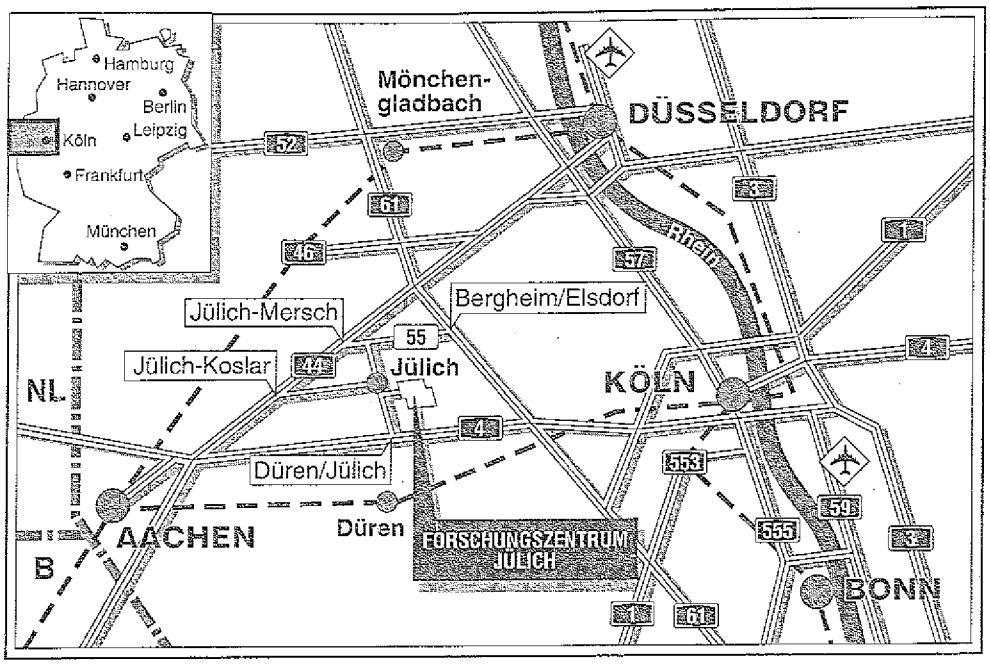
FORSCHUNGSZENTRUM JÜLICH GmbH

Zentralinstitut für Angewandte Mathematik

Multisektionsverfahren zur Bestimmung der Eigenwerte von Tridiagonalmatrizen auf CRAY-Multiprozessorrechnern

Achim Basermann



Berichte des Forschungszentrums Jülich ; 2427
ISSN 0366-0885
Zentralinstitut für Angewandte Mathematik Jül-2427

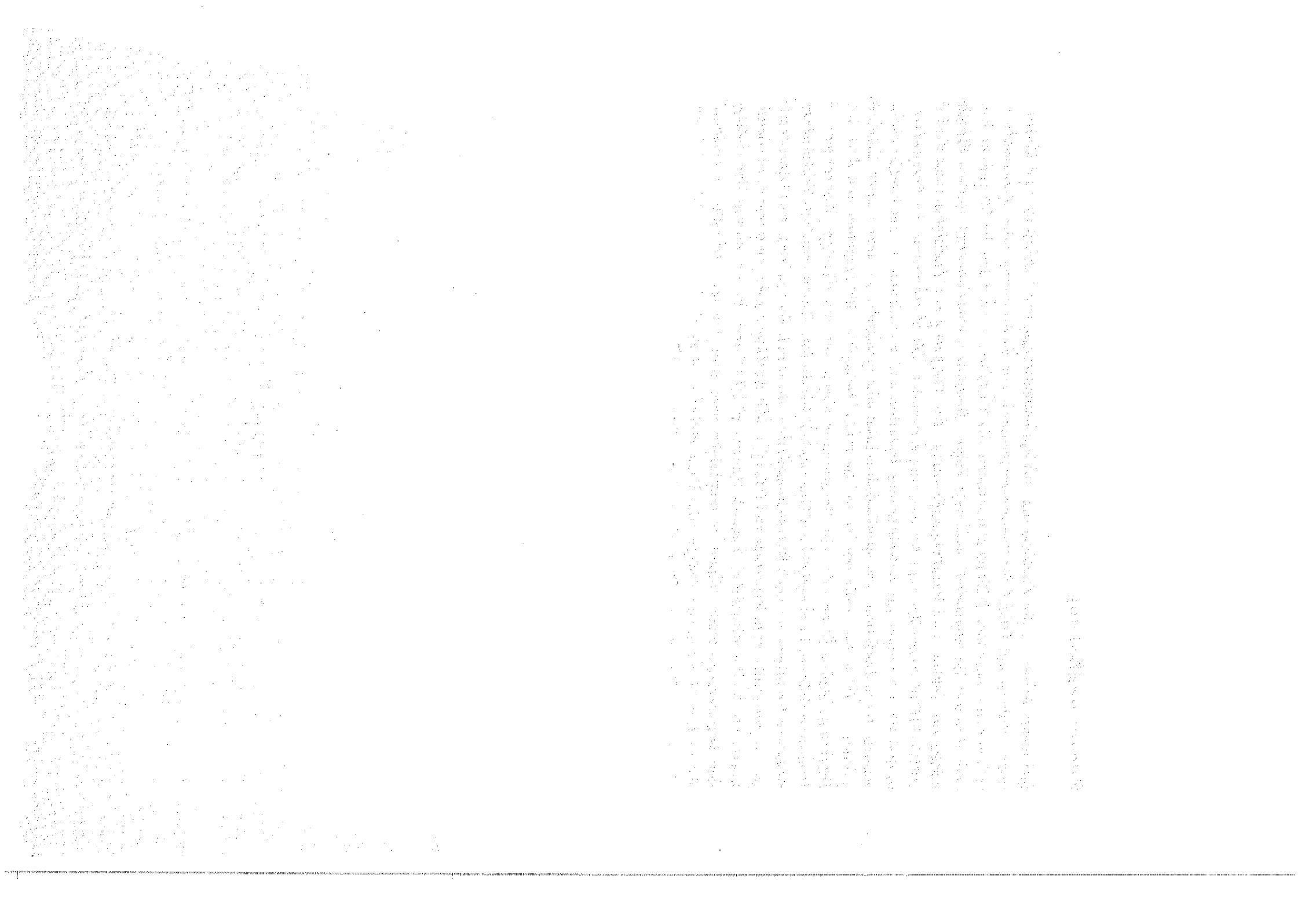
Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek
Postfach 1913 · D-5170 Jülich · Bundesrepublik Deutschland
Telefon: 02461/61-6102 · Telefax: 02461/61-6103 · Telex: 833556-70 kfa d

Multisektionsverfahren zur Bestimmung der Eigenwerte von Tridiagonalmatrizen auf CRAY-Multiprozessorrechnern

Achim Basermann

Zusammenfassung

Ziel dieser Arbeit ist die Entwicklung und Implementierung von Bisektions- und Multisektionsverfahren auf CRAY-Multiprozessorrechnern zur Bestimmung aller Eigenwerte von reellen symmetrischen Tridiagonalmatrizen; die Verfahren verwenden die Sturmsche Kette oder eine aus ihr abgeleitete rationale Folge. Besonders berücksichtigt werden Möglichkeiten zur Vektor- und Parallelverarbeitung. Als Werkzeug zur Parallelisierung wird das CRAY-Multitasking-Konzept Autotasking verwendet. Die Bestimmung der Eigenwerte ist in drei Phasen aufgeteilt: Initialisierung, Isolation und Extraktion. In der Isolationsphase werden durch Bisektion oder Multisektion Intervalle ermittelt, die genau einen Eigenwert enthalten. Diese Intervalle werden an die Extraktionsphase übergegeben und dort durch Bisektion, Multisektion oder ein Iterationsverfahren mit superlinearer Konvergenz, das Pegasusverfahren, in vorgegebener Genauigkeit bestimmt. Durch günstige Vektorisierung sowohl in Isolations- als auch Extraktionsphase sowie durch Verwendung des Pegasusverfahrens werden auf einem Prozessor von CRAY Y-MP Ausführungszeiten erreicht, die deutlich kürzer als die der schnellsten Eispack-Routine (TQL1) sind, die zur Lösung des vorliegenden Eigenwertproblems allgemein empfohlen wird und den QL-Algorithmus verwendet. Durch Kombination von Vektor- und Parallelverarbeitung werden auf CRAY Y-MP mit 8 eingesetzten Prozessoren für große Matrizen Speedup-Werte erreicht, die höher als 7 sind.



Inhaltsverzeichnis

1 Einleitung

Für die Lösung vieler technischer und physikalischer Probleme ist es erforderlich, zu einer reellen vollbesetzten symmetrischen $n \times n$ Matrix A die Lösungen der Gleichung $Ax = \lambda x$, die sogenannten Eigenwerte λ_i und Eigenvektoren x_i, $i = 1, \ldots, n$, zu finden. Dies ist als das reelle symmetrische Eigenwertproblem bekannt. Die Eigenwerte λ_i sind als die Nullstellen des charakteristischen Polynoms $\varphi(\lambda) = \det(A - \lambda I)$ definiert. I ist hierin die Einheitsmatrix der Ordnung n.
Anwendungen treten in der Theorie mechanischer und elektrischer Systeme auf. Beispiele sind in [9], [32] und [34] zu finden.
Der erste Schritt zur Lösung des reellen symmetrischen Eigenwertproblems ist gewöhnlich die Reduktion der Matrix auf Tridiagonalgestalt durch Ähnlichkeitstransformationen. Die Eigenwerte dieser Tridiagonalmatrix können dann mit verschiedenen Methoden bestimmt werden, etwa mit dem Verfahren der Sturmschen Kette [30].
Ziel dieser Arbeit ist die Entwicklung und Implementierung von Bisektions- und Multisektionsverfahren auf CRAY-Multiprozessorrechnern zur Bestimmung aller Eigenwerte von reellen symmetrischen Tridiagonalmatrizen; darin wird die Sturmsche Kette oder eine aus ihr abgeleitete Folge verwendet. Da die CRAY-Rechner X-MP und Y-MP des Forschungszentrums Jülich Vektorrechner mit 4 bzw. 8 Prozessoren sind, ist besonders auf die Vektorisierungs- und Parallelisierungseigenschaften der Verfahren zu achten. Hinsichtlich der Parallelisierung wird eine gleichmäßige Auslastung der Prozessoren angestrebt. Als Werkzeug zur Parallelisierung wird das CRAY-Multitasking-Konzept Autotasking verwendet.
In mehreren Arbeiten wurde bereits auf die Parallelisierung von Bisektions- und Multisektionsverfahren zur Bestimmung von Eigenwerten reeller symmetrischer Tridiagonalmatrizen eingegangen (s. [4], [7], [8] und [22]). In [29] wird ein Ansatz zur Vektorisierung von Multisektionsverfahren beschrieben.
In dieser Arbeit ist die Bestimmung der Eigenwerte in drei Phasen aufgeteilt: Initialisierung, Isolation und Extraktion. In der Initialisierungsphase wird ein Intervall ermittelt, das alle Eigenwerte enthält. Ferner werden Vorbesetzungen für die Isolationsphase vorgenommen. In der Isolationsphase werden Intervalle ermittelt, die genau einen Eigenwert enthalten. Diese Intervalle werden an die Extraktionsphase übergeben. Dort werden die Eigenwerte in vorgegebener Genauigkeit bestimmt. Eigenwerte, die sich um weniger als die vorgegebene Genauigkeit unterscheiden, durchlaufen die Extraktionsphase nicht. Sie werden bereits in der Isolationsphase in vorgegebener Genauigkeit berechnet. Die Trennung

von Isolation und Extraktion der Eigenwerte wird auch in [22] vorgenommen. In [22] wird jedoch lediglich auf die Vektorisierung der Auswertung der Sturmschen Kette eingegangen. Die vorliegende Arbeit untersucht ebenfalls die Vektorisierung der Auswertung der Sturmschen Kette und greift den Ansatz zur Vektorisierung von Multisektionsverfahren aus [29] auf. Ferner werden eigene Ansätze zur Vektorisierung sowohl bei der Isolation als auch bei der Extraktion getestet. In der Extraktionsphase wird neben Bisektion und Multisektion ein Iterationsverfahren mit superlinearer Konvergenz untersucht. Ein Iterationsverfahren mit superlinearer Konvergenz wurde auch in [22] zur Extraktion der Eigenwerte eingesetzt. Da die dazu erforderlichen Funktionswerte des charakteristischen Polynoms mit Hilfe der Sturmschen Kette berechnet wurden, traten häufig Zahlenbereichsüberschreitungen auf. In der vorliegenden Arbeit wird darauf eingegangen, wie man Zahlenbereichsüberschreitungen bei der Berechnung des Funktionswertes des charakteristischen Polynoms entgegenwirken kann. Messungen auf einer CRAY X-MP/48 in [22] ergaben, daß das schnellste dort entwickelte Verfahren zur Bestimmung aller Eigenwerte bei Ausführung auf einem Prozessor ca. 1.25 mal langsamer war als die vektorisierende EISPACK-Routine TQL1, die den QL-Algorithmus verwendet und die schnellste EISPACK-Routine zur Lösung des vorliegenden Eigenwertproblems ist. Bei paralleler Ausführung auf 4 Prozessoren konnte in [22] eine Beschleunigung des Verfahrens um einen Faktor 2.7 erreicht werden. In der vorliegenden Arbeit wird untersucht, ob durch andere Vektorisierungskonzepte als in [22] ein günstigeres Zeitverhalten erreicht werden kann. Da das Vektorisierungskonzept auch das Parallelisierungskonzept beeinflußt, muß darauf geachtet werden, Vektor- und Parallelverarbeitung in günstiger Weise zu kombinieren.
In der vorliegenden Arbeit wird zunächst in einem allgemeinen Kapitel auf mathematische Grundlagen zur Bestimmung der Eigenwerte von reellen symmetrischen Tridiagonalmatrizen eingegangen. Es folgt je ein Kapitel über die CRAY-Rechner des Forschungszentrums Jülich sowie über Werkzeuge für Vektorisierung und Parallelverarbeitung auf CRAY-Rechnern. Anschließend werden die entwickelten Algorithmen beschrieben, im darauf folgenden Kapitel werden die Ergebnisse von Laufzeit- und Speedup-Messungen diskutiert.

2 Eigenwerte reeller symmetrischer Tridiagonalmatrizen

In diesem Kapitel wird auf die mathematischen Grundlagen zur Bestimmung der Eigenwerte von reellen symmetrischen Tridiagonalmatrizen eingegangen. Das Kapitel beginnt mit einem allgemeinen Abschnitt über reelle symmetrische Tridiagonalmatrizen. Im folgenden Abschnitt werden die Sturmsche Kette und ein Bisektionsverfahren zur Bestimmung der Eigenwerte von reellen symmetrischen Tridiagonalmatrizen beschrieben. Anschließend folgt ein Abschnitt über Tridiagonalmatrizen mit speziellen Eigenschaften. Danach wird in je einem Abschnitt darauf eingegangen, wie ein Intervall, das alle Eigenwerte enthält, bestimmt und der Funktionswert des charakteristischen Polynoms berechnet wird. Im letzten Abschnitt wird das Pegasusverfahren, ein Iterationsverfahren mit superlinearer Konvergenz, beschrieben.

2.1 Reelle symmetrische Tridiagonalmatrizen

2.1.1 Bezeichnungen

Reelle symmetrische Matrizen sind definiert durch:

$$A^T = A. \tag{2.1}$$

A^T ist dabei die transponierte Matrix zu A.
Hat A die Gestalt

$$A = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ 0 & \cdots & 0 & \beta_n & \alpha_n \end{pmatrix}, \tag{2.2}$$

so heißt A reelle symmetrische Tridiagonalmatrix der Ordnung n mit den Elementen der Hauptdiagonalen $a_{ii} = \alpha_i$ für $i = 1, \ldots, n$ und den Elementen der Nebendiagonalen $a_{i-1,i} = a_{i,i-1} = \beta_i$ für $i = 2, \ldots, n$.

2.1.2 Transformation einer reellen symmetrischen Matrix auf Tridiagonalgestalt

Eine Matrix B, die durch eine Ähnlichkeitstransformation der Form

$$B = T^{-1}AT \tag{2.3}$$

aus einer Matrix A entsteht, besitzt dasselbe charakteristische Polynom wie A und damit dieselben Eigenwerte λ. T ist dabei eine beliebige nichtsinguläre $n \times n$ Matrix, T^{-1} die Inverse von T. A, B sind $n \times n$ Matrizen. Ziel ist es, die Matrix A durch eine Folge von Ähnlichkeitstransformationen

$$\begin{aligned} A^{(0)} &= A \\ A^{(i)} &= T_i^{-1}A^{(i-1)}T_i, \quad i = 1, 2, \ldots \end{aligned} \tag{2.4}$$

schrittweise in eine Matrix einfacherer Gestalt zu transformieren, deren Eigenwerte man leichter bestimmen kann [31].
Ein häufig verwendetes Verfahren zur Reduktion vollbesetzter symmetrischer Matrizen auf Tridiagonalgestalt in endlich vielen Schritten ist die sogenannte Householder-Transformation. Hier werden geeignete Householder-Matrizen, die symmetrisch und orthogonal ($T^TT = I$) sind, zur Transformation benutzt. Beschreibungen und Algorithmen zu diesem Verfahren sind z.B. in [31], [23], [13] [28] und [15] zu finden.

2.1.3 Eigenschaften der Eigenwerte symmetrischer Tridiagonalmatrizen

Ist eine $n \times n$ Matrix A reell und symmetrisch, so besitzt sie nur reelle Eigenwerte λ_i, $i = 1, \ldots, n$. Setzt man voraus, daß kein Element β_i der Nebendiagonalen der reellen symmetrischen Tridiagonalmatrix A in der Gestalt (2.2) null ist, so besitzen die Eigenwerte folgende Eigenschaften:

1. Alle Eigenwerte sind reell.
2. Alle Eigenwerte als Nullstellen des charakteristischen Polynoms sind einfach.

Der Fall, daß Nullen in den Nebendiagonalen auftreten, wird in 2.3.1 diskutiert.

2.2 Sturmsche Ketten und Bisektionsverfahren

2.2.1 Definition und Eigenschaften der Sturmschen Kette

Sei $p(x)$ ein reelles Polynom n-ten Grades

$$p(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n, \quad a_0 \neq 0. \tag{2.5}$$

Die sogenannte Sturmsche Kette erlaubt Aussagen über die Anzahl und Lage der reellen Nullstellen von $p(x)$. Dabei ist die Anzahl der Vorzeichenwechsel einer mit $p_0(x) = p(x)$ beginnenden Kette von Polynomen $p_i(x)$, $i = 1, \ldots, m$, absteigenden Grades an der Stelle $x = \mu$ von Bedeutung. Verabredet wird, die Anzahl der Vorzeichenwechsel $w(\mu)$ in der Folge $\{p_i(\mu)\}_{i=0,1,\ldots,m}$ abzuzählen, ohne alle $p_i(\mu)$ mit $p_i(\mu) = 0$ zu berücksichtigen [30]. Die Sturmsche Kette ist folgendermaßen definiert [30]:

Def. 2.1 *Die Folge*

$$p(x) = p_0(x), p_1(x), \ldots, p_m(x)$$

reeller Polynome heißt eine Sturmsche Kette, falls gilt:

1. *$p_0(x)$ besitzt nur einfache Nullstellen.*
2. *sign $p_1(\xi) = -$sign $p_0'(\xi)$ für alle reellen Nullstellen ξ von $p_0(x)$.*
3. *Für $i = 1, 2, \ldots, m-1$ gilt*

   $$p_{i+1}(\xi)p_{i-1}(\xi) < 0,$$

   falls ξ reelle Nullstelle von $p_i(x)$ ist.
4. *Das letzte Polynom $p_m(x)$ ändert sein Vorzeichen nicht in ganz $\mathbb{R}$.*

Folgender Satz (s. [30]) macht Aussagen über die Anzahl der reellen Nullstellen von $p(x)$ im Intervall $[a, b[$:

Satz 2.1 *Die Anzahl der reellen Nullstellen von $p(x) = p_0(x)$ im Intervall $a \leq x < b$ ist gleich $w(b) - w(a)$, wobei $w(x)$ die Anzahl der Vorzeichenwechsel der Kette*

$$p_0(x), \ldots, p_m(x)$$

an der Stelle x ist, vorausgesetzt, daß diese Kette eine Sturmsche Kette ist.

2.2.2 Ein Bisektionsverfahren zur Bestimmung der Eigenwerte reeller symmetrischer Tridiagonalmatrizen

Satz 2.1 wird hauptsächlich dazu verwandt, um durch ein Bisektionsverfahren die Eigenwerte von reellen symmetrischen Tridiagonalmatrizen zu bestimmen.
Sei A eine reelle symmetrische Tridiagonalmatrix und A_i ihre i-te Hauptabschnittsmatrix.

$$A_i = \begin{pmatrix} \alpha_1 & \beta_2 & & 0 \\ \beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_i \\ 0 & & \beta_i & \alpha_i \end{pmatrix} \tag{2.6}$$

Das charakteristische Polynom $\varphi(x) = \det(A - xI)$ von A kann dann leicht rekursiv berechnet werden [30], [31]. Man findet durch Entwicklung von $\det(A_i - xI)$ nach der letzten Spalte (Laplacescher Entwicklungssatz) folgende Rekursionsformel:

$$\begin{aligned} p_n(x) &= 1 \\ p_{n-1}(x) &= \alpha_1 - x \\ p_{n-i}(x) &= (\alpha_i - x)p_{n-(i-1)}(x) - \beta_i^2 p_{n-(i-2)}(x), \quad i = 2, 3, \ldots, n. \end{aligned} \tag{2.7}$$

Nun ist $p_0(x) = \varphi(x) = \det(A - xI)$ das charakteristische Polynom von A, dessen Nullstellen die Eigenwerte von A sind.
Wird vorausgesetzt, daß $\beta_i \neq 0$, $i = 2, 3, \ldots, n$, erfüllt ist, so heißen die reellen symmetrischen Tridiagonalmatrizen nichtzerfallend, und es gilt der folgende Satz [31], der teilweise schon in 2.1.3 angegeben wurde:

Satz 2.2 *Jede reelle symmetrische nichtzerfallende n-reihige Tridiagonalmatrix A besitzt n verschiedene reelle Eigenwerte*

$$\xi_1 > \xi_2 > \ldots > \xi_n.$$

Die Polynome $p_{n-i}(x)$ in (2.7) bilden eine Sturmsche Kette.

Die Reihenfolge der Folgenglieder in (2.7) ist umgekehrt zu der in Definition 2.1. Die Anzahl der Vorzeichenwechsel der Folgenglieder an der Stelle x wird dadurch nicht verändert.
Eine weitere Eigenschaft der Folge (2.7) ist die strikte Trennung der Nullstellen von $p_{n-(i+1)}(x)$ durch die von $p_{n-i}(x)$. Seien λ_k die Nullstellen von $p_{n-i}(x)$ und μ_k die von $p_{n-(i+1)}(x)$, so gilt [33]:

$$\mu_1 > \lambda_1 > \mu_2 > \lambda_2 > \ldots > \mu_i > \lambda_i > \mu_{i+1}. \tag{2.8}$$

Für $x = -\infty$ besitzt die Polynomkette (2.7) die Vorzeichenfolge

$$+, +, \ldots, +,$$

also $w(-\infty) = 0$. Aus Satz 2.1 folgt daher, daß $w(\mu)$ gerade die Anzahl der Nullstellen ξ von $p_0(x)$ mit $\xi < \mu$ angibt:

$$w(\mu) \geq n + 1 - k \text{ gilt genau dann, wenn } \xi_k < \mu. \tag{2.9}$$

Hieraus ergibt sich folgendes Bisektionsverfahren, um die k-te Nullstelle von $p_0(x)$ zu bestimmen ($\xi_1 > \xi_2 > \ldots > \xi_n$) [30].
Zunächst ermittelt man ein Intervall $[a_0, b_0]$, das ξ_k sicher enthält. Dann halbiert man sukzessiv dieses Intervall und erhält nach Auswertung der Sturmschen Kette im Teilungspunkt

die Information, in welchem der beiden neuen Teilintervalle ξ_k liegt. Der Algorithmus lautet folgendermaßen:
Für $j = 0, 1, 2, \ldots$ bilde man

$$\begin{aligned} \mu_j &= (a_j + b_j)/2 \\ a_{j+1} &= \begin{cases} a_j, & \text{falls } w(\mu_j) \geq n+1-k \\ \mu_j, & \text{falls } w(\mu_j) < n+1-k \end{cases} \\ b_{j+1} &= \begin{cases} \mu_j, & \text{falls } w(\mu_j) \geq n+1-k \\ b_j, & \text{falls } w(\mu_j) < n+1-k. \end{cases} \end{aligned} \tag{2.10}$$

Es gilt dann stets:

$$\begin{aligned} [a_{j+1}, b_{j+1}] &\subseteq [a_j, b_j] \\ |a_{j+1} - b_{j+1}| &= |a_j - b_j|/2 \\ \xi_k &\in [a_{j+1}, b_{j+1}]. \end{aligned} \tag{2.11}$$

Die a_j konvergieren monoton wachsend, die b_j monoton fallend gegen ξ_i. Die Konvergenz ist linear mit dem Konvergenzfaktor 0.5 [30]. Dieses Verfahren zur Bestimmung der Nullstellen eines reellen Polynoms mit lauter reellen Nullstellen ist zwar langsam aber auch sehr genau. Es ist außerdem vorteilhaft, daß man jede beliebige Nullstelle unabhängig von den übrigen bestimmen kann.
Natürlich ist es auch möglich, auf diese Weise alle Nullstellen zu bestimmen. Die Auswertung der Sturmschen Kette liefert unter Berücksichtigung von Satz 2.1 stets die Information, wieviele Nullstellen in den einzelnen Teilintervallen enthalten sind. Ziel ist es, zunächst die einzelnen Nullstellen in Teilintervallen zu isolieren. Danach ist es unter Umständen von Vorteil, die Nullstellen durch ein schneller als Bisektion konvergierendes Verfahren bis zur gewünschten Genauigkeit zu berechnen.
Die lineare Rekursion in (2.7) ist bei Implementierung auf einer Rechenanlage anfällig für Over- und Underflow. Dies wird deutlich, wenn man das charakteristische Polynom in der Form

$$p_0(x) = \prod_{i=1}^{n} (\lambda_i - x) \tag{2.12}$$

darstellt. λ_i, $i = 1, \ldots, n$, sind die Nullstellen des charakteristischen Polynoms. Liegen z.B. die Nullstellen sehr eng beieinander, so wird $p_0(x)$ für ein x in der Nähe der λ_i sehr klein. Ferner können einige der $p_{n-i}(x)$ sehr klein werden und zu Underflow neigen, ohne daß $p_0(x)$ dieses Verhalten zeigt [5]. Dies ist möglich, da gemäß (2.8) die Nullstellen von $p_{n-i}(x)$ die von $p_{n-(i+1)}(x)$ strikt trennen. Ein möglicher Overflow ist im Gegensatz zu Underflow leicht vorab zu erkennen, indem man den Betrag von $p_0(x)$ in (2.12) durch $|a-b|^n$ für alle $x \in [a, b]$ nach oben abschätzt. Führt die Berechnung von $|a-b|^n$ zu keinem Overflow, so ist bei Auswertung der Sturmschen Kette (2.7) kein Overflow möglich, da wegen (2.8) $|a-b|^i \geq p_{n-i}(x)$, $i = 1, \ldots, n$, für alle $x \in [a, b]$ gilt. Dabei ist $[a, b]$ ein Intervall, das alle Nullstellen enthält.

Diese Schwierigkeiten lassen sich vermeiden, indem man die Sturmsche Kette $\{p_{n-i}(x)\}_{i=0,1,\ldots,n}$ in (2.7) durch die Folge

$$q_{n-i}(x) = \frac{p_{n-i}(x)}{p_{n-(i-1)}(x)}, \qquad i = 1, \ldots, n \tag{2.13}$$

ersetzt [22], [6], [5]. Die lineare Rekursion zweiter Ordnung in (2.7) geht dann über in die nichtlineare Rekursion erster Ordnung

$$\begin{aligned} q_{n-1}(x) &= \alpha_1 - x \\ q_{n-i}(x) &= \alpha_i - x - \frac{\beta_i^2}{q_{n-(i-1)}(x)}, \qquad i = 2, \ldots, n. \end{aligned} \tag{2.14}$$

q_{n-i}, die gleich Null sind, werden inplementierungsabhängig durch sehr kleine positive Zahlen ersetzt [5], [6]. Die Anzahl der Eigenwerte kleiner als μ ist dann gegeben durch die Anzahl negativer Werte in der Folge $\{q_{n-i}(\mu)\}_{i=1,\ldots,n}$. Hier bedarf lediglich der Fall, daß die $q_{n-i}(\mu) = 0$ durch sehr kleine positive Zahlen ersetzt werden, der Überprüfung. Ist ein $p_{n-i}(\mu)$ in (2.7) gleich Null, so soll diesem $p_{n-i}(\mu)$ dasselbe Vorzeichen wie seinem Vorgänger $p_{n-(i-1)}(\mu)$ zugeordnet werden. Aufeinanderfolgende Terme in (2.7) können nicht Null sein (s. Def. 2.1). Dies entspricht dem Vorgehen in 2.2.1, die Terme mit $p_{n-i}(\mu) = 0$ nicht zu berücksichtigen. Ist nun $p_{n-(i-1)}(\mu) = 0$, so wird das Vorzeichen von $p_{n-(i-1)}(\mu)$ wie das von $p_{n-(i-2)}(\mu)$ gewählt. Man erhält dann entsprechend zum Vorgehen in 2.2.1 „$-$“ als Vorzeichen von $q_{n-i}(\mu)$, da $p_{n-i}(\mu)p_{n-(i-2)}(\mu) < 0$ für $p_{n-(i-1)}(\mu) = 0$ gilt (s. Def. 2.1). Das Vorzeichen von $q_{n-i}(\mu)$ ergibt sich direkt durch eine Vorzeichenbetrachtung in (2.13). Folglich läßt sich $q_{n-(i-1)}(\mu)$, falls der Wert des Termes Null ist, durch eine sehr kleine positive Zahl ersetzen. In diesem Fall erhält nämlich wie gefordert $q_{n-i}(\mu) = \alpha_i - \mu - \beta_i^2/q_{n-(i-1)}$ ein negatives Vorzeichen.

2.2.3 Multisektionsverfahren

Bisektion ist ein Spezialfall der Multisektionsverfahren. Bei einem allgemeinen Multisektionsverfahren der Ordnung k wird ein Intervall $I = [a, b]$ durch k innere Teilungspunkte in $k + 1$ Teilintervalle $I_i = [x_i, x_{i+1}]$ mit $x_i = a + i(b - a)/(k + 1)$ für $i = 0, \ldots, k + 1$, unterteilt.
Existiert nur ein Eigenwert im Intervall I und soll dieser mit einem absoluten Fehler ϵ ermittelt werden, so sind

$$n_k = \log_2 \left(\frac{b-a}{2\epsilon} \right) / \log_2(k+1) \tag{2.15}$$

Multisektionen der Ordnung k notwendig [22].

Multisektionsverfahren sind hauptsächlich in Hinsicht auf Parallelverarbeitung interessant. Die Sturmsche Kette kann bei einer Multisektion der Ordnung k beispielsweise an k Punkten gleichzeitig ausgewertet werden.

2.3 Tridiagonalmatrizen mit speziellen Eigenschaften

2.3.1 Reelle symmetrische Tridiagonalmatrizen mit Nullen in den Nebendiagonalen

Sind r Elemente β_i in den Nebendiagonalen einer reellen symmetrischen Tridiagonalmatrix A in der Gestalt (2.2) gleich Null, so läßt sich A in $(r+1)$ reelle symmetrische Tridiagonalmatrizen $A^{(i)}$ niedrigerer Ordnung ohne Nullen in den Nebendiagonalen zerlegen. Die $A^{(i)}$ sind von der Ordnung m_i mit $\sum_{i=1}^{r+1} m_i = n$. Die Vereinigungsmenge der Eigenwerte aller $A^{(i)}$ ergibt dann die Menge der Eigenwerte von A [31], [33].
Die Zerlegung von A wird im folgenden beschrieben. Setzt man voraus, daß β_k gleich Null ist und die übrigen β_j ungleich Null sind, so zerfällt A in zwei unzerlegbare Tridiagonalmatrizen $A^{(1)}$ und $A^{(2)}$.

$$A = \begin{pmatrix} A^{(1)} & 0 \\ 0 & A^{(2)} \end{pmatrix} \tag{2.16}$$

mit

$$A^{(1)} = \begin{pmatrix} \alpha_1 & \beta_2 & & 0 \\ \beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ 0 & & \beta_{k-1} & \alpha_{k-1} \end{pmatrix} \tag{2.17}$$

und

$$A^{(2)} = \begin{pmatrix} \alpha_k & \beta_{k+1} & & 0 \\ \beta_{k+1} & \ddots & \ddots & \\ & \ddots & \ddots & \beta_n \\ 0 & & \beta_n & \alpha_n \end{pmatrix} \tag{2.18}$$

Die Richtigkeit der Behauptung, die Vereinigungsmenge der Eigenwerte von $A^{(1)}$ und $A^{(2)}$ sei die Menge der Eigenwerte von A, zeigt sich durch Bilden des charakteristischen Polynoms von A nach (2.7). Dann gilt:

$$\begin{aligned} &\vdots \quad \vdots \quad \vdots \\ p_{n-(k-1)}(x) &= (\alpha_{k-1} - x)p_{n-(k-2)}(x) - \beta_{k-1}^2 p_{n-(k-3)}(x) \end{aligned}$$

$$\begin{aligned} p_{n-k}(x) &= (\alpha_k - x)p_{n-(k-1)}(x) - 0 \\ p_{n-(k+1)}(x) &= p_{n-(k-1)}[(\alpha_{k+1} - x)(\alpha_k - x) - \beta_{k+1}^2] \\ \vdots \quad &\vdots \qquad \vdots \\ p_0(x) &= p_{n-(k-1)}(x)p_0^{(2)}(x). \end{aligned} \tag{2.19}$$

Hierin sind $p_0(x) = \det(A - xI)$ das charakteristische Polynom von A, $p_{n-(k-1)}(x) = p_0^{(1)}(x) = \det(A^{(1)} - xI)$ das charakteristische Polynom von $A^{(1)}$ und $p_0^{(2)}(x) = \det(A^{(2)} - xI)$ das charakteristische Polynom von $A^{(2)}$. Da $p_0(x)$ als Produkt von $p_0^{(1)}(x)$ und $p_0^{(2)}(x)$ in (2.19) auftritt, folgt die Behauptung.
Zerlegt man die Matrix im Falle $\beta_k = 0$ nicht, so ist folgendes zu beachten:

1. Es können vielfache Nullstellen des charakteristischen Polynoms auftreten. Besitzt das charakteristische Polynom eine Nullstelle der Vielfachheit k, so müssen mindestens $(k-1)$ Elemente der Nebendiagonalen gleich Null sein. Ist andererseits ein Element der Nebendiagonalen β_k gleich Null, so kann daraus nicht geschlossen werden, daß eine vielfache Nullstelle vorhanden ist [33].

2. Zwei aufeinanderfolgende Terme der Sturmschen Kette in (2.7) können zu Null werden. In diesem Fall ist eine Nullstelle des charakteristischen Polynoms als Testpunkt x gewählt worden. Da alle folgenden Terme Null sind, kann das Zählen der Vorzeichenwechsel zu falschen Ergebnissen führen.

Wird die Anzahl der Nullstellen des charakteristischen Polynoms, die kleiner als eine Zahl μ sind, mit Hilfe einer der beiden Folgen (2.7) und (2.14) bestimmt, so werden eventuelle Vielfachheiten von Nullstellen berücksichtigt. Ist z.B. nur eine Nullstelle der Vielfachheit k kleiner als μ, so ergäbe die Auswertung einer der beiden Folgen an der Stelle $x = \mu$, daß k Nullstellen kleiner als μ sind. Bei einem Bisektions- oder Multisektionsverfahren zur Bestimmung aller Eigenwerte einer symmetrischen Tridiagonalmatrix kann mit Hilfe der Folgen (2.7) und (2.14) nicht bestimmt werden, ob sich ein echter vielfacher Eigenwert in einem Intervall befindet oder sich mehrere verschiedene Eigenwerte um weniger als die vorgegebene Genauigkeit unterscheiden. Das Problem in 2. tritt bei Verwendung der Folge (2.14) nicht auf, wenn man Nullen als positiv zählt bzw. vorkommende Nullen durch sehr kleine positive Zahlen ersetzt.
Auf das Zerlegen der Matrix ließe sich also bei Verwendung der Folge (2.14) verzichten. Gewöhnlich ist jedoch beim Auftreten von Nullen in den Nebendiagonalen das Aufteilen des ursprünglichen Problems in kleiner dimensionierte Teilprobleme von Vorteil.

2.3.2 Reelle unsymmetrische Tridiagonalmatrizen

Seien $b_{ii} = \alpha_i$, $b_{i,i+1} = \beta_{i+1}$, $b_{i+1,i} = \gamma_{i+1}$ und $b_{ij} = 0$ sonst die Elemente einer reellen allgemeinen Tridiagonalmatrix B:

$$B = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \gamma_2 & \alpha_2 & \beta_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \gamma_{n-1} & \alpha_{n-1} & \beta_n \\ 0 & \cdots & 0 & \gamma_n & \alpha_n \end{pmatrix} \tag{2.20}$$

Ist

$$\beta_i \gamma_i > 0, \quad i = 2, \ldots, n \tag{2.21}$$

erfüllt, so kann B durch Ähnlichkeitstransformationen mit einer geeigneten Diagonalmatrix D in eine reelle symmetrische Matrix transformiert werden [33]. Ist D definiert durch

$$d_{1,1} = 1, \quad d_{ii} = (\gamma_2 \gamma_3 \ldots \gamma_i / \beta_2 \beta_3 \ldots \beta_i)^{\frac{1}{2}}, \tag{2.22}$$

dann gilt

$$D^{-1} B D = A. \tag{2.23}$$

A ist tridiagonal und besteht aus den Elementen

$$a_{ii} = \alpha_i, \quad a_{i,i+1} = a_{i+1,i} = (\beta_{i+1} \gamma_{i+1})^{\frac{1}{2}}. \tag{2.24}$$

Ist (2.21) erfüllt, so lassen sich die Sturmsche Kette (2.7) und die Folge (2.14) zur Bestimmung der Eigenwerte von B einsetzen. Es ist lediglich β_i^2 durch $\beta_i \gamma_i$ zu ersetzen. Tridiagonalmatrizen, die (2.21) erfüllen, werden quasisymmetrische Tridiagonalmatrizen genannt.
Ist ein β_i oder γ_i gleich Null, so lassen sich die Eigenwerte von B als die Eigenwerte kleinerer Tridiagonalmatrizen bestimmen. Dies kann wie in 2.3.1 über die Folge (2.7) nachgewiesen werden. Sind β_k oder γ_k gleich Null, so verläuft die Zerlegung von B analog zu der in 2.3.1.

$$B = \begin{pmatrix} B^{(1)} & 0 \\ 0 & B^{(2)} \end{pmatrix} \tag{2.25}$$

mit

$$B^{(1)} = \begin{pmatrix} \alpha_1 & \beta_2 & & 0 \\ \gamma_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ 0 & & \gamma_{k-1} & \alpha_{k-1} \end{pmatrix} \tag{2.26}$$

und

$$B^{(2)} = \begin{pmatrix} \alpha_k & \beta_{k+1} & & 0 \\ \gamma_{k+1} & \ddots & \ddots & \\ & \ddots & \ddots & \beta_n \\ 0 & & \gamma_n & \alpha_n \end{pmatrix} \tag{2.27}$$

2.4 Bestimmen eines Intervalls, das alle Eigenwerte enthält

Zum Bestimmen eines Intervalls, in dem alle Eigenwerte einer reellen symmetrischen Tridiagonalmatrix A in der Gestalt (2.2) liegen, lassen sich zum einen bestimmte Matrixnormen, zum anderen die Gerschgorin-Kreise heranziehen.
Hier sei zunächst die Zeilensummennorm als Hilfsmittel zur Abschätzung vorgestellt. Es gilt folgender Satz [31]:

Satz 2.3 *Für die Eigenwerte λ_j der Matrix A gemäß (2.2) gilt*

$$|\lambda_j| \leq \max_{1\leq i\leq n}\{|\beta_i| + |\alpha_i| + |\beta_{i+1}|\}, \qquad \beta_1 = \beta_{n+1} = 0.$$

$\max_{1\leq i\leq n}\{|\beta_i| + |\alpha_i| + |\beta_{i+1}|\}$ ist die Zeilensummennorm $\|A\|_\infty$ und hier auch zugleich die Spaltensummennorm von A. Ein Intervall, in dem alle Eigenwerte von A liegen, ist also durch $[-\|A\|_\infty, \|A\|_\infty]$ gegeben.
Die Gerschgorin-Kreise (s. z.B. [31]) erlauben folgende Aussage über die Eigenwerte reeller symmetrischer Tridiagonalmatrizen: Alle Eigenwerte sind in der Vereinigungsmenge der n Intervalle $[\alpha_i - (|\beta_i| + |\beta_{i+1}|), \alpha_i + (|\beta_i| + |\beta_{i+1}|)]$, $i = 1, \ldots, n$, mit $\beta_1 = \beta_{n+1} = 0$ enthalten [5]. Ein Intervall $[xmin, xmax]$, das alle Eigenwerte von A enthält, ergibt sich also aus

$$\begin{aligned} xmin &= \min_{1\leq i\leq n}\{\alpha_i - (|\beta_i| + |\beta_{i+1}|)\}, \\ xmax &= \max_{1\leq i\leq n}\{\alpha_i + (|\beta_i| + |\beta_{i+1}|)\}. \end{aligned} \tag{2.28}$$

2.5 Berechnung des Funktionswertes des charakteristischen Polynoms

Der Funktionswert des charakteristischen Polynoms φ kann mit Hilfe der Sturmschen Kette berechnet werden; nach (2.7) gilt $\varphi(x) = p_0(x)$.

Die Bestimmung des Funktionswertes des charakteristischen Polynoms ist auch mit Hilfe der Folge (2.14) nach Gleichung

$$\varphi(x) = \prod_{i=1}^{n} q_{n-i}(x) \tag{2.29}$$

möglich, da mit Verwendung von (2.13)

$$p_{n-1} \cdot \frac{p_{n-2}}{p_{n-1}} \cdot \frac{p_{n-3}}{p_{n-2}} \cdots \frac{p_1}{p_2} \cdot \frac{p_0}{p_1} = p_0 = \varphi \tag{2.30}$$

gilt.
Da q_{n-i}, die den Wert Null haben, durch eine sehr kleine positive Zahl ϵ ersetzt werden, ist zu untersuchen, wie sich diese Ersetzung auf die Berechnung von $\varphi(x)$ nach (2.29) auswirkt. Wird angenommen, daß $p_{n-(i-2)} \neq 0$ und $p_{n-(i-1)} = 0$ ist, so gelten die folgenden Gleichungen:

$$p_{n-i} = -\beta_i^2 p_{n-(i-2)} \tag{2.31}$$

und

$$p_{n-(i+1)} = (\alpha_{i+1} - x)p_{n-i} = -(\alpha_{i+1} - x)\beta_i^2 p_{n-(i-2)} \,. \tag{2.32}$$

Für die Folge (2.14) ist in diesem Fall nach (2.13) $q_{n-(i-1)} = 0$. Wird $q_{n-(i-1)}$ durch ϵ ersetzt, so ergibt sich für die Folgenglieder q_{n-i} und $q_{n-(i+1)}$:

$$q_{n-i} = \alpha_i - x - \frac{\beta_i^2}{\epsilon} \tag{2.33}$$

und

$$q_{n-(i+1)} = \alpha_{i+1} - x - \beta_{i+1}^2 \frac{\epsilon}{\epsilon(\alpha_i - x) - \beta_i^2} \,. \tag{2.34}$$

Der exakte Wert für das Produkt aus $q_{n-(i-1)}$, q_{n-i} und $q_{n-(i+1)}$ lautet unter Verwendung von (2.13) und (2.32):

$$q_{n-(i-1)}q_{n-i}q_{n-(i+1)} = \frac{p_{n-(i+1)}}{p_{n-(i-2)}} = -(\alpha_{i+1} - x)\beta_i^2 \,. \tag{2.35}$$

Wird $q_{n-(i-1)}$ durch ϵ ersetzt, so gilt:

$$\begin{aligned} q_{n-(i-1)}q_{n-i}q_{n-(i+1)} &= \epsilon[(\alpha_i - x) - \frac{\beta_i^2}{\epsilon}][\alpha_{i+1} - x - \beta_{i+1}^2 \frac{\epsilon}{\epsilon(\alpha_i - x) - \beta_i^2}] \\ &= [\epsilon(\alpha_i - x) - \beta_i^2]\frac{(\alpha_{i+1} - x)[\epsilon(\alpha_i - x) - \beta_i^2] - \beta_{i+1}^2\epsilon}{\epsilon(\alpha_i - x) - \beta_i^2} \\ &= \epsilon[(\alpha_{i+1} - x)(\alpha_i - x) - \beta_{i+1}^2] - (\alpha_{i+1} - x)\beta_i^2 \\ &\approx (\alpha_{i+1} - x)\beta_i^2 \,. \end{aligned} \tag{2.36}$$

Für den letzten Schritt muß

$$\epsilon|(\alpha_{i+1}-x)(\alpha_i-x)-\beta_{i+1}^2| \ll |(\alpha_{i+1}-x)|\beta_i^2 \tag{2.37}$$

erfüllt sein. (2.37) gilt auf jeden Fall, wenn

$$\epsilon|\alpha_i-x| \ll \beta_i^2 \tag{2.38}$$

und

$$\epsilon \ll |\alpha_{i+1}-x|\frac{\beta_i^2}{\beta_{i+1}^2} \tag{2.39}$$

ist. Hieraus folgt, daß sich $\varphi(x)$ für ein genügend kleines ϵ trotz der Ersetzung von $q_{n-(i-1)}$ durch ϵ ohne großen Fehler nach (2.29) berechnen läßt.
Ist $q_0(x)=0$, so ist auch $p_0(x)=0$, d.h. es liegt eine Nullstelle von $\varphi(x)$ vor. In diesem Fall geht nach (2.29) ϵ als letzter Faktor in die Berechnung von $\varphi(x)$ ein. Ist ϵ genügend klein, so entsteht auch in diesem Fall kein großer Fehler bei der Berechnung von $\varphi(x)$. Abschließend ist zu sagen, daß Nullen in den Gliedern der Folge (2.14) in Anwendungsfällen selten auftreten.

