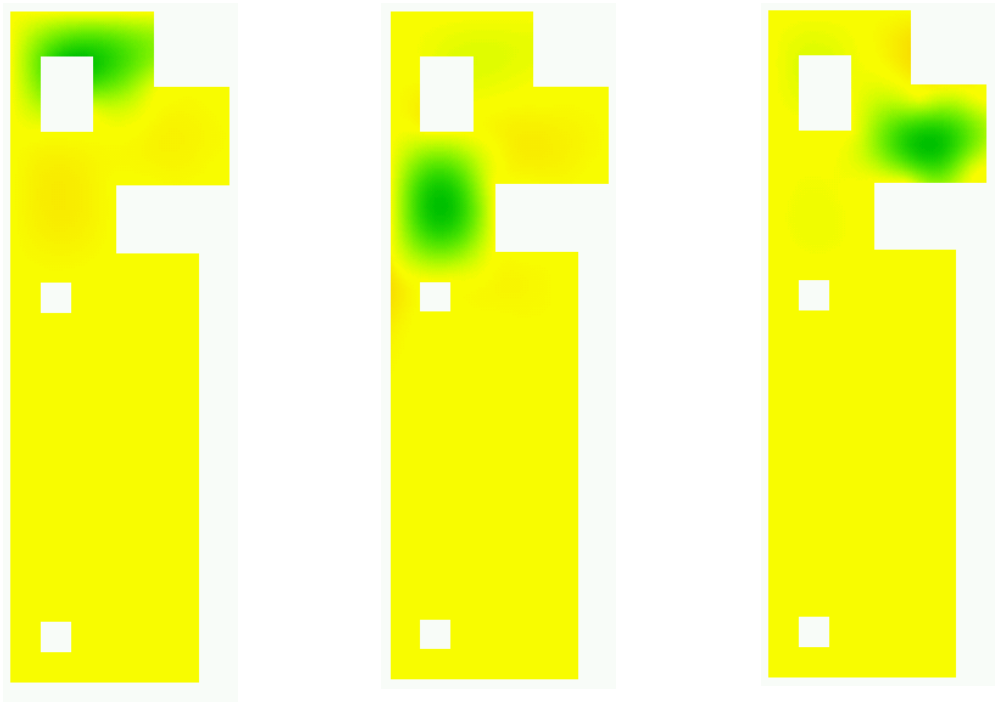


Bild 5.11: Schwingformen der Frequenzen $f_1 = 4.81Hz$, $f_2 = 5.27Hz$, $f_3 = 7.02Hz$

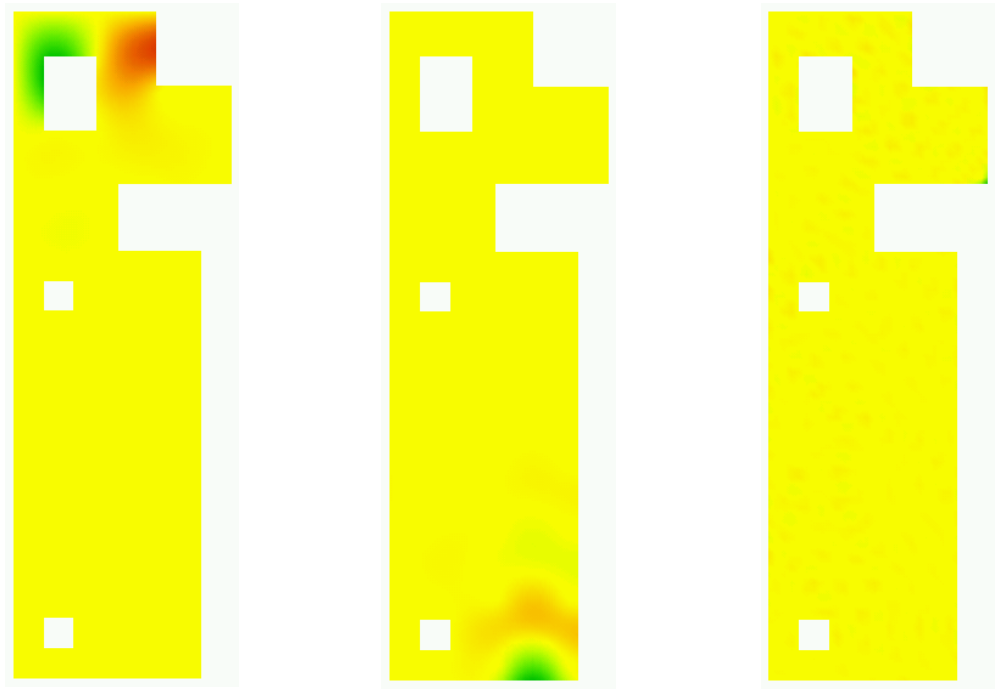


Bild 5.12: Schwingformen der Frequenzen $f_4 = 7.68Hz$, $f_{25} = 14.22Hz$, $f_{493} = 160.83Hz$

5.1.3 Bewertung

Auswirkungen auf Menschen

Die Wahrnehmung von Erschütterungen wird stark durch die physische und psychische Verfassung einer Person geprägt. Für eine Beurteilung der Schwingungsbelastung auf den Körper wird das Zusammenspiel physikalischer Größen, wie Erregerfrequenz, Schwinggeschwindigkeit bzw. Schwingbeschleunigung und Einwirkungsdauer untersucht. Die Schwingungswirkung wird wesentlich durch die Eigenfrequenzen des menschlichen Körpers und der inneren Organe beeinflußt und führt zu Stressreaktionen, Belästigung und potentiell schädigender Belastung.

Die Eigenfrequenzen eines sitzenden Menschen liegen zwischen $4 - 5Hz$. Organe wie Herz, Nieren und Magen besitzen Eigenfrequenzen im Bereich von $2 - 7Hz$. Die Eigenfrequenzen des Kopfes liegen zwischen $20 - 30Hz$ [20]. Im Frequenzbereich bis $10Hz$ ist die Wahrnehmung einer Schwingungsbelastung annähernd proportional zur Schwingbeschleunigung. Im Frequenzbereich zwischen $10 - 100Hz$ wird eine Proportionalität zur Schwinggeschwindigkeit beobachtet. Für die Wahrnehmung von Schwingungen aus Erschütterungen wird ein Frequenzspektrum bis ca. $80Hz$ betrachtet. Über $80Hz$ geht die Empfindlichkeit des Körpers infolge einer inneren Isolierwirkung stark zurück. Frequenzen unterhalb $\sim 0.6Hz$ führen bei längerer Einwirkung (>30min), ab Beschleunigungen von $1m/s^2$ zu Übelkeit und seekrankheitsähnlichen Symptomen [21].

Zur Beurteilung beliebiger periodischer und nichtperiodischer Schwingungen auf Menschen in Gebäuden sind in der DIN4150, Teil 2 "...Anforderungen und Anhaltspunkte genannt, bei deren Einhaltung erwartet werden kann, daß in der Regel erhebliche Belästigungen von Menschen in Wohnungen und vergleichbar genutzten Räumen vermieden werden"[8]. Als Beurteilungsgröße wird in der DIN4150 die bewertete Schwingstärke K verwendet. In den internationalen Richtlinien und den europäischen Normen dient als Beurteilungsgröße die frequenzbewertete Beschleunigung a_w. Auf nationaler Ebene ist diese Größe bereits in der VDI-Richtlinie 2057 berücksichtigt und damit dem internationalen Standard angepasst. Im folgenden werden die Untersuchungen zur "Einwirkung mechanischer Schwingungen auf den Menschen"[46] bezugnehmend auf die VDI-Richtlinie 2057 durchgeführt.

Beurteilungsgrößen Die VDI-Richtlinie 2057 unterscheidet bei der Ermittlung und Beurteilung der Schwingungsbelastung zwischen verschiedenen Körperhaltungen der betroffenen Personen und dem Ort der Einwirkung. Darüber hinaus bietet die Richtlinie auch Anhaltswerte für wechselnde oder unbekannte Körperhaltungen, die im folgenden zu Grunde gelegt werden. Dabei werden zwei Bewertungsgrößen ermittelt :

1. *Frequenzbewertete Beschleunigung* $a_w(t)$: Bewertungsgröße zur Erfassung der momentanen Belastung aus dem zeitlichen Verlauf der Schwingbeschleunigung und dem zugrunde

liegenden Frequenzgehalt.

$$a_w(f) = a(t)\, W_m(f) \tag{5.1.9}$$

$a(t)$ Schwingbeschleunigung zur Zeit t

$W_m(f)$ Frequenzbewertung nach VDI 2057, Blatt 1

2. *Effektivwert der frequenzbewerteten Beschleunigung* a_{wT} : Energieäquivalenter Mittelwert der frequenzbewerteten Beschleunigung über den Zeitraum T.

$$a_{wT} = \sqrt{\frac{1}{T}\int_0^T a_w^2(t)dt} \tag{5.1.10}$$

Die Beurteilung der Schwingungsbelastung erfolgt anhand des Effektivwerts der frequenzbewerteten Beschleunigung a_{wT}. In der VDI-Richtlinie 2057 wird zwischen drei Kriterien unterschieden :

- Wohlbefinden (Komfort)
- Leistungsfähigkeit
- Gesundheit

Wohlbefinden / Komfort : Zur Beurteilung des Wohlbefindens in schwingungsbelasteten Umgebungen stellt die VDI-Richtlinie Erfahrungswerte bereit, die einen Zusammenhang zwischen bewerteter Schwingstärke und subjektiver Wahrnehmung darstellen. Da die Wahrnehmung von Schwingungen stark durch die subjektiven Größen (Alter, Geschlecht, Gesundheitszustand, etc.) geprägt ist, können die Werte aus Tabelle 5.4 lediglich als Anhaltswerte verstanden werden.

