Über die Verbesserung von Schranken für die Eigenwerte symmetrischer Tridiagonalmatrizen

G. Alefeld und J. Herzberger

Universität Karlsruhe, Institut für Angewandte Mathematik 7500 Karlsruhe, Englerstraße 2

Zusammenfassung: *Für die Eigenwerte einer reellen symmetrischen n × n-Tridiagonalmatrix seien Einschließungsintervalle bekannt, die etwa mit dem Bisektionsverfahren oder dem Satz von Gerschgorin berechnet wurden. Wir geben in dieser Arbeit ein Einzelschrittverfahren zur simultanen Verbesserung dieser Einschließungsintervalle an, welches stets gegen alle Eigenwerte konvergiert, falls nur die Einschließungsintervalle disjunkt sind. Das Verfahren konvergiert in diesem Falle schneller als quadratisch. Der vorgeschlagene Algorithmus wird in Form einer ALGOL-60-Prozedur realisiert, bei der sämtliche Rundungsfehler erfaßt werden. Abschließend werden einige numerische Ergebnisse diskutiert.*

Summary: *Let there be known intervals, in which the eigenvalues of a real symmetric n × n-tridiagonal matrix are contained. These intervals could be obtained for example by the bisection method or by application of Gerschgorin's theorem. In this paper we consider a single step method for improving these inclusion sets simultaneously. The method always converges to the eigenvalues if the given inclusion sets are pairwise disjunct. Under these conditions the rate of convergence is higher than quadratic. For the suggested algorithm we give an ALGOL 60 procedure taking account of all rounding errors. At the end of the paper we give some numerical examples.*

1. Einleitung

Will man *alle* Eigenwerte einer symmetrischen vollbesetzten Matrix bestimmen, so bietet sich als Methode das Jacobi-Verfahren an. Vom Aufwand her ist es jedoch im allgemeinen günstiger, die gegebene Matrix zunächst mit einer der Methoden von GIVENS oder HOUSEHOLDER auf Tridiagonalform zu transformieren und anschließend das Bisektionsverfahren oder den QD-Algorithmus zu verwenden (siehe [27], Seite 191 f.). Die beiden zuletzt genannten Verfahren zur Berechnung der Eigenwerte von symmetrischen Tridiagonalmatrizen sind auch bei sämtlich einfachen Eigenwerten nur linear konvergent. Es ist daher wünschenswert, so berechnete grobe Näherungen mit einem schneller konvergenten Verfahren (z. B. Newton-Verfahren) zu verbessern. Das ist im allgemeinen mit erheblichen Schwierigkeiten verbunden, da es nicht einfach ist, den richtigen Zeitpunkt festzustellen, zu welchem man zu einem konvergenten Verfahren höherer Ordnung übergehen kann. Im folgenden machen wir von der Tatsache Gebrauch, daß man mit Hilfe des Bisektionsverfahrens die Eigenwerte einer symmetrischen Tridiagonalmatrix in Intervalle einschließen kann. Wir geben ein Verfahren an, welches – allein unter der Voraussetzung, daß die Eigenwerte in paarweise disjunkte Intervalle eingeschlossen sind – für jeden Eigenwert eine Folge von Einschließungsintervallen liefert, deren Durchmesser gegen Null konvergiert. Die Konvergenzgeschwindigkeit, mit der die Längen der Einschließungsintervalle gegen Null konvergieren, ist größer als 2. Das vorgeschlagene Verfahren wird mit einigen rechnerischen Modifikationen in Form einer Algol-60-Prozedur realisiert, wobei sämtliche Rundungsfehler miterfaßt werden. Abschließend sind einige numerische Beispiele angegeben.

2. Problemstellung und Bezeichnungen

Gegeben sei eine reelle symmetrische n × n-Matrix $\overline{A} = (\overline{a}_{ij})$. Es seien die Eigenwerte der Matrix $\overline{A}$ zu bestimmen, d. h. Zahlen λ, so daß

$$\overline{A}x = \lambda x \text{ mit } x \neq 0.$$

Dazu wird mit Hilfe einer endlichen Folge von orthogonalen Ähnlichkeitstransformationen der Gestalt

$$\widetilde{A} = U^T \overline{A} U$$

die im allgemeinen vollbesetzte Matrix $\overline{A}$ auf die Form

$$A = \begin{pmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & b_{n-1} & a_n \end{pmatrix}$$

gebracht. Als Verfahren dafür bietet sich sowohl vom Rechenaufwand her als auch aus numerischen Gründen die Householdertransformation an. (Siehe etwa [27], [30].)

Die Eigenwerte von A (und damit von $\overline{A}$) berechnen sich dann als Nullstellen des charakteristischen Polynoms

$$p(\lambda) = \det(\lambda I - A)$$

von A. Der Wert von $p(\lambda)$ läßt sich durch die folgenden Rekursionsformeln bestimmen (siehe [27]):

$f_0(\lambda) = 1, f_1(\lambda) = \lambda - a_1,$

$f_k(\lambda) = (\lambda - a_k) f_{k-1}(\lambda) - b_{k-1}^2 f_{k-2}(\lambda), k = 2, 3, \ldots, n,$

$p(\lambda) = f_n(\lambda).$

Die Folge der Polynome

$f_0(\lambda), f_1(\lambda), \ldots, f_n(\lambda)$

bildet bekanntlich eine Sturmsche Kette, und es gilt daher (siehe [27]): Die Anzahl z der Nullstellen von $p(\lambda) = f_n(\lambda)$, die größer als ein Wert a sind, ist gleich der Anzahl der Vorzeichenwechsel $V(a)$ der Folge $f_0(\lambda), f_1(\lambda), \ldots, f_n(\lambda)$ für $\lambda = a$.

Mit Hilfe dieser Eigenschaft lassen sich durch fortgesetztes Halbieren eines Ausgangsintervalles, in welchem alle Eigenwerte von A liegen, n Intervalle bestimmen, in denen jeweils ein Eigenwert liegt. Besitzt A nur einfache Eigenwerte, so kann man durch hinreichend feine Unterteilung erreichen, daß die n Intervalle *disjunkt* sind. Besitzt A mehrfache Eigenwerte, so läßt sich jeder mehrfache Eigenwert in ein Intervall beliebig kleiner vorgegebener Länge einschließen.

Im folgenden können wir daher voraussetzen, daß $n - m > 1$ paarweise *disjunkte* Intervalle bekannt sind, in denen jeweils ein einfacher Eigenwert von A liegt. Für die restlichen m Eigenwerte von A setzen wir voraus, daß m nicht notwendig disjunkte Intervalle bekannt sind, in denen jeweils einer der verbleibenden m Eigenwerte liegt. (Besitzt z. B. A einen zweifachen Eigenwert, so tritt das entsprechende Einschließungsintervall zweimal auf.) Jedes dieser m Intervalle sei jedoch disjunkt mit jedem der $n - m$ Intervalle, in welchem genau ein einfacher Eigenwert liegt.

