MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK

GARCHING BEI MÜNCHEN

EIN VERFAHREN ZUR LÖSUNG DES STABILITÄTS-PROBLEMS FÜR KOMPLEXE MATRIZEN

(A METHOD OF SOLVING THE STABILITY PROBLEM FOR COMPLEX MATRICES)

R. Meyer-Spasche

IPP 6/104 February 1972

Die nachstehende Arbeit wurde im Rahmen des Vertrages zwischen dem Max-Planck-Institut für Plasmaphysik und der Europäischen Atomgemeinschaft über die Zusammenarbeit auf dem Gebiete der Plasmaphysik durchgeführt.

IPP 6/104 · R.Meyer-Spasche

A Method of Solving the Stability Problem for Complex Matrices

(February, 1972
in German)

Abstract

The problem of this paper is to determine the location of the eigenvalues of a general complex matrix. A method to solve this problem and its programming are given. Essentially, the method consists in finding a Hermitian matrix H which has the same inertia as the given matrix M, and in determining the inertia of H.H is found as the solution of the Liapunov equation $H\tilde{M} + \tilde{M}^{*}H = 2I$, where $\tilde{M}$ is a matrix in upper Hessenberg form which is similar to M. The method remains applicable even if the solution of the Liapunov equation is not computed correctly (caused by rounding errors), as long as a given criteria is satisfied. The Program has been run on an IBM 360/91 with satisfactory results. The purpose of this paper is twofold: On the one hand, it is to facilitate the handling of the existing program as a user manual and, on the other, it is to enable the reader to program the method at a minimum of expense.

Keywords: Matrix Investigations, Eigenvalue Problems, Stability of Equations, Programming.

Inhaltsverzeichnis

1. Einleitung

Es ist ein wichtiges Problem, das Verhalten eines physikalischen Systems in der Umgebung eines Gleichgewichtszustandes zu untersuchen. Wenn das System nach kleinen Störungen in seinen ursprünglichen Zustand zurückkehrt, so wird es stabil genannt. Wie z.B. in [1], Kap. 11, gezeigt wird, kann die Dgl

$$(1) \quad \dot{\vec{x}} = A\vec{x}, \ \vec{x}(o) = \vec{c}$$

oft zur Beschreibung des Systems in der Nähe der Gleichgewichtslage benutzt werden. Sein Verhalten ist dann durch die Eigenwerte von A bestimmt, wie folgender Satz zeigt ([1], Kap. 13):

Satz: Die Lösung von (1) strebt genau dann für jedes $\vec{c}$ mit $t \to \infty$ gegen o, wenn alle Eigenwerte von A negativen Realteil haben.

Eine Matrix, deren Eigenwerte alle negativen Realteil haben, heißt deshalb auch Stabilitätsmatrix. Die Aufgabe, die Vorzeichen der Realteile der Eigenwerte zu bestimmen, heißt Stabilitätsproblem.

Dieses Problem wollen wir im Folgenden näher untersuchen.Ausgegangen sind wir von einem Verfahren für reelle Matrizen, das in [4] beschrieben ist. Wir haben zunächst dieses Verfahren programmiert. Als die Ergebnisse recht zufriedenstellend ausfielen, haben wir das Verfahren auf komplexe Matrizen verallgemeinert. Dazu waren einige wesentliche Änderungen nötig.

2. Theoretische Grundlagen

$\mathbb{R}$ bezeichne den Körper der reellen Zahlen, $\mathbb{C}$ den Körper der komplexen Zahlen. Mit $L(\mathbb{K}^{n})$ wollen wir die Menge der nxn-Matrizen mit Elementen aus dem Körper $\mathbb{K}$ bezeichnen.

Def.1: Sei $M \in L(\mathbb{C}^{n})$. Mit In M wollen wir dann das Zahlentripel (l,r,v) bezeichnen, wobei l die Anzahl der Eigenwerte von M mit negativem Realteil, v die Anzahl der Eigenwerte mit verschwindendem Realteil und r die Anzahl der Eigenwerte mit positivem Realteil bezeichne.

Bekanntlich ist es sehr schwierig, für beliebige Matrizen M In M zu bestimmen. Wesentlich einfacher ist es, diese Aufgabe für spezielle Matrizen zu lösen, z.B. für hermitesche. Wir wollen deshalb so vorgehen, daß wir zu einer gegebenen Matrix M einer hermitesche Matrix H erzeugen, für die In M = In H gilt, und dann In H bestimmen. Das ist über die Lösung der Matrixgleichung

$$HM + M^{*}H = 2I$$

möglich, denn es gilt folgender Satz (siehe [7], § 4, S. 78):

Satz 1: Sei $M \in L(\mathbb{C}^n)$ mit den Eigenwerten $\lambda_1, \ldots, \lambda_m$.

Ist $\Delta(M) := \prod_{i,j=1}^{m} (\lambda_i + \overline{\lambda}_j) \neq 0$ und P eine gegebene positiv-definite hermitesche Matrix, so hat die Gleichung

(2) $$HM + M^{*}H = P$$

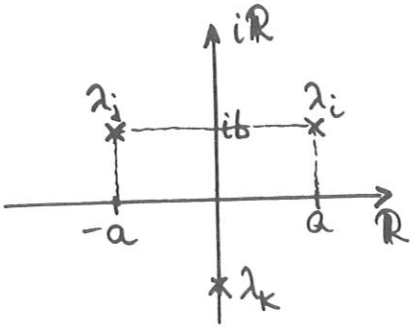
genau eine hermitesche Lösung H, und es ist In H = In M.