Effektivwert $a_{wT}[m/s^2]$	Wahrnehmung
< 0.010	nicht spürbar
$0.010 \ldots 0.015$	Wahrnehmungsschwelle
$0.015 \ldots 0.020$	gerade spürbar
$0.020 \ldots 0.080$	gut spürbar
$0.080 \ldots 0.315$	stark spürbar

Tabelle 5.4: Anhaltswerte für die Wahrnehmung von Schwingungen

Für eine Einschätzung des Komfortempfindens in einer schwingungsbelasteten Umgebung, ist in der ISO 2631, Teil1 ein Zusammenhang zwischen dem Effektivwert der frequenzbewerteten Beschleunigung und Komfortkategorien aufgezeigt. Das Komfortempfinden ist stark situationsbedingt und schließt Gewöhnungseffekte u.ä. nicht aus.

Effektivwert $a_{wT}[m/s^2]$	Wahrnehmung
< 0.315	komfortabel
$0.315 \ldots 0.630$	etwas unkomfortabel
$0.500 \ldots 1.000$	ziemlich unkomfortabel
$0.800 \ldots 1.600$	unkomfortabel
$1.250 \ldots 2.500$	sehr unkomfortabel
> 2.000	extrem unkomfortabel

Tabelle 5.5: Anhaltswerte für das Komfortempfinden

Leistungsfähigkeit : Die Leistungsfähigkeit von Personen wird durch direkte mechanische Störungen oder durch indirekte, physiologische und psychologische Beeinträchtigungen vermindert. Typischerweise treten Ermüdungseffekte, Nervosität und andere Symptome in Erscheinung [46]. Für eine Einschätzung dieses Kriteriums gibt es keine kategorisierten Anhaltswerte. Für Leistungsanforderungen, die "mit dem Führen von Kraftfahrzeugen im Straßenverkehr oder dem Fahren von Erdbaumaschinen vergleichbar sind", ist eine Leistungsbeeinträchtigung "wenig wahrscheinlich"[46], falls der mit der Einwirkungsdauer bewertete Mittelwert $a_{w(8)}$ der Effektivwerte a_{wT} für eine Beurteilungsdauer von $8h$ den Wert $0.3m/s^2$ unterschreitet.

Gesundheit : Gesundheitsschädigende Effekte sind von der Belastungsdauer und Intensität abhängig. Zur Einschätzung der Gesundheitsgefährdung bei einer täglichen Einwirkungsdauer von $T_e = 4 - 8h$ sind in Bild 5.13 zwei Richtwertkurven gezeigt. Für Belastungen oberhalb der ersten Kurve kann von einer möglichen, oberhalb der zweiten Kurve, von einer deutlichen Gefährdung betroffener Personen ausgegangen werden [46].

Auswertung der Berechnung mit Maschinenlast

Für die Ermittlung der maximalen Verschiebungen und Beschleunigungen wurde ein Zeitintervall von $20s$ betrachtet. Das Zeitinkrement der Berechnung beträgt $0.01s$. Effekte aus dem Einschwingverhalten verfälschen die Ergebnisse des Dauerbetriebs. Deshalb wurden alle maßgeblichen Größen aus der zweiten Hälfte des Zeitintervalls ermittelt.

Die dynamische Belastung aus der Erregung durch Maschinen wird als synchron angenommen, ohne Phasenverschiebung.

Die Auswahl der in den folgenden Berechnungen beteiligten Eigenschwingungen erfolgt über die Beteiligungsfaktoren b_i und die Erregerfrequenz der Maschinen. Über die Beteiligungsfaktoren werden Eigenschwingungen mit $|b_i| \geq 10.$ berücksichtigt. Diese Moden werden von der dynamischen Belastung am stärksten angesprochen und sind damit für das Gesamtschwingverhalten der Platte maßgebend. Die Anzahl dieser Moden variiert nur sehr geringfügig mit den

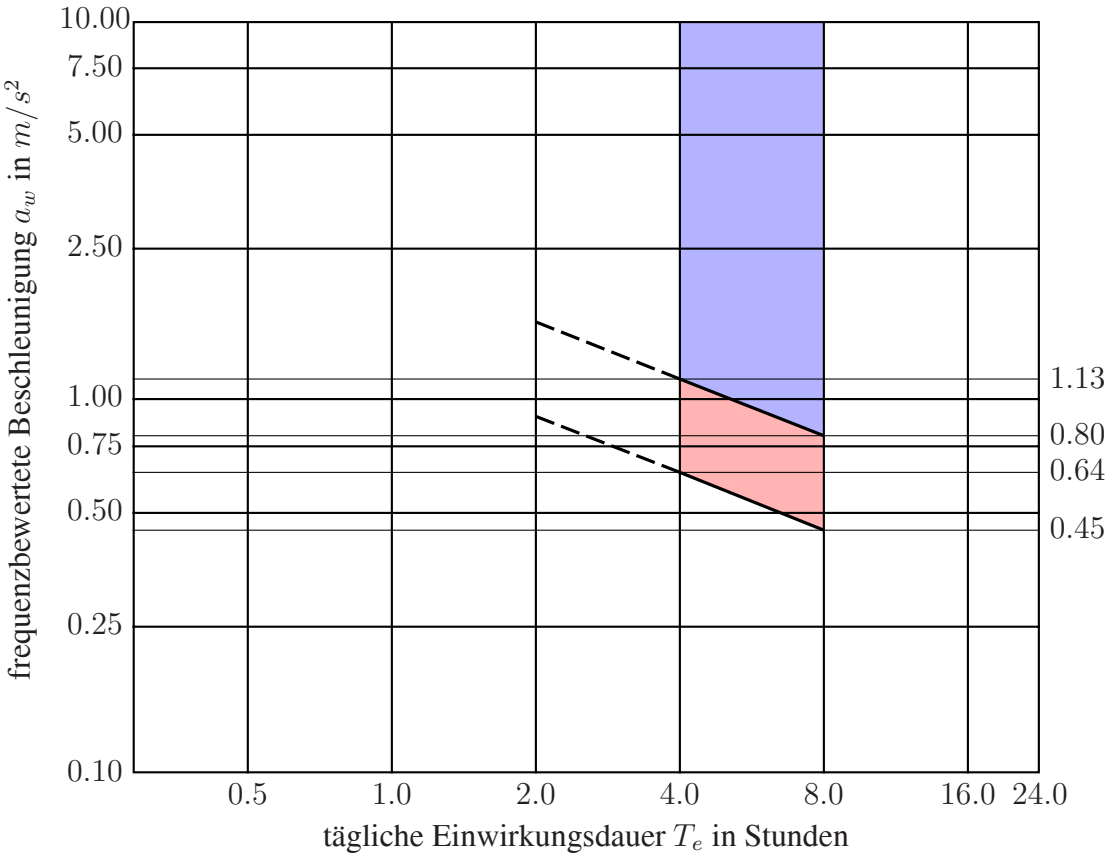


Bild 5.13: Richtwertkurven

verschiedenen Lastvarianten (vergleiche Abschnitt 5.1.1). Zusätzlich werden die zehn Schwingformen berücksichtigt, die am nächsten zur Erregerfrequenz $f = 34Hz$ liegen. Sie besitzen die maximalen dynamischen Lastfaktoren und verursachen Resonanzeffekte (s. Kapitel 2.3).

Bild 5.14 (Seite 208) zeigt die maximalen dynamischen Lastfaktoren $D(t)$, den Zeitpunkt ihres Auftretens, sowie die unskalierten Beteiligungsfaktoren $|b_i|$ der ersten 200 Eigenschwingungen für die Aufstellungsvariante 1. Der Frequenzbereich um $34Hz$ besitzt erwartungsgemäß die maximalen dynamischen Lastfaktoren. Die Beteiligungsfaktoren dieser Eigenfrequenzen sind jedoch sehr gering, sodaß auch im Dauerbetrieb der Maschinen kein Resonanzverhalten der Deckenplatte zu erwarten ist. Die Beteiligung dieser Frequenzen am Gesamtschwingverhalten der Platte wird aber trotzdem signifikant sein, da sie die maximalen generalisierten Verformungen $(max\ g_{82} = 0,199\,mm,$ Lastfall : Störfallunwucht$)$ verursachen. Die generalisierten Verschiebungen der Frequenzen mit großem Beteiligungsfaktor besitzen dieselbe Größenordnung $(max\ g_{13} = 0,121\,mm,$ Lastfall : Störfallunwucht$)$.

Der Zeitpunkt des Auftretens der maximalen dynamischen Lastfaktoren liegt für ungefähr jeweils die Hälfte der Eigenschwingungen bei $t_1 \doteq 10.25\,s$ bzw. bei $t_2 \doteq 19.75\,s$. Da es unwahrscheinlich ist, daß alle Eigenfrequenzen gleichzeitig ihre maximale Verformung aufweisen, werden die maximalen Durchbiegungen nach Gleichung (2.51) überlagert.

Beurteilung der frequenzbewerteten Beschleunigungen Bild 5.15 (Seite 209) zeigt die Effektivwerte der frequenzbewerteten Beschleunigungen a_{wT} als Isolinienmodell für den Lastfall *Störfallunwucht*. Die Aufstellungsvarianten 1 und 2 sind nahezu identisch in ihrem Schwingverhalten. Dies spiegelt sich auch in den bewerteten Beschleunigungen wider. Die Effektivwerte der frequenzbewerteten Beschleunigungen sind für die Aufstellungsvariante 2 im Randbereich geringfügig höher ($\sim 1\%$). Die Aufstellungsvariante 3 erreicht nicht die Maximalwerte der Varianten 1 und 2, wird aber in den maschinenfreien Bereichen zwischen den Achsen IV und X stärker belastet.