Wir bezeichnen im folgenden reelle abgeschlossene Intervalle mit $X = [x_1, x_2]$, $Y = [y_1, y_2], \ldots$. Verknüpfungen zwischen Intervallen sind definiert durch

$X * Y = \{z = x * y \mid x \in X, y \in Y\}, * \in \{+, -, \cdot, /\}.$

Bei der Division ist $0 \in Y$ ausgeschlossen. Die Menge $I(R)$ der Intervalle ist bezüglich dieser Verknüpfungen abgeschlossen, und das Ergebnis läßt sich durch die Schranken von X und Y ausdrücken. Die reelle Zahl $x \in R$ läßt sich als spezielles Intervall der Form $X = [x, x]$ auffassen. R bildet daher eine Teilmenge von $I(R)$. Im folgenden verwenden wir daher für die Verknüpfung von reellen Intervallen und/oder reellen Zahlen die gleichen Symbole. Sämtliche Verknüpfungen von Intervallen genügen der Teilmengeneigenschaft, d. h.

$A_1 \subset A_2, B_1 \subset B_2 \Rightarrow A_1 * B_1 \subset A_2 * B_2, * \in \{+, -, \cdot, /\}.$

Unter dem Durchmesser $d(X)$ eines Intervalles X verstehen wir

$d(X) = x_2 - x_1.$

Wegen weiterer Einzelheiten aus der Intervallrechnung siehe etwa [21] und [22].

3. Algorithmus

Das charakteristische Polynom $p(\lambda)$ von A läßt sich in der Form

$$p(\lambda) = \prod_{j=1}^{n} (\lambda - \lambda_j)$$

schreiben, wobei

$\lambda_1, \lambda_2, \ldots, \lambda_n$

die Eigenwerte von A sind. Daraus erhält man

$$\lambda_i = \lambda - \frac{p(\lambda)}{\prod\limits_{\substack{j=1 \\ j \neq i}}^{n} (\lambda - \lambda_j)}.$$

Es gelte

$\lambda_i \in X_i^{(0)} = [x_{i,1}^{(0)}, x_{i,2}^{(0)}], i = 1(1)n.$

In jedem der $n - m$ paarweise disjunkten Intervalle

$X_1^{(0)}, X_2^{(0)}, \ldots, X_{n-m}^{(0)}$

liege ein einfacher Eigenwert λ_i, $i = 1(1)n - m$.

Die restlichen m Eigenwerte mögen in den Intervallen

$X_{n-m+1}^{(0)}, X_{n-m+2}^{(0)}, \ldots, X_n^{(0)}$

liegen, und es sei

$X_i^{(0)} \cap X_j^{(0)} = \emptyset, i \leqslant n-m, j \geqslant n-m+1.$

Für $i \leqslant n - m$ setzen wir

$\lambda_i^{(0)} = \frac{1}{2} (x_{i,1}^{(0)} + x_{i,2}^{(0)}).$

Wegen $\lambda_j \in X_j^{(0)}$, $j = 1(1)n$, gilt dann

$$\prod_{\substack{j=1 \\ j \neq i}}^{n} (\lambda_i^{(0)} - \lambda_j) \in \prod_{\substack{j=1 \\ j \neq i}}^{n} (\lambda_i^{(0)} - X_j^{(0)}),$$

und aufgrund der Voraussetzungen über die gegenseitige Lage der Intervalle $X_j^{(0)}$, $j = 1(1)n$, enthält das rechts stehende Intervall nicht die Null. Somit gilt

$$\lambda_i = \lambda_i^{(0)} - \frac{p(\lambda_i^{(0)})}{\prod\limits_{\substack{j=1 \\ j \neq i}}^{n} (\lambda_i^{(0)} - \lambda_j)}$$

$$\in \{\lambda_i^{(0)} - \frac{p(\lambda_i^{(0)})}{\prod\limits_{\substack{j=1 \\ j \neq i}}^{n} (\lambda_i^{(0)} - X_j^{(0)})}\} \cap X_i^{(0)} .$$

Somit erhält man durch den letzten Ausdruck ein neues Intervall, welches λ_i enthält. Diese Tatsache gibt Anlaß zu den beiden folgenden Iterationsverfahren.

1. Gesamtschrittverfahren

$$\text{(I)} \quad \begin{cases} \lambda_i^{(k)} = \frac{1}{2}(x_{i,1}^{(k)} + x_{i,2}^{(k)}), \\ X_i^{(k+1)} = \{\lambda_i^{(k)} - \dfrac{p(\lambda_i^{(k)})}{\prod\limits_{\substack{j=1 \\ j \neq i}}^{n-m} (\lambda_i^{(k)} - X_j^{(k)}) \prod\limits_{j=n-m+1}^{n} (\lambda_i^{(k)} - X_j^{(0)})}\} \cap X_i^{(k)}, \\ i = 1(1)n-m,\ k = 0, 1, 2, \ldots . \end{cases}$$

2. Einzelschrittverfahren

$$\text{(II)} \quad \begin{cases} \lambda_i^{(k)} = \frac{1}{2}(x_{i,1}^{(k)} + x_{i,2}^{(k)}), \\ X_i^{(k+1)} = \{\lambda_i^{(k)} - \dfrac{p(\lambda_i^{(k)})}{\prod\limits_{j=1}^{i-1} (\lambda_i^{(k)} - X_j^{(k+1)}) \prod\limits_{j=i+1}^{n-m} (\lambda_i^{(k)} - X_j^{(k)}) \prod\limits_{j=n-m+1}^{n} (\lambda_i^{(k)} - X_j^{(0)})}\} \cap X_i^{(k)}, \\ i = 1(1)n-m,\ k = 0, 1, 2, \ldots . \end{cases}$$

Es gilt der folgende

Satz 1: Unter den getroffenen Voraussetzungen über die Lage der Intervalle $X_i^{(0)}$, $i = 1(1)n$, gilt bei beiden Verfahren

$$\lambda_i \in X_i^{(k)} \text{ und } \lim_{k \to \infty} d(X_i^{(k)}) = 0$$

für $i = 1(1)n-m$.

Diese Aussagen wurden in [5] für $m = 0$ bewiesen. Der Beweis für den Fall $m \neq 0$ erfolgt vollständig analog.

Man erhält also in jedem Iterationsschritt für die Eigenwerte λ_i, $i = 1(1)n-m$, obere und untere Schranken. Diese Schranken konvergieren monoton gegen den jeweiligen Eigenwert, d. h. es gilt $X_i^{(0)} \supset X_i^{(1)} \supset X_i^{(2)} \supset \ldots$ und $\lim\limits_{k \to \infty} X_i^{(k)} = \lambda_i$, $i = 1(1)n-m$.