2.6 Das Pegasusverfahren

Liegen Intervalle vor, die jeweils genau einen Eigenwert einer reellen symmetrischen Tridiagonalmatrix enthalten, so können diese Eigenwerte statt mit Bisektion oder Multisektion auch durch ein Iterationsverfahren mit superlinearer Konvergenz in vorgegebener Genauigkeit berechnet werden. Um verschiedene Iterationsverfahren miteinander vergleichen zu können, führt man den Effizienzindex E ein [21]:

$$E=\omega^{\frac{1}{m}}\ . \tag{2.40}$$

m ist die Zahl der Funktionsauswertungen in jedem Schritt des Iterationsverfahrens, ω die Konvergenzordnung. In Tabelle 2.6 sind die Effizienzindizes einiger Iterationsverfahren zusammengestellt.
Das Pegasusverfahren besitzt den höchsten Effizienzindex der in Tabelle 2.6 zusammengestellten Verfahren. Die Iterationsvorschrift des Pegasusverfahrens zur Bestimmung einer Nullstelle $\overline{x}$ einer Funktion $f(x)$ lautet (s. [21] und [11]):

$$x_{n+1}=x_n-\frac{x_n-z_{n-1}}{f(x_n)-g_{n-1}}f(x_n),\quad n=1,2,3,\dots\ , \tag{2.41}$$

wobei z_{n-1} und g_{n-1} wie folgt gewählt werden:

Iterationsverfahren	Effizienzindex
Bisektion	1
Newtonverfahren	$\sqrt{2} \approx 1.414$
Sekantenverfahren	$(1+\sqrt{5})/2 \approx 1.618$
Pegasusverfahren	≈ 1.642

Tabelle 2.1: Effizienzindizes einiger Iterationsverfahren

a) Ist $f(x_n)f(x_{n-1}) < 0$, so setzt man

$$z_{n-1} = x_{n-1} \text{ und } g_{n-1} = f(x_{n-1}). \tag{2.42}$$

Dies entspricht einem Schritt des Sekantenverfahrens.

b) Ist $f(x_n)f(x_{n-1}) > 0$, so setzt man

$$\begin{aligned} z_{n-1} &= z_{n-2} \text{ und} \\ g_{n-1} &= \frac{f(x_{n-1})f(z_{n-2})}{f(x_n)+f(x_{n-1})} . \end{aligned} \tag{2.43}$$

In Abbildung 2.1 sind einige Schritte des Pegasusverfahrens dargestellt (durchgezogene Linien).

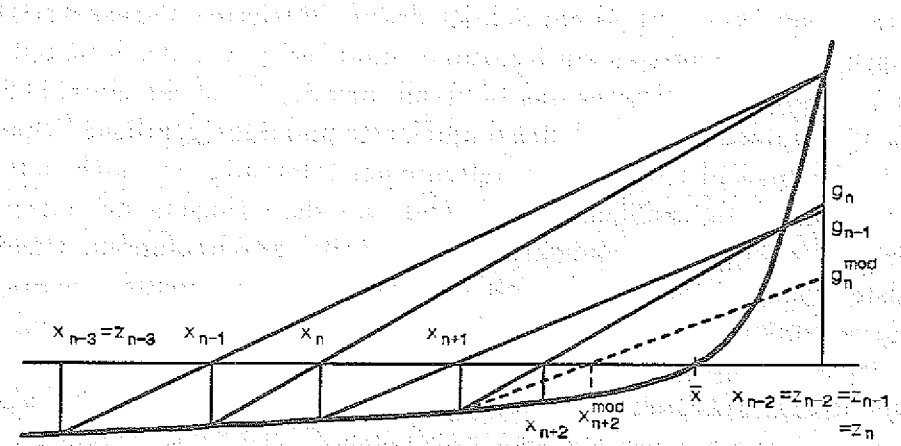


Abb. 2.1: Pegasusverfahren

Die Iterationspunkte x_{n-1} und x_n wurden wie beim Sekantenverfahren ermittelt. Der Fall **b)** tritt zum ersten Mal beim Bestimmen von x_{n+1} ein.

Beginnt man mit einem Intervall, das genau eine Nullstelle $\overline{x}$ einer Funktion $f(x)$ enthält, so schließt jedes Paar von Iterationspunkten des Pegasusverfahrens $\{x_i, z_{i-1}\}$ die Nullstelle ein. Als einschließendes Verfahren konvergiert das Pegasusverfahren immer, wenn $f'(\overline{x}) \neq 0$ gilt. Das Sekantenverfahren besitzt gegenüber dem Pegasusverfahren den Nachteil, daß jedes dritte Paar von Iterationspunkten die Nullstelle nicht einschließt. Ferner konvergieren Newton- und Sekantenverfahren nur, wenn die Startwerte der Verfahren genügend nahe an der Nullstelle liegen und zusätzlich $f'(\overline{x}) \neq 0$ ist.
Bei der Herleitung des Effizienzindexes des Pegasusverfahrens in [11] wird vorausgesetzt, daß die Startwerte des Pegasusverfahrens nahe an der gesuchten Nullstelle liegen. In vielen Anwendungsfällen ist diese Voraussetzung jedoch nicht gegeben. Insbesondere kann der Fall eintreten, daß eine Intervallgrenze des Intervalls, das die Nullstelle enthält, während des gesamten Verfahrens festgehalten wird. Dies geschieht, wenn in Fall **b)** die Steigung der Geraden, die durch die Nullstelle und den Punkt $(z_{n-1}, f(z_{n-1}))$ festgelegt ist, mehr als doppelt so steil ist wie die der Tangente an die Funktion $f(x)$ im Punkt $\overline{x}$. Dieser Fall liegt in Abbildung 2.1 vor und führt zum Verlust der superlinearen Konvergenz des Verfahrens. Ersetzt man jedoch in (2.43) $f(z_{n-2})$ durch g_{n-2}, so wird das Festhalten einer Intervallgrenze verhindert. Man erhält ein modifiziertes Pegasusverfahren. In Abbildung 2.1 unterscheidet sich das modifizierte Pegasusverfahren (gestrichelte Linie) vom eigentlichen Pegasusverfahren (durchgezogene Linie) zum ersten Mal bei der Bestimmung von x_{n+2}. Beim modifizierten Pegasusverfahren wird der Iterationspunkt x_{n+2}^{mod} ermittelt. Dieser liegt näher an der Nullstelle $\overline{x}$ als der Iterationspunkt x_{n+2} des eigentlichen Pegasusverfahrens. Durch die Modifikation in (2.43) wird die Steigung der Geraden zur Bestimmung des nächsten Iterationspunktes mit jedem Schritt des modifizierten Pegasusverfahrens geringer, so daß nach einigen Schritten ein Iterationspunkt rechts von der Nullstelle $\overline{x}$ ermittelt wird. Ist im Verlauf des Verfahrens das Intervall, das die Nullstelle einschließt, genügend klein geworden, so unterscheiden sich das modifizierte und das eigentliche Pegasusverfahren nicht mehr. In diesem Fall tritt das Festhalten einer Intervallgrenze nicht auf.
Ferner ist es in Anwendungsfällen oft lohnend, vor dem Einsatz des eigentlichen oder modifizierten Pegasusverfahrens einige Bisektionsschritte durchzuführen. Diese verkleinern das die Nullstelle einschließende Intervall und führen so zu günstigeren Startwerten für das Iterationsverfahren mit superlinearer Konvergenz. In vielen Fällen sinkt dadurch die Gesamtzahl der Iterationsschritte.
Das charakteristische Polynom $\varphi(x)$ von reellen symmetrischen Tridiagonalmatrizen mit $\beta_i \neq 0$, $i = 2, \ldots, n$, besitzt nur einfache Nullstellen (s. 2.1.3), so daß $\varphi'(\overline{x}) \neq 0$ für jede Nullstelle $\overline{x}$ von $\varphi(x)$ gilt. Sind Intervalle ermittelt, die genau eine Nullstelle von $\varphi(x)$ enthalten, so ist das modifizierte Pegasusverfahren mit einigen vorher durchgeführten Bisektionsschritten aus den oben genannten Gründen ein sehr geeignetes Iterationsverfahren, um die Nullstellen in vorgegebener Genauigkeit zu bestimmen.

3 CRAY-Mehrprozessorrechner

In diesem Kapitel werden die Eigenschaften der CRAY-Rechner des Forschungszentrums Jülich beschrieben. Im ersten Abschnitt wird auf gemeinsame Eigenschaften von CRAY X-MP und CRAY Y-MP eingegangen, in den folgenden Abschnitten sind Unterschiede und Kenndaten beider Rechner dargestellt.
Umfassendere Beschreibungen der CRAY-Rechner sind in [16] und [20] zu finden.

3.1 Klassifizierung und allgemeine Eigenschaften

CRAY X-MP und Y-MP Systeme sind jeweils Mehrprozessorrechner mit gemeinsamem Speicher. Beide Systeme sind Vektor-Supercomputer. Nach der Klassifizierung der Rechnerstrukturen von Flynn [12] zählen beide Systeme zu den MIMD-Rechnern (Multiple Instruktion Stream Multiple Data Stream). Die voneinander unabhängigen Prozessoren der CRAY-Systeme sind in der Lage, gleichzeitig unterschiedliche Maschineninstruktionen auf unterschiedlichen Daten auszuführen. Jeder einzelne Prozessor der CRAY-Systeme stellt nach Flynn einen SIMD-Rechner (Single Instruktion Stream Multiple Data Stream) dar, da jeder Prozessor Funktionseinheiten besitzt, die gleichzeitig unabhängige Operationen auf Vektoren ausführen können.
Jede CPU der beiden CRAY-Systeme ist ein Vektorrechner mit Register-Register-Architektur. Zur Vektorverarbeitung stehen jeder CPU 8 Vektorregister zur Speicherung von jeweils 64 Operanden zur Verfügung. Ferner besitzt jede CPU Vektor-Funktionsheiten (Pipelines), die parallel genutzt werden können. Es ist möglich, verschiedene Funktionseinheiten miteinander zu verbinden. Hierbei wird ein Ausgabewert einer Funktionseinheit zum Eingabewert einer anderen, ohne daß die Zwischenresultate in einem Register gespeichert werden müssen, ehe die zweite Operation beginnen kann. Der Vorgang der Verkettung von Funktionseinheiten wird als „chaining“ bezeichnet. Jede CPU ist mit spezieller Gather/Scatter- und Compress-Index-Hardware zur Vektorisierung von Bedingungsanweisungen ausgerüstet. Dabei wird der vektorielle Zugriff über Indexlisten durch die spezielle Hardware beschleunigt. Neben der Vektoreinheit besitzt jede CPU auch eine sehr schnelle Skalareinheit mit mehreren skalaren Funktionseinheiten. Ferner sind in jeder CPU große Instruktionspuffer vorhanden, in denen die nächsten auszuführenden Befehle bereitgehalten werden.
Die einzelnen Prozessoren eines CRAY-Systems sind über den gemeinsamen Speicher und

gemeinsame Register gekoppelt. Die gemeinsamen Register sind in Gruppen (cluster) aufgeteilt, die von allen Prozessoren gemeinsam genutzt werden können. Jede Gruppe enthält 8 Adreß-, 8 Skalar- und 32 Semaphor-Register. Auf den binären Semaphor-Registern ist eine Test-and-Set-Operation implementiert, die zur Synchronisation zwischen verschiedenen Prozessen verwendet werden kann. Ebenso ermöglichen diese Gruppen von gemeinsamen Registern Hochgeschwindigkeitskommunikation zwischen den einzelnen Prozessoren, wenn Multitasking genutzt wird.

3.2 CRAY X-MP/416

Der Rechner CRAY X-MP/416 verfügt über 4 unabhängige Prozessoren und 16 Megaworte Hauptspeicher. Die Wortlänge beträgt 64 Bit. Die Hardware besteht aus ECL-Bausteinen (Emitter-Coupled Logic) mit lediglich 16 Gattern je Chip und Schaltzeiten von 300-400 ps. Die CPUs werden mit einer Zykluszeit von 8.5 ns getaktet, die Speicherzugriffszeit (Memory Access Time) beträgt 34 ns. Das System besitzt 5 identische Gruppen von gemeinsamen Registern. Jede CPU verfügt über 13 Vektor-Funktionseinheiten.
Der Hauptspeicher ist in 64 Bänke aufgeteilt, die in 4 Sektionen zusammengefaßt werden. Als Hintergrundspeicher steht ein SSD (Solid-State Storage Device) mit einer Kapazität von 32 Megaworten in MOS-Technologie (Metal-Oxide-Semiconductor) zur Verfügung. Die Übertragungsgeschwindigkeit des Hintergrundspeichers zum Hauptspeicher beträgt 1000 Megabytes/s. Die Schnittstelle zwischen dem Rechner und den Peripheriegeräten bildet das physikalisch und logisch vollständig separierte I/O-Subsystem (IOS).

3.3 CRAY Y-MP8/832

CRAY Y-MP ist das Nachfolge-System von CRAY X-MP. Der Rechner CRAY Y-MP8/832 des Forschungszentrums Jülich verfügt über 8 Prozessoren und 32 Megaworte Hauptspeicher (Wortlänge 64 Bit). Die Hardware besteht aus von CRAY selbst entwickelten LSI-Chips (Large Scale Integration) in ECL-Technik. Gegenüber CRAY X-MP wurde die Integrationsdichte wesentlich erhöht (2500-Gate-Arrays). Die Taktzeit der CPUs wurde gegenüber CRAY X-MP auf 6 ns verkürzt. Die Speicherzugriffszeit beträgt 30 ns. Das System verfügt über 9 gemeinsame Registergruppen. Jede CPU besitzt 13 Vektor-Funktionseinheiten. Die logische Struktur der CPUs ist mit der des Vorgänger-Systems identisch. CRAY Y-MP ist X-MP kompatibel im normalen Y-Mode und X-MP/416 Bit-kompatibel im X-Mode.
Die Speicherorganisation wurde gegenüber CRAY X-MP wesentlich verbessert. Der Hauptspeicher ist in 256 Bänke aufgeteilt, die in 4 Sektionen und 32 Untersektionen zusammengefaßt werden. Jede Sektion ist also noch einmal in 8 Untersektionen unterteilt. Durch die

geänderte Speicherorganisation konnte auch die Auflösung von Speicherkonflikten deutlich verbessert werden [17]. Gegenüber CRAY X-MP wurde die Speicherbandbreite auf 42.6 GByte/s nahezu verdreifacht. Als Hintergrundspeicher wird ein SSD mit einer Kapazität von 128 Megaworten verwendet.

4 Werkzeuge für Vektorisierung und Parallelverarbeitung

4.1 Vektorverarbeitung auf CRAY-Rechnern

Zur automatischen Vektorisierung von FORTRAN-Programmen kann auf den CRAY-Rechnern der vektorisierende FORTRAN-Compiler CFT77 allein oder das FORTRAN-Compiling-System CF77, das aus Präprozessor, Midprozessor, Compiler und Segment-Loader besteht, genutzt werden [3]. Die Vektorisierungseigenschaften des CFT77-Compilers allein sind beschränkt, da einige im Prinzip vektorisierbare Programmkonstrukte von CFT77 nicht vektorisiert werden. Im FORTRAN-Compiling-System entscheiden Präprozessor FPP und Compiler CFT77 gemeinsam über die Vektorisierbarkeit von Programmkonstrukten. Der Präprozessor ist in der Lage, Programmtransformationen am Quellcode durchzuführen und Compiler-Direktiven einzufügen, um die Vektorisierbarkeit des FORTRAN-Programms zu verbessern. Auf diese Weise können die Vektorisierungseigenschaften des FORTRAN-Programms gegenüber der alleinigen Benutzung des Compilers deutlich verbessert werden.

Compiler und Compiling-System analysieren vorwiegend DO-Schleifen. Liegen keine Datenabhängigkeiten vor oder können Datenabhängigkeiten durch Programmtransformationen am Quellcode verhindert werden, so werden die entsprechenden Schleifen vektorisiert. Die Vektoren werden in Teilvektoren der Länge 64 zerlegt, da jedes Vektorregister 64 Elemente aufnehmen kann. Zur Laufzeit werden die Teilvektoren in den Vektor-Funktionseinheiten verarbeitet. Zur Vektorverarbeitung geeignet sind eindimensionale Felder, Spalten und Zeilen einer Matrix oder über einen Indexvektor adressierte Feldelemente. Programmteile mit Bedingungsanweisungen können mit Hilfe der Gather/Scatter- und Compress-Index-Hardware vektoriell verarbeitet werden. Es ist günstig und notwendig, schon bei der Programmerstellung auf gute Vektorisierungseigenschafen zu achten. Zur Steuerung der Vektorisierung durch Compiler oder Compiling-System sind Direktiven vorhanden, die in den Quellcode eingefügt werden können. Die Vektorisierung einer DO-Schleife verhindern folgende Anweisungen innerhalb der Schleife: I/O-Anweisungen, GOTO-Anweisungen mit Sprungziel außerhalb der Schleife, ELSEIF-Anweisungen und geschachtelte IF-Strukturen.

Die Vektorisierung eines Programms führt durch Einsatz von Hardware zur Vektorverar-

beitung sowohl zu kürzerer Ausführungszeit als auch kürzerer CPU-Zeit als bei skalarer Bearbeitung des Programms. Die Parallelisierung eines Programms führt hingegen zwar zu kürzerer Ausführungszeit, nicht jedoch zu kürzerer CPU-Zeit, da die Arbeit auf mehrere Prozessoren verteilt wird. Daher ist es auf CRAY-Systemen in der Regel von Vorteil, zunächst die Vektorisierungseigenschaften eines Programms zu untersuchen und dann erst die Parallelisierungseigenschaften.

4.2 Konzepte zur Parallelverarbeitung auf CRAY-Rechnern

In den beiden folgenden Abschnitten werden die CRAY-Multitasking-Strategien Micro- und Macrotasking kurz beschrieben. Auf Autotasking wird im dritten Abschnitt ausführlicher eingegangen, da diese Strategie in dieser Arbeit ausschließlich zur Parallelisierung der entwickelten Programme verwendet wurde. Multitasking ist eine Sonderform des Multiprocessing, die die parallele Ausführung unabhängiger Teile eines Programms durch sogenannte Tasks zuläßt. Der Begriff Task wird im Sinne von User Library Task (s. [2]) verwendet. Da die zeitliche Abfolge der Ausführung der Tasks nichtdeterministisch ist, sind Synchronisations- und Kommunikationsmechanismen erforderlich, um die Bearbeitung eines Programms zeitlich zu koordinieren.
Die Informationen für die folgenden Abschnitte sind [1], [2], [10], [19], [24], [25] und [26] entnommen.

4.2.1 Macrotasking

Macrotasking nutzt als grobgranulares Parallelisierungskonzept Parallelität auf der Ebene von Unterprogrammen. Jeweils ein Unterprogramm wird einer Task zur Bearbeitung zugewiesen, so daß die Unterprogramme gleichzeitig auf unterschiedlichen Prozessoren ausgeführt werden können. Die Aufteilung des Programms geschieht durch den Aufruf von Bibliotheksroutinen. Vom Benutzerprogramm werden Anforderungen an den Bibliotheksscheduler gestellt. Dieser verwaltet die Tasks und stellt Kommunikations- und Synchronisationsroutinen bereit. Damit bildet er die Schnittstelle zum Betriebssystem.
Die Macrotasking-Bibliothek enthält Unterprogramme, die die Erzeugung und Beendigung von Tasks kontrollieren, Unterprogramme, die die Synchronisation von Tasks mit Hilfe von Ereignisvariablen ermöglichen und Unterprogramme, die den wechselseitigen Ausschluß von kritischen Bereichen gewährleisten. Ferner sind Unterprogramme für die Barriere-Synchronisation vorhanden, die sicherstellen, daß eine bestimmte Anzahl von Programmteilen abgearbeitet ist, bevor Anweisungen hinter der Barriere ausgeführt werden.

Durch diese Routinen erlaubt Macrotasking eine flexible, aber auch sehr aufwendige Parallelisierung.
Der Scheduler nutzt die Hardware-Register in den meisten Fällen nicht für Synchronisationsaktivitäten, Kommunikation zwischen parallel ausgeführten Programmteilen geschieht über gemeinsame Variablen im Hauptspeicher. Hieraus folgt ein großer Overhead, der sich bei Programmen mit häufiger Synchronisation und Kommunikation stark auf das Zeitverhalten auswirkt.

4.2.2 Microtasking

Microtasking nutzt als feingranulares Parallelisierungskonzept Parallelität auf der Ebene von Schleifen und Anweisungen innerhalb von Unterprogrammen. Kommunikation und Synchronisation geschehen weitgehend über unmittelbaren Zugriff auf gemeinsame Register-Gruppen. Ferner liegen gegenüber Macrotasking wesentlich schnellere Basisroutinen für Synchronisation und Kommunikation vor. Hieraus folgt ein wesentlich geringerer Overhead.
Parallele Programmteile werden vom Benutzer durch Einfügen von Microtasking-Direktiven gekennzeichnet. Diese Direktiven werden von einem Präprozessor in den notwendigen Assembler-Code, in Bibliotheksprogrammaufrufe und zusätzliche FORTRAN-Anweisungen umgesetzt. Durch die Direktiven werden Kontrollstrukturen festgelegt, die die Zuordnung von parallel ausführbaren Programmteilen zu Prozessoren zur Laufzeit beeinflussen. Diese Zuordnung geschieht dynamisch, d.h. ein Unterprogramm, das als Microtasking-Unterprogramm gekennzeichet ist, wird von allen Prozessoren ausgeführt, die nicht anderweitig beschäftigt, also momentan verfügbar sind. Durch Microtasking-Direktiven können Kontrollstrukturen zur parallelen Bearbeitung von DO-Schleifen und zur expliziten Partitionierung in unabhängige Anweisungsblöcke festgelegt werden. In Unterprogrammen, die Microtasking-Direktiven enthalten, ist die Unterscheidung von global im rufenden Programm und lokal im Unterprogramm deklarierten Variablen von großer Bedeutung, da dadurch weitgehend entschieden wird, welche Variablen für die Tasks gemeinsam oder privat sind [26]. Es ist darauf zu achten, daß allen Tasks gemeinsame Variablen nicht unkontrolliert modifiziert werden. Ferner sollten Werte von privaten Variablen außerhalb der Kontrollstruktur, in der ihnen ein Wert zugewiesen wurde, nicht verwendet werden, da die Verfügbarkeit der Werte nicht garantiert werden kann [26]. Neben den oben genannten Direktiven sind Direktiven zur Anforderung und Rückgabe von CPUs und zur Kennzeichnung kritischer Bereiche, die mehrere Tasks nicht gleichzeitig bearbeiten dürfen, vorhanden. Alle Direktiven beginnen mit „C“ in der ersten Spalte einer Programmzeile, werden also von FORTRAN-Compilern als Kommentar angesehen. Daher sind FORTRAN-Programme mit Microtasking-Direktiven auch in anderen Rechnerumgebungen lauffähig.

4.2.3 Autotasking

Autotasking ist ein feingranulares Parallelisierungskonzept, das parallel ausführbare Blöcke innerhalb eines Programms durch vorherige Programmanalyse automatisch erkennt. Kommunikation und Synchronisation sind über unmittelbaren Zugriff auf gemeinsame Register-Gruppen realisiert. Ein Programm, das Autotasking-Direktiven enthält, wird zunächst von einem einzelnen Prozessor bearbeitet, während zusätzlich verfügbare Prozessoren eine Test-and-Set-Operation auf einem Semaphor-Register ausführen. Wird ein durch Autotasking-Direktiven gekennzeichneter paralleler Bereich erreicht, wird das Semaphor-Register zurückgesetzt, und alle Tasks nehmen an der parallelen Bearbeitung dieses Bereichs teil.
Autotasking wird durch das Compiling-System CF77 realisiert, das aus drei Teilen besteht. Der Präprozessor FPP analysiert FORTRAN-Programme, erkennt parallele Programmteile (z.B. Schleifen) und kennzeichnet diese Teile mit Präprozessor-Direktiven. Diese Direktiven werden durch den zweiten Präprozessor FMP übersetzt. Der Compiler CFT77 verwendet die Ausgabe des FMP als Eingabe und erzeugt für das modifizierte Programm Maschinen-Code.
Nach dem Autotasking-Konzept kann ein paralleler Bereich an jeder Stelle eines Programms gekennzeichnet werden, während bei Microtasking ein paralleler Bereich grundsätzlich ganze Unterprogramme umfaßt. Bei Verwendung von Autotasking erzeugt der FMP für jeden parallelen Bereich ein Unterprogramm. Alle zur Verfügung stehenden Prozessoren beginnen mit der Bearbeitung der ersten Anweisung dieses Unterprogramms. Innerhalb des parallelen Bereichs können durch Direktiven Kontrollstrukturen festgelegt werden. Möglich sind Kontrollstrukturen zur parallelen Bearbeitung von DO-Schleifen und zur parallelen Bearbeitung unabhängiger Programmblöcke. Am Ende der Kontrollstrukturen wird implizit synchronisiert. Durch Optionen zu den Direktiven sind bei den Kontrollstrukturen für DO-Schleifen verschiedene Partitionierungsstrategien möglich (s. [26] und [1]). Ferner sind Direktiven zur Kennzeichnung kritischer Bereiche, die von mehreren Tasks nicht gleichzeitig bearbeitet werden dürfen, vorhanden. Die Autotasking-Direktiven bilden eine Obermenge der Microtasking-Direktiven. Die vorhandenen Erweiterungen verbessern die Funktionalität des Multitasking-Konzepts. Nach Bearbeitung und Terminierung des parallelen Bereichs (s. [26]) setzt ein Prozessor die Bearbeitung des Programms fort, während die übrigen verfügbaren Prozessoren eine Test-and-Set-Operation auf einem Semaphor-Register ausführen, bis erneut ein paralleler Bereich erreicht wird.
Der Gültigkeitsbereich von privaten und gemeinsamen Variablen der Tasks wird automatisch für jeden parallelen Bereich analysiert, kann aber auch vom Benutzer explizit definiert werden. Dies eröffnet dem Benutzer vielfältige Möglichkeiten, den Gültigkeitsbereich von Variablen zu kontrollieren. Ferner ist der Gültigkeitsbereich von Variablen beim Autotasking-Konzept überschaubarer als bei Microtasking.
Bei Autotasking ist der Overhead durch die Parallelisierungsroutinen gering. Messungen

zum Overhead der Autotasking-Direktiven sind [26] zu entnehmen. Ferner ist die Geschwindigkeit der Synchronisation verglichen mit Microtasking unter dem Betriebssystem COS größer. Statt Funktionsaufrufen wie bei Microtasking verwendet Autotasking Inline-Code zur Synchronisation. Autotasking ist auf den CRAY-Rechnern unter dem Betriebssystem UNICOS verfügbar.
Bei Verwendung von Autotasking kann zur Laufzeit festgestellt werden, ob für einen parallelen Block eine ausreichend große Granularität vorliegt, d.h. ob der zeitliche Vorteil durch parallele Bearbeitung den Overhead durch die Parallelisierungsroutinen überwiegt (Threshold Test). Es ist möglich, die parallele Ausführung eines parallelen Bereichs von einer automatisch oder vom Benutzer spezifizierten Bedingung abhängig zu machen. Ist die Bedingung nicht erfüllt, wird der gekennzeichnete Bereich sequentiell ausgeführt.
Autotasking kann in Programmen gemeinsam mit Micro- und Macrotasking genutzt werden. Die Verwendung von Auto- und Microtasking ist jedoch im gleichen Unterprogramm nicht zulässig. Ferner ist die Schachtelung von parallelen Bereichen oder Kontrollstrukturen weder bei Micro- noch bei Autotasking vorgesehen.
In den meisten Fällen verläuft die automatische Parallelisierung durch Autotasking von Programmen nicht optimal. Der Benutzer kann in diesen Fällen eingreifen, indem er selbst Direktiven einfügt. Kleinere Änderungen der Spezifikation der Parallelisierung sind bei Autotasking leichter möglich als bei Microtasking (s. [10]).

5 Beschreibung der Algorithmen

In diesem Kapitel werden die entwickelten Algorithmen und Programme zur Bestimmung aller Eigenwerte von symmetrischen Tridiagonalmatrizen vorgestellt. Da die Auswertung der Sturmschen Kette (2.7) und der Folge (2.14) bei Matrizen hoher Ordnung sehr zeitaufwendig ist, werden im ersten Abschnitt Ansätze zur Vektorisierung dieser Folgen diskutiert. Es folgt ein Abschnitt mit allgemeinen Überlegungen zum Speedup bei Parallelisierung von Multisektionsverfahren der Ordnung $k > 1$ und Bisektionsverfahren. Schließlich werden die entwickelten Algorithmen und Programme in ihrer Struktur beschrieben, wobei besonderer Wert auf die Darstellung der Parallelisierungs- und Vektorisierungseigenschaften gelegt wird. Ansätze zur Vektorisierung und Parallelisierung der Algorithmen bilden jeweils den Abschluß der einzelnen Abschnitte. Genauere Untersuchungen und Messungen folgen in Kapitel 6.

5.1 Vektorisierbarkeit der Sturmschen Kette und der Folge (2.14)

Die Sturmsche Kette (2.7) ist als lineare Rekursion zweiter Ordnung sowohl vektorisierbar als auch parallelisierbar. Parallele Algorithmen zur Lösung von linearen Rekursionen verwenden gewöhnlich das Prinzip des rekursiven Doppelns, das sowohl Vektorisierung als auch Parallelisierung zuläßt [18], [27]. Der Rekurrente Produktform-Algorithmus (s. [18] und [27]), auf den hier nicht weiter eingegangen wird, ist einer der schnellsten parallelen Algorithmen, um eine lineare Rekursion niedriger Ordnung zu lösen. Auf den CRAY-Rechnern des Forschungszentrums Jülich ist eine vektorisierende Hilfsroutine zur Lösung einer linearen Rekursion zweiter Ordnung verfügbar (SOLR).

Die nichtlineare Rekursion erster Ordnung in (2.14) ist in einem Bisektionsverfahren zur Bestimmung der Eigenwerte reeller symmetrischer Tridiagonalmatrizen nicht vektorisierbar. Der folgende Algorithmus in Pseudocode verdeutlicht dies. Er beschreibt die Berechnung und Auswertung der Folge (2.14) als Kern eines Bisektionsverfahrens und ist mit einigen Korrekturen [29] entnommen.

Algorithmus 5.1 Kern eines Bisektionsverfahrens

$q \leftarrow \alpha_1 - x$
$\nu \leftarrow 0$
if $(q < 0)$ *then* $\nu \leftarrow 1$
for $j = 2 \cdots n$ *do*
  if $(q = 0)$ *then* $q \leftarrow \epsilon$
  $q \leftarrow \alpha_j - x - \beta_j^2/q$
  if $(q < 0)$ *then* $\nu \leftarrow \nu + 1$
end do

ϵ ist ein sehr kleiner positiver Wert (maschinenabhängig), ν die Anzahl der Eigenwerte kleiner als x.
Bei einem Multisektionsverfahren der Ordnung k wird ν an k verschiedenen Punkten $x_i, \quad i = 1, \ldots, k,$ berechnet. Durch Vertauschen der inneren (Laufvariable j) und der äußeren Schleife (Laufvariable i) läßt sich die Berechnung der q- und ν-Werte vektorisieren [29]. Der zusätzliche Aufwand an Speicherplätzen besteht aus drei Feldern der Länge k für die Speicherung der x-, q- und ν-Werte. Im untenstehenden Algorithmus in Pseudocode sind die Schleifen bereits vertauscht und einige Korrekturen gegenüber dem Algorithmus in [29] eingefügt. Der Algorithmus beschreibt die Berechnung und Auswertung der Folge (2.14) als Kern eines Multisektionsverfahrens zur Bestimmung der Eigenwerte reeller symmetrischer Tridiagonalmatrizen.

Algorithmus 5.2 Kern eines Multisektionsverfahrens

for $i = 1 \cdots k$ *do*
  $q(i) \leftarrow \alpha_1 - x(i)$
  $\nu(i) \leftarrow 0$
  if $(q(i) < 0)$ *then* $\nu(i) \leftarrow 1$
end do
for $j = 2 \cdots n$ *do*
  for $i = 1 \cdots k$ *do*
    if $(q(i) = 0)$ *then* $q(i) \leftarrow \epsilon$
    $q(i) \leftarrow \alpha_j - x(i) - \beta_j^2/q(i)$
    if $(q(i) < 0)$ *then* $\nu(i) \leftarrow \nu(i) + 1$
  end do
end do

Die innere Schleife vektorisiert. Es bleibt zu untersuchen, für welches k der Zeitaufwand für die obige Multisektionsschleife auf einem Vektorprozessor optimal ist.

Angenommen, ein Eigenwert, der sich im Intervall $[a_0, b_0]$ mit der Länge $l_0 = b_0 - a_0$ befindet, soll mit Hilfe eines Multisektionsverfahrens berechnet werden. In jedem Multisektionsschritt wird die Länge des Intervalls um den Faktor $(k+1)^{-1}$ verkleinert. Um das Intervall auf eine Länge l_j mit $l_j/l_0 < \epsilon_0$ zu reduzieren, sind j Multisektionsschritte mit

$$j = \lceil \frac{-\ln(\epsilon_0)}{\ln(k+1)} \rceil \tag{5.1}$$

notwendig. Der Zeitaufwand der obigen Multisektionsschleife auf einem Vektorprozessor läßt sich für diesen Fall wie folgt optimieren [29]: Die Berechnung der Folge in (2.14) in k Punkten ist in jedem der j Schritte auf einem Vektorrechner proportional zu

$$t = t_0 + t_e k. \tag{5.2}$$

t ist die Ausführungszeit einer Vektorinstruktion, t_0 die „Startup"-Zeit und t_e die Zeit pro Vektorelement (Anzahl der Takte pro Resultat). Der gesamte Zeitaufwand C der Multisektionsschleife ist dann gegeben durch

$$C = (t_0 + t_e k) \lceil \frac{-\ln(\epsilon_0)}{\ln(k+1)} \rceil. \tag{5.3}$$

C wird minimal für einen Wert k_0, der

$$(k_0 + 1)\ln(k_0 + 1) - k_0 = \frac{t_0}{t_e} \tag{5.4}$$

erfüllt. t_0/t_e entspricht Hockneys $n_{1/2}$, der „Performance-Halbwertslänge". Je nach dem Verhältnis t_0/t_e für die obige Multisektionsschleife, das von der Architektur des Rechners abhängt, ist ein optimales k zu wählen. In Tabelle 5.1 sind für einige k die Werte für t_0/t_e berechnet. Für $t_0 = 0$ (Skalarprozessor) ergibt sich aus (5.3) ein minimales C für $k = 1$ (Bisektion). Zur Bestimmung des optimalen k des gesamten Multisektionsverfahrens zur Berechnung aller Eigenwerte einer reellen symmetrischen Tridiagonalmatrix ist eine Laufzeitanalyse des zugehörigen Programms nötig. Da die Anzahl der Auswertungen der Folge (2.14) bei einem Multisektionsverfahren der Ordnung $k > 1$ gegenüber einem Bisektionsverfahren trotz geringerer Anzahl an Iterationsschritten ansteigt, ist zu untersuchen, ob Multisektion überhaupt Vorteile gegenüber Bisektion besitzt. Es ist zu erwarten, daß dies stark rechnerabhängig ist, da die Schnelligkeit der Vektor- gegenüber der Skalareinheit des verwendeten Rechners eine entscheidende Rolle spielt. Multisektion ist nur dann schneller als Bisektion, wenn die innere Schleife des obigen Multisektionsverfahrens schon bei k der Größenordnung 10 von der Vektoreinheit des verwendeten Rechners wesentlich schneller abgearbeitet werden kann als von der Skalareinheit.

t_0/t_e	k	t_0/t_e	k
0.0	1	14.1	9
1.3	2	16.4	10
2.6	3	18.9	11
4.1	4	21.4	12
5.8	5	24.0	13
7.7	6	26.7	14
9.7	7	29.4	15
11.8	8	32.2	16

Tabelle 5.1: Optimale Anzahl von Multisektionspunkten

5.2 Speedup bei paralleler Bisektion und paralleler Multisektion

Der Speedup eines parallelen Algorithmus gegenüber einem sequentiellen Algorithmus ist definiert durch:

$$S_p = \frac{T_1}{T_p}. \tag{5.5}$$

T_p sei die Zeitkomplexität eines parallelen Algorithmus für ein p-Prozessor-System, d.h. die Anzahl der Zeitschritte. T_1 ist dementsprechend die Zeitkomplexität, die derselbe oder ein entsprechender (d.h. das Problem lösender, ggf. dieselbe Methode benutzender) Algorithmus auf einem seriellen Rechner (mit einem Prozessor entsprechend angepaßter Leistung) benötigt [18]. Im allgemeinen gilt $1 \leq S_p \leq p$. Es gibt Sonderfälle, in denen diese Speedup-Grenzen nicht gewährleistet sind.
Gegeben sei ein Intervall $I = [a, b]$, das alle n Eigenwerte einer reellen symmetrischen Tridiagonalmatrix A in der Gestalt (2.2) enthält. Bei einem sequentiellen Verfahren zur Bestimmung der Eigenwerte wird das Intervall an einem inneren Punkt getestet (s. 2.2). Man erhält ein oder zwei kleinere Intervalle, die Eigenwerte enthalten. Unterteilt man das Intervall I in m Teilintervalle, so läßt sich Parallelismus auf zweifache Weise einführen [4]:

1. Einige oder alle der m Teilintervalle werden gleichzeitig an einem inneren Punkt untersucht.

2. Ein Teilintervall wird an mehreren inneren Punkten gleichzeitig getestet.

Bei paralleler Bisektion werden p der m Teilintervalle gleichzeitig an einem Punkt untersucht. p ist die Anzahl der zur Verfügung stehenden Prozessoren. Teilintervalle, die

Eigenwerte enthalten, werden ermittelt, indem man zunächst das Intervall I am Bisektionspunkt testet. Die erhaltenen ein oder zwei nichtleeren Teilintervalle werden wiederum solange geteilt, bis man m nichtleere Teilintervalle von I erhält. Der gegenüber dem sequentiellen Verfahren maximal erreichbare Speedup beträgt p. Dies setzt eine gleichmäßige Verteilung der Eigenwerte in den m Teilintervallen mit $m = ip, \quad i = 1, 2, \ldots$ voraus. Der günstigste Fall liegt vor, wenn sich bei $p = n$ Prozessoren und $m = n$ Teilintervallen genau ein Eigenwert in jedem Teilintervall befindet. Der Speedup beträgt dann n. Im ungünstigsten Fall liegen alle Eigenwerte sehr nah beieinander. Der Speedup ist dann minimal und beträgt 1.
Bei paralleler Multisektion werden die m Teilintervalle von I der Reihe nach behandelt, jedoch wird jedes einzelne Teilintervall an k Punkten gleichzeitig getestet. Die Anzahl der bei diesem Verfahren einsetzbaren Prozessoren p ist beliebig und bestimmt die Zahl der Testpunkte k. Im folgenden ist vorausgesetzt, daß die Teilintervalle, die untersucht werden, Eigenwerte enthalten. Dies ist zu erreichen, indem das Multisektionsverfahren zunächst auf das gesamte Intervall I angewendet wird. Nichtleere Teilintervalle werden dann erkannt und gesammelt. Der ungünstigste Fall liegt vor, wenn alle Eigenwerte sehr nah beieinander liegen. Jeder Multisektionsschritt produziert dann nur ein Teilintervall, das Eigenwerte enthält. Ein sequentielles Bisektionsverfahren würde dasselbe Ergebnis nach $\log_2(k+1)$ Schritten erreichen. Der minimale Speedup beträgt also $\log_2(k+1)$. Im günstigsten Fall zerfällt jedes Teilintervall in jedem Multisektionsschritt so in $(k+1)$ kleinere Intervalle, daß alle neuen Teilintervalle Eigenwerte enthalten. Sequentielle Bisektion kann dies erst in k Schritten erreichen. Der maximale Speedup beträgt also k. Jedoch wird, je mehr Teilintervalle erzeugt werden, die Zahl der enthaltenen Eigenwerte kleiner als k werden. Folglich wird der temporale Speedup im Verlauf des Algorithmus auf $\log_2(k+1)$ fallen.
Eine Verbindung beider Verfahren verspricht eine Verbesserung der unteren Speedupgrenze. Liegen m Teilintervalle vor, die Eigenwerte enthalten, so werden jedem Teilintervall $l = p/m$ Prozessoren zugeteilt. Den Speedup dieses Verfahrens erhält man durch das Produkt aus der Zahl der parallel behandelten Teilintervalle (m) und dem Speedup für das Testen eines einzelnen Intervalls in k Punkten. Die Speedupgrenzen sind also $m\log_2(l+1)$ nach unten und $ml = p$ nach oben. Da $1 \leq m \leq n$ gilt, ist der immer erreichbare Speedup nach unten durch $\log_2(p+1)$ begrenzt und nach oben durch p für $p \leq n$ oder $n\log_2(p/n+1)$ für $p > n$ [4].

5.3 Beschreibung der Algorithmen und Programme

5.3.1 Allgemeine Strukturierung

Die grundlegende Programmstruktur ist in Abbildung 5.1 dargestellt.