Der Vergleich der ermittelten Werte a_{wT} mit den Werten zur Beurteilung der Wahrnehmung aus Tabelle 5.4 (Seite 203) zeigt, daß die maschineninduzierten Schwingungen für alle Varianten in fast allen Bereichen der Deckenplatte wahrgenommen werden. Die Maximalwerte treten in unmittelbarer Nähe der Maschinenstandorte auf. Insbesondere der linke, freie Plattenrand schwingt mit großer Amplitude (Bild 5.17) und verursacht im betrachteten Frequenzbereich eine gut bis stark spürbare Schwingung. Die Auswirkungen der Eigenschwingungen mit großen Beteiligungsfaktoren sind in allen Aufstellungsvarianten gut erkennbar. Das Belastungsmuster in Bild 5.16 (Varianten 1 und 2) entspricht sehr gut der Schwingform der Eigenfrequenzen f_5, f_{13} und f_{18} (Bilder 5.7,5.8, Seiten 197, 198). Zusätzlich sind die Auswirkungen der maximalen dynamischen Lastfaktoren für die Eigenfrequenzen f_{80} und f_{81} am unteren Plattenrand, aber auch zwischen den Achsen II und III, gut erkennbar. Die größte Schwingbelastung verursacht deutlich die Aufstellungsvariante 2, die geringste Schwingbelastung verursacht Aufstellungsvariante 3.

Das Komfortempfinden liegt nach Tabelle 5.5 auch für die Maximalwerte in einem komfortabel wahrgenommenen Bereich.

Für eine Beurteilungsdauer von $8h$ entspricht die Beurteilungsbeschleunigung $a_{w(8)}$ bei Dauerbetrieb der Maschinen den Werten aus Bild 5.16. Der Grenzwert von $0.3m/s^2$ wird nicht erreicht. Eine Leistungsbeeinträchtigung arbeitender Menschen ist nur “wenig wahrscheinlich”, aber wegen individueller Beeinträchtigungen der Psyche nicht auszuschließen. Eine Gesundheitsgefährdung der Personen kann ausgeschlossen werden, da es sich bei den Beurteilungsbeschleunigungen um Maximalwerte handelt, die nicht an jedem Ort der Deckenplatte permanent wirksam sind.

Für den Betriebszustand (Bild 5.16) sind keinerlei Beeinträchtigungen zu erwarten. Die Effektivwerte der frequenzbewerteten Beschleunigungen liegen fast im gesamten Plattenbereich unterhalb der Wahrnehmungsgrenze. Die Bereiche, in denen die Schwingbelastung spürbar ist, beschränken sich auf den äußeren Plattenrand und einzelne Deckenfelder zwischen den Achsen V und X. Eine Gefährdung von Wohlbefindens, der Leistungsfähigkeit oder der Gesundheit kann ausgeschlossen werden.

Gebrauchswert des Tragwerks

Die Beurteilung der Erschütterungseinwirkungen auf die Deckenplatte wird nach DIN 4150-3 [9] über die Schwinggeschwindigkeiten vorgenommen. Die maximalen Verschiebungen und Schwinggeschwindigkeiten für alle drei Aufstellungsvarianten sind in den Bildern 5.17 (Seite 211) und 5.18 (Seite 212) dargestellt.

Die Schwinggeschwindigkeiten liegen für alle maßgeblichen Frequenzen unterhalb $2.5mm/s$, die maximalen Durchbiegungen unterhalb $0.02mm$. Eine "Verminderung des Gebrauchswertes" nach DIN 4150, Teil3 [9], z.B. die Verminderung der Tragfähigkeit der Decke ist für Schwinggeschwindigkeiten $v_z \leq 20mm/s$ nicht zu erwarten. Auch leichtere Schäden werden infolge der geringen Schwinggeschwindigkeiten ausgeschlossen. Ein Spannungsnachweis ist für Schwinggeschwindigkeiten bis $10mm/s$ nach DIN 4150, Teil3 nicht erforderlich, da sie zu keinerlei Schäden führen, "selbst wenn die bei der statischen Bemessung zulässigen Spannungen voll in Anspruch genommen sind" [9].

Ein Nachweis gegen Ermüdungsversagen kann entfallen, "wenn der dynamische Beanspruchungsanteil weniger als 10% der statisch zulässigen Beanspruchung beträgt"[9]. Eine statische Berechnung zeigt, daß die maximale Durchbiegung infolge der Gewichtskraft der Maschinenblöcke ungefähr um den Faktor 30 über dem Maximalwert der dynamischen Belastung liegt. Die zulässige Beanspruchung der Decke wird damit aus der statischen Berechnung ermittelt. Ein Ermüdungsversagen wird ausgeschlossen.

Sämtliche Bewertungen des Schwingungsverhaltens wurden auf Grundlage der maximalen Schwingungen getroffen. Diese stammen, zur Erfassung von Störfallunwuchten beim An- und Abfahren der Maschinen, aus dynamischen Belastungen, die mit einem starken Sicherheitsfaktor behaftet sind. Treten die Rechenwerte der dynamischen Belastung auf, dann sicher nur sehr kurzzeitig, sodaß von einer dauerhaften dynamischen Belastung der Decke durch die aufgestellten Maschinen nicht ausgegangen werden kann.