Die Voraussetzungen für die Konvergenz der beiden angegebenen Verfahren sind sehr schwach. Üblicherweise besitzt man bei ähnlichen Verfahren nur lokale Konvergenzaussagen. (Siehe dazu insbesondere [1], [7], [11], [12], [18], [20], [26].)

Für spätere Zwecke benötigen wir noch das folgende

Korollar:

Gilt für mindestens ein i, $1 \leqslant i \leqslant n$,

$\lambda_i \notin X_i^{(0)}$,

so gibt es eine natürliche Zahl k^* und einen Index i^*, $1 \leqslant i^* \leqslant n-m$, so daß der beim Gesamtschrittverfahren bzw. Einzelschrittverfahren für $k = k^*$ und $i = i^*$ zu bildende Durchschnitt leer ist.

Der einfache Beweis erfolgt durch Herstellung eines Widerspruches. Mit Hilfe dieses Korollars läßt sich einfach kontrollieren, ob die verwendeten Einschließungsintervalle $X_i^{(0)}$, $i = 1(1)n$, tatsächlich die entsprechenden Eigenwerte enthalten.

Das Konvergenzverhalten der beiden Verfahren wird durch folgenden Satz charakterisiert.

Satz 2: Ist $m = 0$ oder

$d(X_i^{(0)}) = 0$, $i = n-m+1, n-m+2, \ldots, n$,

so konvergieren die Durchmesser

$$d(X_i^{(k)}) = x_{i,2}^{(k)} - x_{i,1}^{(k)},\ i = 1(1)\ n-m,$$

beim Gesamtschrittverfahren mindestens quadratisch, beim Einzelschrittverfahren mindestens von der Ordnung $1+\sigma_n > 2$ gegen Null. Dabei ist $\sigma_n > 1$ die eindeutige positive Wurzel des Polynoms

$$p_n(\sigma) = \sigma^n - \sigma - 1.$$

Andernfalls konvergieren beide Verfahren im allgemeinen nur linear.

Als Konvergenzordnung eines Iterationsverfahrens I mit dem Grenzwert x^* verstehen wir dabei (siehe [23], Seite 290) die Größe

$$O_R(I, x^*) = \begin{cases} \infty & \text{falls } R_p(I, x^*) = 0\ \forall p \in [1, \infty) \\ \inf\{p \in [1, \infty) \mid R_p(I, x^*) = 1\} & \text{sonst} \end{cases}$$

mit

$$R_p(I, x^*) = \sup\{R_p\{x^{(k)}\} \mid \{x^{(k)}\} \in C(I, x^*)\},\ 1 \leqslant p < \infty$$

und

$$R_p\{x^k\} = \begin{cases} \lim\limits_{k\to\infty} \sup |x^{(k)} - x^*|^{\frac{1}{k}}, & p = 1 \\ \lim\limits_{k\to\infty} \sup |x^{(k)} - x^*|^{\frac{1}{p^k}}, & p > 1. \end{cases}$$

($C(I, x^*)$ bezeichnet die Menge aller Folgen, die man mit I berechnen kann und die gegen x^* konvergieren.)

Satz 2 besagt, daß man bei sämtlich einfachen Eigenwerten das Einzelschrittverfahren vorziehen kann. Das gleiche ist richtig, falls mehrfache Eigenwerte in Intervalle der Länge Null eingeschlossen, d. h. exakt berechnet sind. Beim praktischen Rechnen muß man sich damit begnügen, mehrfache Eigenwerte im Rahmen der Rechengenauigkeit einzuschließen.

Wir wollen nun das angegebene Einzelschrittverfahren (II) noch etwas modifizieren. Dazu betrachten wir einige Eigenschaften dieses Verfahrens etwas näher. Zunächst führen wir zur Vereinfachung der Schreibweise in (II) folgende Bezeichnung ein:

$$W_i^{(k)} = \prod_{j=1}^{i-1} (\lambda_i^{(k)} - X_j^{(k+1)}) \prod_{j=i+1}^{n-m} (\lambda_i^{(k)} - X_j^{(k)}) \prod_{j=n-m+1}^{n} (\lambda_i^{(k)} - X_j^{(0)}).$$

Damit läßt sich (II) schreiben als

$$(II') \quad \begin{cases} \lambda_i^{(k)} = \frac{1}{2}(x_{i,1}^{(k)} + x_{i,2}^{(k)}), \\ X_i^{(k+1)} = \{\lambda_i^{(k)} - \dfrac{p(\lambda_i^{(k)})}{W_i^{(k)}}\} \cap X_i^{(k)}, \\ i = 1, 2, \ldots, n-m,\ k = 0, 1, 2, \ldots \end{cases}$$

Aus dem vorher Gesagtem folgt, daß stets

$$w_{i,1}^{(k)} \cdot w_{i,2}^{(k)} > 0,\ i = 1, 2, \ldots, n-m,\ k = 0, 1, 2, \ldots,$$

erfüllt sein muß. Außerdem folgert man für (II') bzw. (II) – in Analogie zum intervallmäßigen Newton-Verfahren bei [22] – die Eigenschaft

$$\lambda_i^{(k)} \notin X_i^{(k+1)}$$

und darüberhinaus noch

$$d(X_i^{(k+1)}) \leqslant \tfrac{1}{2}\, d(X_i^{(k)})\,(1 - q_i^{(k)}),$$

$$\text{mit } q_i^{(k)} = \begin{cases} \dfrac{w_{i,1}^{(k)}}{w_{i,2}^{(k)}} & \text{falls } w_{i,1}^{(k)} > 0, \\ \dfrac{w_{i,2}^{(k)}}{w_{i,1}^{(k)}} & \text{falls } w_{i,2}^{(k)} < 0, \end{cases}$$

$$i = 1, 2, \ldots, n-m, \qquad k = 0, 1, 2, \ldots .$$

Diese Eigenschaften stellen eine Verschärfung von Aussagen dar, die beim intervallmäßigen Newtonverfahren zur Lösung nichtlinearer Gleichungssysteme in [3] nur für jeweils mindestens einen Index i ($1 \leqslant i \leqslant n-m$) bewiesen werden konnten.