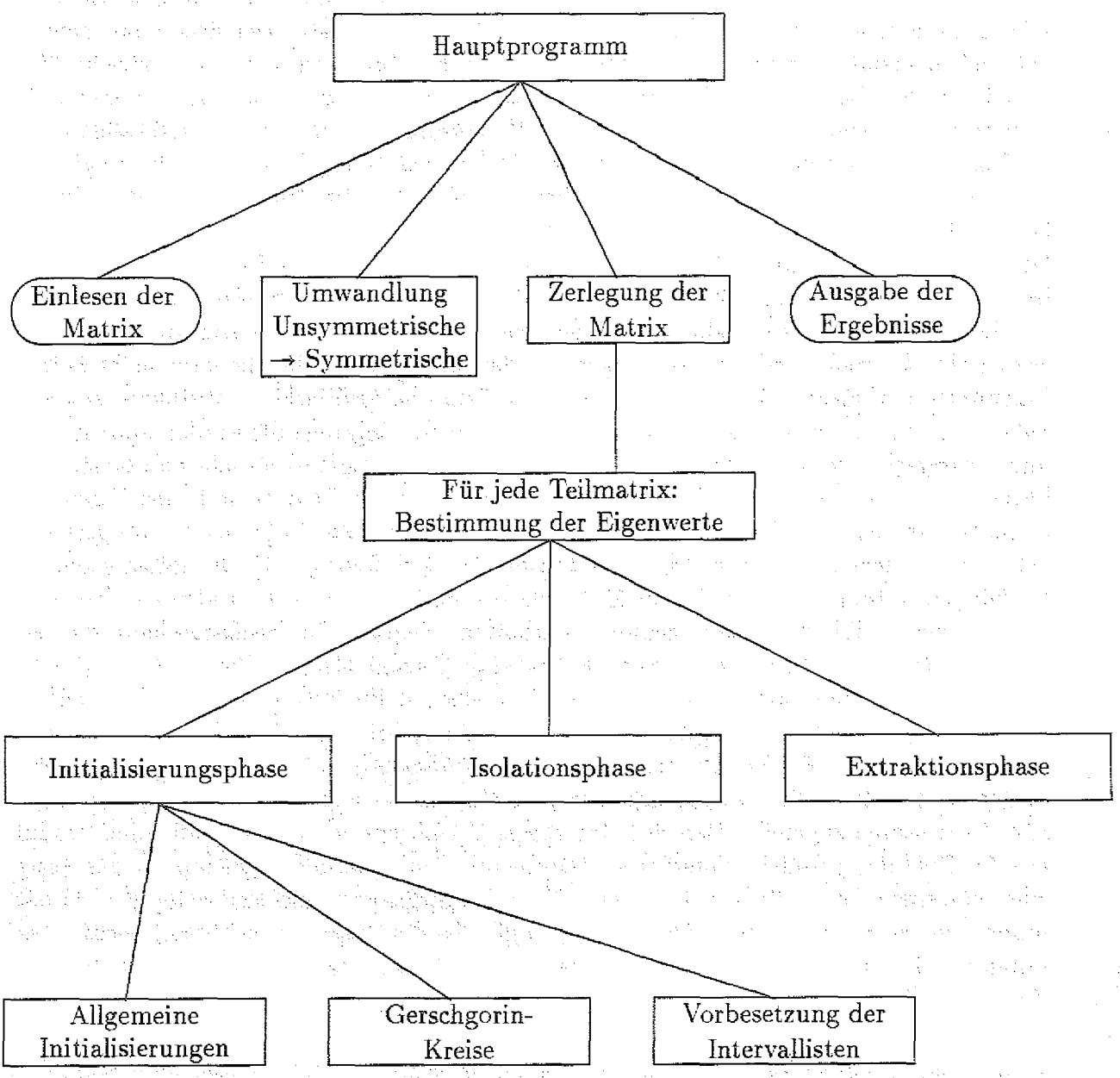


Abb. 5.1: Grundlegende Struktur aller Programmversionen

Zunächst erfolgt das Einlesen der Matrix (Ordnung n). Liegt eine unsymmetrische Tridiagonalmatrix mit $\beta_i\gamma_i \geq 0$ für $i = 2, \ldots, n$, vor, wird diese in eine symmetrische Tridiagonalmatrix mit $\beta_i = \sqrt{\beta_i\gamma_i}$ für $i = 2, \ldots, n$, umgewandelt (s. 2.3.2).
Die Elemente der Diagonalen und einer Nebendiagonalen der symmetrischen Tridiagonalmatrix werden in den eindimensionalen Feldern α und β abgespeichert. Aus programmtechnischen Gründen werden β_1 sowie α_{n+1} und β_{n+1} zu Null gesetzt. Bei der Programmierung der Gerschgorin-Kreise (s. auch 2.4) und der Zerlegung der Matrix (s. auch 2.3.1) führt dies zu Code-Einsparungen. In dem Feld α besitzen die Elemente der Diagonalen einer Matrix der Gestalt 2.2 die Indizes 1 bis n, in dem Feld β die Elemente einer Nebendiagonalen die Indizes 2 bis n.
Treten $\beta_i = 0$ auf, so wird die Matrix in Teilmatrizen zerlegt (s. Abbildung 5.2). Zur Bestimmung der Indizes der β_i mit $\beta_i = 0$ im Feld β steht auf den CRAY-Rechnern die vektorisierende Routine ISRCHEQ zur Verfügung. Die folgenden Phasen werden dann für jede Teilmatrix durchlaufen.
Die erste Programmphase ist die Initialisierungsphase. Sie dient der Bestimmung von Intervallen, die mehrere bzw. einen Eigenwert enthalten, um die Listen der Isolationsphase vorzubesetzen.
In der Isolationsphase werden dann Intervalle ermittelt, in denen sich genau ein Eigenwert befindet.
Die in der Isolationsphase erhaltenen Intervalle werden an die Extraktionsphase übergeben. Dort werden die Eigenwerte in vorgegebener Genauigkeit bestimmt.

5.3.2 Initialisierungsphase

In der Initialisierungsphase wird zunächst die Zahl ϵ bestimmt, durch die q_{n-i}, die gleich Null sind, ersetzt werden (s. die Algorithmen 5.1 und 5.2). ϵ wird in Abhängigkeit von dem maximalen Betrag der Matrixelemente und dem Rundungsfehler der Maschine nach der folgenden Gleichung ermittelt:

$$\epsilon = \max\Big(\max_{i=1,\ldots,n}(|\alpha_i|), \max_{i=2,\ldots,n}(|\beta_i|)\Big) \cdot \delta_f. \tag{5.6}$$

Hierin ist n die Ordnung der Matrix. Auf den CRAY-Rechnern beträgt die Rechengenauigkeit bei einfach genauen REAL-Zahlen 14 Dezimalstellen, der Zahlenbereich reicht von 10^{-2466} bis 10^{2466}. δ_f ist im Programm zu 10^{-14} gesetzt. Dies entspricht ungefähr dem dezimalen Rundungsfehler der Mantisse von einfach genauen REAL-Zahlen auf den CRAY-Rechnern. Durch diese Wahl des ϵ wird Overflow in dem Quotienten von (2.14) sehr unwahrscheinlich, da das Matrixelement mit dem maximalen Betrag in die Berechnung von ϵ eingeht. Eine andere Möglichkeit besteht darin, ϵ gleich 10^{-14} zu setzen. In diesem Fall ist Overflow im Quotienten von (2.14) wesentlich wahrscheinlicher, da im Zähler β_i^2 vorkommt.

Initialisierung: die 1. Teilmatrix beginnt mit dem Index 1 in den Feldern α, β `ERSTER_INDEX:=1`
Bestimmen des Indexes `INDEX` des ersten β_i mit $\beta_i = 0$ ab `ERSTER_INDEX+1` einschließlich
Die Teilmatrix endet mit dem Index `INDEX-1` in den Feldern α, β `LETZTER_INDEX:=INDEX-1`
Ermittlung der Eigenwerte der Teilmatrix
Aktualisierung: die nächste Teilmatrix beginnt mit dem Index `LETZTER_INDEX+1` in den Feldern α, β `ERSTER_INDEX:=LETZTER_INDEX+1`
Bis der Index des letzten Elementes `LETZTER_INDEX` der Ordnung n der ursprünglichen Matrix entspricht

Abb. 5.2: Zerlegung in Teilmatrizen

Als nächstes wird ein Intervall bestimmt, das alle Eigenwerte der Matrix enthält. Die Intervallgrenzen liefern die Gerschgorin-Kreise (s. 2.4).
Das erhaltene Intervall wird dann durch k innere Punkte in $k+1$ Teilintervalle geteilt. Diese Teilintervalle werden mit Hilfe der Sturmschen Kette bzw. der Folge (2.14) daraufhin getestet, wieviele Eigenwerte sie enthalten. Teilintervalle mit mehreren Eigenwerten werden in die Intervalliste der Isolationsphase, Teilintervalle mit genau einem Eigenwert in die Intervalliste der Extraktionsphase eingetragen. Das durch die Gerschgorin-Kreise ermittelte Intervall läßt sich auf zweierlei Weise unterteilen:

1. Man teilt das Intervall äquidistant durch k Punkte (Multisektion der Ordnung k). Hierbei ist keine Aussage darüber möglich, wieviele Teilintervalle überhaupt Eigenwerte enthalten. Sind jedoch mindestens soviele nichtleere Teilintervalle entstanden, wie Prozessoren zur Verfügung stehen, so kann jeder Prozessor schon im ersten Schritt der Isolationsphase mindestens ein Teilintervall bearbeiten. Die Wahrscheinlichkeit, viele nichtleere Teilintervalle nach der Initialisierungsphase zu erhalten, steigt mit der Ordnung der Matrix. Dies ist nicht der Fall, wenn die meisten Eigenwerte sehr

nah beieinanderliegen. Es ist vorteilhaft, schon nach der Initialisierungsphase soviele Teilintervalle, wie Prozessoren vorhanden sind, zur Verfügung zu haben. Da die Mehrprozessorsysteme CRAY X-MP und Y-MP 4 bzw. 8 Prozessoren besitzen, sind als Anzahl der Teilintervalle Vielfache von 4 bzw. 8 zu bevorzugen.

2. Bei der zweiten Variante bestimmt man die inneren Testpunkte als Nullstellen der Glieder $p_{n-1}(x)$ und $p_{n-2}(x)$ der Sturmschen Kette. Hierbei wird vorausgesetzt, daß die Matrix eine Ordnung $n \geq 2$ besitzt. Nullstelle von $p_{n-1}(x)$ ist α_1, die zwei Nullstellen von $p_{n-2}(x)$ können als Lösungen der quadratischen Gleichung $p_{n-2}(x) = 0$ bestimmt werden. Da die Nullstellen von $p_{n-i}(x)$ die von $p_{n-(i+1)}(x)$ strikt trennen, liegt α_1 zwischen den beiden Nullstellen von $p_{n-2}(x)$. Ferner enthalten mindestens 2 der 4 entstandenen Teilintervalle Eigenwerte, da die Nullstellen von $p_{n-2}(x)$ die des charakteristischen Polynoms $p_0(x)$ in mindestens zwei nicht leere Gruppen unterteilen. In den beiden äußeren Teilintervallen - von den Intervallgrenzen des Ausgangsintervalls bis zur jeweils nächstgelegenen Nullstelle von $p_{n-2}(x)$ - befinden sich gemäß (2.8) immer Nullstellen des charakteristischen Polynoms $p_0(x)$. Da die Bestimmung der Nullstellen weiterer Glieder der Sturmschen Kette zu aufwendig wäre (Grad der Polynome größer oder gleich 3), werden bei diesem Verfahren höchstens vier Teilintervalle mit Eigenwerten ermittelt. Daher können im ersten Schritt der Isolationsphase höchstens vier Teilintervalle parallel bearbeitet werden.

Vektorisierung und Parallelisierung

Die Initialisierungsphase läßt sich an mehreren Stellen gut vektorisieren. Bei der Bestimmung von ϵ kann die vektorisierende CRAY-Routine ISAMAX verwendet werden. ISAMAX ermittelt den Index des absolut größten Elementes eines Vektors. Die Intervallgrenzen des Intervalls, das alle Eigenwerte enthält, werden durch die Gerschgorin-Kreise bestimmt. Dies läßt sich mit Hilfe der vektorisierenden CRAY-Routinen ISMIN und ISMAX gut vektorisieren. ISMIN bzw. ISMAX berechnen den Index des kleinsten bzw. größten Elementes eines Vektors. Sowohl bei der Bestimmung von ϵ als auch der Intervallgrenzen beträgt die Vektorlänge n (Ordnung der Matrix), kann also beliebig groß werden. Zur Auswertung der Sturmschen Kette läßt sich die vektorisierende CRAY-Routine SOLR zur Berechnung einer linearen Rekursion zweiter Ordnung heranziehen. Die Vektorlänge ist hier ebensfalls n. Die Berechnung der Folge (2.14) läßt sich mit Hilfe des Multisektionsverfahrens aus Algorithmus 5.2 vektorisieren. Die Vektorlänge hängt hier von der Anzahl der inneren Teilungspunkte des Ausgangsintervalls ab und ist gewöhnlich klein (von der Größenordnung 10).
Die Initialisierungsphase ist nicht parallelisiert, da der Anteil am Gesamtprogramm nur gering ist. Prinzipiell sind die auftretenden vektorisierbaren Schleifen auch parallelisierbar.

5.3.3 Isolationsphase

Bisektions- und Multisektionsverfahren ohne Aufteilung in Gruppen

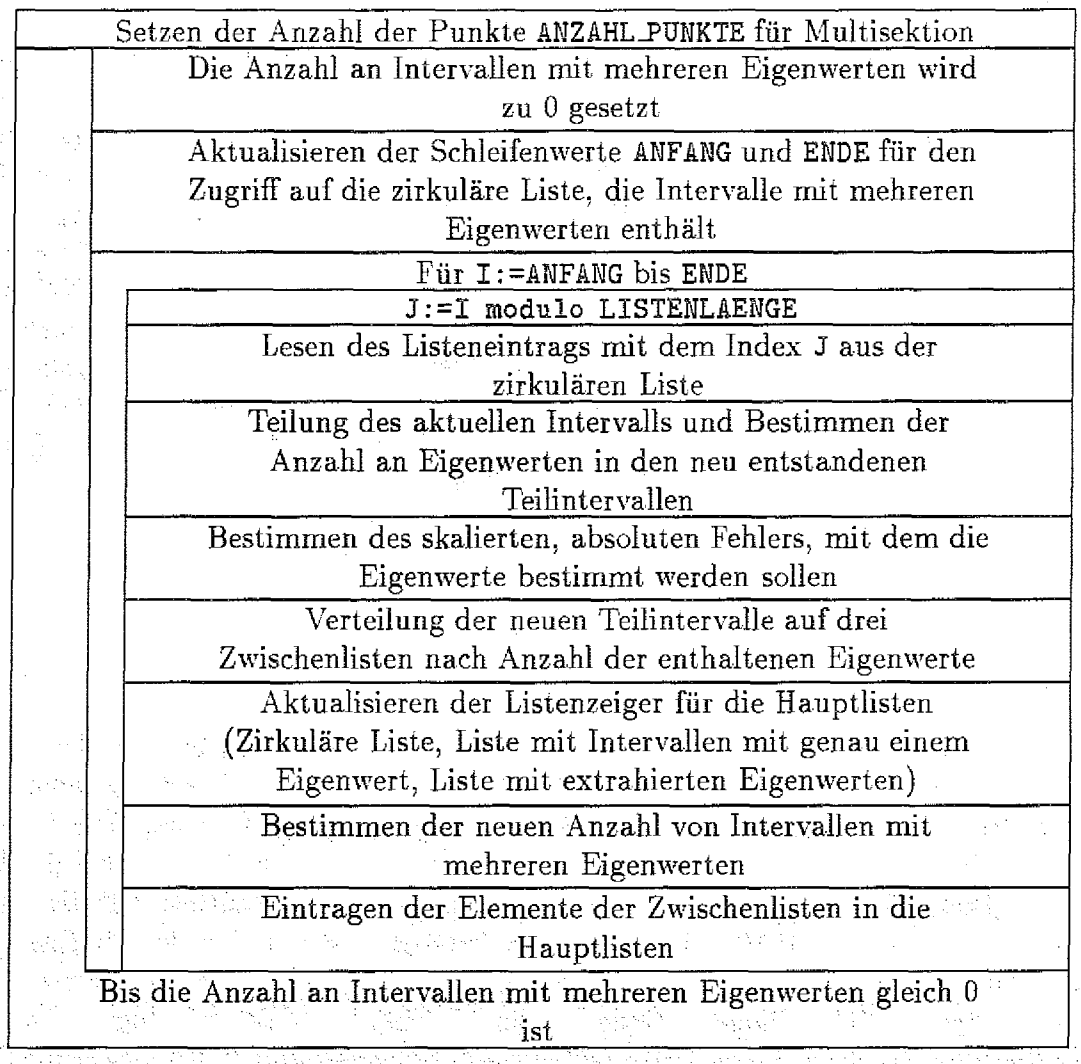


Abb. 5.3: **Struktur der Isolationsphase ohne Aufteilung in Gruppen**

In der Isolationsphase werden Intervalle bestimmt, die genau einen Eigenwert der Matrix enthalten. Die Eigenwerte werden isoliert. Den Aufbau der Isolationsphase zeigt Abbildung 5.3.
Zunächst wird die Anzahl der Punkte gesetzt, durch die ein Intervall, das mehrere Eigenwerte enthält, in Teilintervalle unterteilt wird. Alle Intervalle, die mehrere Eigenwerte

enthalten, werden in dieser Variante der Isolationsphase einzeln bearbeitet. Die Isolationsphase ist beendet, sobald keine Intervalle mit mehreren Eigenwerten mehr vorhanden sind. Dies wird nach jedem Isolationsschritt überprüft (äußere Schleife).
Die Datenstruktur, auf der in der Isolationsphase hauptsächlich gearbeitet wird, ist eine zirkuläre Liste, die nach jedem Isolationsschritt die aktuellen Intervalle mit mehreren Eigenwerten enthält. Nach jedem Isolationsschritt befinden sich die aktuellen Intervalle auf neuen Listenplätzen, die direkt hinter den Listenplätzen der Intervalle aus dem vorherigen Isolationsschritt liegen. Ist der letzte Listenplatz besetzt, so werden weitere aktuelle Intervalle auf den ersten Listenplätzen abgelegt. In den Abbildungen 5.4 und 5.5 ist die zirkuläre Liste in den Zuständen vor und nach einem beliebigen Schritt der Isolationsphase kreisförmig dargestellt.

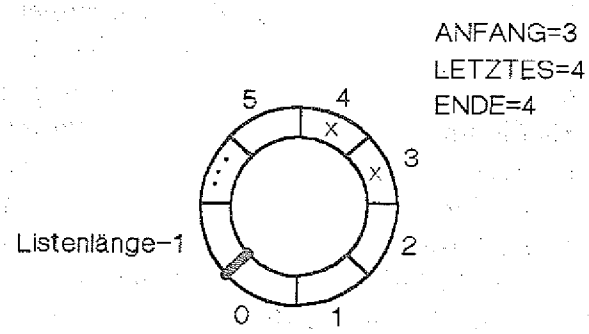


Abb. 5.4: Zirkuläre Liste vor dem nächsten Isolationsschritt. Die mit „x" gekennzeichneten Listenplätze enthalten die aktuellen Intervalle.

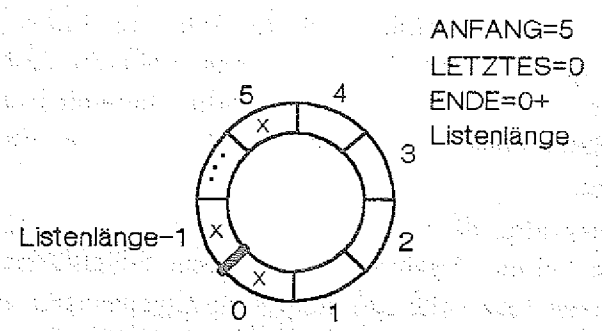


Abb. 5.5: Zirkuläre Liste nach dem nächsten Isolationsschritt. Die mit „x" gekennzeichneten Listenplätze enthalten die neuen aktuellen Intervalle.

Ein Listenelement enthält als Einträge die untere und obere Intervallgrenze des jeweiligen aktuellen Intervalls sowie die Anzahl der Eigenwerte, die kleiner als die untere Intervallgrenze sind und die Anzahl der Eigenwerte, die kleiner als die obere Intervallgrenze sind.

In FORTRAN ist die zirkuläre Liste durch vier Felder realisiert. Die Dimension dieser Felder entspricht der Ordnung der größten Matrix, die bearbeitet werden kann. Die Ordnung dieser Matrix ist bis auf die Beschränkung durch den verfügbaren Speicherplatz frei wählbar und ist in allen Programmversionen zu 10000 gesetzt. Diese Dimensionierung ist ausreichend, da die Liste nur Intervalle mit mehreren Eigenwerten enthält. Die größte Anzahl an Intervallen mit mehreren Eigenwerten liegt vor, wenn alle Intervalle genau zwei Eigenwerte enthalten. Sie läßt sich ermitteln, indem man die Ordnung der größten Matrix, die bearbeitet werden kann, durch 2 dividiert (ganzzahlige Division). Im ungünstigsten Fall werden dann im aktuellen Isolationsschritt keine Eigenwerte isoliert, und es wird wieder die gleiche Anzahl an Intervallen mit je zwei Eigenwerten in die Liste eingetragen. Überschreiben der aktuellen Intervalle durch die neuen Intervalle ist bei der oben genannten Dimensionierung also auszuschließen. Zwei Felder enthalten die Intervallgrenzen, zwei die Anzahl der Eigenwerte, die kleiner als die untere Intervallgrenze sind und die Anzahl der Eigenwerte, die kleiner als die obere Intervallgrenze sind. Eine zirkuläre Liste als Datenstruktur ist Speicherplatz sparend, da die Daten kreisförmig umlaufen, d.h. ist der letzte Listenplatz besetzt, werden die nächsten Daten auf dem ersten Listenplatz abgelegt. Da nicht vorab bestimmt werden kann, wieviele Isolationsschritte benötigt werden und wieviele aktuelle Intervalle mit mehreren Eigenwerten in den einzelnen Isolationsschritten gefunden werden, ist die zirkuläre Liste eine dem Problem angemessene Datenstruktur.

Vor jedem Isolationsschritt werden die Schleifenwerte **ANFANG** und **ENDE** aktualisiert. **ANFANG** wird gleich dem Index des ersten Elementes der zirkulären Liste, die die aktuellen Intervalle mit mehreren Eigenwerten enthält, gesetzt. **ANFANG** enthält damit den Index des ersten aktuellen Intervalls. Ist der Index des letzten Elementes größer oder gleich dem des ersten Elementes, so wird **ENDE** der Index des letzten Elementes zugewiesen (Index des letzten aktuellen Intervalls). Ansonsten erhält **ENDE** den Wert des Indexes des letzten Elementes und wird anschließend um die Listenlänge erhöht (s. auch die Abbildungen 5.4 und 5.5). Dies ist aus programmtechnischen Gründen notwendig, da jeder Isolationsschritt als Zählschleife von **ANFANG** bis **ENDE** realisiert ist. Eine Zählschleife ist zur Parallelisierung jedes Isolationsschrittes notwendig.

Danach werden alle aktuellen Intervalle, die mehrere Eigenwerte enthalten, in der Zählschleife von **ANFANG** bis **ENDE** bearbeitet. Nach Verlassen dieser Schleife ist der aktuelle Isolationsschritt beendet. Sind neue Intervalle mit mehreren Eigenwerten ermittelt worden, so wird der nächste Isolationsschritt eingeleitet. In der Schleife wird zunächst mit Hilfe der MODULO-Funktion auf einen Listeneintrag der zirkulären Liste (Index J) zugegriffen. Es folgt die Teilung des Intervalls und die Bestimmung der jeweiligen Anzahl der Eigenwerte in den neu entstandenen Teilintervallen.

Zur Intervallteilung sind sowohl Multisektion der Ordnung $k > 1$ als auch Bisektion geeignet. Bei Multisektion ist zu untersuchen, welche Anzahl an inneren Teilungspunkten k optimal ist, d.h. den geringsten Zeitaufwand benötigt. Bisektion ist in jedem Isolationsschritt mit weniger Aufwand verbunden, allerdings sind mehr Schritte zu erwarten. Hier muß geprüft werden, welches Verfahren den geringeren Zeitaufwand besitzt. Die Bestimmung der Anzahl an Eigenwerten kann sowohl über die Sturmsche Kette als auch über die Folge (2.14) erfolgen.
Als nächstes wird der Fehler, mit dem die Eigenwerte höchstens berechnet werden sollen, ermittelt. Dies geschieht mit Hilfe der Punkte a und b mit $a < b$ aus dem aktuellen Intervall nach folgender Gleichung:

$$\Delta_{skal} = 2 \cdot \delta_f \cdot (|a| + |b|) \tag{5.7}$$

δ_f ist der in (5.6) auftretende Fehler. Wird Multisektion der Ordnung $k > 1$ durchgeführt, so werden für a und b zwei Teilungspunkte aus der Mitte des aktuellen Intervalls gewählt, bei Bisektion die Intervallgrenzen des aktuellen Intervalls. Δ_{skal} ist der skalierte, absolute Fehler der Eigenwerte. Der Fehler ist skaliert, da die Summe der Beträge der Intervallgrenzen als Faktor in die Berechnung des Fehlers eingeht. Die Summe der Beträge der Intervallgrenzen wird verwendet, da sie nur Null ist, wenn beide Intervallgrenzen den Wert Null haben. In diesem Fall wäre der Eigenwert bestimmt. Wählte man als Skalierungsfaktor den Betrag nur einer Intervallgrenze, ergäbe sich beim Wert Null der Intervallgrenze als Fehler Null. Dies könnte im Laufe der weiteren Rechnung zu Schwierigkeiten führen. Im Algorithmus wird Δ_{skal} mit der Intervallbreite der neuen Teilintervalle verglichen. Δ_{skal} wird hier wie ein absoluter Fehler verwendet. Durch die Skalierung ist gesichert, daß die Anzahl der bestimmten Stellen der Mantisse des Eigenwerts das Genauigkeitskriterium darstellt.
Es folgt die Verteilung der neuen Teilintervalle auf drei Zwischenlisten. Kriterium ist dabei, wieviele Eigenwerte die Teilintervalle enthalten. Die Zwischenlisten sind als Felder der Dimension von der Größenordnung der Anzahl der inneren Teilungspunkte k (maximal 20) realisiert. Den genauen Algorithmus für die Verteilung der Teilintervalle zeigt Abbildung 5.6.
Zunächst werden die drei Listenzeiger für die Zwischenlisten mit Null initialisiert. Ist die Intervallbreite der neuen Teilintervalle kleiner oder gleich dem skalierten, absoluten Fehler Δ_{skal}, so ist die erwünschte Genauigkeit bei der Eigenwertbestimmung erreicht. Jedes neue Teilintervall, das Eigenwerte enthält, wird dann in einer Schleife über alle neuen Teilintervalle einer der drei Zwischenlisten zugeteilt. Ist die Anzahl der enthaltenen Eigenwerte größer als 1 und die Genauigkeit bei der Eigenwertbestimmung noch nicht erreicht, so werden die Intervallgrenzen sowie die Anzahl der Eigenwerte, die kleiner als die untere Intervallgrenze sind und die Anzahl der Eigenwerte, die kleiner als die obere Intervallgrenze sind, in die erste Liste eingetragen. Ist dagegen die Anzahl der enthaltenen Eigenwerte größer als 1 und die Genauigkeit bei der Eigenwertbestimmung bereits erreicht,

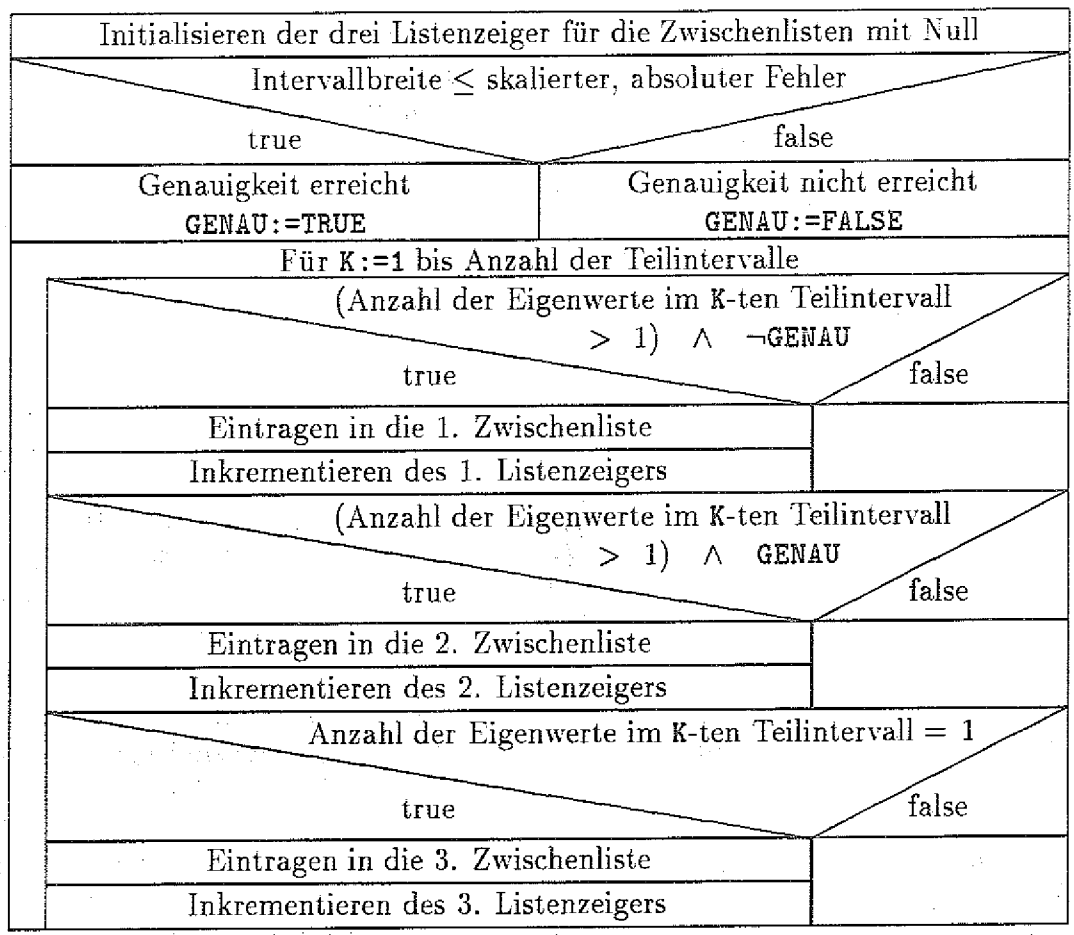


Abb. 5.6: **Verteilung der Teilintervalle auf drei Zwischenlisten**

so werden der Intervallmittelpunkt und die Anzahl der im Intervall enthaltenen Eigenwerte als Vielfachheit des Eigenwerts in die zweite Liste eingetragen. Hierbei kann es sich nur um eine numerische Vielfachheit des Eigenwerts, d.h. verschiedene Eigenwerte liegen sehr nah beieinander, handeln, da echte mehrfache Eigenwerte nicht auftreten können (s. Satz 2.2). Ein Eintrag in die dritte Zwischenliste erfolgt, wenn das Teilintervall genau einen Eigenwert enthält. Eingetragen werden die Intervallgrenzen sowie die Anzahl der Eigenwerte, die kleiner als die untere Intervallgrenze sind und die Anzahl der Eigenwerte, die kleiner als die obere Intervallgrenze sind. Im Falle eines Eintrags wird der Listenzeiger der entsprechenden Liste inkrementiert.

Danach werden die Listenzeiger der drei Hauptlisten aktualisiert. Die Hauptlisten sind

einmal die zirkuläre Liste, die Intervalle mit mehreren Eigenwerten enthält, dann eine Liste mit Intervallen mit genau einem Eigenwert - diese Liste wird nach der Isolationsphase an die Extraktionsphase übergeben - und schließlich die Liste mit bereits extrahierten, d.h. auf die gewünschte Genauigkeit bestimmten Eigenwerten. Diese Listen sind als Felder der Dimension maximale Ordnung der Matrix realisiert. Die Dimension dieser Felder ist bei Matrizen hoher Ordnung wesentlich größer als bei den entsprechenden Feldern der Zwischenlisten. Anschließend wird die Anzahl an Intervallen mit mehreren Eigenwerten aktualisiert, indem die Anzahl an neuen Intervallen mit mehreren Eigenwerten zu dem vorherigen Wert addiert wird.

Schließlich werden die Elemente der Zwischenlisten in die Hauptlisten eingetragen.

Der Isolationsschritt endet, sobald alle aktuellen Intervalle abgearbeitet sind. Sind im letzten Isolationsschritt keine neuen Teilintervalle mit mehreren Eigenwerten gefunden worden, ist die Isolationsphase beendet.

Vektorisierung und Parallelisierung Die Vektorisierung der Sturmschen Kette und der Folge (2.14) bei Multisektion geschieht in gleicher Weise wie in der Initialisierungsphase (s. 5.3.2). Wird Bisektion durchgeführt, ist die Folge (2.14) nicht zu vektorisieren. Ebenso lassen sich nur bei Multisektion das Abspeichern der Teilintervalle in die Zwischenlisten und das Umspeichern der Zwischenlisten in die Hauptlisten vektorisieren. Die Vektorlängen besitzen dabei die Größenordnung der Anzahl der inneren Teilungspunkte (meist 15). Außer bei der Vektorisierung der Sturmschen Kette (Vektorlänge von der Größenordnung Ordnung der Matrix) sind also die Vektorlängen klein. Zusammenfassend ist zu sagen, daß bei Verwendung der Folge (2.14) und Multisektion Vektorisierung nur mit geringen Vektorlängen in der Isolationsphase möglich ist, während die Sturmsche Kette auch Vektorisierung mit großen Vektorlängen zuläßt. Allerdings neigt die Sturmsche Kette, wie schon in 2.2.2 erwähnt, bei der Auswertung zu Zahlenbereichsüberschreitungen (Under- bzw. Overflow).

Zur Parallelisierung bietet sich in der Isolationsphase die Schleife zur Bearbeitung aller aktuellen Intervalle an (s. Abbildung 5.3: Für `I:=ANFANG` bis `ENDE`). Jedes aktuelle Intervall kann unabhängig von den übrigen behandelt werden. Bei paralleler Bearbeitung sind die Zwischenlisten für jede Task (1 Schleifendurchlauf) private, d.h. jede Task besitzt eigene Zwischenlisten. Die Hauptlisten dagegen sind für alle Tasks gemeinsame (shared), d.h. alle Tasks greifen auf den gleichen Speicherbereich zu. Das Aktualisieren der Listenzeiger für die Hauptlisten sowie das Bestimmen der neuen Anzahl von Intervallen mit mehreren Eigenwerten muß in einen kritischen Bereich eingeschlossen werden, den mehrere Tasks nicht gleichzeitig bearbeiten können.

Bisektionsverfahren mit Partitionierung

Dieser Variante der Isolationsphase liegt eine ähnliche Vorgehensweise wie der oben beschriebenen zugrunde. Hier werden jedoch die Intervalle, die mehrere Eigenwerte enthalten, in jedem Isolationschritt in Gruppen aufgeteilt. Alle Elemente einer Gruppe werden zusammen bearbeitet.
Die Datenstrukturen, auf denen in dieser Variante der Isolationsphase hauptsächlich gearbeitet wird, sind zwei lineare Listen. Die erste Liste enthält die Intervalle mit mehreren Eigenwerten, die im vorherigen Isolationsschritt ermittelt wurden. Aus dieser Liste wird im aktuellen Isolationsschritt gelesen, sie wird im folgenden als Eingabeliste bezeichnet. Die zweite Liste enthält die Teilintervalle, die im aktuellen Isolationsschritt entstanden sind und mehrere Eigenwerte enthalten. Diese Liste wird im aktuellen Isolationsschritt beschrieben, sie wird im folgenden als Ausgabeliste bezeichnet. Die Ausgabeliste wird in der Initialisierungsphase mit den Intervallen mit mehreren Eigenwerten, die in dieser Phase ermittelt werden, vorbesetzt. Ein- und Ausgabeliste sind als vier zweidimensionale Felder implementiert. Je eine Spalte dieser Felder enthält die Elemente einer der beiden Listen. In zwei Feldern werden die Intervallgrenzen der Intervalle mit mehreren Eigenwerten abgespeichert, in den zwei anderen die Anzahl der Eigenwerte, die kleiner als die Intervallgrenzen sind. Die maximale Anzahl an Einträgen in jeder Spalte dieser Felder ist durch die Hälfte der Ordnung der größten Matrix, die bearbeitet werden kann, gegeben. In zwei INTEGER-Variablen wird festgehalten, welche Spalte der Felder im aktuellen Isolationsschritt die Ein- bzw. Ausgabeliste darstellt. Die beiden Variablen enthalten den Spaltenindex (1 oder 2) der jeweiligen Liste. Durch Vertauschen der Werte dieser Variablen wird die Ausgabeliste des aktuellen Isolationsschrittes zur Eingabeliste des folgenden und umgekehrt. Die Verwendung der linearen Listen spart gegenüber der zirkulären Liste aus der vorherigen Variante Aufwand zur Listenverwaltung.
Abbildung 5.7 zeigt den Aufbau der Isolationsphase bei Bisektion mit Partitionierung.
Zunächst wird die Anzahl der Intervalle, die mehrere Eigenwerte enthalten, mit Null initialisiert. Danach wechseln Ein- und Ausgabeliste aus dem vorherigen Isolationsschritt ihre Funktion. Die Ausgabeliste des vorherigen Isolationsschritts wird zur Eingabeliste des aktuellen und umgekehrt. Anschließend wird die Anzahl der Gruppen, deren Elemente Intervalle mit mehreren Eigenwerten aus der Eingabeliste sind, bestimmt. Die Anzahl der Gruppen hängt sowohl von der Zahl der eingesetzten Prozessoren als auch von der Anzahl der Intervalle mit mehreren Eigenwerten in der Eingabeliste des aktuellen Isolationschritts ab. Beim Einsatz von 8 Prozessoren werden diese Intervalle in 8 Gruppen aufgeteilt. Enthält jedoch die Eingabeliste weniger als 8 Einträge, so werden entsprechend weniger Gruppen gebildet, die dann jeweils aus einem Element bestehen. Als nächstes folgt eine Zählschleife, in der alle Gruppen des aktuellen Isolationsschritts bearbeitet werden. Innerhalb dieser Schleife werden zu Beginn die Indizes der Gruppenelemente der jeweiligen Gruppe in der Eingabeliste ermittelt.

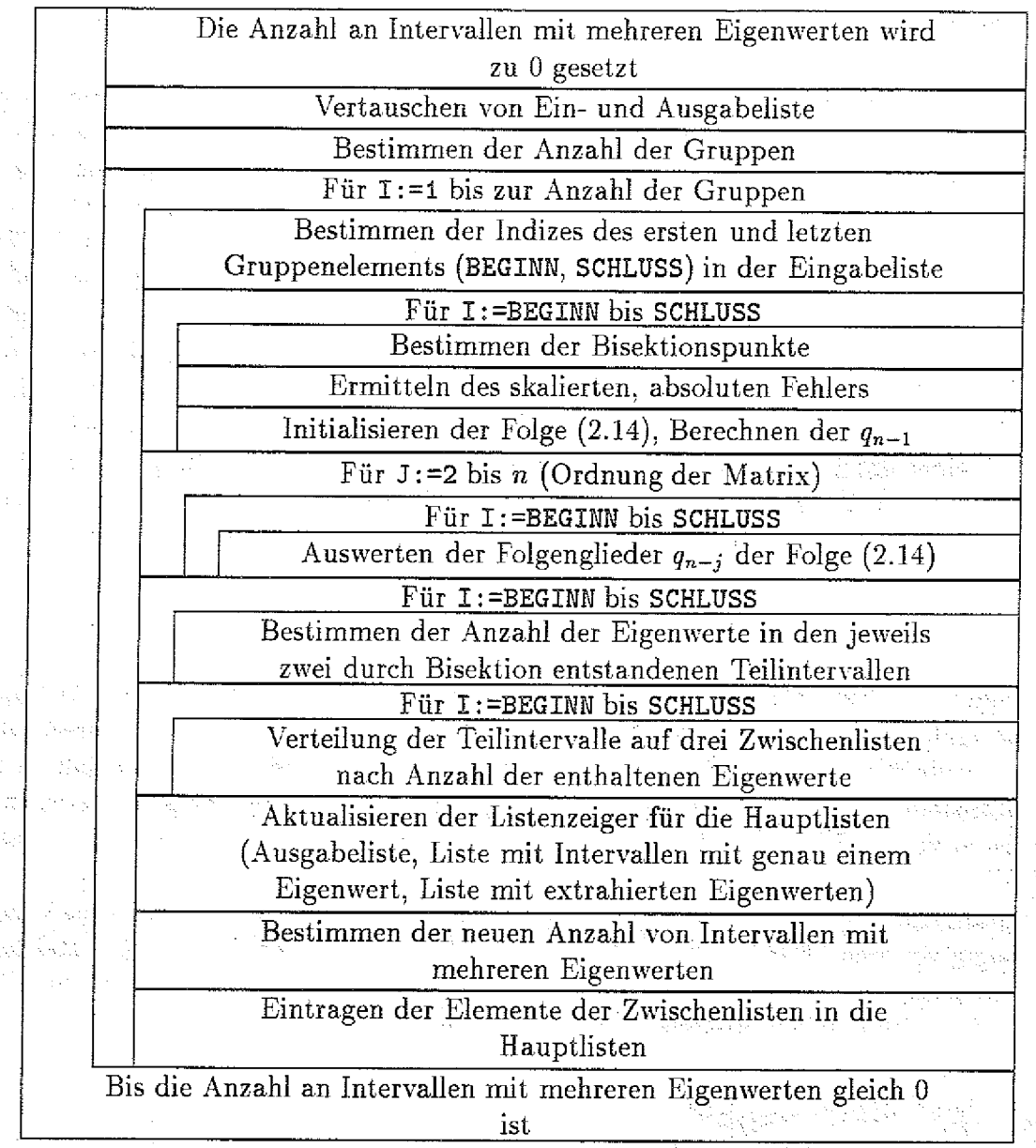


Abb. 5.7: **Struktur der Isolationsphase bei Bisektion mit Partitionierung**

Dies geschieht auf die gleiche Weise wie bei der in [24] gewählten Partitionierung. Dabei entstehen möglichst gleich große Gruppen. Die Gruppengröße wird durch Division der Anzahl der Intervalle mit mehreren Eigenwerten durch die Anzahl der Gruppen ermittelt. Verbleibt ein Rest, so erhalten die ersten Gruppen ein zusätzliches Element, bis alle Intervalle mit mehreren Eigenwerten aufgeteilt sind.
Alle folgenden Operationen werden für alle Gruppenelemente durchgeführt (Schleifen von `BEGINN` bis `SCHLUSS`). Zunächst werden die Bisektionspunkte und die skalierten, absoluten Fehler bestimmt sowie die Folge (2.14) initialisiert. Anschließend wird die Folge (2.14) ausgewertet und danach jeweils die Anzahl der Eigenwerte in den durch Bisektion entstandenen Teilintervallen bestimmt. Als nächstes werden die Teilintervalle auf die Zwischenlisten je nach Anzahl der enthaltenen Eigenwerte verteilt. Dies geschieht wie in der ersten Variante der Isolationsphase (s. auch Abbildung 5.6). In dieser Variante entspricht die Dimension der Zwischenlisten der Ordnung der größten Matrix, die bearbeitet werden kann. Diese Dimensionierung ist notwendig, da, wenn nur eine Gruppe gebildet wird, die Möglichkeit besteht, daß alle Eigenwerte in einem Schritt isoliert werden. Danach werden die Listenzeiger der Hauptlisten (Ausgabeliste, Liste mit Intervallen, die genau einen Eigenwert enthalten, Liste mit bereits extrahierten Eigenwerten) aktualisiert und die neue Anzahl an Intervallen mit mehreren Eigenwerten bestimmt. Schließlich werden die Elemente der Zwischenlisten in die Hauptlisten eingetragen. Die Isolationsphase endet, wenn keine Intervalle mit mehreren Eigenwerten mehr vorhanden sind.

Vektorisierung und Parallelisierung Alle inneren Schleifen, die über alle Gruppenelemente laufen, vektorisieren (s. Abbildung 5.7). Wenn nach einigen Isolationsschritten viele Intervalle mit mehreren Eigenwerten vorliegen, werden große Gruppen gebildet, und die Vektorisierung ist sehr lohnend. Das Eintragen der Elemente der Zwischenlisten in die Hauptlisten vektorisiert ebenfalls. Die Anzahl der Elemente der Zwischenlisten bestimmt jeweils die Vektorlänge.
Die Zählschleife, die über alle Gruppen läuft, kann parallelisiert werden, da jede Gruppe unabhängig von den übrigen bearbeitet werden kann. Das Aktualisieren der Listenzeiger für die Hauptlisten sowie das Bestimmen der neuen Anzahl von Intervallen mit mehreren Eigenwerten muß in einem kritischen Bereich geschehen.

5.3.4 Extraktionsphase

In der Extraktionsphase liegt eine Liste mit Intervallen vor, die genau einen Eigenwert enthalten. Ziel ist es, alle Eigenwerte in vorgegebener Genauigkeit zu berechnen.
In den Algorithmen für die Extraktionsphase treten drei Schleifen auf, deren Schachtelung für die Vektorisierung und Parallelisierung besondere Bedeutung zukommt. Hier ist zunächst die Schleife über alle Intervalle mit genau einem Eigenwert zu nennen. Eine

andere wichtige Schleife ist die Schleife über alle Iterationsschritte des gewählten Iterationsverfahrens. Als dritte relevante Schleife kommt die Schleife für die Auswertung der Sturmschen Kette bzw. der Folge (2.14) hinzu.
Als Iterationsverfahren zur Extraktion der Eigenwerte aus den vorgegebenen Intervallen werden Multisektion (s. auch Algorithmus 5.2), Bisektion mit Verwendung von Algorithmus 5.1(in allen Intervallen, die genau einen Eigenwert enthalten, wird durch Bestimmen der jeweiligen Anzahl der Eigenwerte, die kleiner als die Bisektionspunkte sind und durch Vergleich mit der Anzahl der Eigenwerte, die kleiner als die untere Intervallgrenze sind oder mit der Anzahl der Eigenwerte, die kleiner als die obere Intervallgrenze sind, ermittelt, in welchem der jeweils zwei durch Bisektion entstandenen Teilintervalle sich genau ein Eigenwert befindet), Bisektion mit Auswertung des charakteristischen Polynoms („normale Bisektion") und das Pegasusverfahren (s. 2.6) eingesetzt.
Je nach Schachtelung der oben genannten Schleifen, dem eingesetzten Iterationsverfahren und der Verwendung der Sturmschen Kette bzw. der Folge (2.14) ergeben sich die im folgenden erläuterten Varianten der Extraktionsphase.

Variante 1

Variante 1 der Extraktionsphase enthält drei Zählschleifen über die Anzahl der Intervalle mit genau einem Eigenwert (s. Abbildung 5.8). Die zweite Zählschleife bildet den Hauptteil von Variante 1, die Berechnungen der ersten und dritten Zählschleife werden vor bzw. nach dem Hauptteil durchgeführt, da sie nur dann vektorisiert werden können. Verzichtet man auf die Vektorisierung dieser Berechnungen, so können sie im Hauptteil vor bzw. nach der inneren Schleife eingefügt werden. In Variante 1 wird nur die Folge (2.14) in den Iterationsschritten verwendet. Folgende Iterationsverfahren werden untersucht:

a) Multisektion. Hier ist zu testen, ob Multisektion in der Extraktionsphase Vorteile gegenüber Bisektion besitzt und welche Zahl an inneren Intervallteilungspunkten optimal ist.

b) Bisektion mit Verwendung von Algorithmus 5.1.

c) Bisektion mit Auswertung des charakteristischen Polynoms. Hierzu müssen zu Beginn der Extraktionsphase die Funktionswerte des charakteristischen Polynoms an den Intervallgrenzen aller Intervalle mit genau einem Eigenwert bestimmt werden, da dies in der Isolationsphase nicht erfolgt. Der Funktionswert des charakteristischen Polynoms wird durch Multiplizieren aller q_{n-i} (2.14) berechnet (s. auch 2.5).

d) Pegasusverfahren (s. 2.6). Zu Beginn der Extraktionsphase müssen ebenfalls die Funktionswerte des charakteristischen Polynoms an den Intervallgrenzen aller Intervalle mit genau einem Eigenwert bestimmt werden.