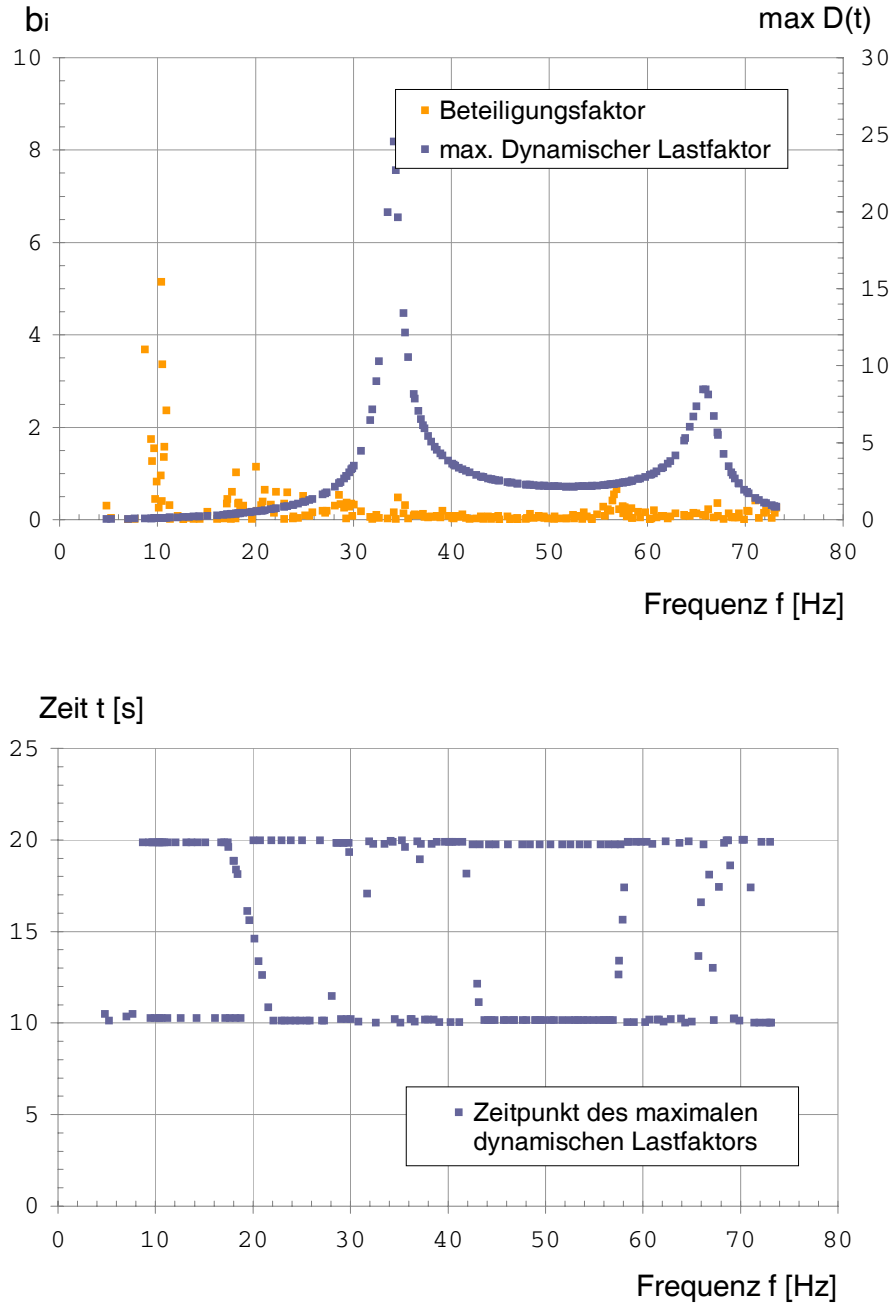


Bild 5.14: Einfluß des Dynamischen Lastfaktors

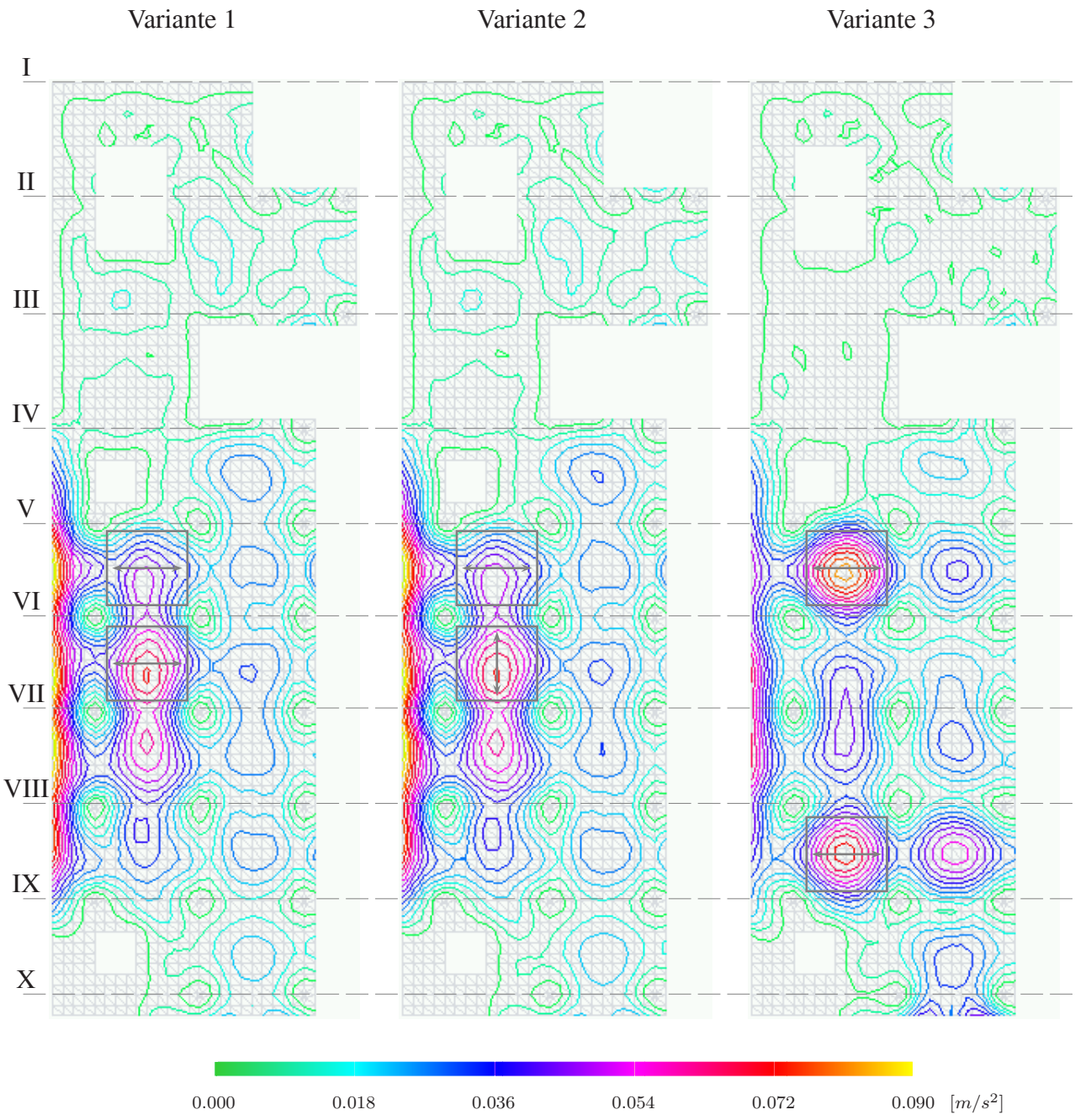


Bild 5.15: Effektivwerte a_{wT}, *Störfallunwucht*-Zustand

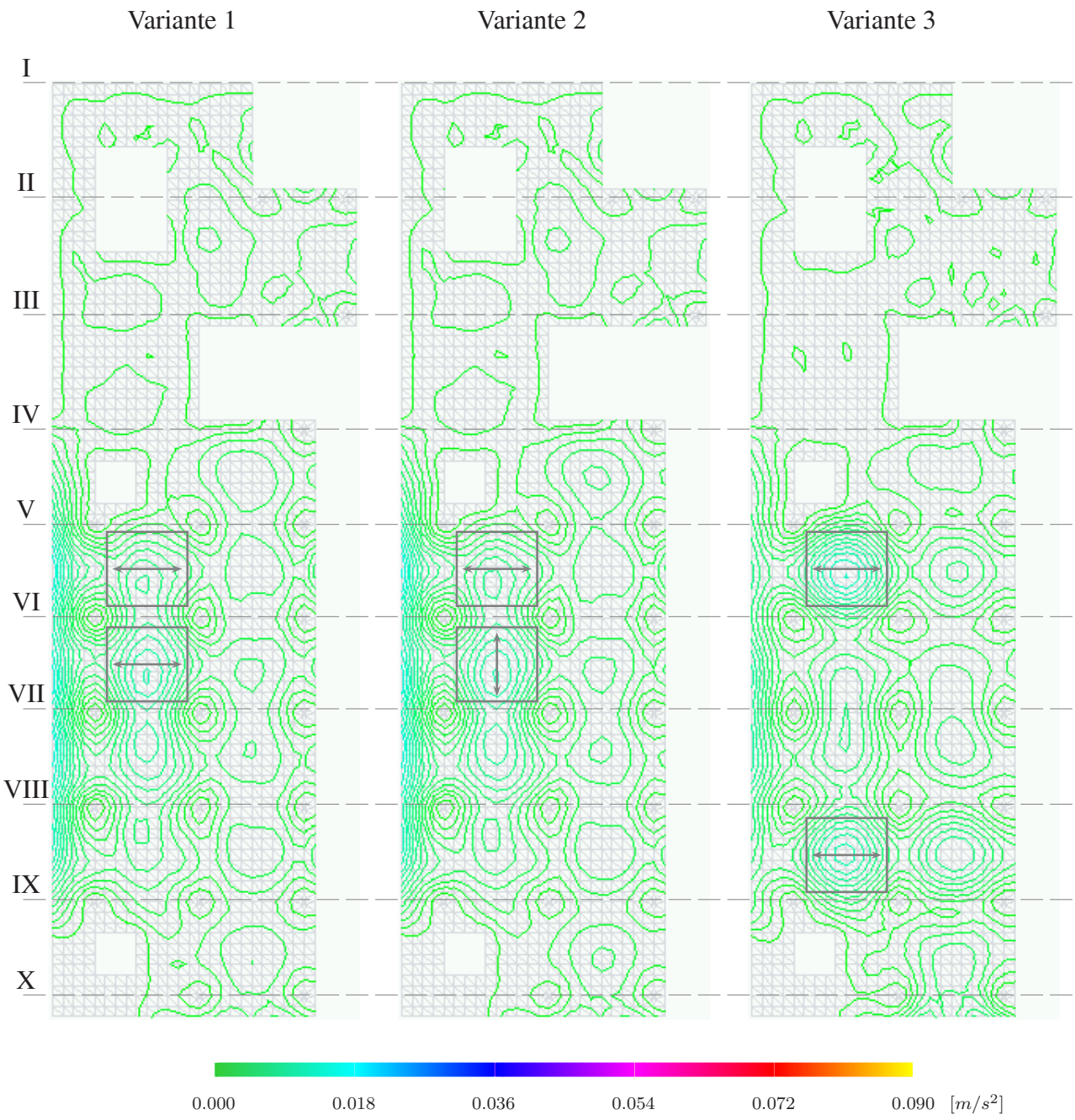


Bild 5.16: Effektivwerte a_{wT}, *Betriebszustand*

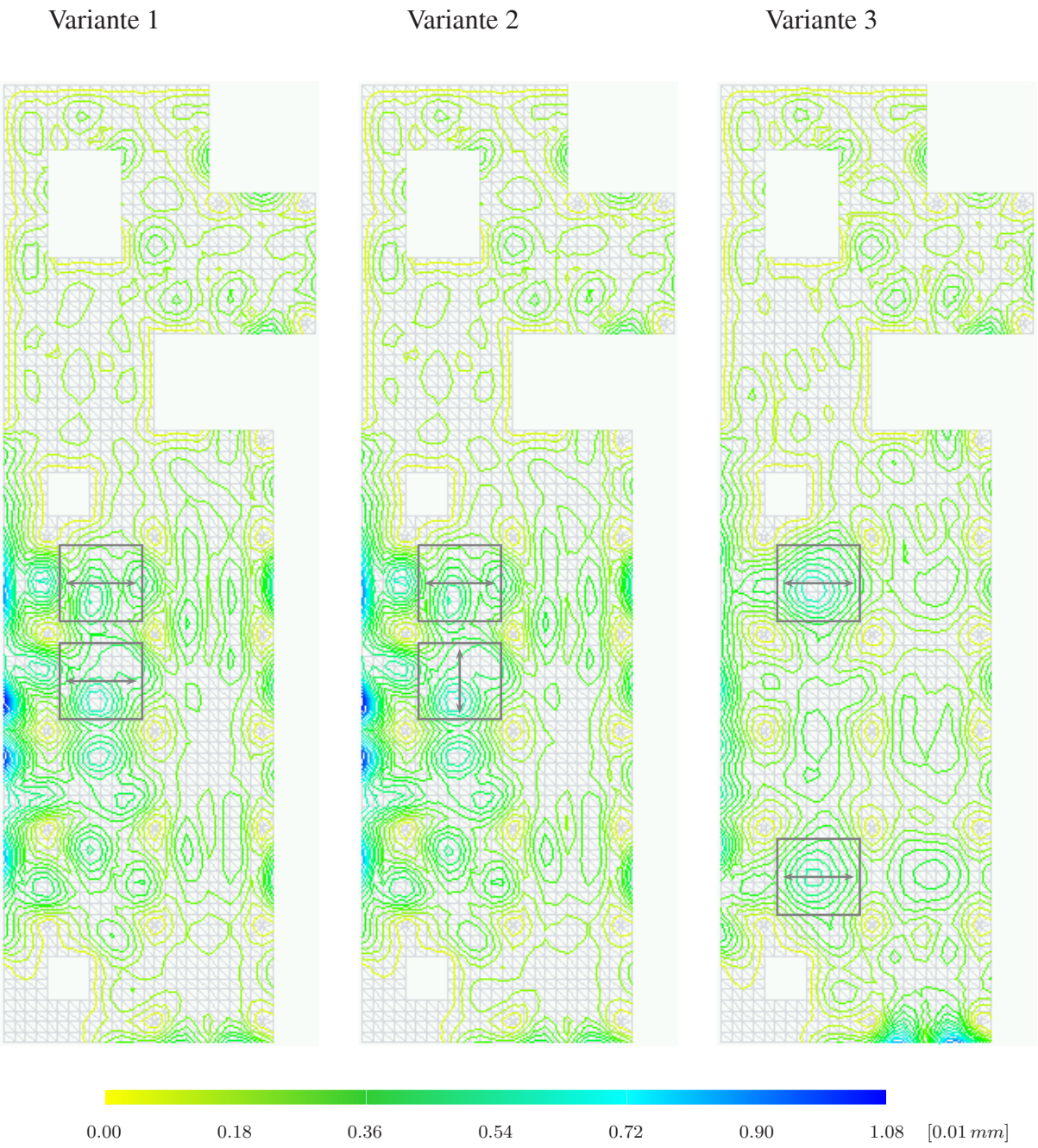


Bild 5.17: Maximalwerte der Verschiebung u_z

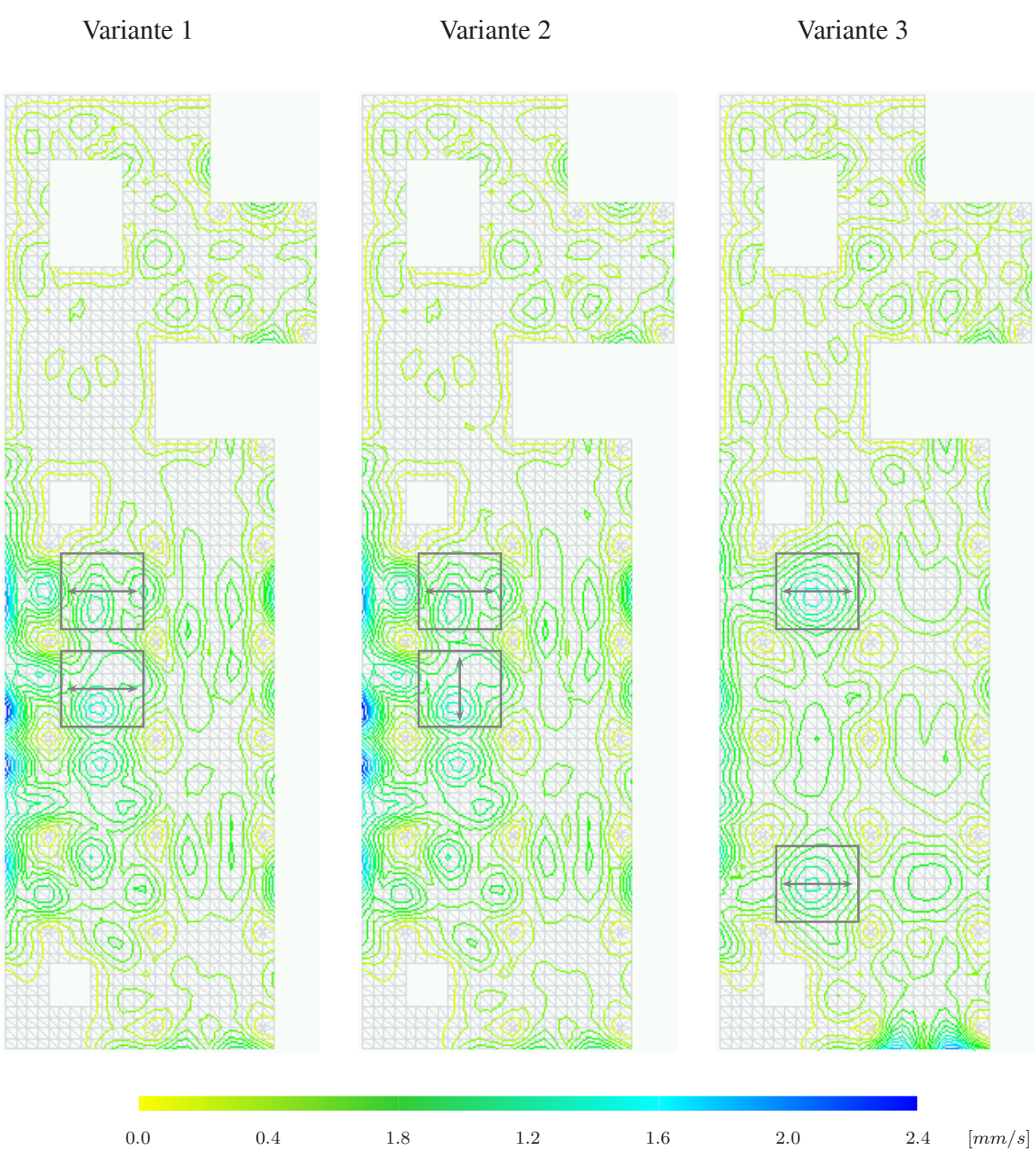


Bild 5.18: Maximalwerte der Schwinggeschwindigkeit v_z

Kapitel 6

Abschließende Betrachtungen

6.1 Zusammenfassung und abschließende Bewertung

Die Bearbeitung von Ingenieuraufgaben erfordert häufig die Bestimmung einer größeren Anzahl von Eigenwerten und Eigenvektoren großer symmetrischer Profilmatrizen. Je nach Art und Umfang der Aufgabenstellung werden dabei Untersuchungen vorgenommen, die eine teilweise oder vollständige Bestimmung der Eigenwerte und Eigenvektoren erfordern. Schwingungs- und Stabilitätsuntersuchungen an Bauwerken und Tragkonstruktionen gehören zu den klassischen Aufgabenbereichen im Bauwesen, die als Teil der Aufgabenstellung die Lösung einer allgemeinen oder speziellen Eigenwertaufgabe beinhalten.

Die etablierten Methoden zur Lösung der Eigenwertaufgabe vereinen nur selten ein hohes Maß an Flexibilität, eine hohe Zuverlässigkeit der Lösung, hohe Genauigkeit und akzeptablen Aufwand. Diese Eigenschaften sind die Zielvorgabe bei der Entwicklung und Untersuchung eines neuen Verfahrens zur Lösung der speziellen Eigenwertaufgabe im Bauwesen.

Die Inverse Matrixiteration ist eine neues, profilerhaltendes Verfahren zur Lösung der speziellen Eigenwertaufgabe großer symmetrischer Profilmatrizen. Das Verfahren basiert auf dem bekannten QR-Algorithmus zur Bestimmung der Eigenzustände vollbesetzter Matrizen. Durch eine Reihe von Erweiterungen wurde eine Methode entwickelt, die sich durch ein hohes Maß an Flexibilität und Zuverlässigkeit hinsichtlich des Berechnungsablaufs und hinsichtlich der Eigenschaften der Lösung, auszeichnet. Die Methode bestimmt eine beliebige Anzahl Eigenwerte und Eigenvektoren unabhängig voneinander und verfügt