Aufgrund der Definition der Intervalle $W_\nu^{(k)}$ folgt auch

$$\operatorname{sign}(W_\nu^{(0)}) = \operatorname{sign}(W_\nu^{(1)}) = \ldots\ldots,\ \nu = 1, 2, \ldots, n-m,$$

wobei für ein Intervall $W_\nu^{(k)}$ die Definition

$$\operatorname{sign}(W_\nu^{(k)}) = \begin{cases} 1 & \text{falls } w_{\nu,1}^{(k)} > 0, \\ -1 & \text{falls } w_{\nu,2}^{(k)} < 0, \\ 0 & \text{sonst,} \end{cases}$$

gilt. Aus diesen aufgezählten Eigenschaften läßt sich nun herleiten, daß in (II') bzw. (II) der fragliche Eigenwert λ_ν schon im Intervall

$$Y_\nu^{(k+1)} = \begin{cases} [x_{\nu,1}^{(k+1)}, \lambda_\nu^{(k+1)}] & \text{falls } \operatorname{sign}(W_\nu^{(k)})\operatorname{sign}(p(\lambda_\nu^{(k+1)})) > 0, \\ [\lambda_\nu^{(k+1)}, x_{\nu,2}^{(k+1)}] & \text{falls } \operatorname{sign}(W_\nu^{(k)})\operatorname{sign}(p(\lambda_\nu^{(k+1)})) < 0 \end{cases}$$

enthalten ist ([19]). Dabei gilt

$$d(Y_\nu^{(k+1)}) = \tfrac{1}{2}\, d(X_\nu^{(k+1)}),\ k = 0, 1, 2, \ldots .$$

Bei der Berechnung der Ausdrücke $W_\nu^{(k)}$, $i < \nu \leqslant n-m$, und der $W_\nu^{(k+1)}$, $1 \leqslant \nu < i$, benutzen wir daher die Intervalle $Y_\nu^{(k+1)}$ anstelle der $X_\nu^{(k+1)}$. Das führt dann auf die folgende Iterationsvorschrift:

(III)
$$
\begin{cases}
\lambda_i^{(0)} = \frac{1}{2}\,(x_{i,1}^{(0)} + x_{i,2}^{(0)}), \\
Y_i^{(0)} = X_i^{(0)}, \\
X_i^{(k+1)} = \{\lambda_i^{(k)} - \dfrac{p(\lambda_i^{(k)})}{W_i^{(k)}}\} \cap X_i^{(k)}, \\
\text{mit } W_i^{(k)} = \prod_{j=1}^{i-1} (\lambda_i^{(k)} - Y_j^{(k+1)}) \prod_{j=i+1}^{n-m} (\lambda_i^{(k)} - Y_j^{(k)}) \prod_{j=n-m+1}^{n} (\lambda_i^{(k)} - Y_j^{(0)}), \\
\lambda_i^{(k+1)} = \frac{1}{2}\,(x_{i,1}^{(k+1)} + x_{i,2}^{(k+1)}), \\
Y_i^{(k+1)} = \begin{cases} [x_{i,1}^{(k+1)}, \lambda_i^{(k+1)}] \text{ falls } \operatorname{sign}(W_i^{(k)}) \operatorname{sign}(p(\lambda_i^{(k+1)})) > 0, \\ [\lambda_i^{(k+1)}, x_{i,2}^{(k+1)}] \text{ falls } \operatorname{sign}(W_i^{(k)}) \operatorname{sign}(p(\lambda_i^{(k+1)})) < 0, \\ X_i^{(k+1)} \quad \text{sonst,} \end{cases} \\
i = 1, 2, \ldots, n-m, \qquad k = 0, 1, 2, \ldots \; .
\end{cases}
$$

Man kann leicht zeigen, daß für Verfahren (III) analoge Aussagen gelten, wie sie für Verfahren (II) in den Sätzen 1 und 2 sowie im Korollar angegeben wurden. Verfahren (III) geht aus Verfahren (II) ohne zusätzlichen Rechenaufwand hervor. Es läßt sich beweisen, daß die Durchmesserabschätzungen der Iterationen gemäß Vorschrift (III) um den Faktor $\frac{1}{2}$ besser ausfallen als bei (II), und wir verwenden deshalb Verfahren (III).

4. Die Prozedur *eigverb*

Bei der praktischen Durchführung von Verfahren (III) müssen alle auftretenden Operationen durch geeignete Maschinenoperationen ausgeführt werden. Wir verwenden im nachfolgenden Programm dazu die in [8] angegebenen ALGOL-Prozeduren

low(x)

zur Abrundung von Maschinenverknüpfungsresultaten (wobei man durch -low(-x) die entsprechende Aufrundung erhält) und

mul(a, b, c)

zur Ausführung von Intervallmultiplikationen. Ganz ähnliche Prozeduren finden sich auch bei [10] und [15]. Die restlichen und einfacher zu realisierenden Intervallverknüpfungen wurden, soweit sie im angegebenen Programm auftreten, ausprogrammiert.

Es ist jedoch zu beachten, daß auch alle reellen Verknüpfungen in (III) als Verknüpfungen von Punktintervallen ausgeführt werden müssen. Nur so kann unter Berücksichtigung aller auftretenden Rundungsfehler die Einschließung der Eigenwerte durch die berechneten iterierten Intervalle gesichert werden. So ergibt z. B. $p([\lambda_i^{(k)}, \lambda_i^{(k)}])$ als intervallmäßige Funktionsauswertung ein Einschließungsintervall für den reellen Wert $p(\lambda_i^{(k)})$. In (III) hat man dann die vorgesehene Fallunterscheidung mit der Funktion

$\operatorname{sign}(p([\lambda_i^{(k)}, \lambda_i^{(k)}]))$,

deren Werte oben erklärt wurden, durchzuführen. Die Elemente der vorgegebenen Matrix sind im angegebenen Programm aus denselben Gründen auch als Intervalle angenommen worden.

Wird Verfahren (III) in der eben beschriebenen Weise ausgeführt, so erhält man eine Folge von Intervallen

$X_i^{(0)} \supset X_i^{(1)} \supset X_i^{(2)} \supset \ldots\ldots \supset X_i^{N(i)} = X_i^{N(i)+1} = \ldots$

$i = 1, 2, \ldots, n-m,$

die nach jeweils endlich vielen Schritten nicht mehr verändert werden und von denen jedes den fraglichen Eigenwert λ_i einschließt. In der Prozedur *eigverb* wird deshalb die Einschließung für den Eigenwert λ_i nicht mehr neu berechnet, sobald

$X_i^{(n)} = X_i^{(n+1)}$

für ein n eingetreten ist.