Abbildung 5.8 zeigt die Struktur der ersten Variante der Extraktionsphase.

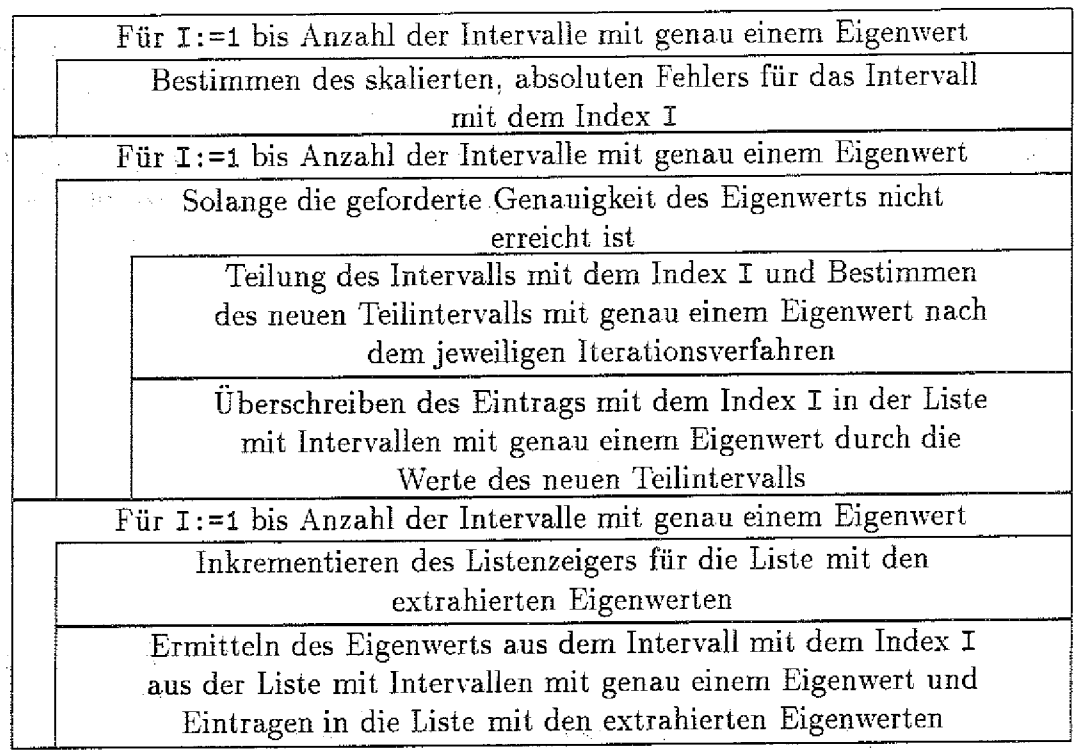


Abb. 5.8: Extraktionsphase: Variante 1

Zunächst werden die skalierten, absoluten Fehler Δ_{skal} für alle Intervalle mit genau einem Eigenwert in einer Schleife bestimmt. Bei Bisektion und Multisektion geschieht dies nach folgender Gleichung:

$$\Delta_{skal} = 2 \cdot \delta_f \cdot (|a| + |b|) \tag{5.8}$$

a und b sind die Intervallgrenzen der Intervalle mit genau einem Eigenwert zu Beginn der Extraktionsphase. In (5.7) stellen a und b im Gegensatz dazu entweder zwei Punkte aus der Mitte des aktuellen Intervalls (Multisektion) oder die Intervallgrenzen des aktuellen Intervalls (Bisektion) dar. δ_f ist die in (5.6) und (5.7) verwendete Größe. Beim Pegasusverfahren fällt der Faktor 2 in (5.8) weg, da hier der letzte Iterationspunkt und nicht wie bei Bisektion und Multisektion der Mittelwert des letzten Intervalls nach Erreichen der geforderten Genauigkeit als Eigenwert gewählt wird.
Im Hauptteil von Variante 1 der Extraktionsphase läuft die äußere Schleife über alle Intervalle mit genau einem Eigenwert. In der inneren Schleife werden alle Iterationsschritte bis zur Extraktion des Eigenwerts durchgeführt. Bei den Verfahren **a**, **b** und **c** ist die

Extraktion des Eigenwerts beendet, sobald die aktuelle Intervallbreite kleiner oder gleich dem skalierten, absoluten Fehler ist. Beim Pegasusverfahren (Verfahren **d**) bricht die innere Schleife ab, wenn der Betrag des aktuellen Iterationspunktes sich von dem Betrag des vorherigen höchstens um den skalierten, absoluten Fehler unterscheidet.
In der inneren Schleife erfolgt zunächst die Teilung des aktuellen Intervalls und das Bestimmen des neuen Teilintervalls, das den Eigenwert enthält, nach einem der oben genannten Iterationsverfahren. Die Intervallgrenzen des neuen Teilintervalls sowie die Anzahl der Nullstellen kleiner als die Intervallgrenzen (Verfahren **a** und **b**) bzw. die Funktionswerte des charakteristischen Polynoms an den Intervallgrenzen (Verfahren **c** und **d**) ersetzen dann die entsprechenden Werte des aktuellen Intervalls.
Nach Beendigung des Hauptteils der Extraktionsphase werden alle Eigenwerte aus der Intervalliste, die alle Intervalle mit genau einem Eigenwert enthält, bestimmt und in die Liste mit den extrahierten Eigenwerten eingetragen. Bei den Iterationsverfahren **a**, **b** und **c** werden die Mittelpunkte der Intervalle mit genau einem Eigenwert in diese Liste übernommen, bei Verfahren **d** der letzte Iterationspunkt, der immer auf dem Speicherplatz der oberen Intervallgrenze abgespeichert ist.

Vektorisierung und Parallelisierung In Variante 1 der Extraktionsphase sind zunächst das Bestimmen des skalierten, absoluten Fehlers und der Listeneintrag in die Liste mit den extrahierten Eigenwerten vektorisierbar (s. die Schleifen in Abbildung 5.8). Die Größenordnung der Vektorlängen entspricht hier gewöhnlich der Ordnung der Matrix. Ansonsten läßt sich lediglich die Auswertung der Folge (2.14) bei Multisektion gemäß Algorithmus 5.2 vektorisieren. Die Vektorlänge entspricht der Anzahl der inneren Teilungspunkte und besitzt die Größenordnung 10.
Zur Parallelisierung bietet sich die Schleife über alle Intervalle mit genau einem Eigenwert an (s. Abbildung 5.8), da jedes Intervall unabhängig von allen anderen behandelt werden kann. Ferner stellt diese Schleife den bei weitem zeitintensivsten Teil der Extraktionsphase dar. Bei den ersten beiden oben genannten vektorisierbaren Schleifen ist zusätzlich Parallelisierung möglich, jedoch ist dies erst bei sehr großen Vektorlängen lohnend. Außerdem ist der zeitliche Anteil dieser Schleifen an der Extraktionsphase gering.

Variante 2

Variante 2 der Extraktionsphase besitzt dieselbe Struktur wie Variante 1 (s. Abbildung 5.8). In den Iterationsschritten wird jedoch statt der Folge 2.14 die Sturmsche Kette verwendet. Folgende Iterationsverfahren werden getestet:

a) Bisektion mit Bestimmen der Anzahl der Eigenwerte in einem der beiden durch den Bisektionspunkt entstandenen Teilintervalle nach Satz 2.1.

b) Bisektion mit Auswertung des charakteristischen Polynoms.

c) Pegasusverfahren.

Auf die Untersuchung von Multisektion wird in Variante 2 verzichtet, da bei Vektorisierung der Sturmschen Kette die Vorteile von Algorithmus 5.2 wegfallen.

Vektorisierung und Parallelisierung Die Sturmsche Kette läßt sich als lineare Rekursion zweiter Ordnung mit Hilfe der CRAY-Routinen SOLR (Verfahren **a**) und SOLRN (Verfahren **b** und **c**) vektorisieren. Die Größenordnung der Vektorlängen entspricht gewöhnlich der Ordnung der Matrix. SORLN liefert nur das Endergebnis einer linearen Rekursion zweiter Ordnung, während SOLR auch alle Zwischenergebnisse zurückgibt, so daß sich die Zahl der Vorzeichenwechsel bestimmen läßt (s. Satz 2.1). Ansonsten gilt für die Vektorisierung und Parallelisierung von Variante 2 das gleiche wie für Variante 1.

Variante 3

In Variante 3 der Extraktionsphase bildet die Schleife über alle Iterationsschritte die äußere Schleife im Hauptteil (s. Abbildung 5.9, Mitte).
Innerste Schleife ist immer die Schleife über alle Intervalle, die genau einen Eigenwert enthalten. Das verwendete Iterationsverfahren wird „parallel" in allen Intervallen mit genau einem Eigenwert durchgeführt. In Variante 3 wird die Folge (2.14) in den Iterationsschritten auswertet. Folgende Iterationsverfahren werden untersucht:

a) Bisektion mit Verwendung von Algorithmus 5.1.

b) Bisektion mit Auswertung des charakteristischen Polynoms.

c) Pegasusverfahren (s. 2.6).

Bei den Verfahren **b** und **c** müssen zu Beginn der Extraktionsphase die Funktionswerte des charakteristischen Polynoms an den Intervallgrenzen aller Intervalle mit genau einem Eigenwert bestimmt werden, da dies in der Isolationsphase nicht erfolgt. Der Funktionswert des charakteristischen Polynoms wird durch Multiplizieren aller q_{n-i} der Folge (2.14) berechnet (s. auch 2.5). Die Struktur von Variante 3 der Extraktionsphase zeigt Abbildung 5.9.
Zunächst werden die skalierten, absoluten Fehler für alle Intervalle mit genau einem Eigenwert wie bei Variante 1 nach (5.8) bestimmt.
Der folgende Hauptteil der Variante ist durch die Schleife über alle Iterationsschritte eingefaßt.

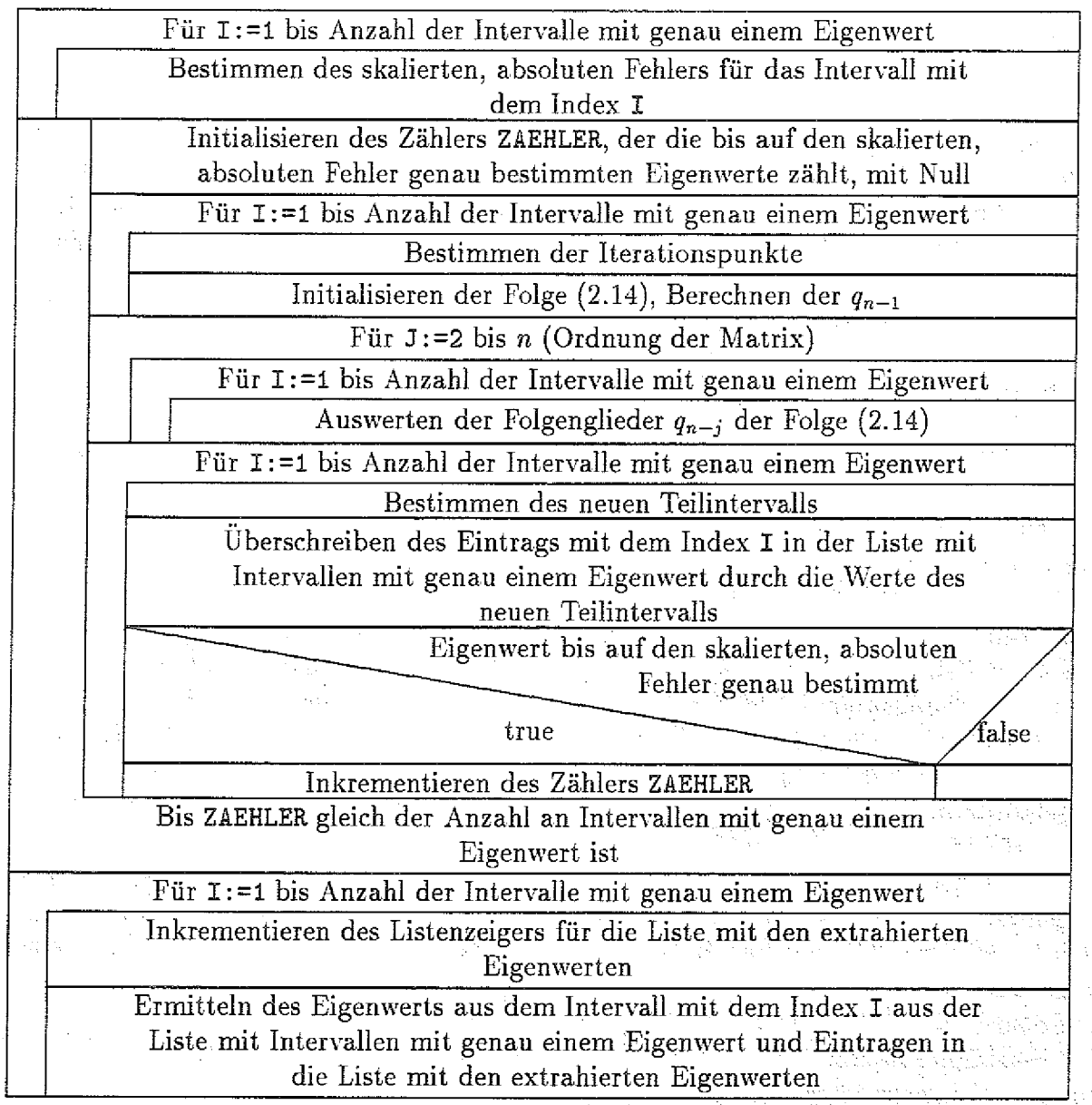


Abb. 5.9: **Extraktionsphase: Variante 3**

Innerhalb dieser äußeren Schleife wird zunächst der Zähler, der die bereits extrahierten, d.h. bis auf den skalierten, absoluten Fehler genau bestimmten Eigenwerte zählt, zu Null gesetzt. Danach werden die Iterationspunkte für alle Intervalle mit genau einem Eigenwert ermittelt und die Folge (2.14) initialisiert. Hierzu werden die q_{n-1} berechnet und die jeweilige Anzahl der Eigenwerte, die kleiner als die Iterationspunkte sind, (Verfahren **a**) oder die Funktionswerte an den Iterationspunkten (Verfahren **b** und **c**) vorbesetzt.
Als nächstes wird jedes Glied der Folge (2.14) an jedem Iterationspunkt bestimmt. Dies wird durch eine Schleife über alle Folgenglieder (äußere Schleife, von 2 bis n) und einer Schleife über alle Intervalle mit genau einem Eigenwert (innere Schleife) realisiert. In der inneren Schleife wird das Folgenglied q_{n-j} (s. Abbildung 5.9) an allen Iterationspunkten bestimmt. Die Werte der q_{n-j} werden bei Verfahren **a** zur Bestimmung der jeweiligen Anzahl der Eigenwerte, die kleiner als die Iterationspunkte sind, verwendet, bei den Verfahren **b** und **c** zur Funktionsauswertung des charakteristischen Polynoms.
Nach Auswertung aller Folgenglieder an allen Iterationspunkten werden in einer Schleife über alle aktuellen Intervalle mit genau einem Eigenwert alle neuen Teilintervalle, die genau einen Eigenwert enthalten, ermittelt. Die neuen Teilintervalle ersetzen dann die entsprechenden aktuellen Intervalle in der Liste, die alle Intervalle mit genau einem Eigenwert enthält. Zusätzlich wird in dieser Schleife geprüft, wieviele Eigenwerte bereits bis auf den skalierten, absoluten Fehler genau bestimmt sind. Die Anzahl dieser Eigenwerte wird in dem Zähler `ZAEHLER` (s. Abbildung 5.9) festgehalten. Bei den Verfahren **a** und **b** wird überprüft, ob die Intervallbreiten der neuen Teilintervalle kleiner oder gleich dem skalierten, absoluten Fehler sind, bei Verfahren **c**, ob sich die Beträge der aktuellen Iterationspunkte und der vorherigen um höchstens den skalierten, absoluten Fehler unterscheiden.
Die Schleife über alle Iterationsschritte (äußere Schleife des Hauptteils, s. Abbildung 5.9) ist beendet, wenn die Anzahl der bis auf den skalierten, absoluten Fehler genau bestimmten Eigenwerte mit der Anzahl der Intervalle mit genau einem Eigenwert übereinstimmt.
Nach Beendigung des Hauptteils von Variante 3 der Extraktionsphase werden wie in den Varianten 1 und 2 alle Eigenwerte aus der Intervalliste mit Intervallen mit genau einem Eigenwert bestimmt und in die Liste mit den extrahierten Eigenwerten eingetragen. Bei den Iterationsverfahren **a** und **b** werden die Mittelpunkte der Intervalle mit genau einem Eigenwert in diese Liste übernommen, bei Verfahren **c** der letzte Iterationspunkt, der immer auf dem Speicherplatz der oberen Intervallgrenze abgespeichert ist.
Bei Bisektion (Verfahren **a** und **b**) läßt sich die Anzahl der Iterationsschritte für ein Intervall $[a, b]$ explizit nach folgender Gleichung bestimmen:

$$\text{Anzahl der Iterationsschritte} = \lceil |\log_2(\frac{|b-a|}{\Delta_{skal}})| \rceil. \tag{5.9}$$

Δ_{skal} ist der skalierte, absolute Fehler für das Intervall $[a, b]$. Zu Beginn der Extraktionsphase ist es bei Bisektion möglich, die Anzahl der Iterationsschritte für jedes Intervall

mit genau einem Eigenwert zu bestimmen. Das Maximum dieser Werte gibt an, wieviele Iterationsschritte durchgeführt werden müssen, um alle Eigenwerte bis auf den skalierten, absoluten Fehler genau zu berechnen. Bei den Verfahren **a** und **b** ist die äußere Schleife im Hauptteil von Variante 3 (s. Abbildung 5.9) auch als Zählschleife von 1 bis zur maximalen Anzahl der Iterationsschritte realisiert.
Variante 3 hat den Nachteil, daß für viele Intervalle mit genau einem Eigenwert zu viele Iterationsschritte durchgeführt werden. In der Isolationsphase werden Intervalle mit genau einem Eigenwerte zu sehr unterschiedlichen Zeitpunkten gefunden (s. auch Abbildung 5.3), so daß Intervalle sehr unterschiedlicher Breite an die Extraktionsphase übergeben werden. In den Varianten 5 und 6 wird versucht, diesen Nachteil zu umgehen. Ferner ist zu untersuchen, ob sich der genannte Nachteil überhaupt stark auf das Zeitverhalten der Extraktionsphase auswirkt.

Vektorisierung und Parallelisierung Variante 3 der Extraktionsphase bietet hervorragende Möglichkeiten zur Vektorisierung, da alle inneren Schleifen über alle Intervalle mit genau einem Eigenwert laufen. Die Größenordnung der Vektorlängen entspricht dabei gewöhnlich der Ordnung der Matrix. Durch die gute Vektorisierbarkeit von Variante 3 ist ein wesentlich günstigeres Zeitverhalten als bei den Varianten 1 und 2 bei sequentieller Programmausführung zu erwarten.
Parallelisierung ist zusätzlich zur Vektorisierung bei allen inneren Schleifen möglich. Dies ist allerdings erst bei Matrizen hoher Ordnung lohnend. Ziel der Parallelisierung muß hier sein, den Anteil des Overheads durch die Parallelisierung möglichst gering zu halten. In allen inneren Schleifen wird auf voneinander unabhängigen Intervallen gearbeitet, nur das Inkrementieren des Zählers für die bereits bis auf den skalierten, absoluten Fehler genau bestimmten Eigenwerte (s. Abbildung 5.9) muß in einem kritischen Bereich geschehen, den nur eine Task zur gleichen Zeit bearbeiten kann.

Variante 4

Variante 4 der Extraktionsphase unterscheidet sich nur darin von Variante 3, daß statt der Folge (2.14) die Sturmsche Kette ausgewertet wird. Abbildung 5.10 zeigt den Ausschnitt von Variante 4, der sich von Variante 3 (s. Abbildung 5.9) unterscheidet.
Nach Bestimmung der Iterationspunkte wird die Sturmsche Kette an allen Iterationspunkten ausgewertet. Dies geschieht mit Hilfe der CRAY-Routinen SOLR bzw. SOLRN (s. auch Variante 2), die intern alle Folgenglieder der Sturmschen Kette bestimmen.

Vektorisierung und Parallelisierung In der zweiten Schleife in Abbildung 5.10 wird die Berechnung aller Folgenglieder der Sturmschen Kette wie in Variante 2 mit Hilfe der

Für I:=1 bis Anzahl der Intervalle mit genau einem Eigenwert
Bestimmen der Iterationspunkte
Für I:=1 bis Anzahl der Intervalle mit genau einem Eigenwert
Auswerten der Sturmschen Kette an dem Iterationspunkt mit dem Index I

Abb. 5.10: **Extraktionsphase: Variante 4, Ausschnitt**

CRAY-Routinen SOLR bzw. SOLRN vektorisiert. Dies geschieht innerhalb der CRAY-Routinen. Die Schleife über alle Intervalle mit genau einem Eigenwert wird hier nicht vektorisiert. Da bei der Bestimmung der Iterationspunkte die Schleife über alle Intervalle, die genau einen Eigenwert enthalten, vektorisiert (erste Schleife in Abbildung 5.10), sind für eine günstige Vektorisierung zwei Schleifen erforderlich. Ansonsten entspricht die Vektorisierung von Variante 4 der von Variante 3.
Die zweite Schleife aus Abbildung 5.10 kann gut parallelisiert werden. Sie stellt den zeitintensivsten Teil von Variante 4 der Extraktionsphase dar, da sie die Auswertung der Sturmschen Kette enthält. Parallelisierung ist in dieser Schleife im Gegensatz zu den inneren Schleifen in Variante 3 schon bei Matrizen wesentlich kleinerer Ordnung lohnend. Sonst gilt für die Parallelisierung von Variante 4 das für Variante 3 Gesagte.

Variante 5

In Variante 5 der Extraktionsphase wird versucht, den Nachteil von Variante 3, daß für viele Intervalle mit genau einem Eigenwert zu viele Iterationsschritte durchgeführt werden, zu umgehen. Hierzu bietet sich an, die Intervalle mit genau einem Eigenwert, die ungefähr gleich viele Iterationsschritte benötigen, in Gruppen zusammenzufassen. Die Gruppen werden nach jedem Isolationsschritt der Isolationsphase gebildet, da der Algorithmus der Isolationsphase garantiert, daß die in jedem Isolationsschritt gefundenen Intervalle mit genau einem Eigenwert die gleiche Intervallbreite besitzen. Die Extraktion der Eigenwerte aus allen Intervallen einer Gruppe erfolgt dann genau wie in Variante 3 der Extraktionsphase. Zur Realisierung der Gruppenbildung und -bearbeitung sind Ergänzungen in der Initialisierungsphase, der Isolationsphase und der Extraktionsphase notwendig.
Die Ergänzungen in der Initialisierungsphase zeigt Abbildung 5.11.
Am Ende der Initialisierungsphase direkt vor der Isolationsphase werden zunächst die letzte Anzahl an Intervallen mit genau einem Eigenwert und die Anzahl der Gruppen mit Intervallen mit genau einem Eigenwert zu Null gesetzt. Ist nun die aktuelle Anzahl an Intervallen mit genau einem Eigenwert, die in der Initialisierungsphase ermittelt wurde, größer oder

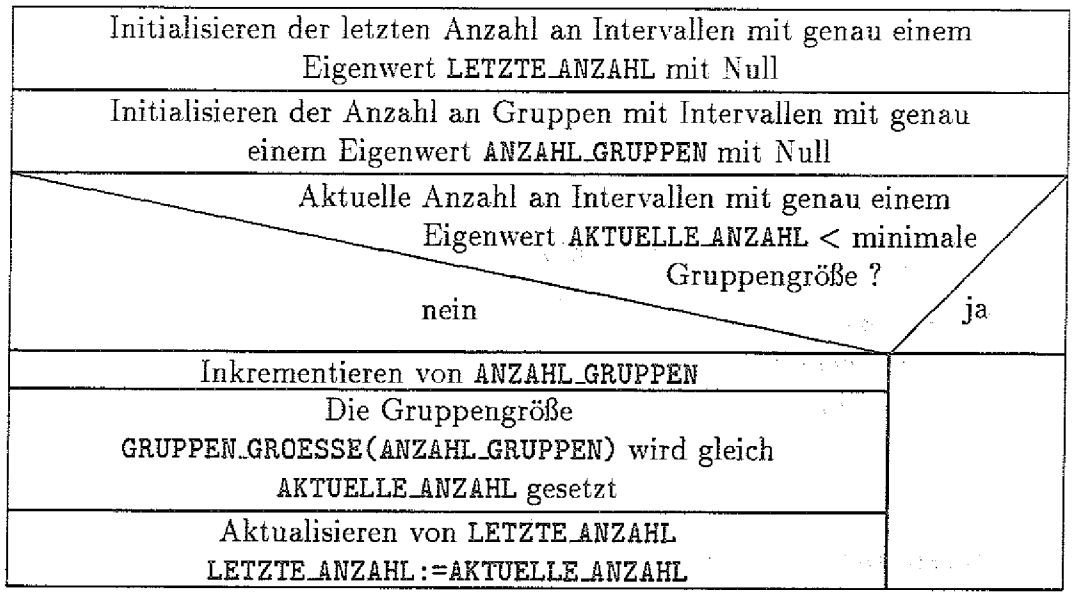


Abb. 5.11: Initialisierungsphase: Ergänzungen zur Gruppenbildung

gleich der minimalen Gruppengröße, so wird eine Gruppe gebildet. Sehr kleine Gruppen sind aus Vektorisierungsgründen ungünstig, deshalb wird als Wert für die minimale Gruppengröße im Programm gewöhnlich 64 (Anzahl der Elemente eines Vektorregisters) gewählt. Bei einer minimalen Gruppengröße von 1 werden die Gruppen so gebildet, wie zufällig Intervalle mit genau einem Eigenwert gefunden werden. Es können dann sehr kleine Gruppen entstehen, die die Vektorisierung stören. Wird eine neue Gruppe gebildet, so wird zunächst der Zähler für die Anzahl der Gruppen `ANZAHL_GRUPPEN` inkrementiert. In der Liste, die die Gruppengrößen enthält, wird das Element mit dem Index `ANZAHL_GRUPPEN` gleich der aktuellen Anzahl an Intervallen mit genau einem Eigenwert gesetzt. Schließlich wird der Wert der letzten Anzahl an Intervallen mit genau einem Eigenwert durch den Wert der aktuellen Anzahl überschrieben. Ist die Anzahl an inneren Teilungspunkten in der Initialisierungsphase kleiner als die minimale Gruppengröße minus 1, so kann die if-Abfrage in Abbildung 5.11 wegfallen, da dann direkt nach der Initialisierungsphase keine Gruppenbildung erfolgen kann. Die Liste für die Gruppengrößen der einzelnen Gruppen ist als Feld der Dimension maximal mögliche Ordnung der Matrix realisiert. Bei fest eingestellter minimaler Gruppengröße ist als Dimension die maximal mögliche Ordnung der Matrix dividiert durch die minimale Gruppengröße (ganzzahlige Division) ausreichend.

In der Isolationsphase wird nach jedem Isolationsschritt (nach Bearbeitung der Schleife von `ANFANG` bis `ENDE` in Abbildung 5.3) geprüft, ob neue Gruppen gebildet werden können.

Die dazu an dieser Stelle notwendigen Ergänzungen zeigt Abbildung 5.12.

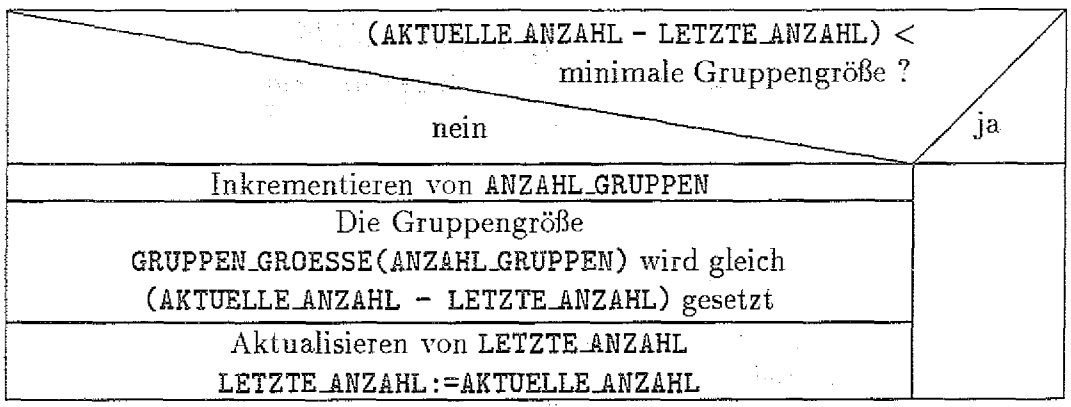


Abb. 5.12: Isolationsphase: Ergänzungen zur Gruppenbildung

Ist die Differenz aus aktueller und letzter Anzahl an Intervallen mit genau einem Eigenwert größer oder gleich der minimalen Gruppengröße, so wird zunächst der Zähler für die Anzahl der Gruppen `ANZAHL_GRUPPEN` inkrementiert. Danach wird in der Liste mit den Gruppengrößen der Wert des Listenplatzes mit dem Index `ANZAHL_GRUPPEN` auf den Wert der Differenz aus aktueller und letzter Anzahl an Intervallen mit genau einem Eigenwert gesetzt. Diese Differenz entspricht der Gruppengröße der neu gebildeten Gruppe. Schließlich wird der Wert der letzten Anzahl an Intervallen mit genau einem Eigenwert aktualisiert, indem ihm der Wert der aktuellen Anzahl zugewiesen wird.
Am Ende der Isolationsphase (s. Abbildung 5.3, nach der äußeren until-Schleife) sind zwei Sonderfälle zu behandeln. Diese sind in Abbildung 5.13 dargestellt.
Sind überhaupt keine Gruppen gebildet worden, da die minimale Gruppengröße nie erreicht wurde, so wird eine Gruppe mit der Gruppengröße der aktuellen Anzahl an Intervallen mit genau einem Eigenwert eingerichtet (erste if-Abfrage in Abbildung 5.13). Danach muß der Wert der letzten Anzahl an Intervallen mit genau einem Eigenwert auf den Wert der aktuellen gesetzt werden, damit der zweite Sonderfall nicht durchlaufen wird (s. Abbildung 5.13). Der zweite Sonderfall liegt dann vor, wenn die Anzahl der in den abschließenden Isolationsschritten gefundenen Eigenwerte kleiner als die minimale Gruppengröße war. Die Anzahl dieser Eigenwerte wird dann zur Gruppengröße der aktuellen Gruppe (Index `ANZAHL_GRUPPEN`) addiert (zweite if-Abfrage in Abbildung 5.13).
Die Extraktionsphase nach Variante 5 wird durch die Schleife über alle Gruppen eingefaßt. Die äußere Struktur der Extraktionsphase nach Variante 5 zeigt Abbildung 5.14.
Zunächst werden die Indizes für das erste und letzte Gruppenelement (`BEGINN` bzw. `SCHLUSS`) mit Null initialisiert. Die Indizes geben das erste und letzte Gruppenelement der

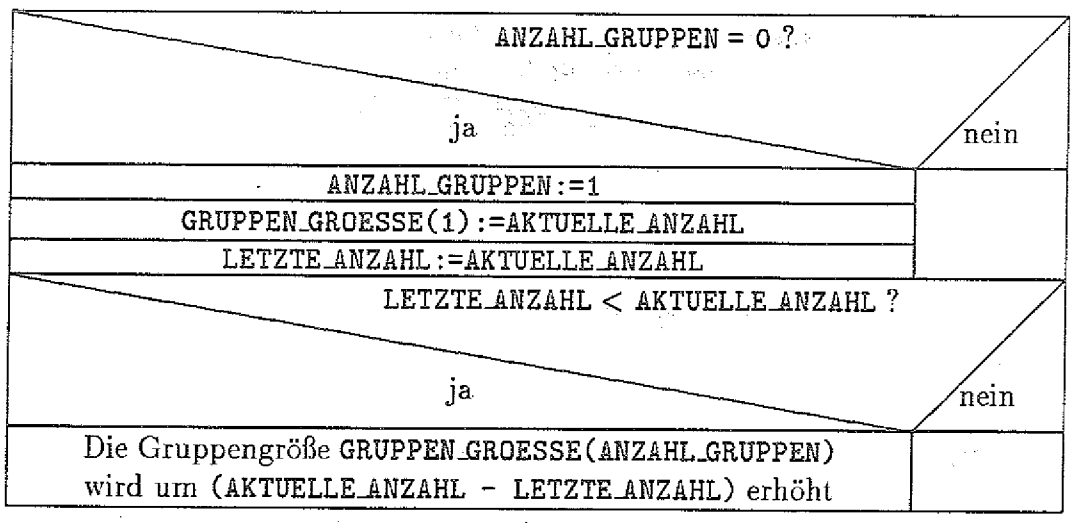


Abb. 5.13: Isolationsphase: Sonderfälle bei der Gruppenbildung

aktuellen Gruppe in der Liste mit allen Intervallen, die genau einen Eigenwert enthalten, an. Es folgt die Zählschleife zur Bearbeitung aller Gruppen (von 1 bis `ANZAHL_GRUPPEN`). In dieser Schleife werden als erstes die Indizes für das erste und letzte Gruppenelement der aktuellen Gruppe aktualisiert. `BEGINN` wird auf `SCHLUSS` plus 1 gesetzt, `SCHLUSS` wird um die Gruppengröße der aktuellen Gruppe erhöht. Danach werden die Eigenwerte aus allen Intervallen der aktuellen Gruppe extrahiert. Hierbei wird genau wie in Variante 3 (s. Abbildung 5.9) verfahren, jedoch laufen alle inneren Schleifen statt über alle Intervalle mit genau einem Eigenwert über alle Intervalle der aktuellen Gruppe (Indizes `BEGINN` bis `SCHLUSS`). Alle inneren Schleifen zur Extraktion der Eigenwerte in Variante 5 enthalten die gleichen Anweisungen wie die entsprechenden Schleifen in Variante 3 (s. Abbildung 5.9). Um nicht alle Anweisungen wiederholen zu müssen, ist dieser Teil in Abbildung 5.14 zu einer Blockanweisung zusammengefaßt.

Vektorisierung und Parallelisierung Gut vektorisierbar sind in Variante 5 wie in Variante 3 der Extraktionsphase die inneren Schleifen, die in Variante 5 über alle Intervalle einer Gruppe laufen. Die Vektorlängen ergeben sich in der Isolationsphase und sind nicht im voraus abzuschätzen, betragen aber mindestens die vorgegebene minimale Gruppengröße. In Variante 5 sind die Vektorlängen gewöhnlich kleiner als in Variante 3, jedoch werden meist weniger Iterationsschritte in der Extraktionsphase benötigt.
Eine Möglichkeit zur Parallelisierung bieten wie in Variante 3 die inneren Schleifen. Lohnend ist dies erst bei Matrizen hoher Ordnung, wenn hohe Gruppengrößen auftreten. In

Initialisieren der Indizes für das erste und letzte Gruppenelement (BEGINN bzw. SCHLUSS) mit Null
Für K:=1 bis ANZAHL_GRUPPEN
BEGINN:=SCHLUSS+1
SCHLUSS:=SCHLUSS+GRUPPEN_GROESSE(K)
Extraktion der Eigenwerte aus allen Intervallen der aktuellen Gruppe; die Intervalle der aktuellen Gruppe besitzen in der Liste mit Intervallen mit genau einem Eigenwert die Indizes BEGINN bis SCHLUSS

Abb. 5.14: Extraktionsphase, Variante 5: äußere Struktur

Variante 5 ist es noch schwieriger als in Variante 3, bei Parallelisierung und Vektorisierung der inneren Schleifen den Anteil des Overheads durch die Parallelisierung gering zu halten. Eine andere Möglichkeit ist die Parallelisierung der äußeren Schleife über alle Gruppen, da alle Gruppen voneinander unabhängig sind. Allerdings sind im voraus keine Aussagen über Anzahl und Größe der Gruppen zu machen. Bei wenigen Gruppen sehr unterschiedlicher Größe ist eine gleichmäßige Lastverteilung nicht zu erreichen. Liegen dagegen sehr viele Gruppen gleicher Größe vor, so ist nur noch die Anzahl an Iterationsschritten in den einzelnen Gruppen unterschiedlich. Bei einer großen Anzahl an Gruppen kann dann durchaus eine gleichmäßige Lastverteilung erhalten werden. In Variante 6 wird der Ansatz, viele Gruppen gleicher Größe zu bilden, aufgegriffen.

Variante 6

In Variante 6 der Extraktionsphase wird die Gruppengröße für die Gruppen mit Intervallen mit genau einem Eigenwert fest vorgegeben. Sind die Gruppengrößen nicht zu groß gewählt, werden für die meisten Intervalle einer Gruppe wie in Variante 5 ungefähr gleich viele Iterationsschritte benötigt. Als Gruppengrößen werden gewöhnlich Vielfache der Anzahl an Elementen eines Vektorregisters festgelegt, meist 64 oder 128. Abbildung 5.15 zeigt die äußere Struktur der Extraktionsphase nach Variante 6. Ergänzungen in der Initialisierungs- und Isolationsphase sind im Gegensatz zu Variante 5 nicht notwendig.
Zunächst wird die Anzahl der Gruppen durch ganzzahlige Division der Anzahl der Intervalle mit genau einem Eigenwert durch die vorgegebene Gruppengröße bestimmt. Die Extraktionsphase wird dann wie in Variante 5 durch die Schleife über alle Gruppen eingefaßt. In dieser Schleife werden zunächst die Indizes für die Intervalle der aktuellen Gruppe in der Liste mit Intervallen mit genau einem Eigenwert bestimmt (BEGINN bis SCHLUSS). SCHLUSS

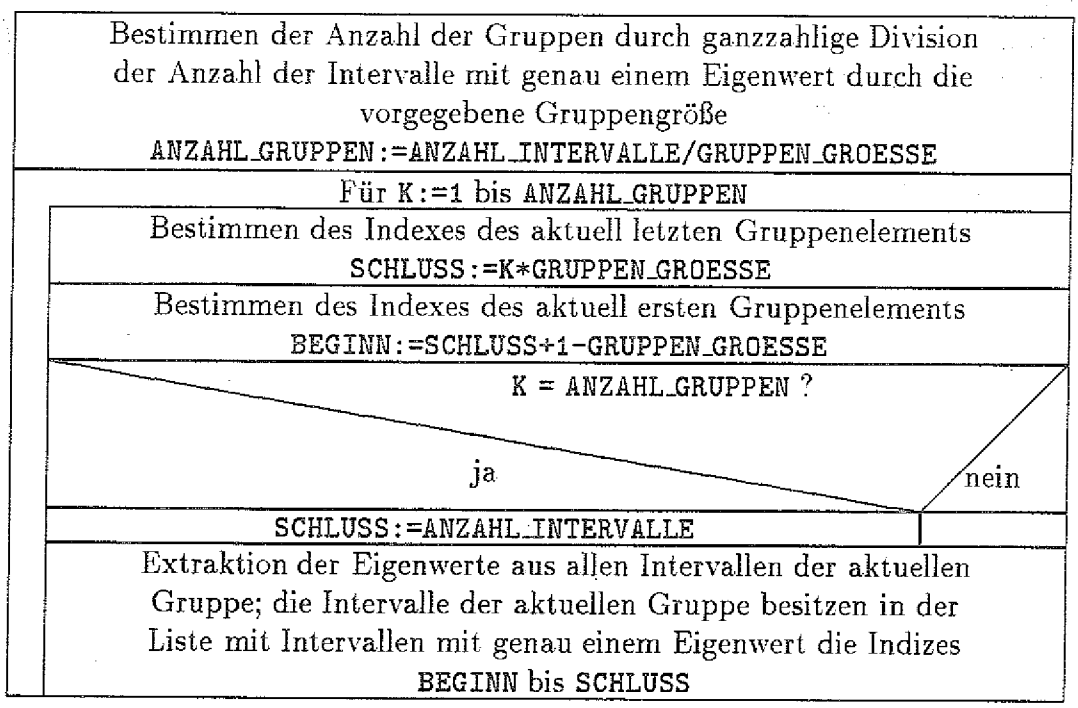


Abb. 5.15: Extraktionsphase, Variante 6: äußere Struktur

wird als Produkt aus aktuellem Gruppenindex K und der Gruppengröße ermittelt. BEGINN ergibt sich aus SCHLUSS plus 1 abzüglich der Gruppengröße. Danach wird abgefragt, ob die aktuelle Gruppe die letzte ist. Ist dies der Fall, wird SCHLUSS gleich der Anzahl an Intervallen mit genau einem Eigenwert gesetzt. Ein bei der Bestimmung der Gruppenanzahl übrig gebliebener Rest wird so berücksichtigt. Es ist sinnvoll, der letzten Gruppe die restlichen Intervalle anzuhängen, da die Intervalle der letzten Gruppe gewöhnlich die wenigsten Iterationsschritte benötigen und damit die Bearbeitung der letzten Gruppe am wenigsten zeitintensiv ist. Zuletzt werden die Eigenwerte aus allen Intervallen der aktuellen Gruppe extrahiert. Hierbei wird genau wie in Variante 5 vorgegangen.

Vektorisierung und Parallelisierung In Variante 6 sind die inneren Schleifen über alle Intervalle einer Gruppe gut vektorisierbar. Die Vektorlängen in Variante 6 entsprechen der vorgegebenen Gruppengröße.
Die äußere Schleife über alle Gruppen bietet sich zur Parallelisierung an (s. Abbildung 5.15). Parallelisierung dieser Schleife ist möglich, sobald die Anzahl an Intervallen mit genau einem Eigenwert größer oder gleich der doppelten Gruppengröße ist. Es entstehen

dann mindestens zwei Gruppen, die parallel bearbeitet werden können. Bei einer Matrix hoher Ordnung ergeben sich bei einer günstig gewählten Gruppengröße sehr viele Gruppen gleicher Größe, so daß trotz der unterschiedlichen Anzahl an Iterationsschritten der einzelnen Gruppen eine günstige Lastverteilung zu erreichen ist. Der Einfluß der Gruppengröße auf die Vektorisierung muß bei der Wahl der Gruppengröße mit berücksichtigt werden, da kleine Gruppengrößen die Vektorisierung stark verschlechtern.

6 Ergebnisse

Dieses Kapitel beschreibt die Ergebnisse der Zeit- und Speedup-Messungen der einzelnen Programmversionen. Zunächst wird auf die Eigenschaften der für die Messungen verwendeten Matrizen eingegangen, dann folgen Beschreibung und Diskussion der Messungen in Initialisierungsphase, Isolationsphase und Extraktionsphase. Der Beschreibung der Ergebnisse in jeder Phase ist jeweils ein kurzer Abschnitt zur Implementierung vorangestellt. Die Abschnitte über Isolations- und Extraktionsphase werden durch eine kurze Zusammenfassung abgeschlossen. Anschließend werden die Ausführungszeiten der schnellsten Programmversion mit denen von Bibliotheksprogrammen verglichen. Den Abschluß des Kapitels bildet die Beschreibung von Sonderfällen, in denen die Leistungsfähigkeit der Programme beeinträchtigt ist.
Alle Messungen wurden auf dedizierter Maschine durchgeführt. Dabei wurde die jeweils kürzeste von mehreren Zeitmessungen verwendet. Bei allen Messungen war δ_f (s. (5.6), (5.7) und (5.8)) zu 10^{-14} gesetzt. Im folgenden bedeutet sequentielle Programmausführung, daß die, wenn möglich, vektorisierte, aber nicht parallelisierte Version der jeweiligen Programme ausgeführt wird.

6.1 Verwendete Matrizen

Zwei der für die Messungen verwendeten Matrizen sind [14] entnommen. Die erste n-reihige Matrix der Gestalt

$$An = \begin{pmatrix} 1 & 2 & & 0 \\ 2 & \ddots & \ddots & \\ & \ddots & \ddots & 2 \\ 0 & & 2 & 1 \end{pmatrix} \tag{6.1}$$

wurde auch bei den Messungen auf einer CRAY X-MP/48 in [22] verwendet. Die Eigenwerte dieser Matrix lassen sich nach folgender Formel berechnen:

$$\lambda_i = 1 + 4\cos\frac{i\pi}{n+1}, \quad i = 1, 2, \ldots, n. \tag{6.2}$$

n ist die Ordnung der Matrix. Wie sich (6.2) entnehmen läßt, liegen für große n sehr viele Eigenwerte in der Nähe von 5 und -3.