damit über die Möglichkeit im Rahmen einer ingenieurmäßigen Untersuchung einer Aufgabe gezielt und selektiv die erforderlichen Informationen zu ermitteln. Die Eigenwerte konvergieren schrittweise und in sortierter Reihenfolge, beginnend mit dem betragskleinsten Eigenwert. Dies ermöglicht den effektiven Einsatz einer Spektralverschiebung zur Bestimmung von Eigenwerten in beliebigen Bereichen des Eigenwertspektrums der Aufgabe. Darüberhinaus wird die Spektralverschiebung zur Verbesserung des Konvergenzverhaltens gewinnbringend eingesetzt. Durch eine kontinuierliche Deflation der Matrix wird die Dimension der Aufgabe stetig reduziert. Der numerische Aufwand des Verfah-

rens nimmt mit zunehmender Anzahl bestimmter Eigenwerte kontinuierlich ab. Das Verfahren bestimmt betragsgleiche Eigenwerte mit gleichen und unterschiedlichen Vorzeichen und Cluster schlecht getrennter Eigenwerte mit hoher Genauigkeit und Zuverlässigkeit.

Für die Entwicklung und Erweiterung des Verfahrens wurde die Mathematik zur Lösung der speziellen Eigenwertaufgabe auf der Grundlage von Ähnlichkeitstransformationen zusammengestellt. Zur Beurteilung der numerischen Eigenschaften und Verhaltensweisen der Iteration wurde schrittweise eine vollständige Beweisführung vorgenommen. Für die Beurteilung der Berechnungsergebnisse wurden Fehlerschranken abgeleitet und soweit erforderlich durch eine Beweisführung verifiziert.

Die Erweiterungen des QR-Verfahrens wurden schrittweise eingeführt und untersucht. Durch den Profilerhalt der Iteration kann das Verfahren alle herkömmlichen Aufgabenstellungen effektiv bearbeiten. Darüberhinaus wird durch den Profilerhalt die Bearbeitung von Aufgaben sehr großer Dimension mit einem über den Iterationsverlauf konstanten Speicherbedarf ermöglicht. Damit erfüllt das Verfahren ein äußerst wichtiges Kriterium zur Beurteilung der Leistungsfähigkeit. Untersuchungen mit Vergleichsverfahren zeigen, daß die populären und etablierten Subraumverfahren über diese wichtige Eigenschaft nicht verfügen und damit an der Bestimmung einer größeren Anzahl von Eigenwerten scheitern.