Als formale Parameter der Prozedur *eigverb* treten auf:

n — Dieser ganzzahlige Parameter gibt den Grad n der behandelten Matrix an.

a — Dieser Parameter steht für den Vektor der Diagonalelemente der Matrix. Die einzelnen Feldkomponenten a[i, 1], a[i, 2] stehen dabei für die intervallmäßigen Diagonalelemente $[a_{i,1}, a_{i,2}]$, $i = 1, 2, \ldots, n$.

b — Der Parameter b steht für den Vektor der Nebendiagonalelemente, und die Feldkomponenten b[i,1], b[i,2] stehen für die Intervallelemente der Nebendiagonalen $[b_{i,1}, b_{i,2}]$, $i = 1, 2, \ldots, n-1$.

x — Der formale Parameter x mit den Feldkomponenten x[i, 1], x[i, 2] bezeichnet den Vektor der Einschließungsintervalle $[x_{i,1}, x_{i,2}] \ni \lambda_i$, $i = 1,$

2, . . . , n. Beim Aufruf der Prozedur handelt es sich dabei um die vorgegebenen Einschließungen $X_i^{(0)}$, i = 1, 2, . . . , n. Nach der Ausführung des Prozeduraufrufs steht x für die nach (III) verbesserten Intervalle.

bo Dieser Parameter steht für einen Booleschen Vektor mit den Komponenten bo[i], i = 1, 2, . . . , n. Dabei hat bo[i] den Wert **true** falls die Einschließung für λ_i im nächsten Iterationsschritt neu zu berechnen ist. Andernfalls hat bo[i] den Wert **false**. Stimmen zwei aufeinanderfolgende Einschließungsintervalle eines Eigenwertes bei Durchführung der Iteration überein, so erhält die entsprechende Komponente von bo den Wert **false**. Beim Aufruf von *eigverb* müssen also alle jene Komponenten des korrespondierenden aktuellen Vektors zu bo den Wert **true** besitzen, deren entsprechende Eigenwerteinschließung nach (III) verbessert werden soll. Alle übrigen Komponenten haben den Wert **false**. Bei Beendigung des Prozeduraufrufs sind also sämtliche Komponenten des bo entsprechenden aktuellen Parameters vom Wert **false**.

eps Der Parameter eps steht für eine reelle Konstante $\epsilon > 0$. Erfüllt der Durchmesser eines Einschließungsintervalls

$$d(X_i^{(k)}) = x_{i,2}^{(k)} - x_{i,1}^{(k)} < \epsilon \cdot \max\{|x_{i,1}^{(k)}|, |x_{i,2}^{(k)}|\},$$

dann erhält bo[i] den Wert **false** zugewiesen, d. h. die Einschließung wird nicht mehr verbessert. Kann die Schranke ϵ nicht unterschritten werden (etwa im Falle $\epsilon = 0$), so wird der Abbruch der Rechnung durch das bei der Beschreibung von bo angegebene Kriterium in jedem Falle erreicht.

empty Die hiermit bezeichnete Marke wird angesprungen, falls während der Durchführung von (III) die Durchschnittsbildung für einen Index i auf die leere Menge führt. Das tritt dann ein, wenn $\lambda_i \notin X_i^{(0)}$ für ein i war.

exit Die hiermit bezeichnete Marke wird angesprungen, falls einer der aktuell eingesetzten Intervallvektoren für die Parameter a, b und x eine Komponente enthält, bei der die obere Schranke kleiner als die untere ist. Das gleiche geschieht, wenn die einzelnen Einschließungsintervalle nicht in der eingangs erwähnten Weise paarweise disjunkt sind. Hier wurde im Programm vorausgesetzt, daß die Eigenwerte nach aufsteigender Größe numeriert sind. Sollte aus numerischen Gründen eine andere Numerierung vorteilhafter sein, dann ist das Programmstück

```
for i:=1 step 1 until n – 1 do
if (bo[i] or bo[i+1]) and x[i,2] notless x[i+1,1]
then goto exit;
```

im Prozedurrumpf sinngemäß abzuändern.

Globale Größen: Die Prozedur benutzt als globale Größen die erwähnten Prozeduren low und mul.

ALGOL-Prozedur:

```
procedure eigverb (n,a,b,x,bo,eps,empty,exit);
        value n,eps;
        integer n;
        real eps;
        array a,b,x;
        boolean array bo;
        label empty,exit;

begin
    integer i,j;
    real n1,n2,u,v,y,z;
    boolean bit;
    array bb,f[1:n,1:2], xm,d[1:n],h,l,m,fl,f2[1:2];

comment eingangsdaten;
for i:=1 step 1 until n do
begin
    if a[i,2] less a[i,1] or b[i,2] less b[i,1]
    then goto exit;
    h[1]:= b[i,1];
    h[2]:= b[i,2];
    mul(h,h,h);
    bb[i,1]:= h[1];
    bb[i,2]:= h[2]
end i;

for   i:= 1 step 1 until n – 1 do
if (bo[i] or bo[i+1]) and x[i,2] notless x[i+1, 1]
then goto exit;

comment startwerte;
for i:= 1 step 1 until n do
if bo[i] then
begin
    xm[i]:=(x[i,1]+x[i,2])/2;
    d[i]:=–low(x[i,1]–x[i,2]);
    f1[1]:= f1[2]:= 1;
    f2[1]:= low(xm[i]– a[1,2]);
    f2[2]:= – low(a[1,1]– xm[i]);
    for   j := 2 step 1 until n do
    begin
        l[1]:=bb[j–1,1];
        l[2]:=bb[j–1,2];
        mul(l,f1,m);
        l[1]:=low(xm[i]–a[j,2]);
        l[2]:=– low(a[j,1]–xm[i]);
        mul(l,f2,l);
        f1[1]:=f2[1];
        f1[2]:=f2[2];
        f2[1]:=low(l[1]–m[2]);
        f2[2]:=–low(m[1]–l[2])
    end j;
    f[i,1]:= f2[1];
    f[i,2]:= f2[2]
end i;

comment iteration;
repeat:
bit:= false;
for i:=1 step 1 until n do
if bo[i] then
begin
    comment nenner;
    h[1]:= h[2]:= 1;
    for j:= 1 step 1 until i–1, i+1 step 1 until n do
    begin
        l[1]:= low(xm[i]–x[j,2]);
        l[2]:=–low(x[j,1]–xm[i]);
        mul(h,l,h)
    end   j;
```

```
comment newton;
n1:=if h[1]greater 0
    then
    (if f[i,1]less 0 then low(f[i,1]/h[1])
                     else low(f[i,1]/h[2]))
    else
    if f[i,2]greater 0 then low(f[i,2]/h[2])
                       else low(f[i,2]/h[1]);
n2:=if h[1]greater 0
    then
    (if f[i,2]greater 0 then -low(-f[i,2]/h[1])
                        else -low(-f[i,2]/h[2]))
    else
    if f[i,1]less 0 then -low(-f[i,1]/h[2])
                    else - low(-f[i,1]/h[1]);
y:=n1;
n1:=low(xm[i]-n2);
n2:=-low(y-xm[i]);
if n1 less x[i,1] then n1:=x[i,1];
if n2 greater x[i,2] then n2:=x[i,2];
if n2 less n1 then goto empty;

comment halbierung;
xm[i]:=(n1+n2)/2;
f1[1]:=f1[2]:=1;
f2[1]:=low(xm[i]-a[1,2]);
f2[2]:=-low(a[1,1]-xm[i]);
for j := 2 step 1 until n do
begin
    l[1]:=bb[j-1,1];
    l[2]:=bb[j-1,2];
    mul(l,f1,m);
    l[1]:=low(xm[i]-a[j,2]);
    l[2]:=-low(a[j,1]-xm[i]);
    mul(l,f2,l);
    f1[1]:=f2[1];
    f1[2]:=f2[2];
    f2[1]:=low(l[1]-m[2]);
    f2[2]:=-low(m[1]-l[2])
end j;
f[i,1]:=f2[1];
f[i,2]:=f2[2];
if f[i,1]greater 0
then
   begin
      if h[1]greater 0
      then
         begin
            x[i,2]:=xm[i];
            x[i,1]:=n1
         end
      else
         begin
            x[i,1]:=xm[i];
            x[i,2]:=n2
         end
   end
else
if f[i,2]less 0
then
   begin
      if h[1]greater 0
      then
         begin
            x[i,1]:=xm[i];
            x[i,2]:=n2
         end
      else
         begin
            x[i,2]:=xm[i];
            x[i,1]:=n1
         end
      end
   else
      begin
         x[i,1]:=n1;
         x[i,2]:=n2
      end;

   comment test;
   z:=-low(x[i,1]-x[i,2]);
   u:=abs(x[i,1]);
   v:=abs(x[i,2]);
   if v less u then v:=u;
   if z notless d[i] or z notgreater eps X v
   then bo[i]:=false else bit:=true;
   d[i]:=z
end iteration;
if bit then goto repeat
end procedure
```