Die zweite Matrix lautet

$$Bn = \begin{pmatrix} 0 & x_1 & & 0 \\ y_1 & \ddots & \ddots & \\ & \ddots & \ddots & x_{n-1} \\ 0 & & y_{n-1} & 0 \end{pmatrix} \tag{6.3}$$

mit

$$\begin{aligned} x_i &= i, \quad i = 1, 2, \ldots, n-1 \\ y_i &= n-i, \quad i = 1, 2, \ldots, n-1. \end{aligned} \tag{6.4}$$

n ist die Ordnung der Matrix. Ist n gerade, so besitzt die Matrix die Eigenwerte

$$\pm(n-1),\ \pm(n-3),\ \ldots,\ \pm 1. \tag{6.5}$$

Die Matrizen Bn sind unsymmetrische Tridiagonalmatrizen. Sie lassen sich jedoch, da $x_i y_i > 0$ gilt, in symmetrische Tridiagonalmatrizen umwandeln (s. 2.3.2).

Außer den oben genannten Matrizen wurden ferner Matrizen verwendet, bei denen die Elemente der Diagonalen und der Nebendiagonalen mit einem Zufallszahlengenerator erzeugt wurden. Benutzt wurde der CRAY-Zufallszahlengenerator RANF, der gleichverteilte Zufallszahlen zwischen 0 und 1 ermittelt, mit unterschiedlichen Startwerten für jede erzeugte Matrix. Diese Matrizen werden im folgenden mit Zn bezeichnet, wobei n die Ordnung der Matrix angibt.

Für die Messungen wurden Matrizen der Ordnung 128, 512, 800, 1000, 3200, 6400 und 10000 verwendet. Alle Ordnungen sind durch 8 teilbar. Bei manchen der untersuchten Programmversionen ist dies zur gleichmäßigen Auslastung aller 8 Prozessoren der CRAY Y-MP von Vorteil, jedoch bei keiner Version notwendig.

Einige Eigenwerte der Matrizen Zn und An liegen sehr nah beieinander. Diese Matrizen bieten sich an, um die Leistungfähigkeit des Isolationsverfahrens zu testen. Für die Messungen wurden daher hauptsächlich die Matrizen Zn und An verwendet. Die Eigenwerte der Matrizen Bn dagegen sind sehr günstig verteilt (Abstand der Eigenwerte immer 2, s. (6.5)). Allerdings führt die Berechnung des Funktionswertes des charakteristischen Polynoms bei diesen Matrizen leicht zu Zahlenbereichsüberschreitungen. Auf die Berechnung der Eigenwerte dieser Matrizen mit hoher Ordnung wird in 6.6 eingegangen.

6.2 Initialisierungsphase

6.2.1 Implementierung

Zur Unterteilung des durch die Gerschgorin-Kreise erhaltenen Intervalls, das alle Eigenwerte enthält, sind drei Varianten implementiert. In der ersten wird Multisektion mit drei inneren Punkten durchgeführt. Als Multisektionspunkte werden α_1 und die zwei Nullstellen der quadratischen Gleichung $p_{n-2} = 0$ verwendet. In der zweiten Variante wird das durch die Gerschgorin-Kreise erhaltene Intervall durch Multisektion mit 15 Punkten in 16 Teilintervalle gleicher Intervallbreite unterteilt. Beide Varianten verwenden zur Bestimmung der Anzahl der Eigenwerte, die in den jeweiligen Teilintervallen enthalten sind, die Folge (2.14). In der dritten untersuchten Variante liegt wie in der zweiten äquidistante Multisektion mit 15 Punkten vor. Die Anzahl der Eigenwerte in den Teilintervallen wird hier jeweils mit Hilfe der Sturmschen Kette ermittelt. Die Auswertung der Sturmschen Kette geschieht unter Verwendung der CRAY-Routine SOLR.

6.2.2 Messungen

Da der zeitliche Aufwand der Initialisierungsphase gegenüber dem der Isolations- und Extraktionsphase gering ist, wurde auf Parallelisierung verzichtet. Abbildung 6.1 zeigt die Ausführungszeiten von sequentiellen Programmläufen der drei Varianten der Initialisierungsphase bei den Matrizen $A512$ und $Z512$. In der Legende weist p auf die Verwendung der Sturmschen Kette (Glieder der Folge: p_{n-i}) hin, q auf die Verwendung der Folge (2.14) (Glieder der Folge: q_{n-i}).
Abbildung 6.1 zeigt, daß Multisektion mit drei Punkten die kürzesten Ausführungszeiten liefert. Dies war zu erwarten, da bei drei Punkten nur ein Fünftel der Auswertungen der Folge (2.14) bzw. der Sturmschen Kette gegenüber Multisektion mit 15 Punkten durchgeführt werden. Die Zeiten bei Multisektion mit 15 Punkten sind jedoch nur um ca. ein Drittel höher als bei Multisektion mit 3 Punkten. Dies ist durch die Vektorisierung der Auswertung der Folge (2.14) nach Algorithmus 5.2 sowie durch die Vektorisierung der Auswertung der Sturmschen Kette mit Hilfe der CRAY-Routine SOLR zu erklären. Abbildung 6.1 ist auch zu entnehmen, daß bei Multisektion mit 15 Punkten die Verwendung der Folge (2.14) zeitlich günstiger ist als die Verwendung der Sturmschen Kette. Dies zeigt, daß die Vektorisierung der Auswertung der Folge (2.14) nach Algorithmus 5.2 bei Multisektion mit 15 Punkten besser ist als die der Auswertung der Sturmschen Kette mit Hilfe der CRAY-Routine SOLR. Ferner läßt sich das Zählen der negativen Vorzeichen der Glieder der Folge (2.14) effizienter codieren als das Zählen von Vorzeichenwechseln bei den Folgengliedern der Sturmschen Kette (s. auch 2.2). Bei Multisektion mit 15 Punkten ist also die Verwendung der Folge (2.14) der Verwendung der Sturmschen Kette in der

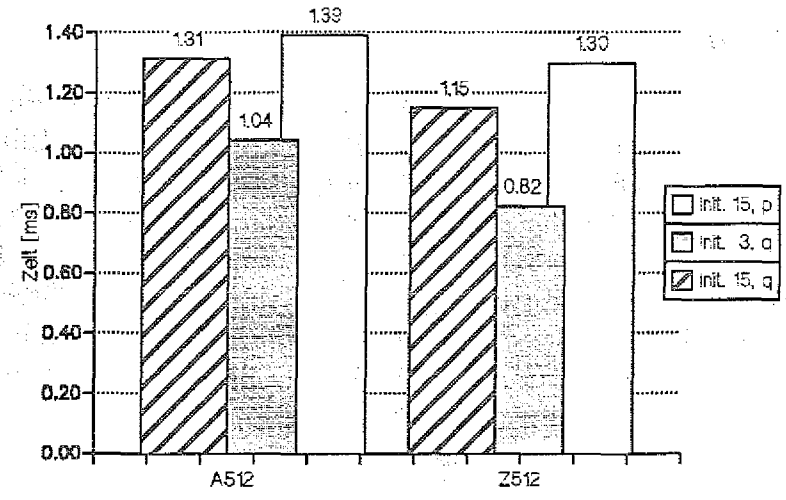


Abb. 6.1: Zeitmessungen der drei Varianten der Initialisierungsphase

Initialisierungsphase vorzuziehen.
Ob Multisektion mit drei oder 15 Punkten günstiger ist, hängt auch von der Verteilung der Eigenwerte im durch die Gerschgorin-Kreise erhaltenen Ausgangsintervall ab. Wie schon in 5.3.2 erwähnt, liefert die oben genannte Multisektion mit drei Punkten in jedem Fall mindestens zwei Intervalle mit Eigenwerten, während man bei Multisektion mit 15 Punkten in manchen Fällen nur ein Intervall mit Eigenwerten erhält. Auf Auswirkungen der Initialisierungsphase auf die Isolationsphase wird in 6.3.2 eingegangen.

6.3 Isolationsphase

6.3.1 Implementierung

In der Isolationsphase wurden verschiedene Multisektions- und Bisektionsverfahren implementiert und getestet. An Multisektionsverfahren liegen zwei Verfahren mit frei wählbarer Punktzahl unter Verwendung der Folge (2.14) bzw. der Sturmschen Kette vor. Das erste Multisektionsverfahren vektorisiert die Auswertung der Folge (2.14) gemäß Algorithmus 5.2, das zweite die Auswertung der Sturmschen Kette mit Hilfe der CRAY-Routine SOLR. Das erste implementierte Bisektionsverfahren verwendet die Folge (2.14). Da nur ein Teilungspunkt vorhanden ist, ist Vektorisierung hier nicht möglich, jedoch läßt sich Bisektion effizienter codieren als Multisektion. Im zweiten Bisektionsverfahren wird die Sturmsche

Kette mit Hilfe der CRAY-Routine SOLR vektorisiert. Ebenso wie im ersten Bisektionsverfahren sind hier Code-Einsparungen gegenüber Multisektion möglich. Auf ein weiteres Bisektionsverfahren wird weiter unten eingegangen.

In allen oben genannten Verfahren wurde jeder Isolationsschritt (s. Abbildung 5.3, Schleife von `ANFANG` bis `ENDE`) unter Verwendung der Autotasking-Direktive DO ALL parallelisiert. Wird dagegen die gesamte Isolationsphase in einen parallelen Bereich eingeschlossen und die Direktive DO ALL durch DO PARALLEL ersetzt, so wird ab dem zweiten Isolationsschritt der Aufwand für Einrichtung und Terminierung des parallelen Bereichs eingespart. Nach [26] ist dies ab drei parallelen DO-Schleifen lohnend, hier also ab drei Isolationsschritten. Da aber in diesem Fall die Aktualisierung der Anzahl an Intervallen mit mehreren Eigenwerten sowie der Schleifenwerte `ANFANG` und `ENDE` (s. Abbildung 5.3) mit Hilfe der Direktiven CASE und END CASE zusätzlich in eine Kontrollstruktur für genau einen Prozeß eingeschlossen werden muß, ergeben sich keine Zeitvorteile. Deshalb ist in allen oben genannten Varianten der Isolationsphase die Direktive DO ALL implementiert. Die Aktualisierung der Listenzeiger für die Hauptlisten sowie die Bestimmung der neuen Anzahl an Intervallen mit mehreren Eigenwerten (s. Abbildung 5.3) sind unter Verwendung der Direktiven GUARD und END GUARD in einen kritischen Bereich eingeschlossen, den mehrere Tasks nicht gleichzeitig bearbeiten können.

Ein weiteres Bisektionsverfahren ist implementiert, das sowohl die Folge (2.14) verwendet als auch vektorisiert. Dieses Verfahren nutzt die Beschleunigung durch Vektorisierung und spart gegenüber Multisektion Code. Letzteres führt ebenfalls zu schnelleren Ausführungszeiten. Alle Intervalle mit mehreren Eigenwerten eines Isolationsschritts werden in 8 möglichst gleich große Gruppen aufgeteilt. Jede innere Schleife eines Isolationsschritts läuft über alle Gruppenelemente und vektorisiert. Nach einigen Bisektionsschritten werden größere Vektorlängen erreicht, so daß die Vektorisierung der inneren Schleifen sehr lohnend wird. Ist die Anzahl an Intervallen mit mehreren Eigenwerten kleiner als 8, so entspricht die Zahl der Gruppen dieser Anzahl. Jede Gruppe enthält dann genau ein Element. Ist die Anzahl an Intervallen mit mehreren Eigenwerten größer als 8, aber nicht durch 8 teilbar, so erhalten die ersten der 8 Gruppen jeweils ein Element mehr als die letzten, bis der nicht durch 8 teilbare Rest aufgeteilt ist. Die Anzahl der Gruppen ist frei wählbar. Bei 8 Gruppen kann jedem der 8 Prozessoren einer CRAY Y-MP eine Gruppe zur Bearbeitung zugeteilt werden. Bei sequentiellen Programmläufen ist eine Gruppe am günstigsten, da dann in jedem Isolationsschritt die größten Vektorlängen erreicht werden.

In dem oben genannten Bisektionsverfahren wird die Schleife über alle Gruppen, die die äußere Schleife jedes Isolationsschritts darstellt, unter Verwendung der Autotasking-Direktive DO ALL parallelisiert. Die Aktualisierung der Listenzeiger für die Hauptlisten sowie die Bestimmung der neuen Anzahl an Intervallen mit mehreren Eigenwerten sind in einen kritischen Bereich unter Verwendung der Direktiven GUARD und END GUARD eingeschlossen.

Werden in der Extraktionsphase Bisektion mit Auswertung des charakteristischen Polynoms oder das Pegasusverfahren verwendet, ist jeweils die Übergabe der Anzahl der Eigenwerte, die kleiner als die unteren Intervallgrenzen der Intervalle mit genau einem Eigenwert sind und der Anzahl der Eigenwerte, die kleiner als die oberen Intervallgrenzen dieser Intervalle sind, an die Extraktionsphase überflüssig. In der Extraktionsphase werden stattdessen die Funktionswerte des charakteristischen Polynoms an den Intervallgrenzen berechnet. Die sich dadurch ergebenden Einsparungen in der Isolationsphase sind gering und beinflussen das Zeitverhalten der Isolationsphase kaum.

6.3.2 Auswirkungen der Initialisierungsphase auf die Isolationsphase

In diesem Abschnitt wird untersucht, wie sich Multisektion mit drei bzw. 15 Punkten in der Initialisierungsphase auf das Zeitverhalten der Isolationsphase auswirkt. In der Isolationsphase werden einerseits Bisektion ohne Vektorisierung, andererseits Multisektion mit 15 Punkten mit Vektorisierung nach Algorithmus 5.2 durchgeführt. Beide Verfahren verwenden die Folge (2.14).
In Abbildung 6.2 werden die Laufzeiten des Bisektionsverfahrens in der Isolationsphase bei Multisektion mit drei bzw. 15 Punkten in der Initialisierungsphase verglichen. Gemessen wurden die Zeiten für die Isolation aller Eigenwerte der Matrizen $A512$ und $Z512$ auf CRAY Y-MP bei sequentieller und paralleler Programmausführung.
Abbildung 6.2 zeigt, daß sich bei Multisektion mit 15 Punkten in der Initialisierungsphase bei allen Messungen etwas niedrigere Zeiten ergeben als bei Multisektion mit 3 Punkten. Die Zeitgewinne in der Isolationsphase bei Multisektion mit 15 Punkten in der Initialisierungsphase gegenüber Multisektion mit 3 Punkten betragen bei dem Matrizen $A512$ und $Z512$ sequentiell 2.3 ms bzw. 6.1 ms sowie parallel 0.8 ms bzw. 1.5 ms. In der Initialisierungsphase dagegen treten Zeitverluste von 0.27 ms bzw. 0.33 ms auf (s. Abbildung 6.1). Die Zeitverluste in der Initialisierungsphase gleichen also die Zeitgewinne in der Isolationsphase bei weitem nicht aus. Bei Multisektion mit 15 Punkten in der Initialisierungsphase werden bei den Matrizen $A512$ und $Z512$ aufwendige Isolationsschritte auf Kosten von wenig Mehraufwand in der Initialisierungsphase gegenüber Multisektion mit 3 Punkten eingespart. Bei paralleler Ausführung erzeugt Multisektion mit 15 Punkten in der Initialisierungsphase bei diesen Matrizen zu Beginn der Isolationsphase zusätzlich mehr Tasks (mehr Intervalle mit mehreren Eigenwerten) als bei Multisektion mit 3 Punkten. Daher können schon im ersten Isolationsschritt mehr Prozessoren beschäftigt werden. Aus den Messungen in Abbildung 6.2 läßt sich der Schluß ziehen, daß bei Bisektion in der Isolationsphase Multisektion mit 15 Punkten in der Initialisierungsphase in der Regel günstiger als Multisektion mit 3 Punkten ist. Dies ist nur dann nicht der Fall, wenn Multisektion mit 3 Punkten in der Initialisierungsphase mehr Intervalle mit mehreren Eigenwerten er-

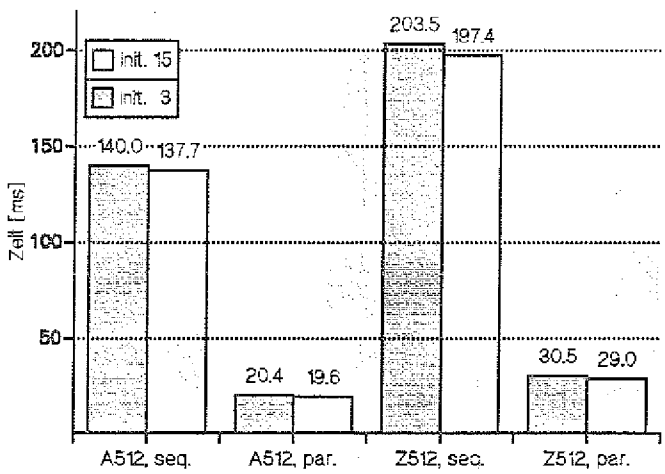


Abb. 6.2: Bisektion in der Isolationsphase: Vergleich der Zeiten bei Multisektion mit drei bzw. 15 Punkten in der Initialisierungsphase

zeugt als Multisektion mit 15 Punkten. Wie schon in 5.3.2 erwähnt, kann dies durchaus vorkommen.

Abbildung 6.3 zeigt entsprechende Zeitmessungen bei Multisekion mit 15 Punkten in der Isolationsphase.

Bei Matrix $A512$ sind die sequentiellen und parallelen Ausführungszeiten bei Multisektion mit 3 Punkten in der Initialisierungsphase wesentlich niedriger als bei Multisektion mit 15 Punkten. Dies weist auf die starke Abhängigkeit des Multisektionsverfahrens in der Isolationsphase von der Verteilung der Eigenwerte in den Ausgangsintervallen hin. Multisektion der Ordnung k ist am effektivsten, wenn in einem Intervall mit $k+1$ Eigenwerten alle Eigenwerte in einem Multisektionsschritt getrennt werden. Dies ist jedoch nur selten der Fall. In ungünstigen Fällen - wird z.B. kein Eigenwert getrennt - wird Multisektion sehr aufwendig. Aus diesen Gründen kann Multisektion in der Isolationsphase bei weniger Ausgangsintervallen mit günstiger Verteilung der Eigenwerte wesentlich bessere Ergebnisse als bei mehr Ausgangsintervallen mit schlechter Verteilung der Eigenwerte liefern. Dies ist bei Matrix $A512$ der Fall.

Bei Matrix $Z512$ ist die sequentielle Ausführungszeit bei Multisektion mit 3 Punkten in der Initialisierungsphase geringfügig niedriger als bei Multisektion mit 15 Punkten. Die parallele Ausführungszeit dagegen ist bei Multisektion mit 15 Punkten in der Initialisierungsphase geringfügig besser als bei Multisektion mit 3 Punkten. Dies ist dadurch zu

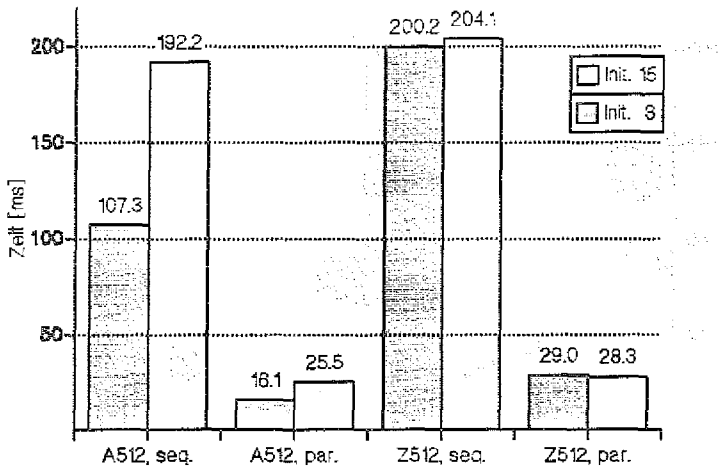


Abb. 6.3: Multisektion mit 15 Punkten in der Isolationsphase: Vergleich der Zeiten bei Multisektion mit drei bzw. 15 Punkten in der Initialisierungsphase

erklären, daß bei Multisektion mit 15 Punkten in der Initialisierungsphase in den meisten Fällen schon im ersten Isolationsschritt alle acht Prozessoren der CRAY Y-MP Intervalle mit mehreren Eigenwerten bearbeiten können, während bei Multisektion mit 3 Punkten im ersten Isolationsschritt maximal 4 Prozessoren eingesetzt werden können.
Da die Effektivität von Multisektion in der Isolationsphase stark von der Verteilung der Eigenwerte abhängt, die im Normalfall unbekannt ist, läßt sich keine Regel für eine günstige Anzahl von Multisektionspunkten in der Initialisierungsphase aufstellen.

6.3.3 Multisektion oder Bisektion in der Isolationsphase?

In diesem Abschnitt wird untersucht, wie sich Bisektion und Multisektion auf das Zeitverhalten der Isolationsphase auswirken. Das getestete Multisektionsverfahren verwendet die Folge (2.14) und vektorisiert nach Algorithmus 5.2. Bei den Matrizen $A512$ und $Z512$ wurden die sequentiellen und parallelen Ausführungszeiten der Isolationsphase bei 1 bis 40 Multisektionspunkten gemessen, bei den Matrizen $A128$ und $Z128$ bei 1 bis 127 Multisektionspunkten. Die Anzahl der Multisektionspunkte, die die kürzesten Ausführungszeiten ergab, wurde jeweils festgehalten. Alle Messungen wurden auf CRAY X-MP und CRAY Y-MP durchgeführt, um zu untersuchen, ob sich die unterschiedlichen Startup-Zeiten beider Rechner bei Vektoroperationen auf die Ergebnisse auswirken. Das zum Vergleich herange-

zogene Bisektionsverfahren verwendet ebenfalls die Folge (2.14) und vektorisiert nicht. Abbildung 6.4 zeigt die sequentiellen Ausführungszeiten der Isolationsphase für die Matrix $A512$ bei 1 bis 40 Multisektionspunkten auf CRAY X-MP und CRAY Y-MP. In der Initialisierungsphase wurde Multisektion mit 3 Punkten durchgeführt. Die diskreten Meßpunkte wurden aus Gründen der Übersichtlichkeit miteinander verbunden.

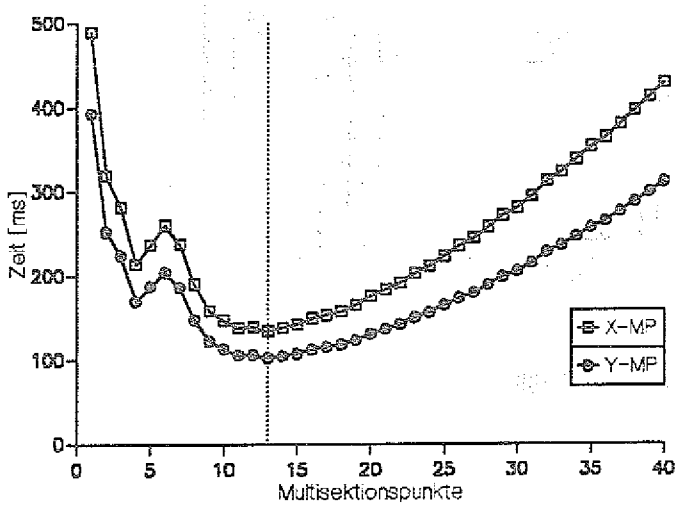


Abb. 6.4: Isolationsphase: Multisektion mit 1 bis 40 Punkten, sequentielle Ausführungszeiten bei Matrix $A512$

Die optimale Anzahl an Multisektionspunkten mit den kürzesten Ausführungszeiten für die Isolationsphase liegt bei beiden Rechnern in dieser Meßreihe bei 13 Punkten. Es ist zu erkennen, daß sich die Zeiten im Bereich von 10 bis 18 Punkten wenig unterscheiden. Bei höherer Anzahl der Multisektionspunkte steigen die Ausführungszeiten langsam an. Es ist nicht auszuschließen, daß bei höherer Anzahl der Multisektionspunkte wieder kürzere Ausführungszeiten auftreten. Dies hängt stark von der Verteilung der Eigenwerte ab, die gewöhnlich unbekannt ist. In den meisten Fällen wächst jedoch bei höherer Anzahl der Multisektionspunkte der zeitliche Aufwand der Multisektion stärker als der Nutzen durch Trennung von mehr Eigenwerten in einem Isolationsschritt. Die auf CRAY X-MP gemessenen Werte liegen wegen der höheren Taktzeit über den auf CRAY Y-MP gemessenenen. Die Abbildungen 6.5 und 6.6 zeigen die sequentiellen Ausführungszeiten der Isolationsphase mit optimaler Anzahl der Multisektionspunkte der jeweiligen Meßreihen auf CRAY X-MP und CRAY Y-MP im Vergleich zu den durch Bisektion erzielten Zeiten. In der Initialisierungsphase wurde Multisektion mit 3 Punkten durchgeführt.

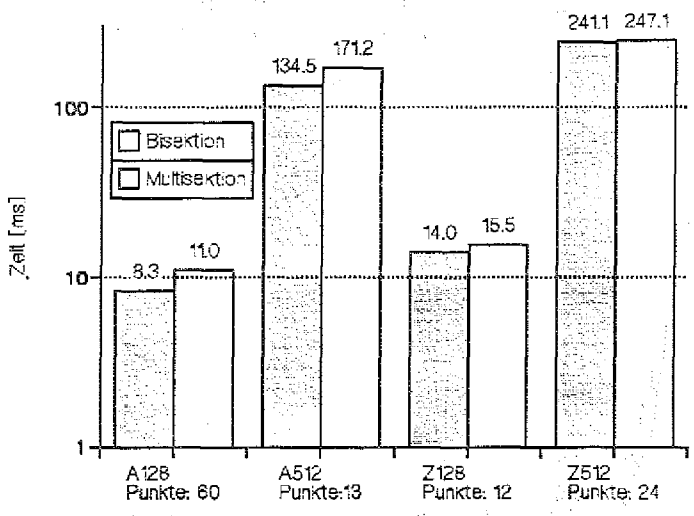


Abb. 6.5: Isolationsphase: Vergleich zwischen Multisektion mit optimaler Punktzahl und Bisektion auf CRAY X-MP, sequentiell

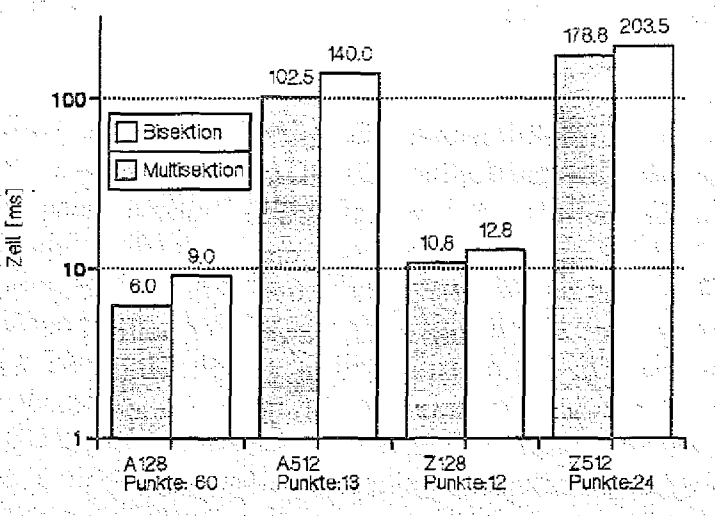


Abb. 6.6: Isolationsphase: Vergleich zwischen Multisektion mit optimaler Punktzahl und Bisektion auf CRAY Y-MP, sequentiell

Die Skalierung der y-Achse ist logarithmisch, um die Messungen mit den Matrizen $A128$, $A512$, $Z128$ und $Z512$ in einem Diagramm darstellen zu können.

Wie den Abbildungen 6.5 und 6.6 zu entnehmen ist, werden bei Multisektion mit in diesen Meßreihen optimaler Punktzahl bei allen Matrizen deutlich kürzere Zeiten als bei Bisektion erzielt. Auf beiden Rechnern ist die jeweils optimale Anzahl der Multisektionspunkte gleich. In der Isolationsphase ist bei sequentieller Ausführung nur die Verteilung der Eigenwerte ausschlaggebend. Die unterschiedlichen Startup Zeiten auf CRAY X-MP und CRAY Y-MP beeinflussen die optimale Anzahl der Multisektionspunkte nicht.

In den Abbildungen 6.7 und 6.8 sind die parallelen Ausführungszeiten der Isolationsphase mit der optimalen Anzahl der Multisektionspunkte der jeweiligen Meßreihen auf CRAY X-MP und CRAY Y-MP im Vergleich zu den durch Bisektion erzielten Zeiten dargestellt.

Bei paralleler Ausführung liegen die Zeiten für Multisektion mit optimaler Punktzahl ebenfalls deutlich unter denen von Bisektion. Es ergeben sich andere optimale Punktzahlen als bei sequentieller Ausführung, da bei paralleler Ausführung neben der Verteilung der Eigenwerte auch die Anzahl der in jedem Isolationsschritt vorhandenen Tasks (entspricht der Anzahl an Intervallen mit mehreren Eigenwerten) und die Aufteilung der Tasks auf die Prozessoren das Zeitverhalten beeinflussen. In den Isolationsschritten kann eine nicht durch 4 bzw. 8 teilbare Anzahl der Intervalle mit mehreren Eigenwerten vorkommen, so daß die 4 bzw. 8 Prozessoren nicht gleichmäßig ausgelastet werden. Die sehr großen Unterschiede zwischen Bisektion und Multisektion bei den Messungen auf CRAY Y-MP lassen sich dadurch erklären, daß bei Bisektion in den ersten Isolationsschritten nur wenige Tasks vorhanden sind und die 8 Prozessoren noch nicht voll ausgelastet werden. Zusätzlich steigt durch die größere Anzahl an Isolationsschritten bei Bisektion gegenüber Multisektion der Overhead durch die Parallelisierung jedes Isolationsschritts. Da auf CRAY X-MP nur 4 Prozessoren vorhanden sind, beeinflussen diese Effekte die Zeiten hier nicht so stark. Durch die unterschiedliche Prozessorzahl auf CRAY X-MP und CRAY Y-MP läßt sich auch eine unterschiedliche optimale Anzahl der Multisektionspunkte beider Rechner wie bei Matrix $A128$ erklären. Die Aufteilung der Tasks auf 4 bzw. 8 Prozessoren kann zu unterschiedlichen Ergebnissen führen.

In den bisherigen Messungen war Bisektion Multisektion mit günstiger Punktzahl unterlegen. Eine günstige Anzahl von Multisektionspunkten kann jedoch nicht ohne Kenntnis der Verteilung der Eigenwerte vor der Programmausführung ermittelt werden. Die Verteilung der Eigenwerte ist in den meisten Anwendungen unbekannt. In den Abbildungen 6.9 und 6.10 werden daher Multisektion mit 15 Punkten in der Isolationsphase und Bisektion bei sequentieller und paralleler Programmausführung auf CRAY Y-MP bei Matrizen hoher Ordnung verglichen. 15 Multisektionspunkte wurden gewählt, da diese Zahl im Bereich der bisher gemessenen optimalen Multisektionspunkte liegt, der Aufwand der Multisektion bei 15 Punkten nicht sehr hoch ist und die sich im günstigsten Fall in jedem Multisektionsschritt ergebende Anzahl von 16 Intervallen mit mehreren Eigenwerten durch 8 teilbar ist.

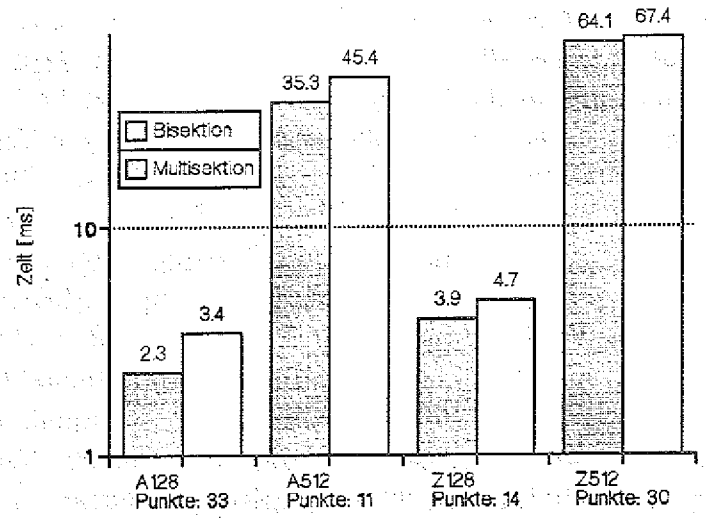


Abb. 6.7: Isolationsphase: Vergleich zwischen Multisektion mit optimaler Punktzahl und Bisektion auf CRAY X-MP, parallel

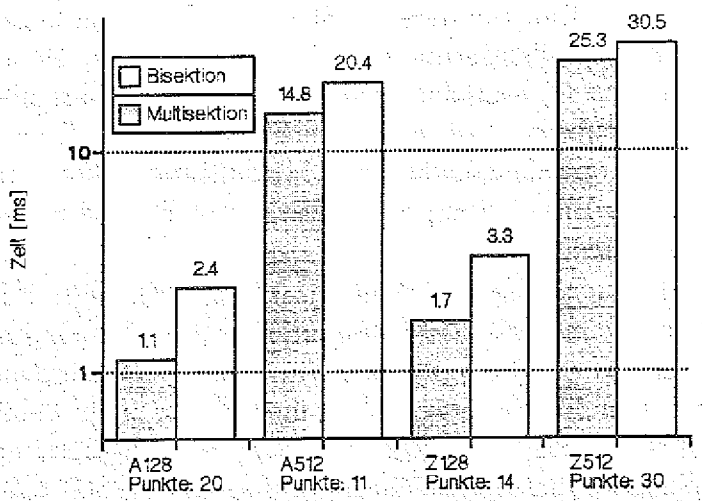


Abb. 6.8: Isolationsphase: Vergleich zwischen Multisektion mit optimaler Punktzahl und Bisektion auf CRAY Y-MP, parallel

In der Initialisierungsphase wurde Multisektion mit 15 Punkten durchgeführt. Dies ist, wie schon in 6.3.2 erwähnt, für Bisektion in der Regel günstiger. Die Messungen wurden mit den Matrizen $A3200$, $A6400$, $Z3200$ und $Z6400$ durchgeführt.
Die Abbildungen zeigen, daß sich für die Matrizen $A6400$ und $Z6400$ bei Bisektion sequentiell und parallel geringfügig niedrigere Zeiten als bei Multisektion mit 15 Punkten ergeben. Bei Matrix $A3200$ liegen die Ausführungszeiten für Multisektion sequentiell und parallel deutlich unter denen für Bisektion. Die Ausführungszeiten für die Matrix $Z3200$ bei Multisektion sind dagegen nur sehr geringfügig niedriger als bei Bisektion. Hier zeigt sich, daß Bisektion und Multisektion mit 15 Punkten in der Isolationsphase bei für Multisektion nicht günstigen Verteilungen der Eigenwerte gleichwertig sind. Ist jedoch die Verteilung der Eigenwerte für Multisektion mit 15 Punkten günstig, so ergeben sich deutliche Zeitvorteile für Multisektion.

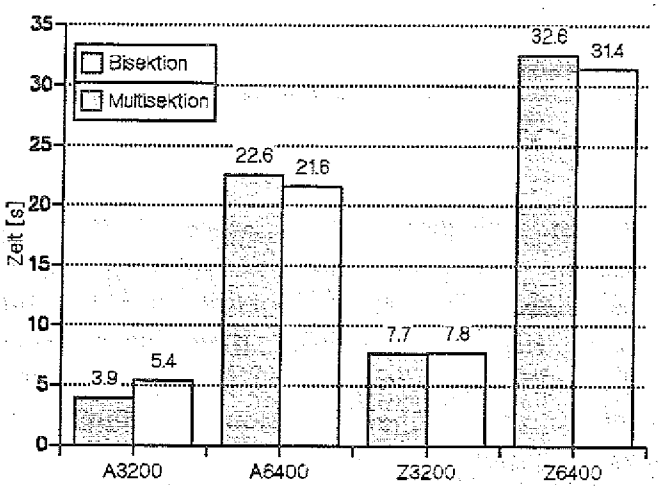


Abb. 6.9: **Isolationsphase: Vergleich zwischen Multisektion mit 15 Punkten und Bisektion auf CRAY Y-MP bei Matrizen hoher Ordnung, sequentiell**

Zusammenfassend läßt sich sagen, daß sich das Zeitverhalten von Multisektion und Bisektion nur geringfügig unterscheidet, wenn die Anzahl der Multisektionspunkte nicht gut auf die Verteilung der Eigenwerte abgestimmt ist. Da die Verteilung der Eigenwerte in den meisten Anwendungsfällen unbekannt ist, ist eine gute Abstimmung vor Programmausführung nicht möglich. Ist jedoch die Anzahl der Multisektionpunkte für die Verteilung der Eigenwerte günstig, lassen sich durch Multisektion in der Isolationsphase deutlich kürzere Zeiten erzielen als durch Bisektion. Bei paralleler Programmausführung besitzt Multisektion in

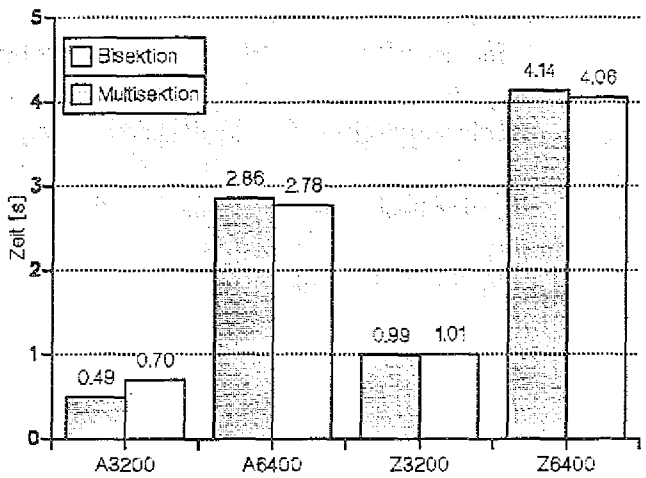


Abb. 6.10: Isolationsphase: Vergleich zwischen Multisektion mit 15 Punkten und Bisektion auf CRAY Y-MP bei Matrizen hoher Ordnung, parallel

den ersten Isolationsschritten Vorteile gegenüber Bisektion, da mehr Tasks erzeugt werden. Da jedoch auch bei Bisektion nach einigen Isolationsschritten genügend Tasks vorhanden sind, wirkt sich dies vor allem bei Matrizen hoher Ordnung kaum auf das Zeitverhalten der Isolationsphase aus (s. auch Abbildung 6.10). Bei den hier verwendeten Matrizen liefert Multisektion mit 15 Punkten kürzere oder ungefähr gleiche Zeiten wie Bisektion. Der Mehraufwand der Multisektion gegenüber Bisektion wird durch die Vektorisierung nach Algorithmus 5.2 in Grenzen gehalten.

6.3.4 Verwendung der Sturmschen Kette bei Multisektion mit 15 Punkten

In diesem Abschnitt wird die Verwendung der Sturmschen Kette bei Multisektion mit 15 Punkten in der Isolationsphase untersucht. Die Auswertung der Sturmschen Kette wird mit Hilfe der CRAY-Routine SOLR vektorisiert. In den Abbildungen 6.11 und 6.12 werden die sequentiellen Ausführungszeiten der Isolationsphase auf CRAY Y-MP bei Multisektion mit 15 Punkten unter Verwendung der Sturmschen Kette sowie der Folge (2.14) verglichen. Die untersuchten Matrizen sind $A128$, $A512$, $A3200$, $Z128$, $Z512$ und $Z3200$.
Den Abbildungen 6.11 und 6.12 ist zu entnehmen, daß die Ausführungszeiten bei Verwendung der Sturmschen Kette höher sind als bei Verwendung der Folge (2.14). Dies zeigt,

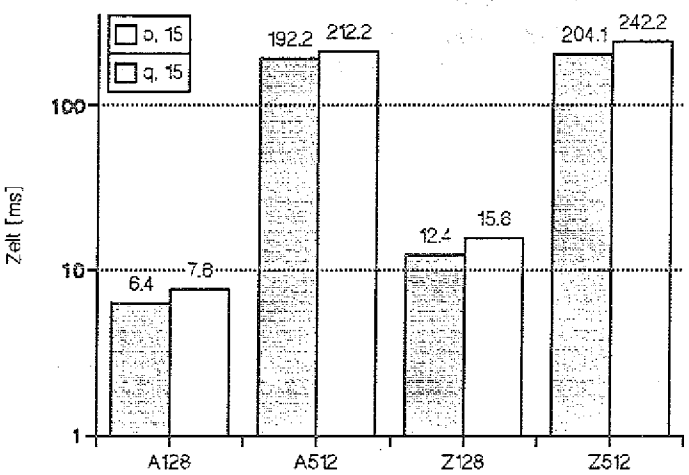


Abb. 6.11: Isolationsphase: Multisektion mit 15 Punkten bei Verwendung der Sturmschen Kette und der Folge (2.14), Matrizen niedriger Ordnung

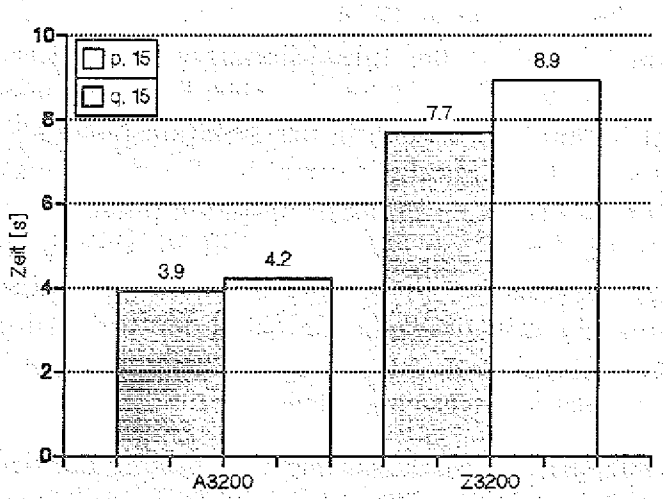


Abb. 6.12: Isolationsphase: Multisektion mit 15 Punkten bei Verwendung der Sturmschen Kette und der Folge (2.14), Matrizen hoher Ordnung

wie schon in 6.2.2 erwähnt, daß die Vektorisierung der Auswertung der Folge (2.14) nach Algorithmus 5.2 bei Multisektion mit 15 Punkten günstiger ist als die der Auswertung der Sturmschen Kette mit Hilfe der CRAY-Routine SOLR. Ferner läßt sich das Zählen der negativen Vorzeichen der Glieder der Folge (2.14) effizienter codieren als das Zählen von Vorzeichenwechseln bei den Folgengliedern der Sturmschen Kette (s. auch 2.2). Bei Multisektion mit 15 Punkten ist also die Verwendung der Folge (2.14) der Verwendung der Sturmschen Kette in der Isolationsphase vorzuziehen.

6.3.5 Verwendung der Sturmschen Kette bei Bisektion in der Isolationsphase

Während bei dem bisher erwähnten Bisektionsverfahren mit Verwendung der Folge (2.14) keine Vektorisierung in der Isolationsphase möglich ist, läßt sich bei Bisektion mit Verwendung der Sturmschen Kette die Auswertung der Sturmschen Kette mit Hilfe der CRAY-Routine SOLR vektorisieren. Abbildung 6.13 zeigt die sequentiellen Ausführungszeiten auf CRAY Y-MP von Bisektion mit Verwendung der Sturmschen Kette im Vergleich zu Multisektion mit 15 Punkten und für die Verteilung der Eigenwerte günstiger Punktzahl unter Verwendung der Folge (2.14) sowie Bisektion mit Verwendung der Folge (2.14). Bei den Multisektionsverfahren wurde in der Initialisierungsphase Multisektion mit 3 Punkten durchgeführt. Dies erwies sich in 6.3.2 bei den auch hier verwendeten Matrizen $A512$ und $Z512$ bei Multisektion mit 15 Punkten in der Isolationsphase als günstiger als Multisektion mit 15 Punkten in der Initialisierungsphase. Bei den Bisektionsverfahren wird Multisektion mit 15 Punkten in der Initialisierungsphase durchgeführt. In der Legende weist q auf die Verwendung der Folge (2.14) hin, p auf die Verwendung der Sturmschen Kette. Die zweite Angabe bezeichnet die Zahl der verwendeten Multisektionspunkte bzw. das verwendete Verfahren, die dritte Angabe die Zahl der Multisektionspunkte in der Initialisierungsphase. Bei Multisektion mit günstiger Punktzahl („opt.“) ist die Zahl der Multisektionspunkte unter der x-Achse angegeben.

Abbildung 6.13 zeigt, daß sich bei Bisektion mit Verwendung der Sturmschen Kette die deutlich kürzesten Laufzeiten ergeben. Dies ist durch die Vektorisierung der Auswertung der Sturmschen Kette zu erklären, die neben der effizienten Codierung der Bisektion die schnellen Ausführungszeiten bewirkt. Die Verwendung der Sturmschen Kette hat jedoch den Nachteil, daß ihre Auswertung leicht zu Zahlenbereichsüberschreitungen führen kann (s. auch 2.2).