Die kontinuierlich veränderte Spektralverschiebung der Matrix während der Iteration beschleunigt wesentlich die Konvergenz, ohne zusätzlichen multiplikativen Aufwand. Der Einsatz einer Spektralverschiebung zur Konvergenzbeschleunigung wird auch von Subraummethoden genutzt, ist aber durch die erforderliche Zerlegung der spektralverschobenen Matrix mit jeder Änderung des Verschiebungsparameters von einem erheblichen numerischen Mehraufwand gekennzeichnet. Der Verschiebungsparameter wird daher im allgemeinen zumindest über einen Teil der Berechnung konstant gehalten. Das Konvergenzverhalten kann damit nur teilweise verbessert werden.

Mit jedem konvergierten Eigenwert ist implizit ein Eigenzustand bekannt, um den die iterierte Matrix deflationiert wird. Da die Eigenwerte in der letzen Zeile und Spalte der Matrix konvergieren wird die effektive Dimension für die nachfolgenden Iterationen jeweils um eins reduziert. Der Aufwand der Iteration verringert sich kontinuierlich, die Effektivität des Verfahrens steigt. Für die Subraumverfahren ist eine Deflation unbedingt erforderlich, falls betragsgleiche Eigenwerte Teil des zu bestimmenden Spektrums sind. Eine Verringerung des numerischen Aufwands erfolgt dabei nicht bzw. nur unwesentlich.

Zwei wesentliche Verbesserungen des Konvergenzverlaufs der Iteration wurden durch eine wiederholte Prekonditionierung der Matrix und die Jacobi-Randkorrektur erreicht. Beide Erweiterungen sind profilerhaltend und werden lokal zur Vermeidung einer Stagnation der Konvergenz eingesetzt. Die Prekonditionierung fördert eine rasche und hohe Diagonaldominanz in den Bereichen der Matrix mit fortgeschrittener Konvergenz. Die Sortierung der Eigenwerte wird dadurch global nicht beeinflußt, lokal aber stark begünstigt. Die Problemkoeffizienten der Iteration, die teilweise eine erhebliche Stagnation im Konvergenzverlauf verursachen, werden vollständig aus der Berechnung entfernt. Bereiche der Matrix, die die Prekonditionierung infolge des Matrixprofils nicht oder nur unzureichend erfaßt, werden durch die Jacobi-Randkorrektur effektiv besei-

tigt. Die Jacobi-Randkorrektur ermöglicht eine gezielte Behandlung von Problemkoeffizienten und verbessert das Konvergenzverhalten maßgeblich und nachhaltig. Beide Verfahren wurden mathematisch beschrieben. Die Funktions- und Wirkungsweise wurde anhand eines Beispiels aufgezeigt und durch mehrere Vergleichsrechnungen an Matrizen großer Dimension verifiziert. Der Aufwand der beiden Verfahren ist gemessen am Gesamtaufwand der Iteration klein bis vernachlässigbar klein.

Mit den bestimmten Eigenwertnäherungen werden selektiv die Eigenvektoren bestimmt. Im Falle mehrfacher Eigenwerte reicht teilweise eine QR-Zerlegung zur Bestimmung der dazugehörigen Eigenvektornäherungen. Um eine hohe Zuverlässigkeit der Lösung auch bei schlecht getrennten Eigenwerten zu gewährleisten wurde eine lokale Iteration eingeführt, die teilweise eine zusätzliche QR-Zerlegung vermeidet und damit den Aufwand erheblich reduziert, die Genauigkeit der Ergebnisse aber nicht negativ beeinflußt. Damit ist die selektive Eigenvektorbestimmung mit der Inversen Matrixiteration der Eigenvektorbestimmung mit der Inversen Vektoriteration, hinsichtlich der Zuverlässigkeit und der Genauigkeit der Ergebnisse, trotz eines erhöhten numerischen Aufwands, klar überlegen. Das Orthogonalitätslevel der Eigenvektoren ist im allgemeinen für beide Verfahren gleichwertig, muß aber bei der Inversen Vektoriteration häufig explizit durch Reorthogonalisierungsprozesse sichergestellt werden und kann nicht in jedem Fall garantiert werden. Sind die Eigenwertnäherungen nur schlecht bestimmt, kann eine Konvergenz zum falschen Eigenvektor teilweise nicht verhindert werden. Die Eigenvektorbestimmung mit der Inversen Matrixiteration erweist sich als wesentlich stabiler und verfügt über Mechanismen zur Nachiteration, die häufig numerisch günstig und lokal begrenzt bleiben. Eine Nachiteration mit vollen QR-Zerlegungen zur Verbesserung der Eigenwertnäherung bzw. zur Sicherstellung der Orthogonalität unter den Eigenvektornäherungen sind numerisch teuer, aber im Vergleich zum Reorthogonalisierungsprozess der Inversen Vektoriteration zuverlässig im Ergebnis. Die Genauigkeit der berechneten Eigenvektornäherungen war in allen Fällen besser als für die mit der Inversen Vektoriteration bestimmten Näherungen. Ein aufwendiges Monitoring ist nicht erforderlich.

Das entwickelte Verfahren wurde in einer objektorientierten Entwicklungsumgebung implementiert. Existierende Softwareumgebungen zur Entwicklung und Forschung im Bereich der Linearen Algebra stammen häufig aus dem Gebiet der Mathematik und verfolgen keinen objektorientierten Ansatz. Für die rechnergestützte Bearbeitung von Aufgaben des Bauingenieurwesens hat sich die objektorientierte Vorgehensweise jedoch als zweckmäßig und notwendig erwiesen. Objektorientierte Softwarepakete, die eine Entwicklung und Untersuchung neuartiger Konzepte auf dem Gebiet der Bauinformatik unterstützen, sind derzeit noch unzureichend verbreitet oder bieten keine ausreichende Flexibilität. Für die Umsetzung und Untersuchung des neu entwickelten Verfahrens und der Vergleichsverfahren wurde ein Softwarepaket entwickelt und implementiert, das gestützt auf die Finite-Element-Methode als Untersuchungs- und Entwicklungsumgebung zur Bearbeitung numerischer Problemstellungen dient. Die Implementierung und ihre Konzepte wurden detailliert dokumentiert.

Zur Beurteilung und Einordnung der Inversen Matrixiteration wurden zwei Vergleichsverfahren implementiert, die den Entwicklungsstand der Methoden zur Lösung der reellen, symmetrischen

Eigenwertaufgabe wiederspiegeln. Das implementierte Givens-QRI-Verfahren ist eine der Standardmethoden zur vollständigen Bestimmung der Eigenwerte und zur selektiven Bestimmung der Eigenvektoren. Das IRL-Verfahren ist ein Subraumverfahren, das auf dem Algorithmus von Lanczos basiert. Beide Verfahren sind Teil der großen Softwarepakete der Linearen Algebra. Die Theorie beider Verfahren wurde dokumentiert und der numerische Aufwand abgeschätzt.

Untersucht wurde der numerische Aufwand der Verfahren, der Speicherbedarf, das Konvergenzverhalten, sowie die Zuverlässigkeit und Genauigkeit der Berechnungsergebnisse. Der Vergleich zeigt sehr unterschiedliche Ergebnisse. Der Vergleich des numerischen und zeitlichen Aufwands der Berechnungen zeigt, daß für eine kleine bis moderate Anzahl von Eigenwerten die Lanczos-Implementierung sehr effektiv ist. Der numerische und zeitliche Aufwand ist gering und die Berechnungsergebnisse verfügen über eine ausreichende Genauigkeit. Mit den implementierten Deflationsprozessen werden auch mehrfache Eigenwerte zuverlässig approximiert. Das Verfahren verliert für eine größere Anzahl von Eigenwerten jedoch sehr schnell an Wirtschaftlichkeit. Durch den Aufbau eines Subraums der in seiner Dimension im allgemeinen um den Faktor 2 größer ist als die Anzahl gesuchter Eigenwerte, erfordert das Verfahren sehr schnell ein Vielfaches des Speicherbedarfs der Inversen Matrixiteration. Damit steigt auch der operative Aufwand und wird durch die wiederholte Reorthogonalisierung der Subraumbasis dominiert. Die Anzahl der Eigenwerte, die das Verfahren bestimmen kann, wird damit wesentlich eingeschränkt. Das Konvergenzverhalten wird stark von ad-hoc Entscheidungen geprägt die als a priori Informationen in die Berechnung einfließen. Eine Wiederholung der Berechnung mit veränderten Randbedingungen war in mehreren Fällen erforderlich.

Wird eine große Anzahl Eigenwerte gesucht, zeigt der Vergleich, daß häufig die vollständige Bestimmung des Gesamtspektrums mit dem Givens-QR-Verfahren günstiger ist, als eine Berechnung mit Lanczos oder der Inversen Matrixiteration. Der Hauptaufwand des Givens-QR-Verfahren wird durch die Tridiagonalisierung in einer endlichen Anzahl von Schritten vorgenommen. Die Inverse Matrixiteration ist ein iteratives Verfahren und beinhaltet eine relativ teure Zerlegung der Matrix in jedem Iterationszyklus. Trotz der wesentlichen Verbesserungen des Verfahrens im Konvergenzverhalten bleibt das Verfahren im operativen Vergleich zur vollständigen Bestimmung der Eigenwertaufgabe dem Givens-QR-Verfahren unterlegen. Der Speicheraufwand der beiden Verfahren ist annähernd gleichwertig.

Der Verfahrensvergleich läßt zwei wesentliche Stärken des neu entwickelten Verfahrens feststellen, die einen Einsatz für die Aufgabenstellungen des Bauingenieurwesens rechtfertigen.