5. Numerische Beispiele

Wir wollen zur Illustration hier einige der mit Hilfe der Prozedur *eigverb* gerechneten numerischen Beispiele wiedergeben. Die dazu nötigen Rechnungen wurden an der elektronischen Rechenanlage EL X8 am Rechenzentrum der Universität Karlsruhe ausgeführt. Die Mantissenlänge der dort vorhandenen Gleitkommazahlen beträgt 40 Bit.

a) Als erstes Beispiel wählen wir die Matrix

$$A = \begin{pmatrix} -2 & 0.5 & 0 \\ 0.5 & 0 & 0.7 \\ 0 & 0.7 & 2 \end{pmatrix}.$$

Die angegebenen Intervalle $X_i^{(0)}$, $1 \leqslant i \leqslant 3$, sind Obermengen der nach GERSCHGORIN berechneten und enthalten die Eigenwerte von A. Wir geben im folgenden die einzelnen Iterationsschritte bis zum Stillstand an ($\epsilon = 0$).

Startwerte:

$X_1^{(0)} = [-0.3500000000000_{10}1,\ -0.1500000000000_{10}1]$
$X_2^{(0)} = [-0.1200000000000_{10}1,\ \ 0.1200000000000_{10}1]$
$X_3^{(0)} = [\ 0.1300000000000_{10}1,\ \ 0.2700000000000_{10}1]$

1. Schritt:

$X_1^{(1)} = [-0.2278846153855_{10}1,\ -0.1958755060724_{10}1]$
$X_2^{(1)} = [-0.1332574834133_{10}0,\ -0.7801218940221_{10}-1]$
$X_3^{(1)} = [\ 0.2214726735652_{10}1,\ \ 0.2226492877257_{10}1]$

2. Schritt:

$X_1^{(2)} = [-0.2125267660522_{10}1,\ -0.2122551286407_{10}1]$
$X_2^{(2)} = [-0.1015013969698_{10}0,\ -0.1014000393717_{10}0]$
$X_3^{(2)} = [\ 0.2226122470285_{10}1,\ \ 0.2226122593995_{10}1]$

3. Schritt:

$X_1^{(3)} = [-0.2124636207984_{10}1,\ -0.2124636192154_{10}1]$
$X_2^{(3)} = [-0.1014863409916_{10}0,\ -0.1014863409816_{10}0]$
$X_3^{(3)} = [\ 0.2226122537849_{10}1,\ \ 0.2226122537861_{10}1]$

4. Schritt:

$X_1^{(4)} = [-0.2124636196874_{10}1,\ -0.2124636196866_{10}1]$
$X_2^{(4)} = [-0.1014863409890_{10}0,\ -0.1014863409842_{10}0]$
$X_3^{(4)} = [\ 0.2226122537849_{10}1,\ \ 0.2226122537861_{10}1]$

5. Schritt:

$X_1^{(5)} = [-0.2124636196874_{10}1,\ -0.2124636196866_{10}1]$
$X_2^{(5)} = [-0.1014863409890_{10}0,\ -0.1014863409842_{10}0]$
$X_3^{(5)}$ wurde nicht mehr neu berechnet.

b) Für n = 30 wählen wir die z. B. in [31] S. 254 ff. behandelte Matrix mit den Elementen

$a_i = i^4, \quad 1 \leqslant i \leqslant 30,$
$b_i = i, \quad 1 \leqslant i < 30.$

Die Intervalle $X_i^{(0)} = [i(i^3-2)+1,\ i(i^3+2)-1],\ 1 \leqslant i \leqslant 30$, schließen als Obermengen der nach dem Satz von GERSCHGORIN berechneten Intervalle die Eigenwerte der Matrix ein. Nach 2 Iterationsschritten für 3 Einschließungsintervalle, 3 Schritten für 25 Einschließungsintervalle und 4 Schritten für die restlichen 2 Einschließungsintervalle ergab sich bei $\epsilon = 0$ folgendes Ergebnis (vergleiche auch [31] S. 255):

$X_1 = [0.9334070848644_{10}0,\ \ 0.9334070848673_{10}0]$
$X_2 = [0.1600506537031_{10}2,\ \ 0.1600506537036_{10}2]$
$X_3 = [0.8101010054542_{10}2,\ \ 0.8101010054561_{10}2]$
$X_4 = [0.2560080668913_{10}3,\ \ 0.2560080668923_{10}3]$
$X_5 = [0.6250061023711_{10}3,\ \ 0.6250061023730_{10}3]$
$X_6 = [0.1296004678577_{10}4,\ \ 0.1296004678582_{10}4]$
$X_7 = [0.2401003670609_{10}4,\ \ 0.2401003670617_{10}4]$
$X_8 = [0.4096002945057_{10}4,\ \ 0.4096002945067_{10}4]$
$X_9 = [0.6561002410116_{10}4,\ \ 0.6561002410136_{10}4]$
$X_{10} = [0.1000000200625_{10}5,\ \ 0.1000000200630_{10}5]$
$X_{11} = [0.1464100169472_{10}5,\ \ 0.1464100169477_{10}5]$
$X_{12} = [0.2073600144973_{10}5,\ \ 0.2073600144981_{10}5]$
$X_{13} = [0.2856100125383_{10}5,\ \ 0.2856100125392_{10}5]$
$X_{14} = [0.3841600109485_{10}5,\ \ 0.3841600109495_{10}5]$
$X_{15} = [0.5062500096410_{10}5,\ \ 0.5062500096429_{10}5]$
$X_{16} = [0.6553600085544_{10}5,\ \ 0.6553600085563_{10}5]$
$X_{17} = [0.8352100076399_{10}5,\ \ 0.8352100076418_{10}5]$
$X_{18} = [0.1049760006862_{10}6,\ \ 0.1049760006867_{10}6]$
$X_{19} = [0.1303210006199_{10}6,\ \ 0.1303210006203_{10}6]$
$X_{20} = [0.1600000005626_{10}6,\ \ 0.1600000005631_{10}6]$
$X_{21} = [0.1944810005130_{10}6,\ \ 0.1944810005135_{10}6]$
$X_{22} = [0.2342560004693_{10}6,\ \ 0.2342560004702_{10}6]$
$X_{23} = [0.2798410004311_{10}6,\ \ 0.2798410004321_{10}6]$
$X_{24} = [0.3317760003974_{10}6,\ \ 0.3317760003984_{10}6]$
$X_{25} = [0.3906250003678_{10}6,\ \ 0.3906250003689_{10}6]$
$X_{26} = [0.4569760003410_{10}6,\ \ 0.4569760003420_{10}6]$
$X_{27} = [0.5314410003174_{10}6,\ \ 0.5314410003194_{10}6]$
$X_{28} = [0.6146560002953_{10}6,\ \ 0.6146560002972_{10}6]$
$X_{29} = [0.7072810002764_{10}6,\ \ 0.7072810002783_{10}6]$
$X_{30} = [0.8100000081867_{10}6,\ \ 0.8100000081886_{10}6]$