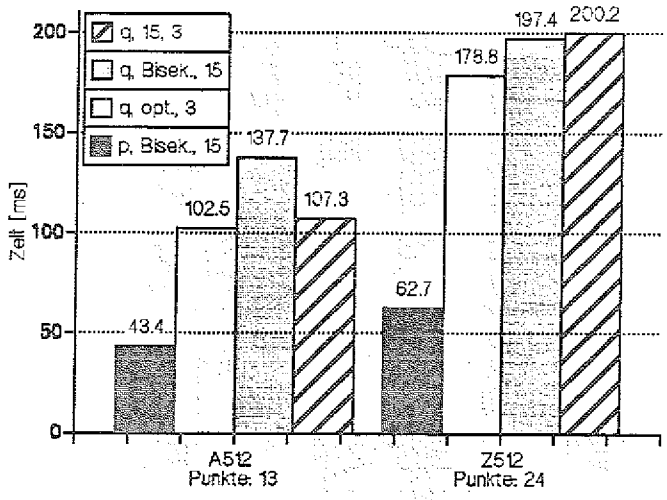


Abb. 6.13: Isolationsphase: Vergleich von verschiedenen Bisektions- und Multisektionsverfahren, sequentiell

6.3.6 Speedup-Messungen bei Matrizen unterschiedlicher Ordnung

In diesem Abschnitt wird die Abhängigkeit der Speedup-Werte von der Ordnung der Matrizen untersucht. Bei dem hier untersuchten Verfahren wird in der Initialisierungsphase Multisektion mit 15 Punkten durchgeführt, in der Isolationsphase Multisektion mit 15 Punkten unter Verwendung der Folge (2.14). Zur Berechnung der Speedup-Werte wurde die kürzeste von mehreren sequentiellen Ausführungszeiten durch die kürzeste von mehreren parallelen Ausführungszeiten dividiert. Bei Matrizen bis einschließlich der Ordnung 1000 wurden je 4 Messungen durchgeführt, bei höheren Ordnungen je 2 Messungen. Gemessen wurde auf CRAY Y-MP bei dedizierter Maschine mit den Matrizen $A128$, $A512$, $A800$, $A1000$, $A3200$ und $A6400$ sowie $Z128$, $Z512$, $Z800$, $Z1000$, $Z3200$ und $Z6400$. Bei den parallelen Messungen wurden alle 8 Prozessoren eingesetzt. Die Abbildungen 6.14 und 6.15 zeigen die jeweiligen Speedup-Werte.

Den Abbildungen 6.14 und 6.15 ist zu entnehmen, daß die Speedup-Werte mit zunehmender Ordnung der Matrizen steigen und ab der Ordnung 512 in der Nähe des maximal möglichen Speedups 8 liegen. Die Abweichungen vom maximalen Speedup bei diesen Ordnungen sind zum einen durch den Overhead durch die Parallelisierungsroutinen, zum anderen durch eine geringfügig ungleiche Auslastung der Prozessoren in einigen Isolations-

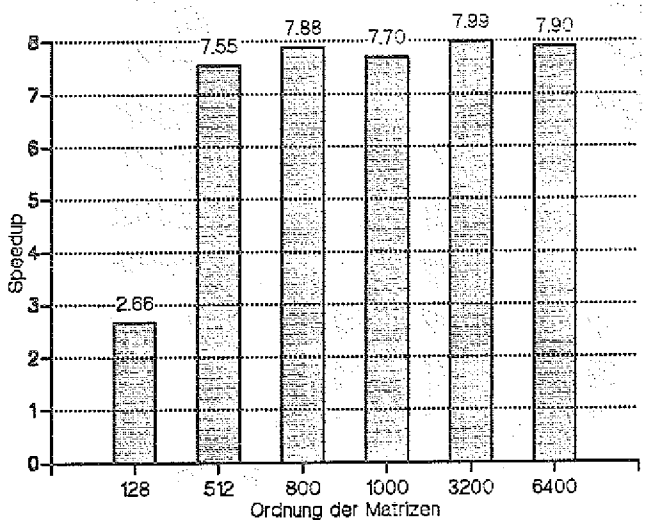


Abb. 6.14: Isolationsphase: Speedup-Werte bei den Matrizen An

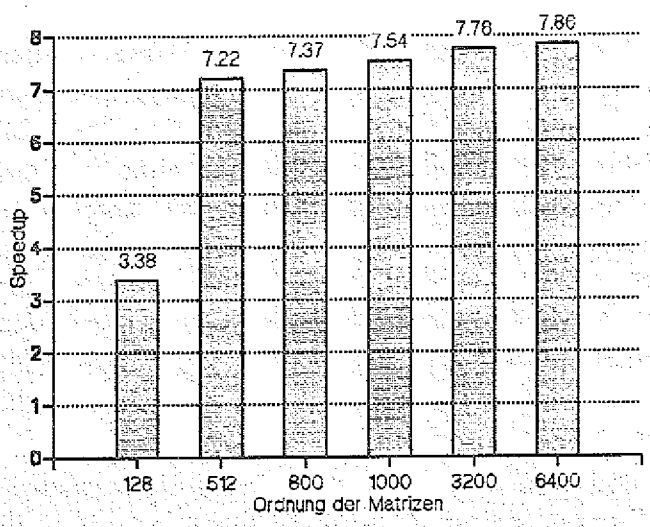


Abb. 6.15: Isolationsphase: Speedup-Werte bei den Matrizen Zn

schritten zu erklären. Eine ungleiche Auslastung der Prozessoren kann vor allem in den ersten und letzten Isolationsschritten auftreten, da dort in der Regel nur wenige Intervalle mit mehreren Eigenwerten vorliegen. In den mittleren Isolationsschritten sind die Prozessoren durch die sehr hohe Anzahl an Intervallen mit mehreren Eigenwerten bei diesen Matrizen gleichmäßig ausgelastet. An sich stellt jede Matrix ein eigenes Problem dar, da die Parallelisierung der Isolationsphase stark von der Verteilung der Eigenwerte abhängt. Liegen alle Eigenwerte einer Matrix sehr nah beieinander, so ist die Isolationsphase sehr schlecht parallelisierbar. Die Messungen mit den Matrizen An und Zn, die keine besonders günstige Verteilung der Eigenwerte aufweisen, zeigen jedoch, daß außer bei sehr extremen Verteilungen der Eigenwerte gute Speedup-Werte erreicht werden können.

Der schlechte Speedup bei den Matrizen $A128$ und $Z128$ kommt zustande, da selbst bei 4 aufeinanderfolgenden Messungen die Ausführungszeit des Programms kleiner ist als die Zeit, bis alle Prozessoren zur Verfügung stehen. Diese Zeit liegt im Bereich von einigen 10 Millisekunden. Hieraus folgt, daß sich maximale Speedups erst ab Ausführungszeiten im Bereich von Sekunden erreichen lassen. Um dies zu überprüfen, wurde vor dem eigentlichen Programm ein paralleles Dummy-Unterprogramm mit einer Ausführungszeit von einigen Sekunden ausgeführt. Die Speedup-Werte der Isolationsphase ergaben daraufhin 5.31 bei Matrix $A128$ und 5.45 bei Matrix $Z128$. Die jetzt noch auftretenden Speedup-Verluste sind dadurch zu erklären, daß sich der Overhead durch die Parallelisierung und eine geringfügig unterschiedliche Auslastung der Prozessoren bei Matrizen niedriger Ordnung stärker auswirken als bei Matrizen hoher Ordnung. Parallelisierung von Routinen, deren Ausführungszeiten bei den vorliegenden Problemgrößen im Millisekunden-Bereich liegen, ist auf CRAY-Rechnern in Anwendungenfällen nur dann sinnvoll, wenn die Routinen sehr oft ausgeführt werden oder Teil eines Gesamtprogramms sind, in dem vorher parallele Abschnitte mit höheren Ausführungszeiten durchlaufen werden.

6.3.7 Speedup-Messungen bei 1 bis 8 Prozessoren

Abbildung 6.16 zeigt die auf CRAY Y-MP mit 1 bis 8 Prozessoren erzielten Speedup-Werte der Isolationsphase bei Matrix $A3200$. In der Initialisierungsphase wurde Multisektion mit 15 Punkten durchgeführt, in der Isolationsphase Multisektion mit 15 Punkten unter Verwendung der Folge (2.14).

Alle Speedup-Werte in Abbildung 6.16 liegen nahe am maximal erreichbaren Wert. Da in den Isolationsschritten sehr viele Tasks (entspricht der Anzahl an Intervallen mit mehreren Eigenwerten) erzeugt werden, können die jeweils angeforderten Prozessoren gut ausgelastet werden. Trotzdem auftretende Speedup-Verluste sind mit dem Overhead durch die Parallelisierungsroutinen und geringfügig unterschiedliche Auslastungen der jeweils angeforderten Prozessoren in einzelnen Isolationsschritten zu erklären.

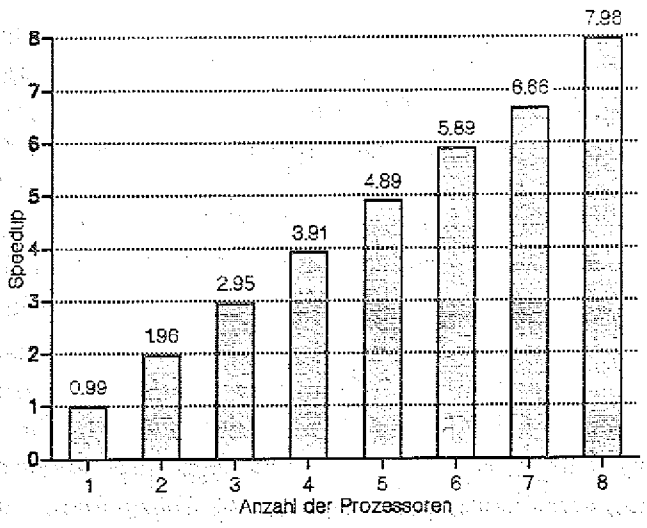


Abb. 6.16: Isolationsphase: Speedup-Werte bei 1 bis 8 Prozessoren, Matrix $A3200$

6.3.8 Parallele Bisektion mit Partitionierung in der Isolationsphase

In diesem Abschnitt wird das Zeitverhalten von paralleler Bisektion mit Partitionierung in der Isolationsphase untersucht. Dieses Bisektionsverfahren teilt alle vorliegenden Intervalle mit mehreren Eigenwerten in Gruppen möglichst gleicher Größe auf. Alle inneren Schleifen des Verfahrens laufen über die Anzahl der Gruppenelemente und vektorisieren. Die Vektorlänge entspricht der Anzahl der Gruppenelemente im jeweiligen Isolationsschritt. Die äußere Schleife jedes Isolationsschritts, die über die Anzahl der Gruppen läuft, wird parallelisiert. Zur Bestimmung der jeweiligen Anzahl der Eigenwerte in den durch Bisektion entstandenen Teilintervallen wird die Folge (2.14) verwendet. In Abbildung 6.17 werden die sequentiellen Ausführungszeiten auf CRAY Y-MP von Bisektion mit Verwendung der Sturmschen Kette, Bisektion ohne Partitionierung (entspricht einer Gruppe), Bisekton mit Partitionierung in 8 Gruppen und Multisektion mit 15 Punkten verglichen. Die 3 zuletzt genannten Verfahren verwenden die Folge (2.14). In allen Verfahren wurde in der Initialisierungsphase Multisektion mit 15 Punkten durchgeführt. Gemessen wurde mit den Matrizen $A512$, $A800$ und $A1000$. In der Legende weist p auf die Verwendung der Sturmschen Kette hin, q auf die Verwendung der Folge (2.14).

Abbildung 6.17 zeigt, daß die sequentiellen Ausführungszeiten von Bisektion ohne Partitionierung die bei weitem kürzesten sind. Bei Bisektion ohne Partitionierung ergeben

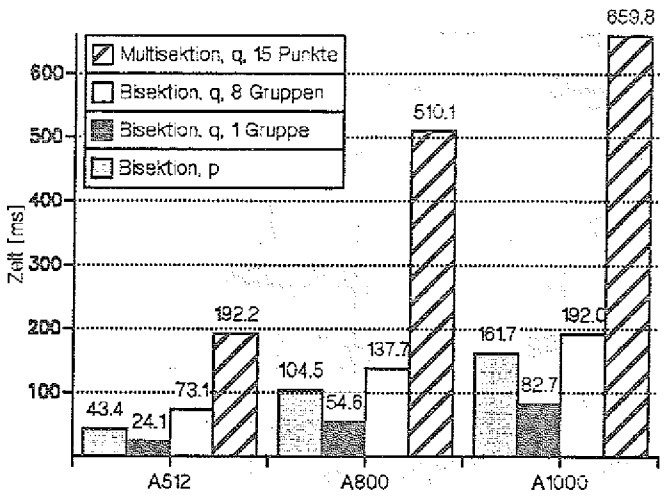


Abb. 6.17: Isolationsphase: Vergleich von Bisektion mit Verwendung der Sturmschen Kette, Bisektion ohne Partitionierung, Bisektion mit Partitionierung in 8 Gruppen und Multisektion mit 15 Punkten, sequentiell

sich ca. 8 mal höhere Vektorlängen als bei Bisektion mit Partitionierung in 8 Gruppen. Dies erklärt die Zeitunterschiede dieser beiden Verfahren. Abbildung 6.17 ist ferner zu entnehmen, daß die Vektorisierung bei Bisektion ohne Partitionierung besser ist als die Vektorisierung der Auswertung der Sturmschen Kette mit Hilfe der CRAY-Routine SOLR. Bisektion mit Partitionierung in 8 Gruppen führt dagegen zu höheren Laufzeiten als Bisektion mit Verwendung der Sturmschen Kette. Bei höheren Ordnungen der Matrizen werden jedoch die Unterschiede geringer, da dann bei Bisektion mit Partitionierung höhere Vektorlängen erreicht werden. Bisektion mit bzw. ohne Partitionierung sind auf jeden Fall Multisektion mit 15 Punkten vorzuziehen, wie Abbildung 6.17 deutlich zeigt. Multisektion mit günstiger Punktzahl (s. auch Abbildung 6.13, Matrix $A512$) führt ebenfalls zu höheren Ausführungszeiten als Bisektion mit bzw. ohne Partitionierung.

Abbildung 6.18 vergleicht die sequentiellen und parallelen Ausführungszeiten von Bisektion mit Verwendung der Sturmschen Kette und Bisektion mit Partitionierung in 8 Gruppen bei den Matrizen $A3200$ und $Z3200$ für CRAY Y-MP. Ferner sind die sequentiellen Ausführungszeiten von Bisektion ohne Partitionierung ergänzt.

Die Ausführungszeiten von Bisektion mit und ohne Partitionierung in Abbildung 6.18 liegen deutlich unter denen von Bisektion mit Verwendung der Sturmschen Kette. Dies ist durch die hohe Anzahl der Intervalle mit mehreren Eigenwerten, die nach einigen Isolati-

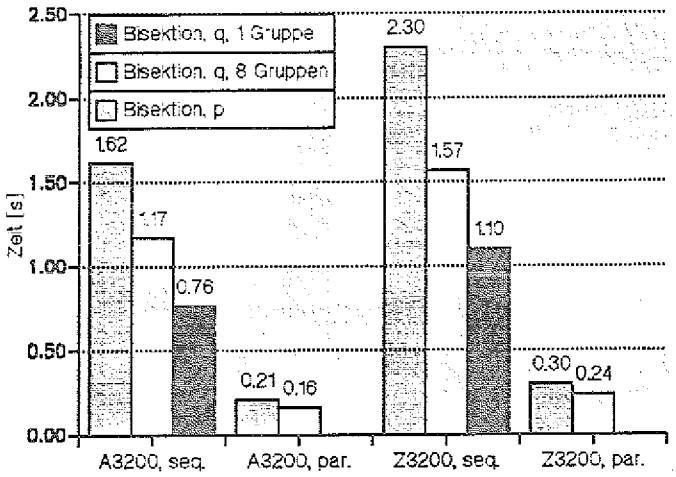


Abb. 6.18: Isolationsphase: Vergleich von Bisektion mit Verwendung der Sturmschen Kette, Bisektion ohne Partitionierung und Bisektion mit Partitionierung in 8 Gruppen

onsschritten bei diesen Matrizen auftritt und die Vektorlängen bei den Bisektionsverfahren mit und ohne Partitionierung bestimmt, zu erklären. Abbildung 6.18 zeigt ferner, daß die sequentiellen Ausführungszeiten von Bisektion mit und ohne Partitionierung bei Matrizen hoher Ordnung näher beieinander liegen als bei Matrizen niedriger Ordnung (zum Vergleich s. Abbildung 6.17).

Speedup-Messungen bei Matrizen unterschiedlicher Ordnung

Die Abbildungen 6.19 und 6.20 zeigen die Speedup-Werte auf CRAY Y-MP mit 8 Prozessoren für die Matrizen $A512$, $A800$, $A1000$, $A3200$ und $A6400$ sowie $Z512$, $Z800$, $Z1000$, $Z3200$ und $Z6400$ bei Bisektion mit Partitionierung in 8 Gruppen. Zur Berechnung der Speedup-Werte wurden die sequentiellen Ausführungszeiten des Bisektionsverfahrens mit Partitionierung in 8 Gruppen durch die parallelen Ausführungszeiten dividiert.
In den Abbildungen 6.19 und 6.20 nähern sich die Speedups bei den Matrizen hoher Ordnung dem Maximalwert 8. Speedup-Verluste treten durch den Overhead durch die Parallelisierungsroutinen und unterschiedliche Auslastung der Prozessoren vor allem in den ersten und letzten Isolationsschritten auf. Ein Bisektionsverfahren benötigt gegenüber einem Multisektionsverfahren zu Beginn der Isolation mehr Isolationsschritte, um eine große Anzahl an Intervallen mit mehreren Eigenwerten zu erzeugen. Ist eine große Zahl an Intervallen mit mehreren Eigenwerten erzeugt und enthalten dann die ersten Gruppen ein zusätzliches

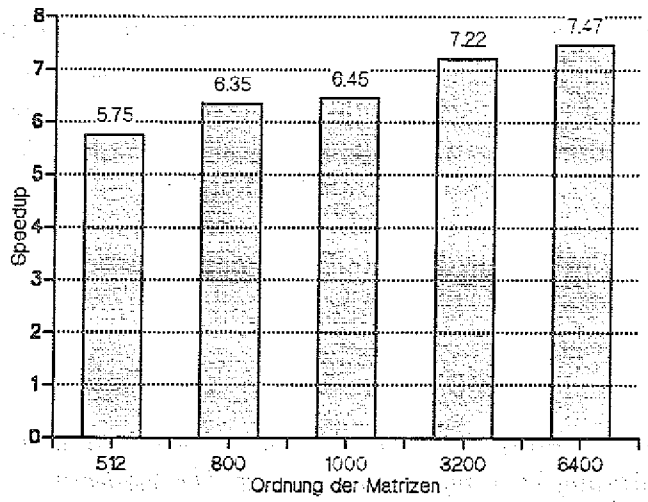


Abb. 6.19: Isolationsphase: Speedup-Werte bei Matrizen unterschiedlicher Ordnung, Matrizen An

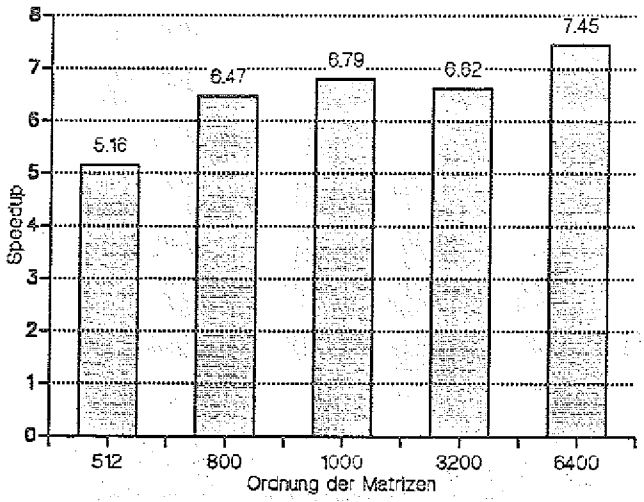


Abb. 6.20: Isolationsphase: Speedup-Werte bei Matrizen unterschiedlicher Ordnung, Matrizen Zn

Element, so beeinflußt die daraus folgende geringfügig unterschiedliche Auslastung der Prozessoren das Speedup-Verhalten kaum. Ferner sind in den letzten Isolationsschritten häufig sehr wenige Eigenwerte zu isolieren. Die Isolation dieser Eigenwerte erfordert bei einem Bisektionsverfahren in der Regel mehr Schritte als bei einem Multisektionsverfahren, so daß sich eine unterschiedliche Auslastung der Prozessoren in den letzten Isolationsschritten stärker auswirkt. Dies macht sich gerade bei Matrizen niedriger Ordnung bemerkbar, da der zeitliche Anteil der ersten und letzten Isolationsschritte an der Gesamtzeit der Isolationsphase in der Regel größer ist als bei Matrizen höherer Ordnung. Bei sehr ungünstiger Verteilung der Eigenwerte, z.B. wenn alle Eigenwerte sehr nah zusammenliegen, gelten die obigen Aussagen nicht. In diesem Fall sind nur kleine Speedups durch Parallelisierung möglich.

Speedup-Messungen bei 1 bis 8 Prozessoren

Abbildung 6.21 zeigt die Speedup-Werte auf CRAY Y-MP mit 1 bis 8 Prozessoren bei Bisektion mit Partitionierung in 1 bis 8 Gruppen in der Isolationsphase für die Matrix $A3200$. Zur Speedup-Berechnung wurde die sequentielle Ausführungszeit durch die parallele dividiert. Sequentiell und parallel entsprach die Anzahl der Gruppen der Zahl der für die parallele Ausführung angeforderten Prozessoren.

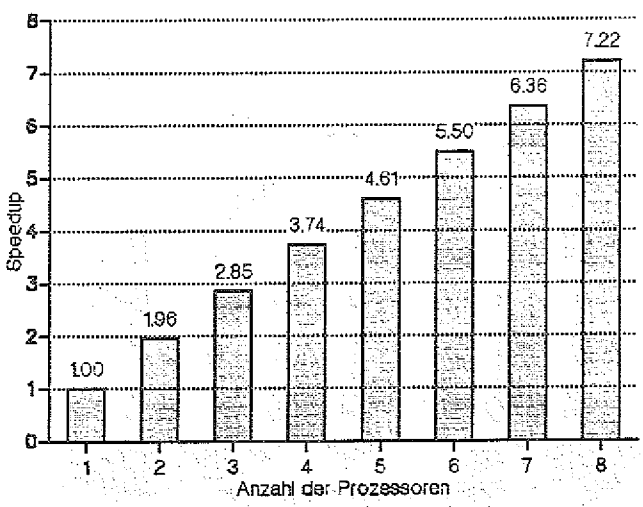


Abb. 6.21: Isolationsphase: Speedup-Werte bei 1 bis 8 Prozessoren, Matrix $A3200$

In Abbildung 6.21 liegen die Speedup-Werte desto näher am Maximalwert, je kleiner die Anzahl der Prozessoren ist. Bei wenigen Prozessoren und der entsprechenden Anzahl an

Gruppen lassen sich die Intervalle mit mehreren Eigenwerten eines Isolationsschritts günstiger verteilen und damit die Prozessoren besser auslasten. Speedup-Verluste haben ansonsten die gleichen Ursachen wie die im vorherigen Abschnitt erwähnten. Bei einem Prozessor und entsprechend einer Gruppe ist der Speedup tatsächlich eins, da eine Schleife mit einem Durchlauf nicht parallelisiert werden kann und dies auch zur Laufzeit durch die Parallelisierungroutinen erkannt wird. In den Routinen wird die Anzahl der Schleifendurchläufe abgefragt. Liegt nur ein Schleifendurchlauf vor, werden keine weiteren Operationen zur Parallelisierung der Schleife durchgeführt [26].
Die Speedup-Werte in Abbildung 6.22 wurden durch Division der sequentiellen Ausführungszeit bei einer Gruppe durch die parallele Ausführungszeit bei der Anzahl an Gruppen berechnet, die jeweils der Anzahl der angeforderten Prozessoren entsprach. Es liegt hier also im Sequentiellen ein anderes Verfahren als im Parallelen vor, wenn zur parallelen Bearbeitung mehr als ein Prozessor angefordert wird. Das sequentielle und das parallele Verfahren sind jedoch sehr ähnlich.

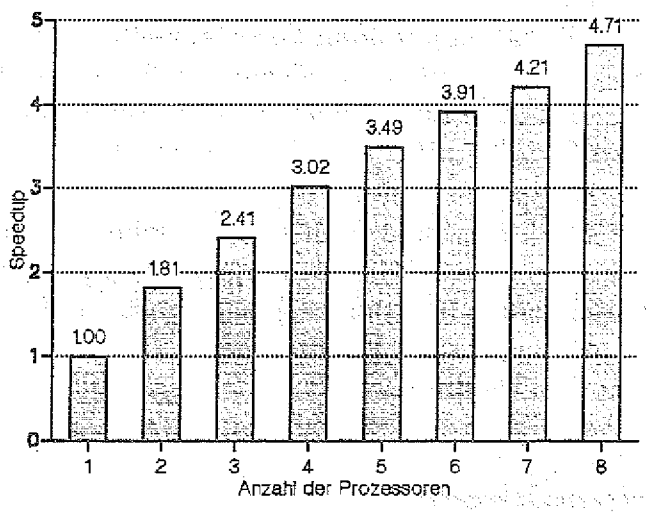


Abb. 6.22: Isolationsphase: Speedup-Werte bei 1 bis 8 Prozessoren, Matrix $A3200$

Die Speedup-Werte in Abbildung 6.22 liegen bei mehr als einem Prozessor deutlich unter dem Maximalwert, der durch die Anzahl der Prozessoren gegeben ist. Dies ist durch die größeren Vektorlängen bei Bisektion ohne Partitionierung gegenüber Bisektion mit Partitionierung in eine der Prozessoranzahl entsprechende Anzahl an Gruppen zu erklären. Bei 2 und mehr Gruppen wird Beschleunigung durch Vektorisierung eingebüßt. Es liegt der Fall vor, daß der beste hier untersuchte sequentielle Algorithmus nicht ohne leichte

Änderungen des Verfahrens parallelisiert werden kann. Eine mögliche Konsequenz des Speedup-Verhaltens in Abbildung 6.22 wäre, zur Parallelisierung der Isolationsphase nicht alle Prozessoren anzufordern (z.B. nur 4 oder 5). Die übrigen Prozessoren könnten dann im Multiprogramming-Betrieb anderweitig genutzt werden. Die durch Parallelisierung erreichbaren Speedups werden jedoch höher, je größer die Ordnung der Matrix ist. Bei Matrix $A6400$ wird z.B. bei Einsatz von 8 Prozessoren ein Speedup von 5.79 gegenüber dem sequentiellen Verfahren ohne Partitionierung erreicht (Matrix $Z6400$: 5.96). Sollten Matrizen sehr hoher Ordnung vorliegen, so kann sich auch der Einsatz aller Prozessoren ohne große Speedup-Verluste lohnen.

6.3.9 Zusammenfassung

Zusammenfassend läßt sich sagen, daß Bisektion mit Partitionierung in Gruppen sequentiell und parallel das schnellste der hier untersuchten Verfahren für die Isolationsphase darstellt. Da dieses Verfahren die Folge (2.14) verwendet, kann in der Isolationsphase ohne Geschwindigkeitsverluste auf die Verwendung der Sturmschen Kette, die leicht zu Zahlenbereichsüberschreitungen führt, verzichtet werden.
Multisektion hat gegenüber Bisektion nur dann Vorteile, wenn die Anzahl der Multisektionspunkte für die vorliegende Verteilung der Eigenwerte günstig ist. Ohne Kenntnis der Verteilung der Eigenwerte ist diese Anzahl jedoch vor der Programmausführung nicht abzuschätzen. Bisektion stellt dagegen für jede Verteilung eine gute „mittlere" Lösung dar, da dieses Verfahren sehr effizient zu codieren ist.
Die Parallelisierung der Bisektions- und Multisektionsverfahren ohne Aufteilung der Intervalle mit mehreren Eigenwerten in Gruppen ergibt günstigere Speedup-Werte als die von Bisektion mit Partitionierung in Gruppen. Die kurzen Laufzeiten von Bisektion mit Partitionierung in Gruppen gerade bei Matrizen hoher Ordnung werden jedoch von den übrigen Verfahren bei weitem nicht erreicht.

6.4 Extraktionsphase

6.4.1 Auswirkungen der Isolationsphase auf die Extraktionsphase

Multisektion der Ordnung $k > 1$ in der Isolationsphase führt zu einem anderen Zeitverhalten der Extraktionsphase als Bisektion in der Isolationsphase. Bei Multisektion in der Isolationsphase werden in der Regel Intervalle kleinerer Intervallbreite an die Extraktionsphase übergeben als bei Bisektion in der Isolationsphase. Die Intervallbreite dieser Intervalle, die genau einen Eigenwert enthalten, bestimmt die Anzahl der Iterationsschritte

in der Extraktionsphase. Bei kleinerer Intervallbreite sind in der Extraktionsphase weniger Iterationsschritte erforderlich, um die vorgegebene Genauigkeit zu erreichen. Wird Bisektion statt Multisektion mit 15 Punkten in der Isolationsphase durchgeführt, so ist in der Extraktionsphase beim Pegasusverfahren in der Regel ein zusätzlicher Iterationsschritt notwendig. Bisektion in der Extraktionsphase erfordert ca. 4 zusätzliche Iterationsschritte. Die Auswirkungen dieser zusätzlichen Iterationsschritte auf das Zeitverhalten der Extraktionsphase sind jedoch nicht sehr groß, da bei dem vorgegebenen Genauigkeitskriterium (s. (5.8) mit $\delta_f = 10^{-14}$) beim Pegasusverfahren, wenn vorher 10 Bisektionsschritte durchgeführt werden, um bessere Startwerte für dieses Verfahren zu erhalten, insgesamt ungefähr 15 Iterationsschritte notwendig sind, bei Bisektion bis zu 40 Iterationsschritte. Die Zahl der Iterationsschritte schwankt bei den einzelnen Intervallen mit genau einem Eigenwert, da die Intervallbreite dieser Intervalle von der Zahl der Isolationsschritte, die erforderlich sind, um die jeweiligen Eigenwerte zu isolieren, abhängt.

Um Auswirkungen der Isolationsphase auf das Zeitverhalten der Extraktionsphase richtig einschätzen zu können, muß auch der Anteil der Laufzeiten von Extraktionsphase und Isolationsphase an der Ausführungszeit des Gesamtprogramms berücksichtigt werden. Der zeitliche Aufwand der schnellsten Verfahren der Extraktionsphase - die einzelnen Verfahren der Extraktionsphase werden in den folgenden Abschnitten diskutiert - ist bei den in den Messungen verwendeten Matrizen 4 bis 9 mal höher als der der schnellsten Verfahren der Isolationsphase. Der zeitliche Anteil des Isolationsverfahrens an der Ausführungszeit des Gesamtprogramms steigt jedoch, wenn eine geringere Genauigkeit vorgegeben wird ($\delta_f > 10^{-14}$) und dadurch die Laufzeit des Extraktionsverfahrens verkürzt wird.

In den Abbildungen 6.23 und 6.24 werden die sequentiellen und parallelen Ausführungszeiten der Gesamtprogramme bei Bisektion mit Partitionierung in 8 Gruppen in der Isolationsphase und Multisektion mit 15 Punkten in der Isolationsphase verglichen. In der Initialisierungsphase wurde Multisektion mit 15 Punkten durchgeführt, in der Extraktionsphase Variante 6 mit Verwendung des Pegasusverfahrens. Die Ausführungszeiten der Gesamtprogramme wurden bei den Matrizen $A3200$, $A6400$, $Z3200$ und $Z6400$ gemessen.

Die Abbildungen 6.23 und 6.24 zeigen, daß sich bei Bisektion mit Partitionierung in Gruppen in der Isolationsphase deutlich kürzere Ausführungszeiten des Gesamtprogramms ergeben als bei Multisektion mit 15 Punkten in der Isolationsphase. Die Zeitgewinne durch das wesentlich schnellere Isolationsverfahren (s. auch 6.3.8) überwiegen die geringen Zeitverluste in der Extraktionsphase bei weiten. Die Isolationsverfahren aus 6.3.8 sind folglich die für den gesamten Algorithmus günstigsten der untersuchten Isolationsverfahren. Die etwas geringeren Zeitunterschiede bei Matrix $A3200$ sind dadurch zu erklären, daß 15 Multisektionspunkte bei dieser Matrix eine für die Verteilung der Eigenwerte günstige Anzahl sind (s. auch die Abbildungen 6.9 und 6.10).

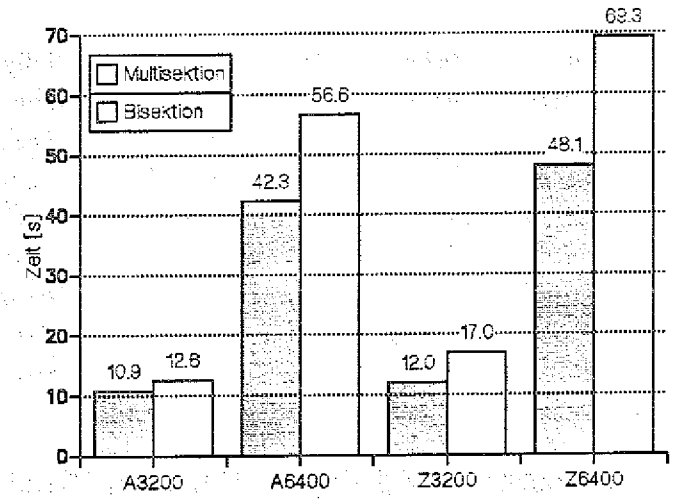


Abb. 6.23: Vergleich der Ausführungszeiten der Gesamtprogramme bei Multisektion mit 15 Punkten und Bisektion mit Partitionierung in 8 Gruppen in der Isolationsphase, sequentiell

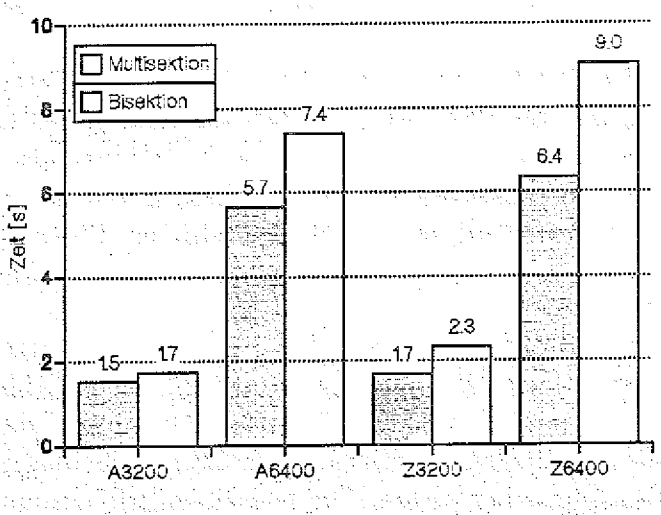


Abb. 6.24: Vergleich der Ausführungszeiten der Gesamtprogramme bei Multisektion mit 15 Punkten und Bisektion mit Partitionierung in 8 Gruppen in der Isolationsphase, parallel

6.4.2 Variante 1

Implementierung

In Variante 1 der Extraktionsphase werden 4 unterschiedliche Iterationsverfahren miteinander verglichen. Implementiert sind Multisektion mit frei wählbarer Anzahl der Multisektionspunkte, Bisektion mit Verwendung von Algorithmus 5.1, Bisektion mit Auswertung des charakteristischen Polynoms und das Pegasusverfahren. Bei Bisektion mit Auswertung des charakteristischen Polynoms werden zu Beginn der Extraktionsphase die Funktionswerte des charakteristischen Polynoms an den unteren Intervallgrenzen aller Intervalle, die genau einen Eigenwert enthalten, berechnet. Beim Pegasusverfahren werden die Funktionswerte des charakteristischen Polynoms an den unteren und oberen Intervallgrenzen ermittelt. Vor dem Pegasusverfahren werden, wie in allen anderen Varianten auch, 10 Bisektionsschritte durchgeführt, um günstigere Startwerte für dieses Verfahren zu erhalten. Die Funktionswerte des charakteristischen Polynoms werden durch das Produkt aller q_{n-i}, $i = 1, \ldots, n$, berechnet (s. auch 2.5).
Das Multisektionsverfahren vektorisiert nach Algorithmus 5.2, während die übrigen Iterationsverfahren nicht vektorisieren.
Im Hauptteil von Variante 1 der Extraktionsphase wird die Schleife über alle Intervalle mit genau einem Eigenwert (s. Abbildung 5.8, Mitte) mit Hilfe der Autotasking-Direktive DO ALL parallelisiert. Die Bearbeitung jedes Intervalls mit genau einem Eigenwert stellt eine Task dar. Die beiden Schleifen zur Bestimmung des skalierten, absoluten Fehlers sowie zum Eintragen der Eigenwerte in die Liste mit den extrahierten Eigenwerten (s. Abbildung 5.8, oben und unten) werden mit Hilfe der Direktive DO ALL VEKTOR parallelisiert. Der zeitliche Anteil dieser Schleifen an der Laufzeit der Extraktionsphase ist jedoch sehr gering, so daß auf ihre Parallelisierung auch verzichtet werden könnte.

Ermittlung der Performance-Halbwertslänge der Multisektionsschleife

Um die Anzahl an Multisektionspunkten abzuschätzen, bei der sich die kürzesten Laufzeiten des Multisektionsverfahrens in der Extraktionsphase nach Variante 1 ergeben, wurden die Performance-Halbwertslängen $n_{1/2}$ der inneren Multisektionsschleife in Algorithmus 5.2 auf CRAY X-MP und CRAY Y-MP ermittelt. Die Bearbeitung der inneren und äußeren Schleife in Algorithmus 5.2 hat den höchsten zeitlichen Anteil an der gesamten Laufzeit des Multisektionsverfahrens. Zur Abschätzung der Performance-Halbwertslängen wurde die lineare Näherung aus 5.2 verwendet. Durch Umformung erhählt man die Gleichung

$$t = t_e(k + \frac{t_0}{t_e}) \text{ mit } \frac{t_0}{t_e} = n_{1/2}. \tag{6.6}$$

(6.6) ist zu entnehmen, daß $k = -n_{1/2}$ für $t = 0$ gilt, die Performance-Halbwertslänge also dem negierten x-Achsenabschnitt entspricht. In den Abbildungen 6.25 und 6.26 sind die

Laufzeiten der inneren Schleife aus Algorithmus 5.2 bei den Vektorlängen (k) 10 bis 50 in 5-er Schritten aufgetragen. Durch die Meßpunkte wurde eine Ausgleichgerade gelegt (lineare Regression) und der Schnittpunkt dieser Geraden mit der x-Achse ermittelt.

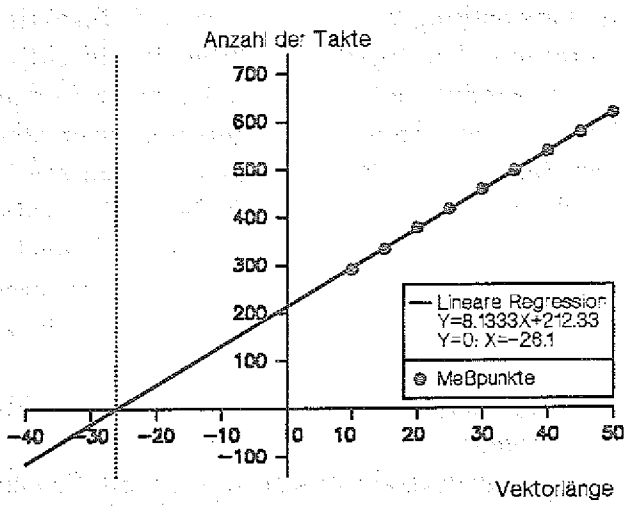


Abb. 6.25: Abschätzung der Performance-Halbwertslänge, X-MP

Die Abschätzung der Performance-Halbwertslänge für die innere Multisektionsschleife in Algorithmus 5.2 ergibt für CRAY X-MP ungefähr 26 (s. Abbildung 6.25) und für CRAY Y-MP ungefähr 30 (s. Abbildung 6.26). Mit Hilfe der Tabelle 5.1 läßt sich die jeweils optimale Anzahl der Multisektionspunkte abschätzen. Für CRAY Y-MP erhält man 15 oder 16 Multisektionspunkte, für CRAY X-MP 13 oder 14 Multisektionspunkte. In den folgenden Messungen wird u. a. untersucht, ob diese Werte gute Abschätzungen für das gesamte Multisektionsverfahren der Extraktionsphase sind.

Multisektion oder Bisektion in der Extraktionsphase?

In diesem Abschnitt wird das Zeitverhalten von Multisektion und Bisektion in der Extraktionsphase untersucht. Zu diesem Zweck wurden die Ausführungszeiten des Multisektionsverfahrens mit 1 bis 40 Punkten bei den Matrizen $A512$ und $Z512$ sowie mit 1 bis 127 Punkten bei den Matrizen $A128$ und $Z128$ ermittelt. Die Anzahl der Multisektionspunkte, die die jeweils kürzeste Ausführungszeit ergab, wurde festgehalten.
In den Abbildungen 6.27 und 6.28 werden die sequentiellen und parallelen Ausführungszeiten von Multisektion mit jeweils optimaler Anzahl der Multisektionspunkte und von

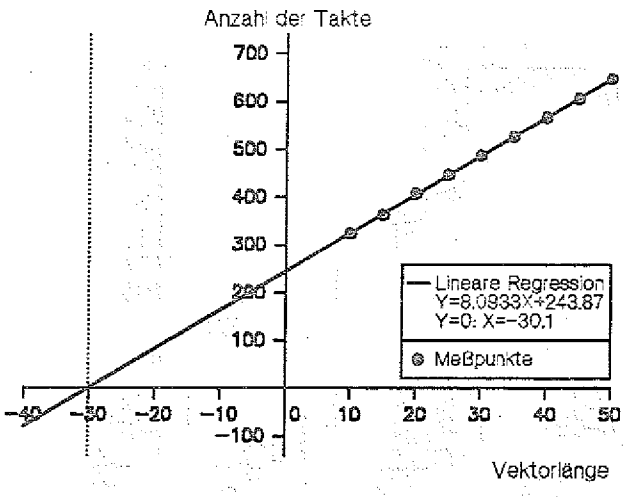


Abb. 6.26: Abschätzung der Performance-Halbwertslänge, Y-MP

Bisektion auf CRAY X-MP bei den Matrizen $A128$, $A512$, $Z128$ und $Z512$ verglichen. Die jeweils optimale Anzahl der Multisektionspunkte ist unter der x-Achse vermerkt. In der Initialisierungsphase und in der Isolationsphase wurde, wie auch bei allen Zeitmessungen der Extraktionsphase in den folgenden Varianten, Multisektion mit 15 Punkten durchgeführt.

Die Abbildungen 6.27 und 6.28 zeigen, daß auf CRAY X-MP die sequentiellen und parallelen Ausführungszeiten von Bisektion bei allen Matrizen geringfügig kürzer sind als die von Multisektion mit optimaler Punktzahl. Dies läßt sich durch die effiziente Codierung des Bisektionsverfahrens erklären. Der zeitliche Aufwand des Multisektionsverfahrens ist auf CRAY X-MP höher, obwohl gegenüber Bisektion Iterationsschritte eingespart werden und die Auswertung der Folge (2.14) nach Algorithmus 5.2 vektorisiert. Die höhere Anzahl an Auswertungen der Folge (2.14) bei Multisekton führt trotz der Vektorisierung zu höheren Ausführungszeiten als bei Bisektion. Bisektion in der Extraktionsphase ist auf CRAY X-MP also das etwas günstigere Verfahren als Multisektion. Als optimale Anzahl der Multisektionspunkte ergeben sich bei sequentieller Programmausführung 12, 13 und 14 Punkte. Die Abschätzung aus dem vorherigen Abschnitt von 13 oder 14 Punkten stimmt also ziemlich genau mit den gemessenen Werten überein. Bei den Matrizen $A128$ und $A512$ sind bei paralleler Programmausführung 13 bzw. 16 Multisektionspunkte optimal. Die Unterschiede zu den bei sequentieller Ausführung ermittelten optimalen Punktzahlen sind durch eine unterschiedliche Auslastung der Prozessoren bei der Bearbeitung der letzten Intervalle mit genau einem Eigenwert zu erklären. Zur Extraktion der Eigenwerte kann

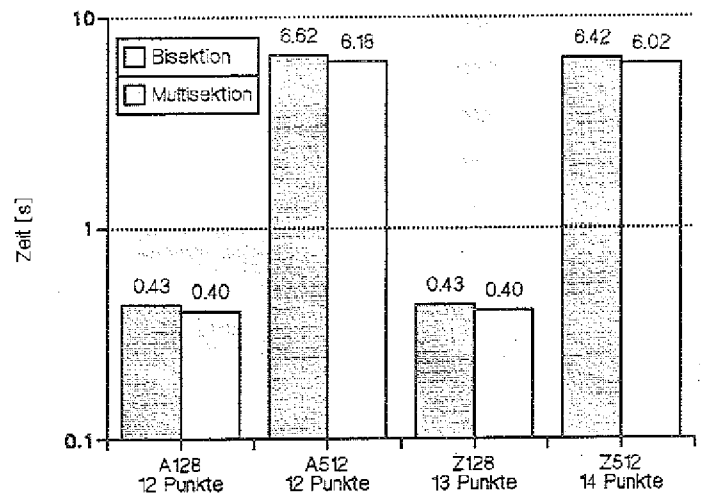


Abb. 6.27: Extraktionsphase, Variante 1: Vergleich zwischen Bisektion und Multisektion mit optimaler Punktzahl, X-MP, sequentiell

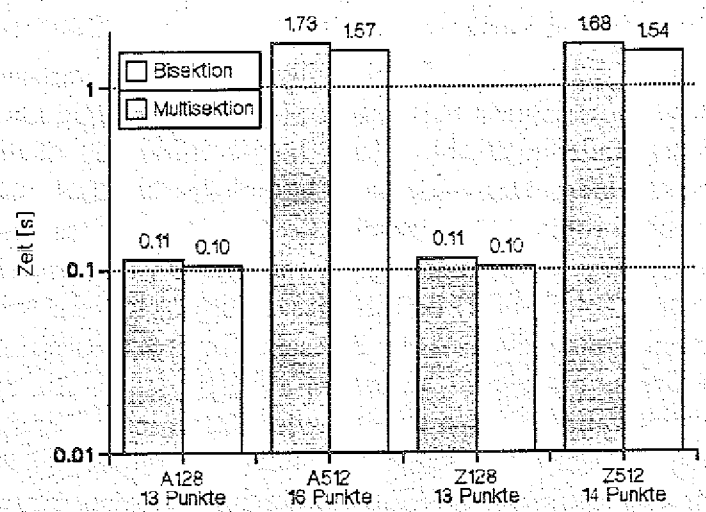


Abb. 6.28: Extraktionsphase, Variante 1: Vergleich zwischen Bisektion und Multisektion mit optimaler Punktzahl, X-MP, parallel

eine unterschiedliche Anzahl an Multisektionsschritten erforderlich sein. Dies hängt zum einen davon ab, in welchem Isolationsschritt die Eigenwerte isoliert wurden, zum anderen von der Zahl der Multisektionspunkte. Bei einer höheren Anzahl der Multisektionspunkte ist unter Umständen eine günstigere Auslastung der Prozessoren bei der Bearbeitung der letzten Intervalle mit genau einem Eigenwert zu erreichen, wenn sich die Anzahl der Multisektionsschritte bei diesen Intervallen und dieser Anzahl der Multisektionspunkte wenig oder gar nicht unterscheidet. Dies kann bei paralleler Ausführung zu geringfügig kürzerer Laufzeit der gesamten Extraktionsphase führen.
Die Abbildungen 6.29 und 6.30 zeigen die Ergebnisse der entsprechenden Messungen auf CRAY Y-MP.