Bezogen auf den Aufwand positioniert sich das Verfahren klar zwischen dem Lanczos-Verfahren und dem Givens-QR-Verfahren. Nach unten hin wird der Einsatz der Methode durch den sehr guten numerischen Aufwand des Lanczos-Verfahrens begrenzt, nach oben durch den Aufwand des Givens-QR-Verfahren. Dazwischen liegt ein Bereich, in dem die Inverse Matrixiteration dem Speicheraufwand des Lanczos-Verfahrens und dem numerischen Aufwand des Givens-QR-Verfahren, überlegen ist. Dabei handelt es sich um einen Bereich der von keinem der beiden Vergleichsverfahren wirtschaftlicher bestimmt werden kann. Das Verfahren ist in der Lage diese Anzahl Eigenwerte in beliebigen Bereichen des Eigenwertspektrums zu ermitteln. Für eine kleine Anzahl von Eigenwerten ist der zeitliche Aufwand, trotz der Überlegenheit des Lanczos-

Verfahrens, immer noch in einem akzeptablen Bereich. Nach oben hin bleibt die Anzahl der gewünschten Eigenwerte, trotz der Überlegenheit des Givens-QR-Verfahren, unbeschränkt.

Ein zweiter, wichtiger Aspekt zur Beurteilung des Verfahrens ist die Zuverlässigkeit. Die Iteration ist in jeder Form stabil und terminiert selbständig. Damit unterscheidet sich die Methode wesentlich vom Lanczos-Verfahren, daß zur zuverlässigen Bestimmung der Eigenwerte ein Monitoring des Berechnungsablaufs und der Berechnungsergebnisse erfordert. Die hohe Zuverlässigkeit und Stabilität, aber auch die Genauigkeit der Berechnungsergebnisse sind eine klare Abgrenzung zur Eigenvektorbestimmung mit der Inversen Vektoriteration. Insbesondere bietet die Inverse Matrixiteration im Vergleich zur Inversen Vektoriteration eine uneingeschränkte Bestimmung aller Eigenvektoren bei guten Eigenschaften hinsichtlich der Orthogonalität zwischen den Vektoren. Für die vollständige Bestimmung der Eigenwerte und eine selektive Bestimmung der Eigenvektoren ist eine Kombination der Inversen Matrixiteration und dem Givens-QR-Verfahren denkbar.

Für das neu entwickelte Verfahren wurde mit der Untersuchung des Schwingungsverhaltens von Bauwerken ein Anwendungsfall beispielhaft dokumentiert. Die dynamischen Bestimmungsgleichungen der Aufgabe und die zu lösende Eigenwertaufgabe wurden hergeleitet. Zur Untersuchung von Bauwerken auf ihr dynamisches Verhaltens wurde die modale Analyse formuliert und am Beispiel einer Deckenplatte unter dynamischer Maschinenlast angewandt. Die Auswirkungen der Schwingungsbelastung auf die Konstruktion der Deckenplatte und auf die von der Schwingungsbelastung betroffenen Personen wurden nach gültigen Richtlinien untersucht und bewertet.

6.2 Ausblick

Trotz der wesentlichen Verbesserung des Konvergenzverhaltens bei der Bestimmung der Eigenwerte ist die Inverse Matrixiteration für eine sehr große Anzahl von Eigenwerten noch immer numerisch teuer. Selbst eine Leistungssteigerung des Computers in 10 Jahren um den Faktor 100 [1] führt für Matrizen großer Dimension bei der Berechnung vieler Eigenwerte nicht zu einer befriedigenden Rechenzeit. Es bleibt zu untersuchen, ob das vorgestellte Verfahren durch sogenannte *Schnelle Givens-Transformationen* verbessert werden kann. Die Systemmatrix $\mathbf{A}$ wird dabei in faktorisierter Form $(\mathbf{D}\hat{\mathbf{A}})$ iteriert. $\mathbf{D}$ ist dabei eine reguläre Diagonalmatrix, die das Produkt $\mathbf{R}_{ik}^T\mathbf{D}\hat{\mathbf{A}}$ zur Elimination einzelner Koeffizienten so vereinfacht, daß sich die Anzahl Multiplikationen von 4 auf 2 halbiert. Der vollständige Prozess ist in [39] gezeigt. Der Nachteil dieser Methode ist, daß die Speicherung der Information über die durchgeführte Rotation zwei Speicherstellen erfordert. Der Speicherbedarf steigt damit um bN Koeffizienten. Dies bedeutet eine wesentliche Verschlechterung des Verfahrens. Außerdem sind die durchgeführten Rotationen auf $\hat{\mathbf{A}}$ nicht mehr orthogonal und verfügen damit über andere numerische Eigenschaften hinsichtlich der Genauigkeit. Der Aufwand für die Prekonditionierung würde durch die faktorisierte

[1] $\sim 2^{(10\cdot 12/18)}$ *Moore's Law :* Verdoppelung der Transistorzahl pro Chip, alle 18 Monate

Form ansteigen. Der Einsatz einer Jacobi-Randkorrektur kann wegen der erforderlichen Rücktransformation nicht effektiv erfolgen. Eine Untersuchung zur Lösung der aufgeführten Problemstellungen umfaßt auch eine kritische Beurteilung der numerischen Stabilität im Vergleich zum vorhandenen Verfahren. Für die Eigenvektorberechnung ist die *Schnelle Givens-Transformation* ungeeignet, da pro Eigenwert im allgemeinen nur eine Zerlegung erforderlich ist und damit der Aufwand sowohl zur Speicherung der Rotationsinformationen, als auch zur Bestimmung der Rotationsparameter aus der faktorisierten Form, dem vorhandenen Verfahren stark unterlegen ist.

Für die Bestimmung des vollständigen Eigenwertspektrums und einer selektiven Auswahl der Eigenvektoren wird ein gemischtes Verfahren vorgeschlagen. Eine effektive Eigenwertbestimmung mit dem Givens-QR-Verfahren, gekoppelt mit den guten numerischen Eigenschaften der Inversen Matrixiteration zur Eigenvektorbestimmung verspricht eine leistungsfähige Methode auch für Matrizen großer Dimension zu sein. Zu untersuchen bleibt, ob die maximal erreichbare Genauigkeit des Givens-QR-Verfahrens in jedem Fall für den zuverlässigen Einsatz der Inversen Matrixiteration zur Eigenvektorbestimmung ausreicht.

Bei den durchgeführten Untersuchungen mit der Inversen Matrixiteration wurde teilweise mit skalierten Matrizen gerechnet. Eine kleine Matrixnorm reduziert nach Kapitel 3.3.3 den Fehler in der Eigenwertnäherung. Die hohe Genauigkeit der skalierten Eigenwertnäherung geht bei der Rückskalierung vollständig verloren und kann bei Verwendung eines unzweckmäßigen Skalierungsfaktors das Ergebnis wesentlich verschlechtern. Gelingt es jedoch eine geeignete Skalierung vorzunehmen, die dem Berechnungsergebnis der unskalierten Berechnung gleichwertig ist, so kann die hohe Genauigkeit der skalierten Eigenwertnäherungen zweckmäßig für die Berechnung der Eigenvektoren verwendet werden. Wegen der freien Skalierbarkeit der Eigenvektoren, sind diese in ihrer Richtung identisch mit den Eigenvektoren der unskalierten Aufgabe. Die skalierten Eigenwertnäherungen vereinfachen zusätzlich eine sichere und numerisch einfache Bestimmung der Multiplizität von geclusterten Eigenwerten. Die zweckmässige Wahl eines Skalierungsfaktors und die damit verbundenen numerischen Eigenschaften der Iteration sind zu untersuchen.

Die Analyse zur Eigenvektorberechnung mit der Inversen Matrixiteration hat gezeigt, daß der Fehler im Fall einer großen Restnorm $\|\mathbf{r}\|_2 (= \|\mathbf{Ax} - \lambda\mathbf{x}\|_s)$ immer in den letzten Koeffizienten des Restvektors auftritt und damit aus den letzten Koeffizienten des Eigenvektors stammt. Die Eigenvektorberechnung beginnt mit den letzten Rotationsmatrizen der Zerlegung $\mathbf{A} = \mathbf{QR}$. Die fehlerbehafteten Koeffizienten am Ende des Eigenvektors werden durch diese Rotationsmatrizen bestimmt. Diese Beobachtung ist gleichbedeutend mit der Tatsache, daß in den Fällen schlechter Konvergenz, nach der Zerlegung keine Nullzeile am Ende der Rechtsdreiecksmatrix $\mathbf{R}$ existiert. Es bleibt zu untersuchen, ob anstelle einer Konvergenzverbesserung in der Zerlegung, eine Korrektur der wenigen Stellen des Eigenvektors $(i.a. \leq 3)$ mit weit weniger Aufwand zu einem gleichwertigen Ergebnis führt. Eine mögliche Korrektur könnte durch eine Orthogonalisierung zu einem guten Eigenvektor erfolgen, die lediglich Einfluß auf die fehlerbehafteten Koeffizienten nimmt.

Mit der Lösung der speziellen Eigenwertaufgabe kann nur ein Teilbereich der Aufgabenstellungen im Ingenieurwesen bearbeitet werden. Stabilitätsuntersuchungen an Tragwerken erfor-

dern im allgemeinen Berechnungen am verformten System, aus denen verschiebungsabhängige Steifigkeiten resultieren. Die Bestimmung der kritischen Systembelastung oder die Berücksichtigung imperfekter Geometrie erfordern häufig die Lösung einer allgemeinen Eigenwertaufgabe, die nicht profilerhaltend in eine spezielle Eigenwertaufgabe transformiert werden kann. Eine effektive Lösung der allgemeinen Eigenwertaufgabe mit einem Verfahren, das auf der Inversen Matrixiteration basiert, scheitert möglicherweise am numerischen Aufwand. Eine vorteilhafte Prekonditionierung des Gleichungssystems mit den profilerhaltenden Iterationen der Inversen Matrixiteration oder eine schrittweise profilerhaltende Überführung der Aufgabe in eine spezielle Eigenwertaufgabe sind aber denkbar und zu untersuchen.