c) Es wurden für die 100 × 100-Matrix die Diagonalelemente

$a_i = i, \quad 1 \leqslant i \leqslant 100,$

sowie die Nebendiagonalelemente

$b_i = 0.1, \quad 1 \leqslant i \leqslant 99$

vorgegeben. Die Intervalle $X_i = [i-0.79999,\ i+0.2]$, i = 1(1)100, sind Obermengen der nach GERSCHGORIN berechneten Intervalle und schließen sämtliche Eigenwerte der Matrix ein. Nach 5 Schritten für 96 der Einschließungsintervalle und 6 Schritten für die restlichen 4 Einschließungen kam die Iteration bei vorgegebenem $\epsilon = 0$ zum Stillstand. Der maximale relative Durchmesser

$$d = \max_{1 \leqslant i \leqslant 100} \frac{x_{i,2} - x_{i,1}}{\max\{|x_{i,1}|, |x_{i.2}|\}}$$

der resultierenden Einschließungen ergab sich dabei zu

$d = 0.35 \cdot 10^{-10}$.

d) Wir betrachten eine 14 × 14-Testmatrix, deren auf Tridiagonalgestalt transformierte Form etwa in [16], [28], [29] zu finden ist. Zum Zwecke der Fehlererfassung wurden die Matrizenelemente als echte Intervalle vorgegeben, wobei die in [16] angegebenen Zahlenwerte jeweils um eine Dezimaleinheit der letzten Stelle nach unten bzw. oben verändert wurden. Somit ergaben sich im einzelnen:

$A_1 = [0.249999999,\ \ 0.250000001]$
$A_2 = [0.768491172,\ \ 0.768491174]$
$A_3 = [0.919556755,\ \ 0.919556757]$
$A_4 = [0.230938894,\ \ 0.230938896]$
$A_5 = [0.133053787,\ \ 0.133053789]$
$A_6 = [0.222549574,\ \ 0.222549576]$
$A_7 = [0.116127855,\ \ 0.116127857]$
$A_8 = [0.120339372,\ \ 0.120339374]$
$A_9 = [0.123719911,\ \ 0.123719913]$
$A_{10} = [0.128561406,\ \ 0.128561408]$
$A_{11} = [0.107768088,\ \ 0.107768090]$
$A_{12} = [0.137039202,\ \ 0.137039204]$
$A_{13} = [0.138057029,\ \ 0.138057031]$
$A_{14} = [0.103796942,\ \ 0.103796944]$

und

$B_1 = [0.153660745,\ \ 0.153660747]$
$B_2 = [0.467260327,\ \ 0.467260329]$
$B_3 = [0.119256497,\ \ 0.119256499]$
$B_4 = [0.080763538,\ \ 0.080763540]$
$B_5 = [0.033947195,\ \ 0.033947197]$
$B_6 = [0.036090903,\ \ 0.036090905]$
$B_7 = [0.035022374,\ \ 0.035022376]$
$B_8 = [0.029157560,\ \ 0.029157562]$
$B_9 = [0.037453704,\ \ 0.037453706]$
$B_{10} = [0.016090598,\ \ 0.016090600]$
$B_{11} = [0.023826466,\ \ 0.023826468]$
$B_{12} = [0.029468448,\ \ 0.029468450]$
$B_{13} = [0.007646393,\ \ 0.007646395].$

Aufgrund der in der Literatur angegebenen Näherungen für die Eigenwerte der Matrix wurden folgende Einschließungsintervalle als Startwerte verwendet:

$X_1^{(0)} = [0.0, \quad 0.07]$
$X_2^{(0)} = [0.071, \quad 0.08]$
$X_3^{(0)} = [0.081, \quad 0.09]$
$X_4^{(0)} = [0.091, \quad 0.099]$
$X_5^{(0)} = [0.1, \quad 0.11]$
$X_6^{(0)} = [0.12, \quad 0.13]$
$X_7^{(0)} = [0.135, \quad 0.15]$
$X_8^{(0)} = [0.151, \quad 0.169]$
$X_9^{(0)} = [0.17, \quad 0.175]$
$X_{10}^{(0)} = [0.176, \quad 0.2]$
$X_{11}^{(0)} = [0.21, \quad 0.24]$
$X_{12}^{(0)} = [0.25, \quad 0.4]$
$X_{13}^{(0)} = [0.45, \quad 0.8]$
$X_{14}^{(0)} = [0.9, \quad 2.0]$.

Bei vorgegebenem $\epsilon = 0$ kam die Iteration für 2 Einschließungen nach 5 Schritten, für 7 nach 6 Schritten, für 3 nach 7 Schritten und für die restlichen 2 nach 8 Schritten zum Stillstand. Dabei ergaben sich im einzelnen folgende Resultate:

$X_1 = [0.6437975905537_{10} - 1, \quad 0.6438005909923_{10} - 1]$
$X_2 = [0.7359686569825_{10} - 1, \quad 0.7359737182715_{10} - 1]$
$X_3 = [0.8422510621749_{10} - 1, \quad 0.8422543065272_{10} - 1]$
$X_4 = [0.9720917735721_{10} - 1, \quad 0.9720925990014_{10} - 1]$
$X_5 = [0.1032157548572_{10} - 1, \quad 0.1032157658138_{10} \quad 0]$
$X_6 = [0.1227875161428_{10} \quad 0, \quad 0.1227875301207_{10} \quad 0]$
$X_7 = [0.1434228692123_{10} \quad 0, \quad 0.1434228895304_{10} \quad 0]$
$X_8 = [0.1663243939337_{10} \quad 0, \quad 0.1663248090367_{10} \quad 0]$
$X_9 = [0.1713071407377_{10} \quad 0, \quad 0.1713079789443_{10} \quad 0]$
$X_{10} = [0.1773561018011_{10} \quad 0, \quad 0.1773565711463_{10} \quad 0]$
$X_{11} = [0.2316394748722_{10} \quad 0, \quad 0.2316394930204_{10} \quad 0]$
$X_{12} = [0.2677332841326_{10} \quad 0, \quad 0.2677333086459_{10} \quad 0]$
$X_{13} = [0.4627661966269_{10} \quad 0, \quad 0.4627662067147_{10} \quad 0]$
$X_{14} = [0.1334034839710_{10} \quad 1, \quad 0.1334034845201_{10} \quad 1]$.