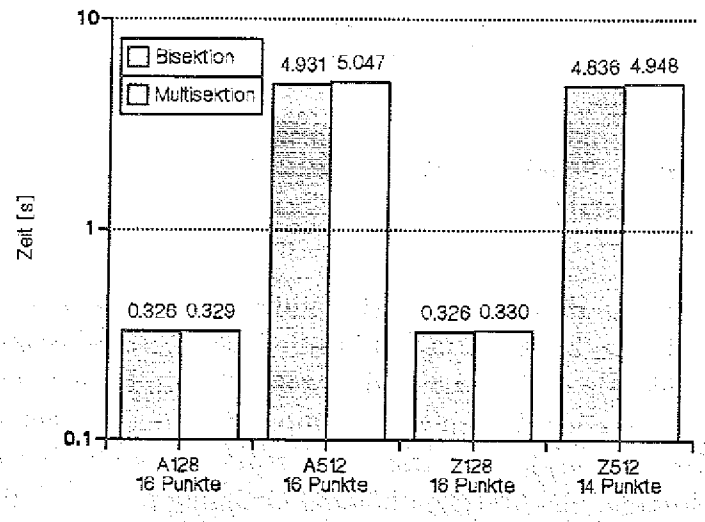


Abb. 6.29: **Extraktionsphase, Variante 1: Vergleich zwischen Bisektion und Multisektion mit optimaler Punktzahl, Y-MP, sequentiell**

Die sequentiellen Ausführungszeiten auf CRAY Y-MP in Abbildung 6.29 sind bei Multisektion mit optimaler Punktzahl geringfügig besser als bei Bisektion. Als optimale Anzahl der Multisektionpunkte ergaben sich dreimal 16 und einmal 14 Punkte. Dies stimmt gut mit der Abschätzung aus dem vorherigen Abschnitt (15 oder 16 Punkte) überein. Die jeweils optimale Anzahl der Multisektionspunkte ist wegen der größeren Performance-Halbwertslänge (s. vorheriger Abschnitt) höher als auf CRAY X-MP. Ferner zeigt sich bei den sequentiellen Messungen, daß auf CRAY Y-MP im Gegensatz zu CRAY X-MP Multisektion mit optimaler Punktzahl bei sequentieller Ausführung geringfügig günstiger als Bisektion ist. Bei paralleler Ausführung liegen die Ausführungszeiten bei Bisektion

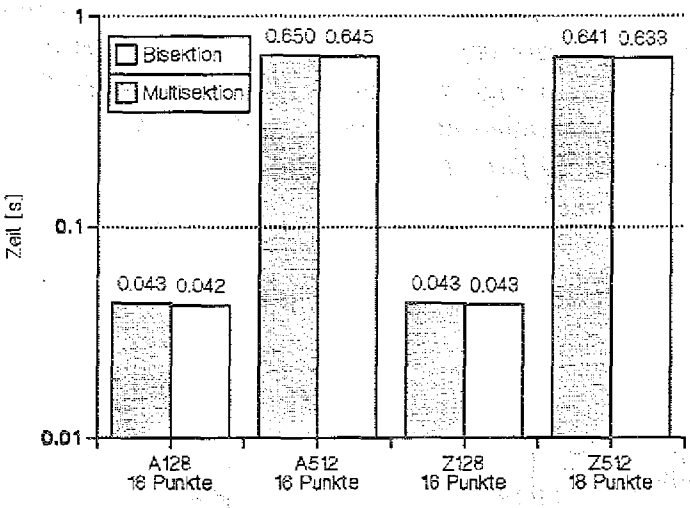


Abb. 6.30: Extraktionsphase, Variante 1: Vergleich zwischen Bisektion und Multisektion mit optimaler Punktzahl, Y-MP, parallel

geringfügig unter denen von Multisektion (s. Abbildung 6.30). Diese den Ergebnissen aus Abbildung 6.29 gegenläufige Tendenz ist durch unterschiedliche Auslastung der Prozessoren bei der Bearbeitung der letzten Intervalle mit genau einem Eigenwert zu erklären. Eine unterschiedliche Anzahl an aufwendigen Multisektionsschritten wirkt sich auf das Zeitverhalten der gesamten Extraktionsphase in der Regel stärker aus als eine unterschiedliche Anzahl an Bisektionsschritten. Bei Matrizen höherer Ordnung wirkt sich die unterschiedliche Auslastung der Prozessoren bei der Bearbeitung der letzten Intervalle mit genau einem Eigenwert weniger auf das Zeitverhalten der gesamten Extraktionsphase aus. In diesem Fall müßte Multisektion auch bei paralleler Ausführung zu geringfügig kürzeren Laufzeiten als Bisektion führen. Um dies zu bestätigen, werden in Abbildung 6.31 die sequentiellen und parallelen Ausführungszeiten von Multisektion mit 16 Punkten und von Bisektion auf CRAY Y-MP bei den Matrizen $A800$, $A1000$ und $A3200$ verglichen.

In Abbildung 6.31 sind die sequentiellen und parallelen Ausführungszeiten von Multisektion mit 16 Punkten geringfügig kürzer als die von Bisektion. Dies bestätigt, daß auf CRAY Y-MP Multisektion mit optimaler Punktzahl im Gegensatz zu CRAY X-MP etwas günstiger als Bisektion ist.

Zusammenfassend ist zu sagen, daß sich die Ausführungszeiten von Bisektion und Multisektion mit optimaler Punktzahl sowohl auf CRAY X-MP als auch auf CRAY Y-MP wenig unterscheiden.

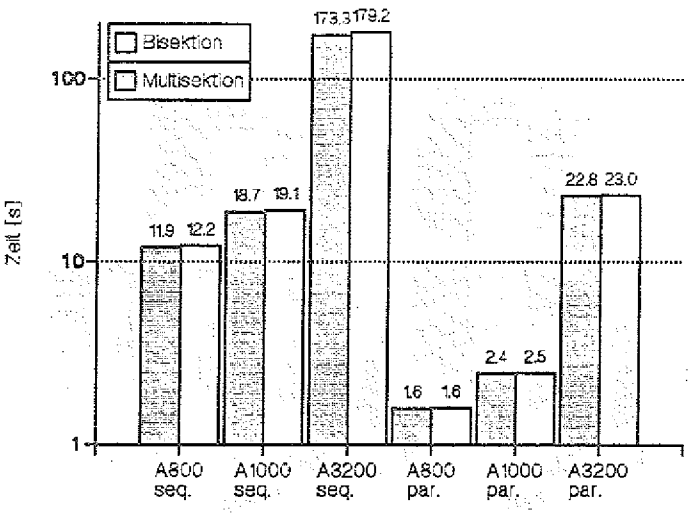


Abb. 6.31: Extraktionsphase, Variante 1: Vergleich zwischen Bisektion und Multisektion mit 16 Punkten, Y-MP, sequentiell und parallel

Auf beiden Rechnern sind diese Iterationsverfahren als gleichwertig zu beurteilen.

Vergleich der 4 implementierten Iterationsverfahren

In diesem Abschnitt werden Multisektion mit optimaler Punktzahl, Bisektion, Bisektion mit Auswertung des charakteristischen Polynoms und das Pegasusverfahren in ihrem Zeitverhalten verglichen. In den Abbildungen 6.32 und 6.33 sind die sequentiellen und parallelen Ausführungszeiten dieser Iterationsverfahren auf CRAY Y-MP bei den Matrizen $A512$ und $Z512$ dargestellt.
In den Abbildungen 6.32 und 6.33 liegen die sequentiellen und parallelen Laufzeiten von Bisektion mit Auswertung des charakteristischen Polynoms (Bisek. m. F.) deutlich unter denen von Bisektion und Multisektion mit optimaler Punktzahl. Bisektion mit Auswertung des charakteristischen Polynoms spart gegenüber diesen Verfahren den Aufwand, der sich durch das Zählen der negativen Vorzeichen in den Gliedern der Folge (2.14) ergibt. Stattdessen wird das Produkt aller q_{n-i} gebildet (s. auch 2.5). Dies ist zeitlich weniger aufwendig. Das Pegasusverfahren ist das bei weitem schnellste Verfahren, da es als Iterationsverfahren mit superlinearer Konvergenz (s. auch 2.6) die wenigsten Iterationsschritte benötigt. Ein Iterationsschritt des Pegasusverfahrens ist nur wenig aufwendiger als ein Bisektionsschritt. Wie den Abbildungen 6.32 und 6.33 zu entnehmen ist, betragen die sequentiellen und parallelen Ausführungszeiten des Pegasusverfahrens ungefähr ein Drittel

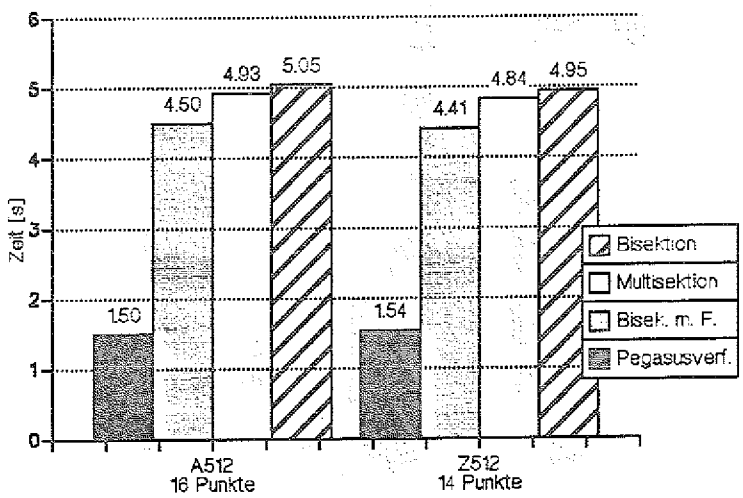


Abb. 6.32: Extraktionsphase, Variante 1: Vergleich der 4 implementierten Verfahren, sequentiell

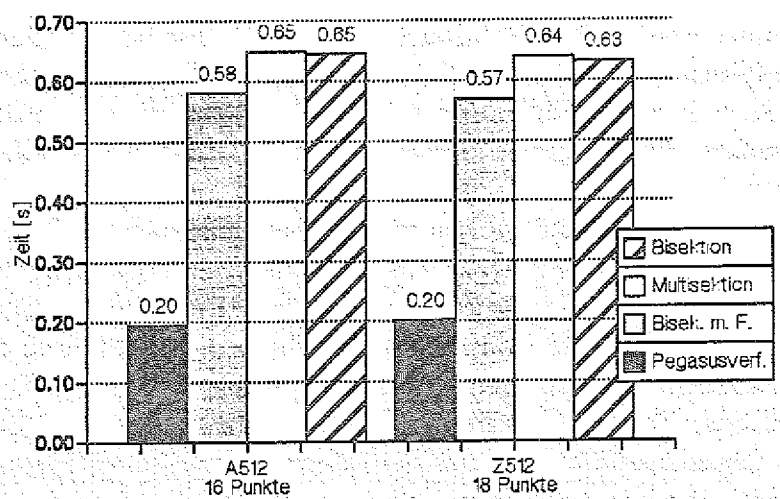


Abb. 6.33: Extraktionsphase, Variante 1: Vergleich der 4 implementierten Verfahren, parallel

der Ausführungszeiten von Bisektion mit Auswertung des charakteristischen Polynoms. Wird der Funktionswert des charakteristischen Polynoms berechnet, so sind Zahlenbereichsüberschreitungen möglich, da sich unter Umständen sehr hohe Werte ergeben. Dies ist z.B. bei den Matrizen $A10000$ und $Z10000$ der Fall. In 6.6 wird auf eine Möglichkeit eingegangen, Zahlenbereichsüberschreitungen bei der Berechnung des Funktionswertes des charakteristischen Polynoms nach 2.5 zu verhindern.

Speedup-Werte bei Matrizen unterschiedlicher Ordnung

In Abbildung 6.34 sind die Speedup-Werte von Bisektion in der Extraktionsphase auf CRAY Y-MP mit 8 Prozessoren bei den Matrizen $A128$, $A512$, $A800$, $A1000$ und $A3200$ dargestellt.

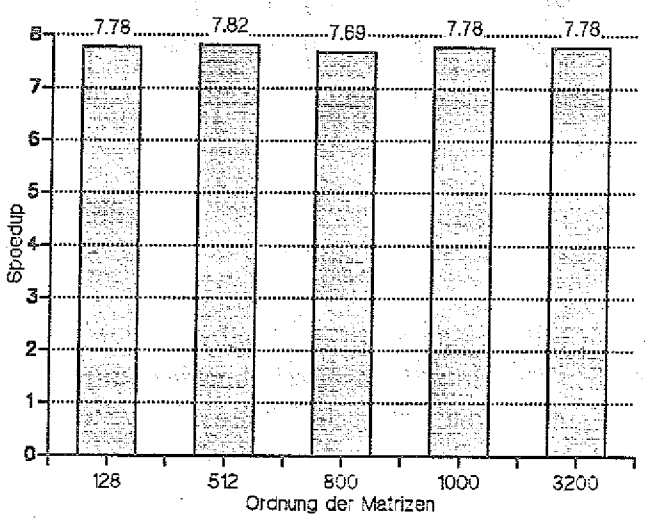


Abb. 6.34: **Extraktionsphase, Variante 1: Speedup-Werte bei Matrizen unterschiedlicher Ordnung, Matrizen An**

Die Speedup-Werte in Abbildung 6.34 liegen bei allen Matrizen nahe am Maximalwert 8. Speedup-Verluste treten durch den Overhead durch die Parallelisierungsroutinen und durch unterschiedliche Auslastung der Prozessoren bei der Bearbeitung der letzten Intervalle mit genau einem Eigenwert auf. Da auch die übrigen Iterationsverfahren von Variante 1 der Extraktionsphase auf die gleiche Weise wie das Bisektionsverfahren parallelisiert sind, ergeben sich ähnliche Speedup-Werte.

Speedup-Werte bei 1 bis 8 Prozessoren

Abbildung 6.35 zeigt die Speedup-Werte des Pegasusverfahrens auf CRAY Y-MP mit 1 bis 8 Prozessoren bei Matrix $A512$.

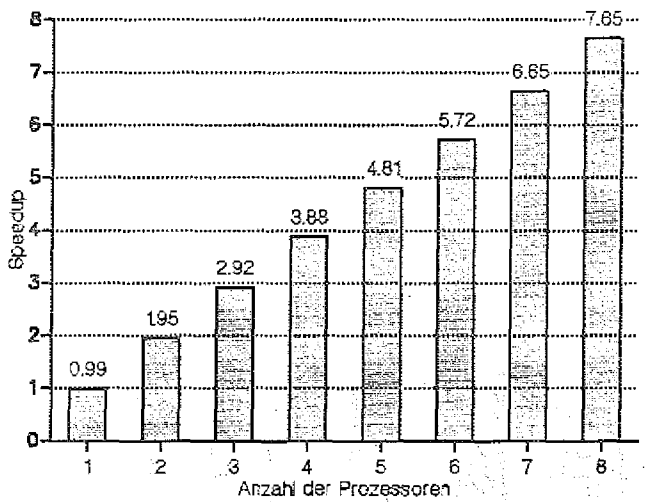


Abb. 6.35: Extraktionsphase, Variante 1: Speedup-Werte bei 1 bis 8 Prozessoren, Matrix $A512$

Die Speedup-Werte liegen jeweils nahe am Maximalwert, der der Anzahl der angeforderten Prozessoren entspricht. Speedup-Verluste ergeben sich durch den Overhead durch die Parallelisierungsroutinen und durch unterschiedliche Auslastung der Prozessoren bei der Bearbeitung der letzten Intervalle mit genau einem Eigenwert.

6.4.3 Variante 2

Implementierung

In Variante 2 der Extraktionsphase sind Bisektion, Bisektion mit Auswertung des charakteristischen Polynoms und das Pegasusverfahren implementiert. Statt der Folge (2.14) wie in Variante 1 wird hier die Sturmsche Kette verwendet. Die Auswertung der Sturmschen Kette geschieht mit Hilfe der vektorisierenden CRAY-Routinen SOLR und SOLRN. SOLRN liefert lediglich das Endergebnis einer linearen Rekursion zweiter Ordnung, während SOLR auch alle Zwischenergebnisse an das die Routine aufrufende Programm zurückgibt. Bei Bisektion wird die Routine SOLR verwendet, da die Zahl der

Vorzeichenwechsel in den Folgengliedern der Sturmschen Kette bestimmt wird. Bisektion mit Funktionsauswertung und das Pegasusverfahren benötigen dagegen lediglich den Funktionswert des charakteristischen Polynoms. Daher ist in diesen Verfahren die Routine SOLRN implementiert. Ansonsten stimmt die Implementierung von Variante 2 mit der von Variante 1 überein.

Vergleich mit Variante 1

In den Abbildungen 6.36 und 6.37 werden die sequentiellen und parallelen Ausführungszeiten der Verfahren von Variante 1 und von Variante 2 auf CRAY Y-MP bei den Matrizen $A512$ und $Z512$ verglichen. Die Ausführungszeiten der Verfahren von Variante 1 und der entsprechenden Verfahren von Variante 2 sind in Paaren nebeneinander dargestellt. Der linke Balken entspricht jeweils der Ausführungszeit des Verfahrens von Variante 1, der rechte der des Verfahrens von Variante 2. In der Legende wird auf die jeweilige Variante der Extraktionsphase (V. 1 oder V. 2) und das Iterationsverfahren hingewiesen (Pegasusverfahren: Pegasus., Bisektion mit Auswertung des charakteristischen Polynoms: Bi. m. F., Bisektion: Bisek.).

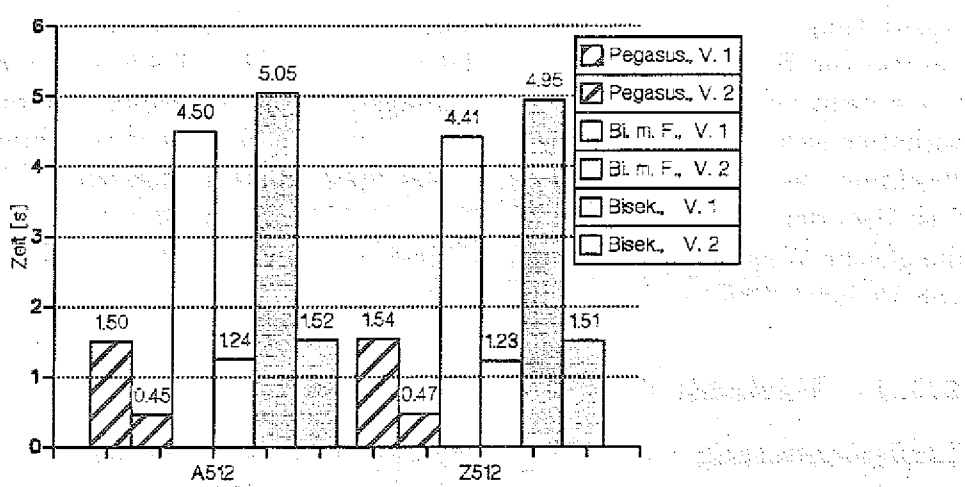


Abb. 6.36: **Extraktionsphase, Variante 2: Vergleich der 3 implementierten Verfahren mit den entsprechenden Verfahren von Variante 1, sequentiell**

Die Abbildungen 6.36 und 6.37 zeigen, daß die Verfahren von Variante 2 mehr als dreimal schneller sind als die entsprechenden Verfahren von Variante 1. Die Vektorisierung der

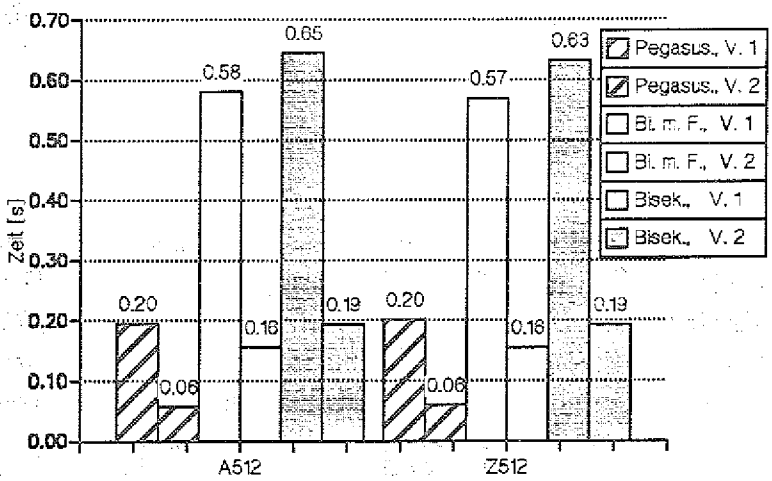


Abb. 6.37: Extraktionsphase, Variante 2: Vergleich der 3 implementierten Verfahren mit den entsprechenden Verfahren von Variante 1, parallel

Auswertung der Sturmschen Kette mit Hilfe der CRAY-Routinen SOLR bzw. SOLRN bewirkt die Beschleunigung der Verfahren von Variante 2 gegenüber denen von Variante 1, die nicht vektorisieren. Die Ausführungszeiten der einzelnen Verfahren von Variante 2 verhalten sich zueinander wie die von Variante 1. Das Pegasusverfahren ist das schnellste, Bisektion das langsamste Verfahren. Das Speedup-Verhalten der Verfahren von Variante 2 ist dem der Verfahren von Variante 1 sehr ähnlich, da die Verfahren von Variante 2 auf die gleiche Weise parallelisiert sind. Daher wird auf das Speedup-Verhalten der Verfahren von Variante 2 hier nicht weiter eingegangen.

6.4.4 Variante 3

Implementierung

In Variante 3 der Extraktionsphase sind Bisektion, Bisektion mit Auswertung des charakteristischen Polynoms und das Pegasusverfahren implementiert. Alle inneren Schleifen dieser Verfahren laufen über alle Intervalle, die genau einen Eigenwert enthalten und vektorisieren (s. Abbildung 5.9). Zur parallelen Ausführung auf CRAY Y-MP mit 8 Prozessoren werden alle inneren Schleifen mit Hilfe der Autotasking-Direktive DO PARALLEL NUMCHUNKS(8) parallelisiert. Die gesamte Extraktionsphase nach Variante 3 ist in einen parallelen Bereich eingeschlossen. Durch die Verwendung der Option NUMCHUNKS(8)

liegt bei paralleler Programmausführung ein anderer Algorithmus vor als bei sequentieller Programmausführung. Beide Algorithmen sind aber sehr ähnlich. Bei paralleler Programmausführung werden die Schleifendurchläufe aller inneren Schleifen in 8 möglichst gleich große Gruppen aufgeteilt. Die Aufteilung geschieht genauso wie in 6.3.8 bei Bisektion mit Partitionierung in 8 Gruppen beschrieben. Stehen alle 8 Prozessoren von CRAY Y-MP zur Verfügung, so übernimmt jeder die Bearbeitung einer Gruppe. Es ist zu beachten, daß auf diese Weise die Vektorlängen bei der Vektorverarbeitung aller inneren Schleifen ca. achtmal kürzer sind als bei sequentieller Programmausführung. Dies führt zu Speedup-Verlusten.

Bei Bisektion sind drei verschiedene Verfahren zur Bestimmung des Abbruchkriteriums des Iterationsverfahrens implementiert. Im ersten Verfahren wird die Anzahl der Bisektionsschritte für jedes Intervall, das genau einen Eigenwert enthält, nach (5.9) berechnet. Das Maximum dieser Werte gibt an, wieviele Iterationsschritte durchgeführt werden müssen, um alle Eigenwerte bis auf den skalierten, absoluten Fehler genau zu berechnen. Daher ist in diesem Verfahren eine Zählschleife von 1 bis zur maximalen Anzahl der Bisektionsschritte implementiert. Während im ersten Verfahren die Berechnung der maximalen Anzahl der Bisektionsschritte sequentiell mit Hilfe der vektorisierenden CRAY-Routine ISMAX erfolgt, wird diese Berechnung in dem zweiten Verfahren durch Partitionierung parallelisiert. Das Feld, das die jeweilige Anzahl der Iterationsschritte aller Intervalle mit genau einem Eigenwert enthält, wird in 8 Teile zerlegt und jeder Teil mit Hilfe der Routine ISMAX bearbeitet. Der größte der 8 Maximalwerte der Teilfelder ist die maximale Anzahl der Bisektionsschritte. Im dritten Verfahren ist eine until-Schleife implementiert, die beendet ist, wenn alle Eigenwerte extrahiert sind. Die aktuelle Anzahl der extrahierten Eigenwerte wird in einem Zähler festgehalten (s. Abbildung 5.9). Das Inkrementieren dieses Zählers, der bei paralleler Ausführung allen Tasks gemeinsam ist, geschieht in einem kritischen Bereich, der von mehreren Tasks nicht gleichzeitig bearbeitet werden kann (Autotasking-Direktiven GUARD und END GUARD).

Vergleich der Bisektionsverfahren mit unterschiedlicher Bestimmung des Abbruchkriteriums

In Abbildung 6.38 werden die sequentiellen und parallelen Ausführungszeiten der 3 oben genannten Bisektionsverfahren mit unterschiedlicher Bestimmung des Abbruchkriteriums auf CRAY Y-MP bei den Matrizen $A6400$ und $Z6400$ verglichen.

Abbildung 6.38 zeigt, daß sich bei allen 3 Verfahren ungefähr gleiche Ausführungszeiten ergeben. Alle drei Verfahren sind also als gleichwertig zu beurteilen. Ferner ist zu sagen, daß der zeitliche Anteil der Bestimmung des Abbruchkriteriums an der Ausführungszeit der Extraktionsphase gering ist.

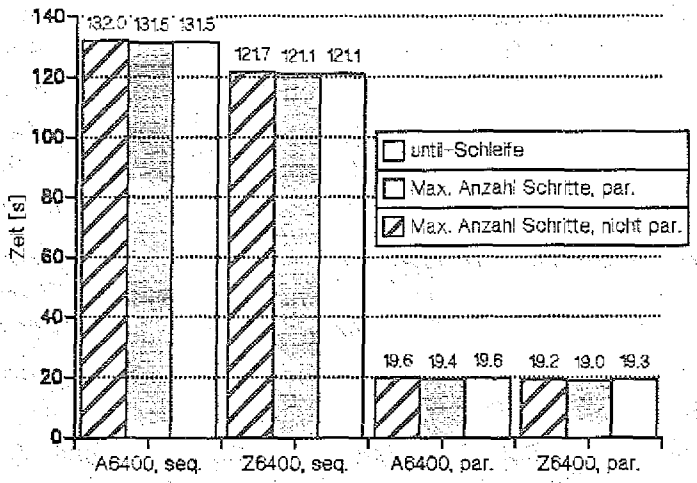


Abb. 6.38: Extraktionsphase, Variante 3: Vergleich dreier Bisektionsverfahren mit unterschiedlicher Bestimmung des Abbruchkriteriums, sequentiell und parallel

Vergleich mit Variante 2

In den Abbildungen 6.39 und 6.40 werden die sequentiellen und parallelen Ausführungszeiten von Bisektion, Bisektion mit Auswertung des charakteristischen Polynoms und des Pegasusverfahrens von Variante 3 mit denen der entsprechenden Verfahren von Variante 2 auf CRAY Y-MP bei den Matrizen $A3200$ und $Z3200$ verglichen.
In den Abbildungen 6.39 und 6.40 liegen die Ausführungszeiten von Variante 3 mit Ausnahme der parallelen Ausführungszeiten des Pegasusverfahrens deutlich unter denen von Variante 2. Dies zeigt, daß die Vektorierung der inneren Schleifen von Variante 3 günstiger ist als die Vektorisierung der Sturmschen Kette mit Hilfe der CRAY-Routinen SOLR bzw. SOLRN. Die parallele Ausführungszeit des Pegasusverfahrens von Variante 3 ist bei Matrix $A3200$ etwas höher als die von Variante 2, bei Matrix $Z3200$ sind beide Zeiten ungefähr gleich. Dies ist durch das ungünstigere Speedup-Verhalten der Verfahren von Variante 3 gegenüber dem der Verfahren von Variante 2 zu erklären. Das Speedup-Verhalten der Verfahren von Variante 3 wird in den folgenden Abschnitten näher untersucht.

Speedup-Messungen bei Matrizen unterschiedlicher Ordnung

In Abbildung 6.41 sind die Speedup-Werte des Pegasusverfahrens auf CRAY Y-MP mit 8 Prozessoren bei den Matrizen $A512$, $A800$, $A1000$, $A3200$ und $A6400$ dargestellt.

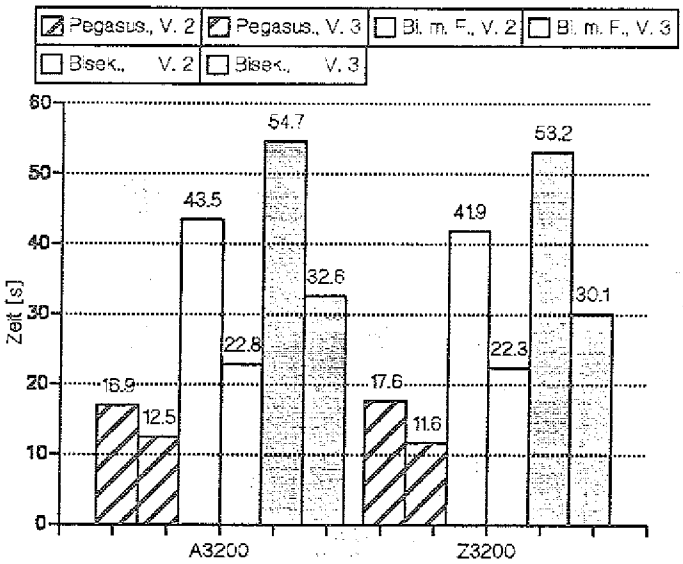


Abb. 6.39: Extraktionsphase, Variante 3: Vergleich der 3 implementierten Verfahren mit den entsprechenden Verfahren von Variante 2, sequentiell

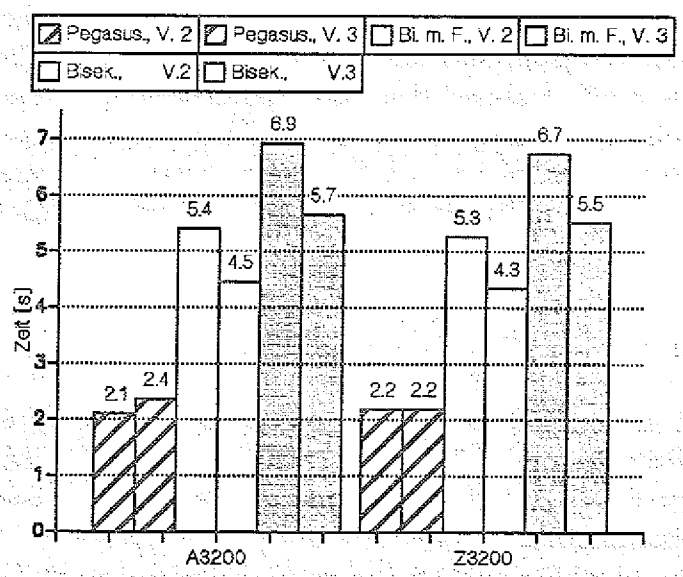


Abb. 6.40: Extraktionsphase, Variante 3: Vergleich der 3 implementierten Verfahren mit den entsprechenden Verfahren von Variante 2, parallel

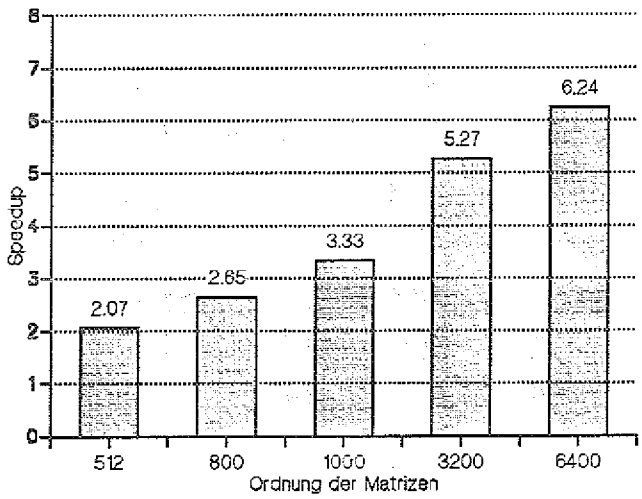


Abb. 6.41: Extraktionsphase, Variante 3: Speedup-Werte bei Matrizen unterschiedlicher Ordnung, Matrizen An

Die Speedup-Werte in Abbildung 6.41 liegen bei allen Matrizen deutlich unter dem Maximalwert 8. Zu Speedup-Verlusten führt der Overhead durch die Parallelisierungsroutinen. Dieser ist bei Variante 3 höher als bei den bisher untersuchten Varianten, da jede innere Schleife in eine Kontrollstruktur eingeschlossen ist. Ferner werden bei paralleler Programmausführung von jedem der 8 Prozessoren Vektoren ca. achtmal kürzerer Länge bearbeitet als bei sequentieller Programmausführung von einem Prozessor. Dies führt zu deutlichen Speedup-Verlusten.

Speedup-Werte bei 1 bis 8 Prozessoren

Abbildung 6.42 zeigt die Speedup-Werte des Pegasusverfahrens auf CRAY Y-MP mit 1 bis 8 angeforderten Prozessoren bei Matrix $A3200$. Der Option NUMCHUNKS(**anz**) der Autotasking-Direktive DO PARALLEL wurde als Parameter **anz** jeweils die Anzahl der angeforderten Prozessoren übergeben.
Die Speedup-Werte in Abbildung 6.42 liegen deutlich unter dem Maximalwert, der durch die Anzahl der angeforderten Prozessoren gegeben ist. Die Speedup-Verluste werden desto höher, je mehr Prozessoren eingesetzt werden, da der Overhead durch die Parallelisierungsroutinen (s. auch [26], Option NUMCHUNKS) steigt und die Länge der Vektoren, die jeder Prozessor bearbeitet, kürzer wird.

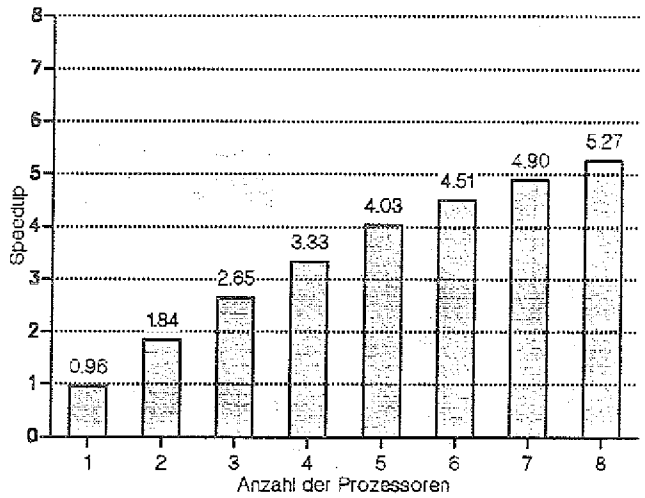


Abb. 6.42: Extraktionsphase, Variante 3: Speedup-Werte bei 1 bis 8 Prozessoren, Matrix $A3200$

6.4.5 Variante 4

Implementierung

Die Implementierung von Variante 4 ist der von Variante 3 sehr ähnlich. Der wesentliche Unterschied besteht darin, daß in Variante 4 statt der Folge (2.14) die Sturmsche Kette an allen Bisektionspunkten aller Intervalle, die genau einen Eigenwert enthalten, ausgewertet wird. Die Schleife, in der die Sturmsche Kette an allen Bisektionspunkten ausgewertet wird, wird mit Hilfe der Autotasking Direktive DO PARALLEL parallelisiert (s. auch Abbildung 5.10, zweite Schleife). Der zeitliche Aufwand dieser Schleife bestimmt in großem Maße das Zeitverhalten der gesamten Extraktionsphase, da die Auswertung der Sturmschen Kette sehr zeitintensiv ist.

Vergleich mit Variante 3

In Abbildung 6.43 sind die sequentiellen und parallelen Ausführungszeiten der Bisektionsverfahren von Variante 3 und Variante 4 auf CRAY Y-MP bei den Matrizen $A3200$ und $Z3200$ dargestellt.
Alle Ausführungszeiten des Bisektionsverfahrens von Variante 4 liegen in Abbildung 6.43 deutlich über denen von Variante 3. Dies zeigt, daß die Vektorisierung der Variante 3 günstiger ist als die der Auswertung der Sturmschen Kette mit Hilfe der CRAY-Routine

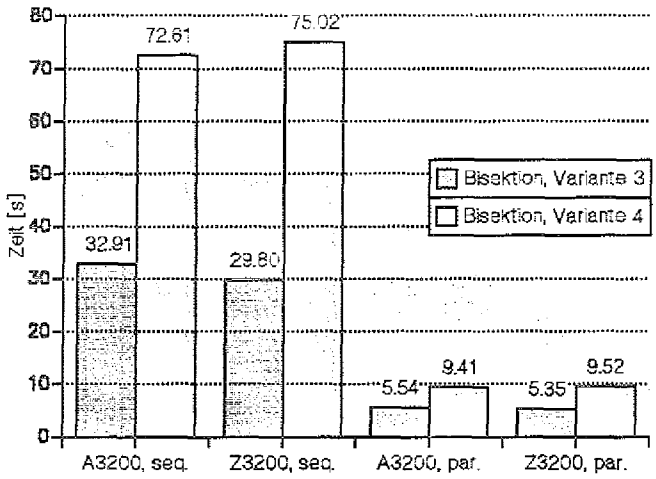


Abb. 6.43: Extraktionsphase, Variante 4: Vergleich mit Variante 3 bei Bisektion, sequentiell und parallel

SOLR.
Die Speedup-Werte des Bisektionsverfahrens von Variante 4 betragen 7.71 bei Matrix $A3200$ und 7.88 bei Matrix $Z3200$. Sie sind deutlich höher als die Speedup-Werte des Bisektionsverfahrens von Variante 3 (5.72 bei Matrix $A3200$ und 5.47 bei Matrix $Z3200$). Dies ist dadurch zu erklären, daß bei paralleler Ausführung der zeitintensivsten Schleife von Variante 4, die die Auswertung der Sturmschen Kette enthält, keine Speedup-Verluste durch kürzere Vektorlängen als bei sequentieller Ausführung auftreten. Ferner wirkt sich bei dieser Schleife der Overhead durch die Parallelisierungsroutinen nur sehr geringfügig auf das Speedup-Verhalten aus, da der Overhead im Vergleich zum zeitlichen Aufwand der Auswertung der Sturmschen Kette bei Matrizen hoher Ordnung sehr gering ist. Trotz des günstigeren Speedup-Verhaltens liegen die parallelen Ausführungszeiten von Variante 4 deutlich über denen von Variante 3 (s. Abbildung 6.43). Durch die ungünstigere Vektorisierung besitzt Variante 4 keine Vorteile gegenüber Variante 3.

6.4.6 Variante 5

Implementierung

In Variante 5 der Extraktionsphase sind Bisektion, Bisektion mit Auswertung des charakteristischen Polynoms und das Pegasusverfahren implementiert. Die Extraktion der

Eigenwerte erfolgt in Gruppen. Die Intervalle mit genau einem Eigenwert, die derselben Gruppe angehören, erfordern ungefähr gleich viele Iterationsschritte zur Extraktion der Eigenwerte. Die Gruppen werden nach jedem Isolationsschritt gebildet, da die in demselben Isolationsschritt ermittelten Intervalle mit genau einem Eigenwert die gleiche Intervallbreite besitzen (s. auch 5.3.4, Variante 5). Die minimale Größe der Gruppen ist frei wählbar. Da zu kleine Gruppen aus Vektorisierungsgründen ungünstig sind und die Vektorregister der CRAY-Rechner 64 Elemente aufnehmen können, ist die minimale Gruppengröße in den Verfahren von Variante 5 zu 64 gesetzt. Die einzelnen Gruppen werden nacheinander bearbeitet. Bei der Extraktion der Eigenwerte einer Gruppe laufen alle inneren Schleifen über alle Intervalle mit genau einem Eigenwert, die dieser Gruppe angehören. Die inneren Schleifen in Variante 5 sind auf die gleiche Weise vektorisiert und parallelisiert wie die inneren Schleifen in Variante 3, die im Gegensatz zu den inneren Schleifen in Variante 5 über alle Intervalle mit genau einem Eigenwert laufen.

Vergleich mit Variante 3

In den Abbildungen 6.44 und 6.45 werden die sequentiellen und parallelen Ausführungszeiten auf CRAY Y-MP der Verfahren von Variante 5 mit den entsprechenden Verfahren von Variante 3 bei den Matrizen $A3200$ und $Z3200$ verglichen.

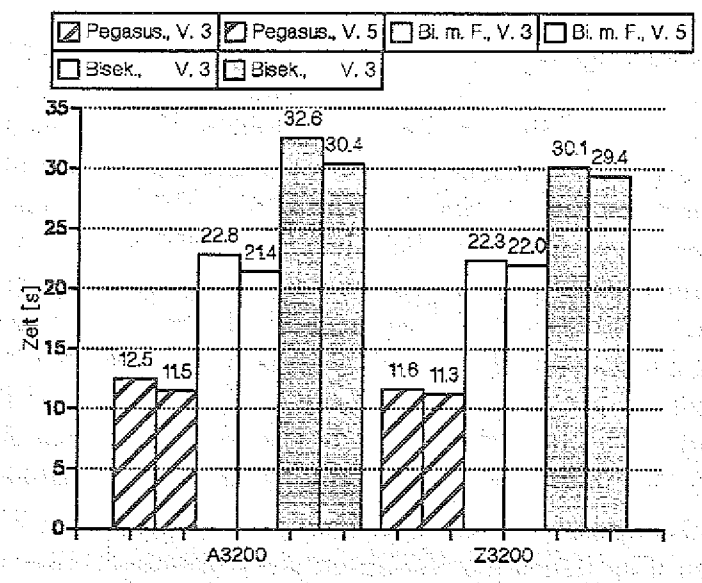


Abb. 6.44: **Extraktionsphase, Variante 5: Vergleich der 3 implementierten Verfahren mit den entsprechenden Verfahren von Variante 3, sequentiell**

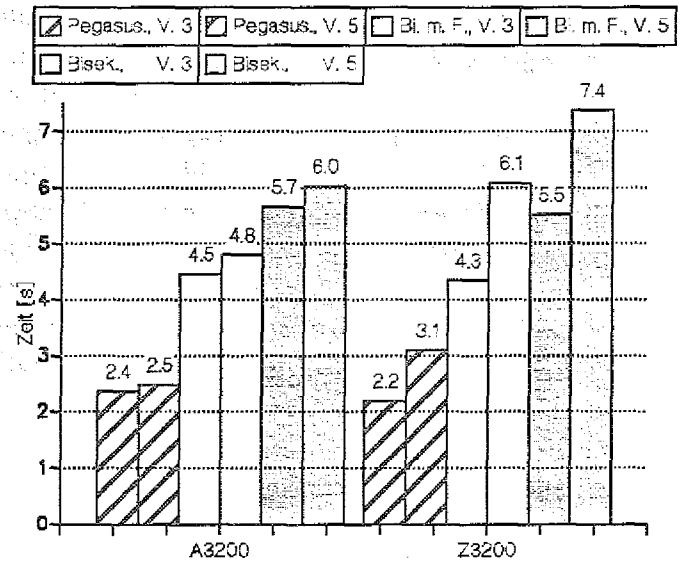


Abb. 6.45: Extraktionsphase, Variante 5: Vergleich der 3 implementierten Verfahren mit den entsprechenden Verfahren von Variante 3, parallel

Die sequentiellen Ausführungszeiten der Verfahren von Variante 5 in Abbildung 6.44 sind bei beiden Matrizen kürzer als die der entsprechenden Verfahren von Variante 3. Die Verfahren von Variante 5 benötigen in der Regel weniger Iterationsschritte als die Verfahren von Variante 3 (s. auch 5.3.4, Variante 5). Dies führt gegenüber den Verfahren von Variante 3 zu weniger Rechenaufwand. Die kürzeren Vektorlängen, die sich durch die Aufteilung in Gruppen bei den inneren Schleifen der Verfahren von Variante 5 ergeben, verursachen jedoch eine gegenüber den Verfahren von Variante 3 geringere Beschleunigung durch Vektorisierung. Ob der geringere Rechenaufwand oder die geringere Beschleunigung durch Vektorisierung das Zeitverhalten der Verfahren von Variante 5 entscheidend beeinflussen, hängt davon ab, wie unterschiedlich die Intervallbreiten der Intervalle mit genau einem Eigenwert sind. Die Intervallbreiten dieser Intervalle ergeben sich im Verlauf der Isolationsphase und sind damit von der Verteilung der Eigenwerte abhängig.