Untersuchungen an räumlichen Stabwerken haben gezeigt, daß teilweise sehr hohe Frequenzen, die in den üblichen Stabilitätsuntersuchungen unberücksichtigt bleiben, zu starken Schwingungen einzelner Stäbe anregen. Ein Systemversagen infolge lokalen Bruchversagens bzw. eine erhebliche lokale Schädigung infolge Einzelstabschwingens sind auszuschließen. Eine systematische Untersuchung dieses Phänomens und seinen möglichen Auswirkungen kann nur mit der Kenntnis des vollständigen Eigenwertspektrums bzw. der Kenntnis aller Eigenformen zuverlässig erfolgen.

Literaturverzeichnis

[1] Argyris, J., Mlejnek, H.-P.
Die Methode der Finiten Elemente in der elementaren Strukturmechanik
Band II - Kraft- und gemischte Methoden, Nichtlinearitäten
Vieweg & Sohn, Braunschweig 1987

[2] Argyris, J., Mlejnek, H.-P.
Die Methode der Finiten Elemente in der elementaren Strukturmechanik
Band III - Einführung in die Dynamik
Vieweg & Sohn, Braunschweig 1987

[3] Bathe K.-J.
Finite-Elemente-Methoden
2., vollständig überarbeitete und erweiterte Auflage
Springer Verlag Berlin Heidelberg 2002

[4] Bischof C.H.
A Framework for Symmetric Band Reduction and Tridiagonalization
Mathematics and Computer Science Division
Argonne National Laboratory, Argonne

[5] Calvetti D., Reichel L., Sorensen D.C.
An Implicitly Restarted Lanczos Method For Large Symmetric Eigenvalue Problems
Electronic Transactions on Numerical Analysis
Volume 2, pp.1-21
Kent State University, March 1994

[6] Clough R.W., Penzien J.
Dynamics of Structures
McGraw-Hill, Inc, 1975

[7] Cook D.R., Malkus D.S., Plesha M.E.
Concepts and Applications of Finite Element Analysis
Third Edition
John Wiley & Sons, Inc 1988

[8] DIN 4150
Erschütterungen im Bauwesen
Teil 2 : Einwirkungen auf Menschen in Gebäuden
Beuth Verlag GmbH, Berlin
Juni 1999

[9] DIN 4150
Erschütterungen im Bauwesen
Teil 3 : Einwirkungen auf bauliche Anlagen
Beuth Verlag GmbH, Berlin
Februar 1999

[10] Dhillon I.S.
Current inverse iteration software can fail
IBM Almaden Research Center
San Jose, California
1998

[11] Dhillon I.S.
A New (N^2) Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem
PhD thesis
University of California, Berkeley
1999

[12] Fowler M., Scott K.
UML Distilled
Sencond Edition
Addison Wesley
August 1999

[13] Golub G.H., Van Loan C.F.
Matrix Computations
Second Edition, Forth Printing
The John Hopkins University Press
Baltimore and London, 1993

[14] Golub G.H., van der Vorst H.A.
Eigenvalue Computation in the 20th Century
Journal of Computational and Applied Mathematics
Volume 123(2000), pp. 35-65
Elsevier Science Publisher B.V., 2000

[15] Hanff J.
Abhängigkeiten zwischen Objekten in ingenieurwissenschaftlichen Anwendungen
genehmigte Dissertation zum Doktor der Ingenieurwissenschaften

Technische Universität Berlin
Shaker Verlag
Aachen, 2003

[16] Hartmann D. (Hrg)
Objektorientierte Modellierung in Planung und Konstruktion
Forschungsbericht der Deutschen Forschungsgemeinschaft (DFG)
Wiley-VCH GmbH
Weinheim, 2000

[17] Kuijlaars A.B.J.
Which Eigenvalues are found by the Lanczos Method?
SIAM Journal on Matrix Analysis and Applications
June 2000

[18] Lang B.
Efficient Algorithms for Reducing Banded Matrices to Bidiagonal and Tridiagonal Form
Bergische Universität GH Wuppertal
Januar 1997

[19] Lehoucq R.B., Sorensen D.C., Yang C.
ARPACK Users' Guide
Rice University, Houston
Oktober 1997

[20] Melzig-Thiel R., Kinne J., Schatte M.
Schwingungsbelastung in der Bauwirtschaft
Bundesanstalt für Arbeitsschutz und Arbeitsmedizin
Gruppe AS 4.2 "Technischer Schwingungsschutz"
Dortmund, Dresden, 2001

[21] Müller G., Möser M.
Taschenbuch der Technischen Akustik
Springer-Verlag
Berlin, 2004

[22] Noble B.
Applied Linear Algebra
Prentice-Hall, Inc.,
Englewood Cliffs, N.J. 1969

[23] Pahl P.J.
Nichtlineare Elastizitätstheorie
als Skript veröffentlicht, Institut für Allgemeine Bauingenieurmethoden,
Technische Universität Berlin, Juni 2000

[24] Pahl P.J.
Eigenstates of Large Systems
Vortrag an der Tsinghua University,
Beijing, 1998

[25] Pahl P.J.
Algebraische Algorithmen
als Skript veröffentlicht, Institut für Allgemeine Bauingenieurmethoden,
Technische Universität Berlin, April 1994

[26] Pahl P.J.
Algebraische Methoden
als Skript veröffentlicht, Institut für Allgemeine Bauingenieurmethoden,
Technische Universität Berlin, April 2002

[27] Pahl P.J.
Plates-Finite Element Theory and Computer Implementation
als Skript veröffentlicht, Institut für Allgemeine Bauingenieurmethoden,
Technische Universität Berlin, Januar 2001

[28] Pahl P.J., Ruess M.
Eigenstates of Large Profiled Matrices
Invited Lecture at SEMC 2001,
Cape Town, 2001

[29] Pahl P.J., Werner H.
Computing in Civil and Building Engineering
Proceedings of the Sixth International Conference on Computing in Civil and Building Engineering
Berlin, 1995

[30] Parlett B.N.
The Symmetric Eigenvalue Problem
Society for Industrial and Applied Mathematics (SIAM)
Prentice-Hall, Englewood Cliffs, NJ, 1980

[31] Petersen C.
Dynamik der Baukonstruktionen
Vieweg Verlagsgesellschaft mbH
Braunschweig/Wiesbaden 1996

[32] Ruess M., Pahl P.J.
Die Bestimmung von Eigenzuständen mit dem Verfahren der Inversen Matrizeniteration
Vortrag und Veröffentlichung,
Internationales Kolloquium über die Anwendungen der Informatik und der Mathematik in

Architektur und Bauwesen (IKM),
Bauhaus-Universität Weimar,
Juni 2000

[33] Ruess M.
Matrixiteration For Large Symmetric Eigenvalue Problems
Vortrag und Veröffentlichung,
The Ninth International Conference on Computing in Civil and Building Engineering,
Taipei, Taiwan
April 2002

[34] Ruess M.
FELINA-Finite Elements for Linear and Nonlinear Analysis
Application Programming Interface, Software Documentation
Institut für Allgemeine Bauingenieurmethoden,
Technische Universität Berlin, April 2004

[35] Saad, Y.
Numerical Methods For Large Eigenvalue Problems
Manchester University Press Series in Algorithms for Advanced Scientific Computing
December 1991

[36] Simon H., Kasheng W.
Thick-Restart Lanczos Method for Symmetric Eigenvalue Problems
Lawrence Berkeley National Laboratory/NERSC
Berkeley, CA, February 1998

[37] Sorensen D.C.
Implicitly Restarted Arnoldi/Lanczos Methods For Large Scale Eigenvalue Calculations
Departement of Computational and Applied Mathematics
Rice University, Houston

[38] Sorensen D.C.
Deflation For Implicitly Restarted Arnoldi Methods
Center for Research on Parallel Computation
Rice University, Houston, 1998

[39] Schwarz H.R.
Numerische Mathematik
B.G. Teubner, Stuttgart 1997

[40] Simon H.D., Lewis J.H., Grimes R.G.
A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems
SIAM Journal on Matrix Analysis and Applications
1994

[41] Stewart G.W.
Matrix Algorithms, Vol I: Basic Decompositions
Society for Industrial and Applied Mathematics (SIAM),
Philadelphia, 1998

[42] Stewart G.W.
Matrix Algorithms, Vol II: Eigensystems
Society for Industrial and Applied Mathematics (SIAM),
Philadelphia, 2001

[43] Stoer J., Bulirsch R.
Numerische Mathematik 2
3. Auflage
Springer-Verlag, Berlin - Heidelberg, 1990

[44] van der Vorst H.A.
Computational Methods for Large Eigenvalue Problems
Mathematical Institute
Utrecht University
Utrecht, Netherlands, 2000

[45] van der Vorst H.A., Golub G.H.
150 Years old and still alive : Eigenproblems
Veröffentlichung zum 10. Todestag von J.H. Wilkinson
in *The State of the Art in Numerical Analysis*, Duff I.S., Watson G.A.(eds)
Clarendon Press, Oxford, 1997

[46] VDI-Richtlinie 2057
Einwirkung mechanischer Schwingungen auf den Menschen
Blatt 1 - Ganzkörper-Schwingungen
VDI-Gesellschaft Entwicklung Vertrieb
Fachbereich Schwingungstechnik
Beuth Verlag GmbH
Berlin, 2002

[47] Wilhelms G., Kopp M.
JAVA professionell
MITP-Verlag GmbH, Bonn, 1999

[48] Wilkinson J.H.
The Algebraic Eigenvalue Problem
Clarendon Press, Oxford, 1965

[49] Wilkinson J.H., Reinsch C.
Handbook for Automatic Computation, Vol.2, Linear Algebra
Springer Verlag, Heidelberg - Berlin - New York, 1971

[50] Zienkiewicz O.C.
The Finite Element Method in Engineering Science
McGraw-Hill Publishing Company Limited, London, 1971

[51] http://www.netlib.org/benchmark/linpackjava

Index