Es zeigt sich, daß von den in [16], [29] auf 16 Dezimalstellen angegebenen Näherungen für die Eigenwerte der obigen Matrix die Näherung für λ_{14} nicht in der berechneten Einschließung X_{14} liegt, d. h. von den angegebenen 16 Dezimalstellen können nur die ersten 8 richtig sein (vergleiche auch [28]). Da der in [16], [29] für λ_{14} angegebene Näherungswert in $X_{14}^{(0)}$ enthalten ist, müßte er andernfalls wegen der Teilmengeneigenschaft der Intervallverknüpfungen (siehe Abschnitt 2) auch in X_{14} liegen.

Literatur

[1] *O. Aberth:* Iteration methods for finding all zeros of polynomials simultaneously. Technical Report, Texas A & M University (1971), Math. Comp. 27, 339–334 (1973).

[2] *G. Alefeld:* Eine Modifikation des Newton-Verfahrens zur Bestimmung der reellen Nullstellen einer reellen Funktion. Z. Angew. Math. Mech. 50, T 32–T33 (1970).

[3] *G. Alefeld, J. Herzberger:* Über das Newton-Verfahren bei nichtlinearen Gleichungssystemen. Z. Angew. Math. Mech. 50, 773–774 (1970).

[4] *G. Alefeld, J. Herzberger:* Nullstelleneinschließung mit dem Newton-Verfahren ohne Invertierung von Intervallmatrizen. Numer. Math. 19, 56–64 (1972).

[5] *G. Alefeld, J. Herzberger:* Zur simultanen Berechnung von Polynomwurzeln durch monotone Folgen. Interner Bericht (1971), erscheint in ZAMM.

[6] *W. Barth:* Nullstellenbestimmung mit der Intervallrechnung. Computing 8, 320–328 (1971).

[7] *W. Börsch-Supan:* A Posteriori Error Bounds for the Zeros of Polynomials. Numer. Math. 5, 380–398 (1963).

[8] *H. Christ:* Realisierung einer Maschinenintervallarithmetik mit beliebigen ALGOL 60 Compilern. Elektron. Rechenanlagen 10, 217–222 (1968).

[9] *R. H. Dargel, F. R. Loscalzo, T. H. Witt:* Automatic Error Bounds on Real Zeros of Rational Functions. Comm. ACM 9, 806–809 (1966).

[10] *J. K. S. Dewar:* Procedure for Interval Arithmetic. Comp. J. 14, 447–450 (1970).

[11] *K. Dochev, P. Byrnev:* Certain Modifications of Newton's Method for the Approximate Solution of Algebraic Equations. Zh. Vych. Mat. 4, 915–920 (1964).

[12] *E. Durand:* Solutions Numeriques des Equations Algebriques (Tome I, Chap. IX). Paris: Masson et. Cie (1960).

[13] *L. W. Ehrlich:* A Modified Newton Method for Polynomials. Comm. ACM, 10, 107–108 (1967).

[14] *I. Gargantini, P. Henrici:* Circular Arithmetic and the Determination of Polynomial Zeros. Numer. Math. 18, 305–320 (1972).

[15] *D. I. Good, R. L. London:* Computer Interval Arithmetic: Definition and Proof of Correct Implementation. J. ACM. 17, 603–612 (1970).

[16] *R. T. Gregory, D. L. Karney:* A Collection of Matrices for Testing Computational Algorithms. New York: Wiley-Interscience (1969).

[17] *R. J. Hanson:* Automatic Error Bounds for Real Roots of Polynomials having Interval Coefficients. Comp. J. 13, 284–288 (1970).

[18] *P. Henrici:* Circular Arithmetic and the Determination of Polynomial Zeros. In "Conference on Application of Numerical Analysis", Lecture Notes in Mathematics 228, 96–92 (1971), Springer-Verlag.

[19] *J. Herzberger:* Bemerkungen zu einem Verfahren von R. E. MOORE. Z. Angew. Math. Mech. 53, 356–358 (1973).

[20] *I. O. Kerner:* Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen. Numer. Math. 8, 290–294 (1966).

[21] *U. Kulisch:* Grundzüge der Intervallrechnung. In Überblicke Mathematik 2 (1969), Bibliographisches Inst. Mannheim.

[22] *R. E. Moore:* Interval Analysis. Englewood Cliffs N. J.: Prentice Hall Inc.

[23] *J. M. Ortega, W. C. Rheinboldt:* Iterative Solution of Nonlinear Equations in Several Variables. New-York–London: Academic Press (1971).

[24] *G. Peters, J. H. Wilkinson:* Eigenvalues of Ax = λBx with band symmetric A and B. Comp. J. 12, 398–404 (1969).

[25] *G. Peters, J. H. Wilkinson:* Practical Problems Arising in the Solution of Polynomial Equations. J. Inst. Maths. Applics. 8, 16–35 (1971).

[26] *J. W. Schmidt, H. Dressel:* Fehlerabschätzungen bei Polynomgleichungen mit dem Fixpunktsatz von Brouwer. Numer. Math. 10, 42–50 (1967).

[27] *H. R. Schwarz, H. Rutishauser, E. Stiefel:* Matrizen Numerik. Stuttgart: Teubner Verlag (1968).

[28] *J. H. Wilkinson:* The Calculation of the Eigenvectors of Codiagonal Matrices. Comp. J. 1, 90–96 (1958).

[29] *J. H. Wilkinson:* The Evaluation of Zeros of Ill-conditioned Polynomials, Part II. Numer. Math. 1, 167–180 (1959).

[30] *J. H. Wilkinson:* The Algebraic Eigenvalue Problem. London: Oxford Clarendon Press (1965).

[31] *J. H. Wilkinson, C. Reinsch:* Handbook for Automatic Computation, Vol. II: Linear Algebra. Berlin-Heidelberg-New York: Springer-Verlag (1971).

[32] *G. Zielke:* ALGOL-Katalog Matrizenrechnung. München: R. Oldenbourg Verlag (1972).