Die parallelen Ausführungszeiten der Verfahren von Variante 5 der Extraktionsphase in Abbildung 6.45 liegen über denen der Verfahren von Variante 3. Dies wird durch das gegenüber den Verfahren von Variante 3 deutlich schlechtere Speedup-Verhalten der Verfahren von Variante 5 verursacht. Speedup-Verluste bei paralleler Ausführung der Verfahren von Variante 5 haben die gleichen Ursachen wie in Variante 3, jedoch treten in Variante 5 bei paralleler Bearbeitung der inneren Schleifen durch die Aufteilung in Gruppen kürzere Vektorlängen auf. Ferner steigt der Overhead durch die Parallelisierungsroutinen

gegenüber den Verfahren von Variante 3, da die Elemente jeder Gruppe parallel bearbeitet werden. Aus diesen Gründen besitzen die Verfahren von Variante 5 bei paralleler Programmausführung in der Regel keine zeitlichen Vorteile gegenüber den Verfahren von Variante 3. Auf das Speedup-Verhalten der Verfahren von Variante 5 wird daher nicht weiter eingegangen.

6.4.7 Variante 6

Implementierung

In Variante 6 der Extraktionsphase sind Bisektion, Bisektion mit Auswertung des charakteristischen Polynoms und das Pegasusverfahren implementiert. Die Extraktion der Eigenwerte erfolgt in Gruppen gleicher Größe. Lediglich die letzte Gruppe kann mehr Elemente als die übrigen enthalten (s. auch Abbildung 5.15). Die Größe der Gruppen ist frei wählbar. Bei den untersuchten Verfahren ist sie zu 64 gesetzt, da 64 der Anzahl der Elemente eines Vektorregisters der CRAY-Rechner entspricht. Die Gruppenelemente sind Intervalle, die genau einen Eigenwert enthalten. Die Bearbeitung der Gruppenelemente wird vektorisiert. Die Vektorlänge entspricht dabei der fest vorgegebenen Gruppengröße. Die äußere Schleife der Extraktionsphase nach Variante 6 (s. Abbildung 5.15), die über alle Gruppen läuft, wird mit Hilfe der Autotasking-Direktive DO ALL parallelisiert. In Variante 6 werden die einzelnen Gruppen und nicht wie in Variante 5 die Gruppenelemente parallel bearbeitet.

Vergleich mit den schnellsten der bisher untersuchten Verfahren

In Abbildung 6.46 werden die sequentiellen Ausführungszeiten der Verfahren von Variante 6 auf CRAY Y-MP mit denen der entsprechenden Verfahren von Variante 5 bei den Matrizen $A3200$ und $Z3200$ verglichen.
Wie Abbildung 6.46 zeigt, ergeben sich bei den Verfahren von Variante 6 im Vergleich zu den entsprechenden Verfahren von Variante 5 deutlich kürzere oder ungefähr gleiche Laufzeiten. Bei nicht zu hohen Gruppengrößen (hier 64) enthalten die einzelnen Gruppen in der Regel Intervalle, die ungefähr gleich viele Iterationsschritte zur Extraktion der Eigenwerte erfordern. Dies trifft bei den Gruppen nicht zu, die Intervalle enthalten, die in aufeinanderfolgenden Isolationsschritten ermittelt wurden. Die überflüssigen Iterationsschritte bei einigen Intervallen dieser Gruppen wirken sich jedoch nicht stark auf das Zeitverhalten der gesamten Extraktionsphase aus, da sehr viele kleine Gruppen bearbeitet werden. In Variante 6 liegen zwar in der Regel kürzere Vektorlängen als in den Varianten 3 und 5 vor, jedoch werden bei den untersuchten Matrizen insgesamt deutlich weniger Iterationsschritte zur Extraktion aller Eigenwerte durchgeführt. Variante 5 hat gegenüber Variante 6 den Nachteil, daß bei vorgegebener minimaler Gruppengröße von 64 unter Umständen Intervalle

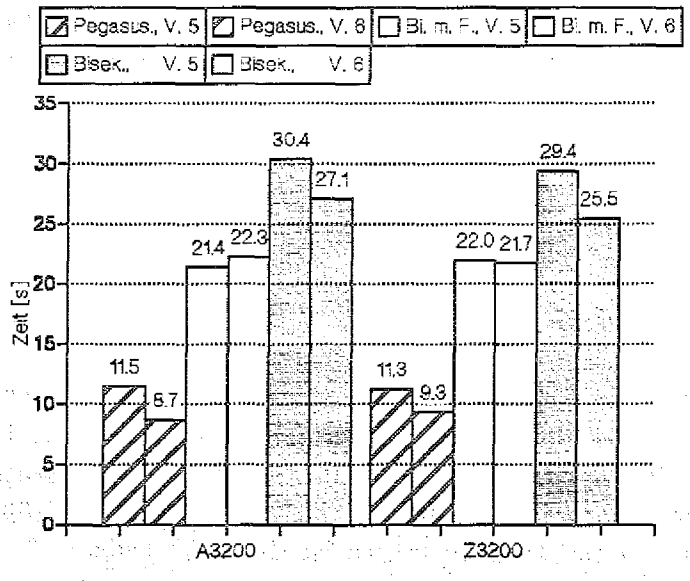


Abb. 6.46: Extraktionsphase, Variante 6: Vergleich der 3 implementierten Verfahren mit den entsprechenden Verfahren von Variante 5, sequentiell

in einer großen Gruppe zusammengefaßt werden, die jeweils eine deutlich unterschiedliche Anzahl der Iterationsschritte erfordern. Dieser Fall tritt ein, wenn in einigen Isolationsschritten sehr wenige Intervalle isoliert werden. Bei einer minimalen Gruppengröße von 1 würden diese Intervalle jeweils zu sehr kleinen Gruppen zusammengefaßt, jedoch ergäben sich infolge der sehr kleinen Vektorlängen deutliche Einbußen an Beschleunigung durch Vektorisierung.

In Abbildung 6.47 sind die parallelen Ausführungszeiten der Verfahren von Variante 6 auf CRAY Y-MP im Vergleich zu denen der entsprechenden Verfahren von Variante 2 bzw. Variante 3 bei den Matrizen $A3200$ und $Z3200$ dargestellt. Beim Pegasusverfahren ergaben sich in der bisherigen Untersuchung bei Variante 2 die kürzesten parallelen Laufzeiten, während bei den beiden anderen Verfahren Variante 3 das bisher günstigste Zeitverhalten aufwies.

In Abbildung 6.47 liegen die parallelen Ausführungszeiten der Verfahren von Variante 6 deutlich unter denen der Verfahren von Variante 2 bzw. Variante 3. Bei Variante 6 entsteht gegenüber Variante 3 wesentlich weniger Overhead durch die Parallelisierungsroutinen, da nur eine Schleife parallelisiert wird. Ferner wirkt sich der Overhead in Variante 6 kaum auf das Zeitverhalten der Extraktionsphase aus, da er gegenüber dem zeitlichen Aufwand zur Bearbeitung einer Gruppe sehr gering ist. Gegenüber Variante 2 führt hauptsächlich die bessere Vektorisierung in Variante 6 zu einem günstigeren Zeitverhalten.

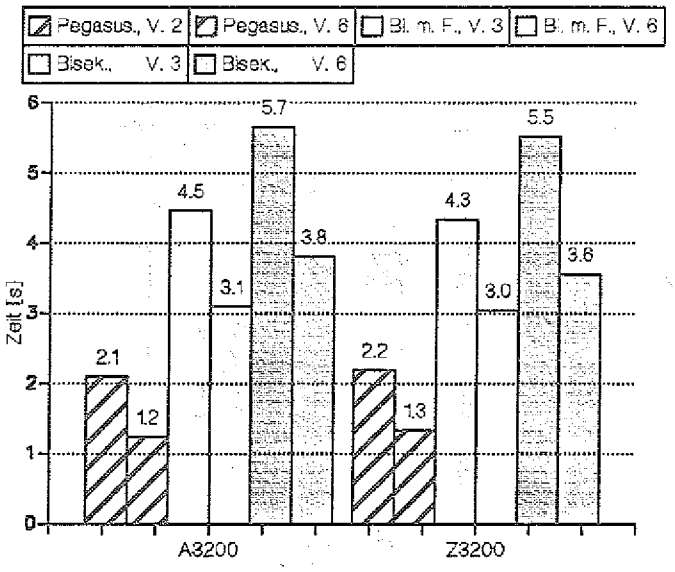


Abb. 6.47: Extraktionsphase, Variante 6: Vergleich der 3 implementierten Verfahren mit den schnellsten der bisher untersuchten Verfahren, parallel

Speedup-Werte bei Matrizen unterschiedlicher Ordnung

In Abbildung 6.48 sind die Speedup-Werte des Pegasusverfahrens von Variante 6 bei den Matrizen $A128$, $A512$, $A800$, $A1000$, $A3200$ und $A6400$ dargestellt.
In Abbildung 6.48 sind die Speedup-Werte desto höher, je höher die Ordnung der Matrizen ist. Bei Matrizen hoher Ordnung werden sehr viele Gruppen bearbeitet, so daß sich eine unterschiedliche Auslastung der Prozessoren bei der Bearbeitung der letzten Gruppen nicht stark auf das Speedup-Verhalten auswirkt. Eine unterschiedliche Auslastung der 8 Prozessoren von CRAY Y-MP bei der Bearbeitung der letzten Gruppen tritt zum einen dann auf, wenn die Anzahl der Gruppen nicht durch 8 teilbar ist. Dies ist bei den hier untersuchten Matrizen außer bei Matrix $A512$ der Fall. Zum anderen kann eine jeweils unterschiedliche Anzahl der Iterationsschritte, die die letzten Gruppen zur Extraktion der Eigenwerte der einzelnen Gruppen erfordern, zu unterschiedlicher Auslastung der Prozessoren führen, auch wenn die Anzahl der Gruppen durch 8 teilbar ist. Neben der unterschiedlichen Auslastung der Prozessoren bei der Bearbeitung der letzten Gruppen verursacht der Overhead durch die Parallelisierungsroutinen geringfügige Speedup-Verluste.
Da bei Matrix $A128$ nur zwei Gruppen der Größe 64 gebildet werden, liegt der maximal erreichbare Speedup bei 2. Dies erklärt den niedrigen Speedup-Wert von 1.76. Der hohe Speedup von 7.43 bei Matrix $A512$ kommt zustande, da bei dieser Matrix 8 Gruppen der Größe 64 gebildet werden und damit die 8 Prozessoren von CRAY Y-MP gut ausgelastet

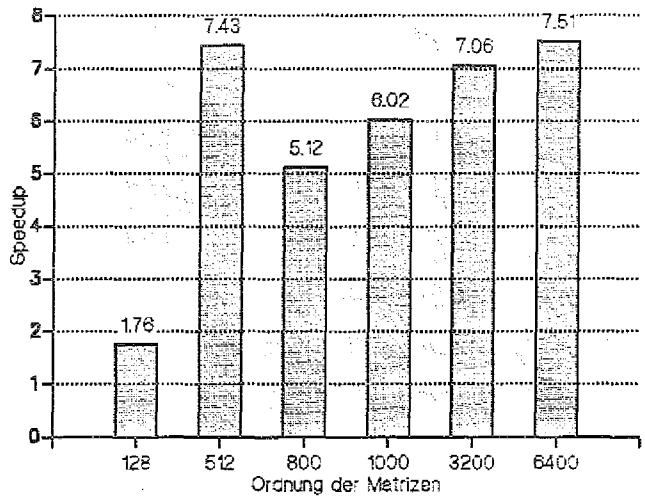


Abb. 6.48: Extraktionsphase, Variante 6: Speedup-Werte bei Matrizen unterschiedlicher Ordnung, Matrizen An

werden können.

Ein Vergleich von Abbildung 6.48 mit 6.41 zeigt, daß das Speedup-Verhalten bei Variante 6 der Extraktionsphase deutlich günstiger ist als bei Variante 3.

Speedup-Werte bei 1 bis 8 Prozessoren

Abbildung 6.49 zeigt die Speedup-Werte des Pegasusverfahrens auf CRAY Y-MP mit 1 bis 8 angeforderten Prozessoren bei Matrix $A3200$.

Die Speedup-Werte in Abbildung 6.49 liegen außer bei 7 Prozessoren desto näher am Maximalwert, der der Zahl der angeforderten Prozessoren entspricht, je weniger Prozessoren eingesetzt werden. Die Bearbeitung der Gruppen kann bei weniger Prozessoren in der Regel günstiger auf die eingesetzten Prozessoren verteilt werden, so daß ein bessere Auslastung entsteht. Da bei Matrix $A3200$ 50 Gruppen der Größe 64 gebildet werden, wird bei 7 eingesetzten Prozessoren nach der Bearbeitung der ersten 49 Gruppen im letzten Schritt nur eine Gruppe von einem Prozessor bearbeitet. Dies erklärt den vom sonstigen Speedup-Verlauf etwas abweichenden Wert bei 7 eingesetzten Prozessoren.

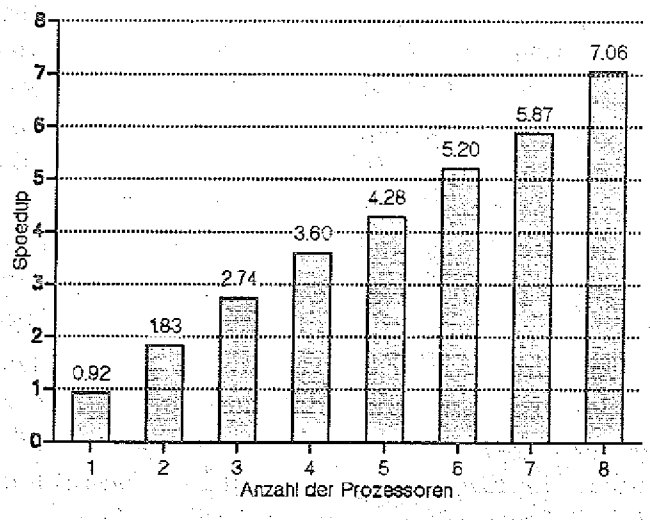


Abb. 6.49: Extraktionsphase, Variante 6: Speedup-Werte bei 1 bis 8 Prozessoren, Matrix $A3200$

6.4.8 Zusammenfassung

Die Verfahren von Variante 6 der Extraktionsphase sind bei sequentieller und paralleler Programmausführung die schnellsten der untersuchten Verfahren. Da diese Verfahren die Folge (2.14) verwenden, kann wie in der Isolationsphase auf die Verwendung der Sturmschen Kette ohne Geschwindigkeitsverluste verzichtet werden.
Bisektion und Multisektion in der Extraktionsphase sind auf CRAY X-MP und CRAY Y-MP gleichwertige Verfahren. Auf CRAY X-MP ergeben sich geringfügige zeitliche Vorteile für Bisektion, auf CRAY Y-MP dagegen für Multisektion. Dies zeigt, daß Multisektion ein Verfahren ist, dessen Zeitverhalten stark maschinenabhängig ist. Die optimale Anzahl an Multisektionspunkten läßt sich in der Extraktionsphase mit Hilfe der Überlegungen in 5.1 gut abschätzen.

6.5 Vergleich mit Bibliotheksprogrammen

In diesem Abschnitt werden die Ausführungszeiten des schnellsten in dieser Arbeit untersuchten Verfahrens mit denen der Bibliotheksroutinen BISECT und IMTQL1 aus der CRAY-Programmbibliothek SCI (SCIentific Library) auf CRAY Y-MP bei den Matrizen $A3200$, $A6400$, $Z3200$ und $Z6400$ verglichen. Beim schnellsten in dieser Arbeit untersuchten Verfahren wird in der Initialisierungsphase Multisektion mit 15 Punkten durchgeführt,

in der Isolationsphase Bisektion mit Partitionierung in 8 Gruppen sowie das Pegasusverfahren von Variante 6 in der Extraktionsphase. Da die Routinen BISECT und IMTQL1 die Eigenwerte der Größe nach geordnet an das rufende Programm zurückgeben, wurden die Eigenwerte bei dem in dieser Arbeit entwickelten Verfahren aus Gründen der Vergleichbarkeit mit Hilfe der vektorisierenden CRAY-Routine ORDERS nach der Extraktionsphase geordnet. Der zeitliche Aufwand für das Ordnen der Eigenwerte ist in den Ausführungszeiten in Abbildung 6.50 enthalten. Die Bibliotheksroutinen sind den CRAY-Rechnern angepaßte Routinen aus dem Programmpacket EISPACK. Sie sind, wenn möglich, auf Assembler-Ebene vektorisiert und können nur sequentiell ausgeführt werden. Die Routine BISECT wird zur Bestimmung einiger Eigenwerte von reellen symmetrischen Tridiagonalmatrizen empfohlen. Alle Eigenwerte, die in dem an die Routine übergebenen Intervall enthalten sind, werden berechnet. BISECT vektorisiert nicht. In den Messungen wurde an BISECT jeweils ein Intervall übergeben, das alle Eigenwerte der verwendeten Matrix enthält. Die Routine führt zur Eigenwertberechnung ein Bisektionsverfahren unter Verwendung der Folge (2.14) durch. Die Routine IMTQL1 wird zur Bestimmung aller Eigenwerte von reellen symmetrischen Tridiagonalmatrizen empfohlen. Die Routine verwendet den QL-Algorithmus, in dessen Verlauf eine Folge symmetrischer Tridiagonalmatrizen gebildet wird, die durch Ähnlichkeitstransformationen aus der ursprünglichen Matrix entstehen. Diese Folge konvergiert gegen eine Diagonalmatrix. Die Routine IMTQL1 vektorisiert.

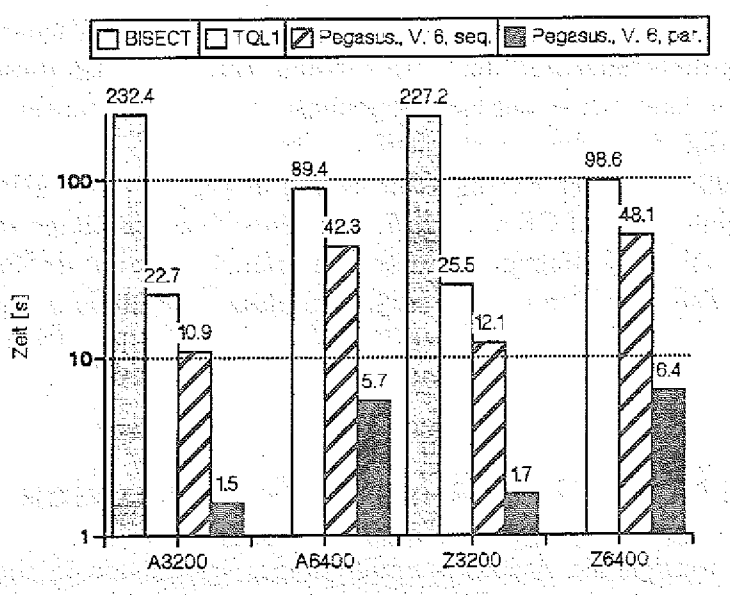


Abb. 6.50: Vergleich der Ausführungszeiten des schnellsten in dieser Arbeit untersuchten Verfahrens mit denen der Bibliotheksroutinen BISECT und TQL1

In Abbildung 6.50 sind die Ausführungszeiten von BISECT, IMTQL1 und des schnellsten in dieser Arbeit untersuchten Verfahrens dargestellt. Die Laufzeiten von BISECT wurden lediglich bei den Matrizen $A3200$ und $Z3200$ gemessen, da sie wesentlich höher sind als die der übrigen Routinen. Beim Pegasusverfahren sind die parallelen Ausführungszeiten auf CRAY Y-MP mit 8 eingesetzten Prozessoren ergänzt.
Abbildung 6.50 zeigt, daß die sequentiellen Ausführungszeiten des schnellsten in dieser Arbeit untersuchten Verfahrens weniger als die Hälfte von denen der Routine IMTQL1 betragen und ca. ein Zwanzigstel von denen der Routine BISECT. Die gute Vektorisierung des in dieser Arbeit entwickelten Verfahrens sowie das Iterationsverfahren mit superlinearer Konvergenz in der Extraktionsphase führen zu den wesentlich kürzeren Ausführungszeiten. Die parallelen Ausführungszeiten des in dieser Arbeit entwickelten Verfahrens sind um einen Faktor größer als 7 kürzer als die sequentiellen. Das in dieser Arbeit entwickelte Verfahren ist gut auf die Eigenschaften zur Vektor- und Parallelverarbeitung der CRAY-Rechner abgestimmt. Da es durch die günstige Vektorisierung sequentiell schneller als die Routine IMTQL1 ist und zusätzlich gut parallelisiert, kann es auf den CRAY-Rechnern zur Bestimmung aller Eigenwerte von reellen symmetrischen Tridiagonalmatrizen durchaus mit dem QL-Algorithmus konkurrieren.

6.6 Problemfälle

Liegen sehr viele Eigenwerte einer Matrix sehr nah beieinander, ist bei allen untersuchten Verfahren der zeitliche Aufwand zur Isolation der Eigenwerte groß. Die Laufzeit der Isolationsphase nähert sich in diesem Fall der der Extraktionsphase an. Das Zeitverhalten des Gesamtprogramms wird ungünstiger, da in der Isolationsphase nur Bisektion oder Multisektion eingesetzt werden können. In der Extraktionsphase, deren zeitlicher Aufwand in diesem Fall sinkt, kann dagegen ein Iterationsverfahren mit superlinearer Konvergenz durchgeführt werden. Unterscheiden sich alle Eigenwerte einer Matrix um weniger als den skalierten, absoluten Fehler, liegen aber keine echten mehrfachen Eigenwerte vor, so ist die Ausführung der Extraktionsphase nicht erforderlich, da alle Eigenwerte bereits in der Isolationsphase bis auf den skalierten, absoluten Fehler genau bestimmt werden.
Sind echte mehrfache Eigenwerte vorhanden, so führt die Zerlegung der Matrix zu einem günstigen Zeitverhalten. In diesem Fall sind Elemente der Nebendiagonalen der Matrix gleich Null (s. auch 2.3.1). In Abbildung 6.51 werden die sequentiellen und parallelen Ausführungszeiten eines Verfahrens mit Zerlegung der Matrix und eines Verfahrens ohne Zerlegung der Matrix verglichen. Bei beiden Verfahren wird in der Initialisierungsphase und in der Isolationsphase Multisektion mit 15 Punkten durchgeführt, in der Extraktionsphase das Pegasusverfahren von Variante 3 ohne vorher ausgeführte Bisektionsschritte. Matrix $A512*2$ ist aus zwei Matrizen $A512$ zusammengesetzt, besitzt also die Ordnung 1024. Alle Eigenwerte sind zweifach. Die Matrix $A512 + Z512$ besteht aus den Matrizen $A512$ und

$Z512$, ist ebenfalls von der Ordnung 1024 und besitzt nur einfache Eigenwerte.

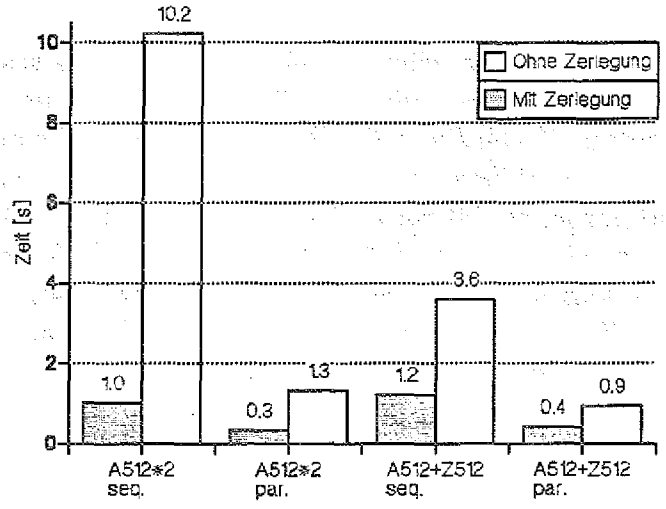


Abb. 6.51: Auswirkung der Zerlegung der Matrix auf das Zeitverhalten

Abbildung 6.51 zeigt, daß sich für Matrix $A512 * 2$ bei Zerlegung der Matrix sequentiell ca. zehnmal kürzere Laufzeiten ergeben als ohne Zerlegung und parallel ca. 4 mal kürzere Laufzeiten. Ohne Zerlegung werden alle Eigenwerte in der Isolationsphase bestimmt. Die Extraktionsphase wird nicht durchlaufen. Die superlineare Konvergenz des Pegasusverfahrens kann nicht genutzt werden. Mit Zerlegung werden nacheinander die Eigenwerte von zwei Matrizen $A512$ berechnet und jeweils Isolations- und Extraktionsphase durchlaufen. Bei paralleler Ausführung sind die Unterschiede der Verfahren mit und ohne Zerlegung geringer, da die Extraktionsphase bei Variante 3 ein schlechteres Speedup-Verhalten aufweist als die Isolationsphase bei Multisektion mit 15 Punkten (vergl. die Abbildungen 6.14 und 6.41). Abbildung 6.51 ist ebenfalls zu entnehmen, daß die Zerlegung der Matrix auch bei Matrix $A512 + Z512$ zu deutlich kürzeren Ausführungszeiten führt. Dies zeigt, daß die Zerlegung der Matrix, wenn Nullen in den Nebendiagonalen vorliegen, lohnend ist, auch wenn keine mehrfachen Eigenwerte vorliegen.
Wie schon in 2.2.2 erwähnt, ist die Auswertung der Sturmschen Kette anfällig für Zahlenbereichsüberschreitungen. Bei den untersuchten Verfahren, die die Sturmsche Kette verwenden, treten bei der Berechnung der Eigenwerte der Matrizen $A6400$ und $Z6400$ Zahlenbereichsüberschreitungen auf, während dies bei den Verfahren, die die Folge (2.14) auswerten, nicht der Fall ist. Daher ist die Verwendung der Folge (2.14) der der Sturmschen Kette vorzuziehen. Auf Zahlenbereichsüberschreitungen bei Auswertung der Sturmschen

Kette wird auch in [22] hingewiesen.

Bei Bisektion mit Auswertung des charakteristischen Polynoms und dem Pegasusverfahren wird der Funktionswert des charakteristischen Polynoms berechnet. Dabei können auch bei Verwendung der Folge (2.14) Zahlenbereichsüberschreitungen auftreten, da die Funktionswerte des charakteristischen Polynoms an manchen Punkten sehr groß oder sehr klein werden. Die Berechnung der Eigenwerte der Matrizen $A10000$ und $Z10000$ sowie $B1000$, $B3200$, $B6400$ und $B10000$ führt zu Zahlenbereichsüberschreitungen, wenn der Funktionswert des charakteristischen Polynoms bestimmt werden muß. Bei Bisektion ohne Auswertung des charakteristischen Polynoms treten bei keiner der untersuchten Matrizen Zahlenbereichsüberschreitungen auf. Alle implementierten Verfahren, die dieses Iterationsverfahren verwenden, sind für Zahlenbereichsüberschreitungen nicht anfällig. Bei Bisektion mit Auswertung des charakteristischen Polynoms und dem Pegasusverfahren können Zahlenbereichsüberschreitungen jedoch abgefangen werden, wenn der Funktionswert des charakteristischen Polynoms nach 2.5 berechnet wird. Zum Abfangen von Zahlenbereichsüberschreitungen ist ein Verfahren implementiert, in dem nach jedem Zwischenschritt der Produktbildung des Produktes aller q_{n-i} (s. 2.5) abgefragt wird, ob der Betrag des aktuellen Wertes 10^{2000} überschreitet bzw. 10^{-2000} unterschreitet (Zahlenbereich der CRAY-Rechner bei einfach genauen REAL-Zahlen: von 10^{-2466} bis 10^{2466}). Ist dies der Fall, wird der aktuelle Wert durch 10^{2000} dividiert bzw. mit 10^{2000} multipliziert. Beim Pegasusverfahren muß darauf geachtet werden, daß die Funktionswerte, die zur Berechnung des nächsten Iterationspunktes verwendet werden (s. auch 2.6), mit den gleichen Faktoren gewichtet sind. Eine Wichtung der Funktionswerte mit gleichen Faktoren beeinflußt den weiteren Verlauf des Pegasusverfahrens nicht. Die Wichtung mit gleichen Faktoren wird erreicht, indem bei Berechnung jedes Funktionswertes in einem Zähler festgehalten wird, wie oft durch 10^{2000} dividiert bzw. mit 10^{2000} multipliziert wird. Bei unterschiedlichen Zählerständen kann durch Multiplikation mit 10^{2000} bzw. durch Division durch 10^{2000} bei den Funktionswerten, die zur Berechnung des nächsten Iterationspunktes verwendet werden, dafür gesorgt werden, daß diese mit den gleichen Faktoren gewichtet sind. Bei Anwendung dieses Verfahrens zum Abfangen von Zahlenbereichsüberschreitungen traten bei keiner der untersuchten Matrizen Zahlenbereichsüberschreitungen auf. Sicherlich sind auch andere Methoden zum Abfangen von Zahlenbereichsüberschreitungen möglich, so könnte z.B. auch die Matrix selbst gewichtet werden. Auf diese Methoden wird in dieser Arbeit jedoch nicht eingegangen.

Das oben beschriebene Verfahren zum Abfangen von Zahlenbereichsüberschreitungen führt zu zusätzlichem Rechenaufwand. Dieser Aufwand wirkt sich deutlich auf das Zeitverhalten der Extraktionsphase aus, da er in den zeitintensivsten Schleifen der Extraktionsphase auftritt. In Abbildung 6.52 sind die sequentiellen und parallelen Ausführungszeiten des Pegasusverfahrens mit und ohne Zusatzaufwand zum Abfangen von Zahlenbereichsüberschreitungen sowie von Bisektion nach Variante 6 der Extraktionsphase auf CRAY Y-MP

bei Matrix $Z6400$ dargestellt.

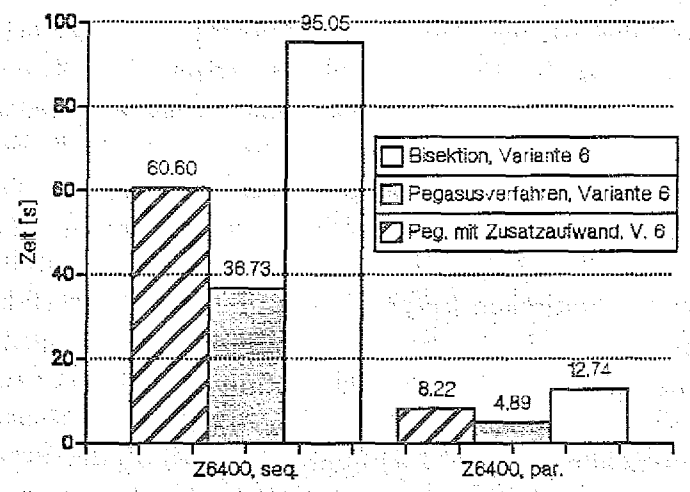


Abb. 6.52: **Extraktionsphase, Variante 6: Auswirkung des Zusatzaufwandes zur Verhinderung von Zahlenbereichsüberschreitungen auf das Zeitverhalten des Pegasusverfahrens**

Abbildung 6.52 zeigt, daß die Ausführungszeiten des Pegasusverfahrens mit Zusatzaufwand zum Abfangen von Zahlenbereichsüberschreitungen ca. 1.7 mal höher sind als die des Pegasusverfahrens ohne zusätzlichen Aufwand zur Vermeidung von Zahlenbereichsüberschreitungen. Es ist zu beachten, daß das Pegasusverfahren von Variante 6 trotz dieses Zusatzaufwandes deutlich schneller als Bisektion ist. Ebenso ist es trotz der ca. 1.7 mal höheren Laufzeit der Extraktionsphase immer noch schneller als die Routine IMTQL1, da die Laufzeit dieser Routine mehr als doppelt so hoch wie die Laufzeit des schnellsten Gesamtprogramms mit Verwendung des Pegasusverfahrens von Variante 6 in der Extraktionsphase ist (s. Abbildung 6.50). Die Verwendung des Pegasusverfahrens ist also auch mit zusätzlichem Aufwand zur Vermeidung von Zahlenbereichsüberschreitungen lohnend.

7 Schlußbemerkungen

In dieser Arbeit wurden verschiedene Bisektions- und Multisektionsverfahren zur Bestimmung aller Eigenwerte von reellen symmetrischen Tridiagonalmatrizen entwickelt und auf den CRAY-Rechnern des Forschungszentrums Jülich implementiert. Dabei wurden die Vektorisierungs- und Parallelisierungseigenschaften dieser Verfahren besonders berücksichtigt. Zur Parallelisierung der Verfahren wurde das Multitasking-Konzept Autotasking verwendet.
Alle entwickelten Verfahren sind in drei Phasen aufgeteilt: Initialisierung, Isolation und Extraktion. In der Initialisierungsphase konnte durch ein spezielles Multisektionsverfahren sichergestellt werden, daß zu Beginn der Isolationsphase mindestens zwei Intervalle vorliegen, die Eigenwerte enthalten. Neben diesem Verfahren wurde äquidistante Multisektion untersucht. In der Isolationsphase konnten bei Bisektion mit Partitionierung in Gruppen gute Vektorisierungseigenschaften bei der jeweiligen Bearbeitung der Gruppenelemente erreicht werden, während die parallele Bearbeitung der von einander unabhängigen Gruppen möglich ist. Die Extraktionsphase konnte durch Verwendung des modifizierten Pegasusverfahrens sowie durch eine spezielle Gruppenaufteilung optimiert werden. Diese Gruppenaufteilung führt sowohl zu guten Vektorisierungs- als auch Parallelisierungseigenschaften der Extraktionsphase und spart zusätzlich gegenüber den Verfahren, die keine Aufteilung in Gruppen vornehmen, Iterationsschritte. Durch das Genauigkeitskriterium des skalierten, absoluten Fehlers konnte eine hohe Genauigkeit bei der Bestimmung der Eigenwerte erreicht werden.
Durch gute Vektorisierungseigenschaften in allen Phasen sowie den Einsatz des Pegasusverfahrens für die Extraktion konnten auch bei sequentieller Ausführung des schnellsten der untersuchten Verfahren deutlich kürzere Laufzeiten als die der EISPACK-Routinen BISECT und IMTQL1 erzielt werden, die als den CRAY-Rechnern angepaßte Versionen in der Programmbibliothek SCILIB vorliegen. Bei paralleler Bearbeitung auf CRAY Y-MP mit 8 eingesetzten Prozessoren wurden für Matrizen hoher Ordnung Speedup-Werte erreicht, die größer als 7 waren.
Es wurde gezeigt, daß auf die Sturmsche Kette ohne Geschwindigkeitsverluste verzichtet und statt dieser die für Zahlenbereichsüberschreitungen weniger anfällige rationale Folge verwendet werden kann, auch wenn die Berechnung des Funktionswertes des charakteristischen Polynoms erforderlich ist. Dennoch auftretende Zahlenbereichsüberschreitungen bei der Berechnung des Funktionswertes des charakteristischen Polynoms können abgefangen werden.

In der Isolationsphase erwies sich erwartungsgemäß das Zeitverhalten von Multisektion als stark von der Verteilung der Eigenwerte abhängig. Lediglich bei günstiger Verteilung der Eigenwerte ergaben sich nennenswerte zeitliche Vorteile gegenüber Bisektion. In der Extraktionsphase unterschied sich das Zeitverhalten von Bisektion und Multisektion wenig. Die optimale Zahl der Multisektionspunkte konnte hier gut mit Hilfe der Performance-Halbwertslänge abgeschätzt werden.
Alle entwickelten Verfahren können ohne großen Aufwand auf die Bestimmung der Eigenwerte in einem vorgegebenen Intervall umgestellt werden. Hierzu müssen lediglich die Sturmsche Kette oder die entsprechende rationale Folge zu Beginn der Initialisierungsphase an den Intervallgrenzen des vorgegebenen Intervalls ausgewertet werden. In diesem Fall entfällt die Bestimmung des Intervalls, das alle Eigenwerte enthält.
Bisektions- und Multisektionsverfahren ohne Aufteilung in Gruppen in der Isolationsphase und die Varianten 1 und 2 der Extraktionsphase führen auf den CRAY-Rechnern zu schlechteren Laufzeiten als die übrigen Verfahren, da sie keine günstigen Vektorisierungseigenschaften besitzen. Auf hochparallelen Systemen ohne Vektorverarbeitung sind diese Verfahren jedoch geeigneter als die übrigen Verfahren, da sehr viele parallele Prozesse entstehen. Ferner könnte auf hochparallelen Systemen die Trennung von Isolations- und Extraktionsphase aufgehoben werden, und ermittelte Intervalle, die genau einen Eigenwert enthalten, könnten direkt zur Extraktion des Eigenwerts an verfügbare Prozessoren übergeben werden.

Literaturverzeichnis

[1] *Autotasking User's Guide.* Cray Research, Inc., Oktober 1988. SN-2088.

[2] *Multitasking Programmer's Manual.* Cray Research, Inc., 1989. SR-0222 F.

[3] N. Attig et al. *Einführung in das Betriebssystem UNICOS.* KFA-ZAM-BHB-0093, Zentralinstitut für Angewandte Mathematik, Forschungszentrum Jülich, Juli 1990.

[4] R.H. Barlow, D.J. Evans, J. Shanehchi. Parallel multisection applied to the eigenvalue problem. *The Computer Journal*, 26:6–9, 1983.

[5] W. Barth, R.S. Martin, J.H. Wilkinson. *Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection. Contribution II/5, in Handbook for Automatic Computation, Linear Algebra*, Springer Verlag, Berlin Heidelberg New York, 1971.

[6] H.J. Bernstein. An accelerated bisection method for the calculation of eigenvalues of a symmetric tridiagonal matrix. *Numerische Mathematik*, 43:153–160, 1984.

[7] H.J. Bernstein, M. Goldstein. Optimizing givens' algorithm for multiprocessors. *SIAM Journal on Scientific and Statistical Computing*, 9:601–602, 1988.

[8] H.J. Bernstein, M. Goldstein. Parallel implementation of bisection for the calculation of eigenvalues of tridiagonal symmetric matrices. *Computing*, 37:85–91, 1986.

[9] L. Collatz. *Eigenwertaufgaben mit technischen Anwendungen.* Akademische Verlagsgesellschaft Geest & Portig K.-G., Leipzig, zweite, durchgesehene Auflage, 1963.

[10] U. Detert. Experience with autotasking on CRAY Y-MP. *Proceedings of the Cray User Group*, 1989.

[11] M. Dowell, P. Jarratt. The "pegasus" method for computing the root of an equation. *BIT*, 12:503–508, 1972.

[12] M.J. Flynn. Very high-speed computing systems. *Proceedings of the IEEE*, 54:1901–1909, 1966.

[13] G.H. Golub, C.F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 1983.

[14] R.T. Gregory, D.L. Karney. *A Collection of Matrices for Testing Computational Algorithms.* Krieger Verlag, Huntington, New York, 1978.

[15] I. Gutheil, W. Rönsch, H. Strauß. *Lineare Algebra Software für SUPRENUM.* Jül-2345, Zentralinstitut für Angewandte Mathematik, Forschungszentrum Jülich, Januar 1990.

[16] R.W. Hockney, C.R. Jesshope. *Parallel Computers 2 - Architecture, Programming and Algorithms.* Adam Hilger, 1988.

[17] G. Hofemann. *Vergleich der Speicherarchitekturen der Mehrprozessor-Vektorrechner CRAY X-MP und CRAY Y-MP anhand vektorisierender Algorithmen.* Jül-Spez-555, Zentralinstitut für Angewandte Mathematik, Forschungszentrum Jülich, März 1990.

[18] F. Hoßfeld. *Parallele Algorithmen. Informatik-Fachberichte*, Springer Verlag, Berlin Heidelberg New York Tokyo, 1983.

[19] F. Hoßfeld, R. Knecht, W.E. Nagel. Multitasking: Experience with applications on a CRAY X-MP. *Parallel Computing*, 12:259–283, 1989.

[20] K. Hwang, F.A. Briggs. *Computer Architecture and Parallel Processing.* McGraw-Hill, 1985.

[21] R. Jeltsch. *Numerische Mathematik I für Ingenieure (Teil A).* Institut für Geometrie und Praktische Mathematik, RWTH - Aachen, 1985.

[22] S. Lo, B. Philippe, A. Sameh. A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problem. *SIAM Journal on Scientific and Statistical Computing*, 8:155–165, 1987.

[23] R.S. Martin, C. Reinsch, J.H. Wilkinson. *Householder's Tridiagonalization of a Symmetric Matrix. Contribution II/2, in Handbook for Automatic Computation, Linear Algebra*, Springer Verlag, Berlin Heidelberg New York, 1971.

[24] W. Nagel, S. Knecht. Einsatzmöglichkeiten des Multitasking am Beispiel von Programmkernen. *Mitteilungen - Gesellschaft für Informatik e. V., Parallelalgorithmen und -rechnerstrukturen*, 4:75–90, 1987. ISSN 0177-0454.

[25] W.E. Nagel. *Prinzipien der Parallelverarbeitung auf Rechnern mit gemeinsamem Speicher.* KFA-ZAM-IB-9013, Zentralinstitut für Angewandte Mathematik, Forschungszentrum Jülich, Mai 1990.

[26] H. Reger. *Ein Vergleich der Multitasking-Implementierungen auf CRAY X-MP und IBM 3090.* Jül-Spez-542, Zentralinstitut für Angewandte Mathematik, Forschungszentrum Jülich, Dezember 1989.

[27] U. Schendel. *Einführung in die parallele Numerik.* R. Oldenbourg Verlag, München Wien, 1981.

[28] H.R. Schwarz. *Tridiagonalization of a Symmetric Band Matrix. Contribution II/8, in Handbook for Automatic Computation, Linear Algebra,* Springer Verlag, Berlin Heidelberg New York, 1971.

[29] H.D. Simon. Bisection is not optimal on vector processors. *SIAM Journal on Scientific and Statistical Computing,* 10:205–209, 1989.

[30] J. Stoer. *Einführung in die Numerische Mathematik I.* Springer Verlag, Berlin Heidelberg New York, zweite, neubearbeitete und erweiterte Auflage, 1976.

[31] J. Stoer, R. Bulirsch. *Einführung in die Numerische Mathematik II.* Springer Verlag, Berlin Heidelberg New York, 1973.

[32] R. Unbehauen. *Systemtheorie.* R. Oldenbourg Verlag, München Wien, 3. Auflage, 1980.

[33] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[34] L.A. Zadeh, C.A. Desoer. *Linear System Theory.* McGraw-Hill, 1963.

Die vorliegende Arbeit wurde als Diplomarbeit am Lehrstuhl für Technische Informatik und Computerwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen angefertigt.

Dem Lehrstuhlinhaber Herrn Prof. Dr. F. Hoßfeld danke ich für die Möglichkeit, die Arbeit am Zentralinstitut für Angewandte Mathematik (ZAM) des Forschungszentrums Jülich zu erstellen. Mein besonderer Dank gilt Herrn Dr. P. Weidner für die Betreuung dieser Arbeit sowie hilfreiche Anregungen und Ratschläge. Bei meinem Zimmerkollegen Herrn Dr. B. Steffen bedanke ich mich für viele interessante Diskussionen. Ferner danke ich Frau R. Knecht und Herrn W. Nagel für die Beratung bei der Verwendung von Multitasking.

FORSCHUNGSZENTRUM JÜLICH GmbH

KFA

Jül-2427
Januar 1991
ISSN 0366-0885