Nun weiß man natürlich nicht von vornherein, ob $\Delta(M) \neq 0$ ist. Das kann aber nicht zu falschen Ergebnissen führen, denn man weiß weiter (siehe [7], § 3, S. 76):

Satz 2: Sei $M \in L(\mathbb{C}^m)$. Notwendig und hinreichend für die Existenz einer hermiteschen Matrix H mit $HM + M^{*}H$ positiv definit ist, daß M keine rein imaginären Eigenwerte hat (dh $v(M) = 0$). Und dann gilt In H = In M.

Wenn man zu vorgegebenen P und M eine Lösung von (2) findet, gilt also immer In H = In M.

Wenn man keine Lösung von (2) findet, so bedeutet das, daß $\Delta(M) = 0$ ist, dh M hat zwei Eigenwerte λ_i und λ_j, für die $\lambda_i + \overline{\lambda}_j = 0$ gilt, oder einen Eigenwert λ_k mit $\lambda_k + \overline{\lambda}_k = 0$.



λ_j geht dann aus λ_i durch Spiegelung an der imaginären Achse hervor. λ_k ist rein imaginär.

In diesem Fall kann man wenigstens feststellen, ob M Eigenwerte mit positivem Realteil hat, jedoch nicht mehr die exakte Anzahl.

Dazu geht man so vor: Man unterwirft M einer Translation, dh man betrachtet statt M die Matrix $M - \mu I$, $\mu > 0$.

Die Eigenwerte von $M - \mu I$ gehen aus denen von M durch Subtraktion von μ hervor, wegen

$$\det (M - \mu I) = \det ((M - \mu I) - (\lambda - \mu) I).$$

Geht das Verfahren auch für $M - \mu I$ nicht, dh ist $\Delta(M - \mu I) = 0$, so hat $M - \mu I$ einen Eigenwert α mit $\operatorname{Re} \alpha \geq 0$, also M einen Eigenwert $\lambda := \alpha + \mu$ mit $\operatorname{Re} \lambda \geq \mu > 0$.

Ist $\Delta(M - \mu I) \neq 0$, so kann man $r(M - \mu I)$ ermitteln und weiß, daß $r(M) \geq r(M - \mu I)$ ist. Ist $r(M - \mu I) = 0$, so kann man dasselbe mit kleinerem μ noch einmal versuchen und eventuell so lange weitermachen, bis μ so klein ist, daß der Schluß $r(M) = r(M - \mu I) = 0$ gerechtfertigt ist.

In diesem Fall ist $\nu(M) \geq 1$.

Wir wollen uns jetzt einem Lösungsverfahren für die Gleichung (2) zuwenden.

Def. 2: Eine hermitesche Matrix H und eine reelle Diagonalmatrix D heißen Ljapunowpaar bzgl. M, wenn sie die Gleichung

(4) $$HM + M^{*}H = 2D$$

erfüllen.

In [6] findet man einen Beweis für das folgende Lemma:

Lemma: Zwei hermitesche Matrizen H und D, D diagonal, sind genau dann ein Ljapunowpaar bzgl. M, wenn es eine komplexe Matrix T, $T^{*} = -T$, gibt, s.d.

$$HM = T + D$$

gilt.

Diese Aussage erlaubt es, statt Gl. (4) die Gleichung

(5) $HM = T + D$, $T^{*} = -T$, $D^{*} = D$ diag.

zu lösen, was sehr viel einfacher ist, wenn M in oberer Hessenbergform gegeben ist. (dh $m_{ij} = o$ für $i > j + 1$) Dann kann man sich nämlich nach geeigneter Vorgabe der ersten Zeile von H eine eindeutig bestimmte Matrix H erzeugen, die sich von einer hermiteschen Matrix nur in der Hauptdiagonale unterscheidet, und die die Gleichung (5) löst. T und D sind durch Vorgabe der ersten Zeile von H eindeutig bestimmt.

Wie in [6] gezeigt ist, kann man mehrere Lösungen mit verschiedenen Anfangsbedingungen zu einer hermiteschen Lösung von (5) linear kombinieren. Auf diese Weise hat man sich dann ein Ljapunowpaar bzgl. M erzeugt. Nun gilt weiter (siehe [6], § 2).

Satz 3:

Ist $M \in L(\mathbb{C}^n)$ eine Matrix in oberer Hessenbergform mit nicht verschwindenden Elementen in der unteren Nebendiagonalen, und ist $\Delta(M) \neq 0$, so gibt es bzgl. M mindestens n über $\mathbb{R}$ linear unabhängige Ljapunowpaare $\{H_j, D_j\}_{j=1,\dots,n}$.

$\{H_j, D_j\}$, $j = 1, \dots, n$ wollen wir über $\mathbb{R}$ linear unabhängige Ljapunowpaare nennen, wenn

1) H_j, D_j für jedes j ein Ljapunowpaar ist,

2) $H_1, \dots, H_n$ über $\mathbb{R}$ linear unabhängig sind (dh $\sum_{j=1}^{n} \alpha_j H_j = 0, \alpha_j \in \mathbb{R} \Rightarrow \alpha_j = 0, j = 1, \dots, n$) und

3) $D_1, \dots, D_n$ über $\mathbb{R}$ linear unabhängig sind.

Ist $\Delta(M) \neq 0$ und M in der vorgeschriebenen Form, so kann man sich also linear unabhängige reelle Diagonalmatrizen $D_1, \dots, D_n$ erzeugen. Man hat nur darauf zu achten, daß die ersten Zeilen der zu erzeugenden H's voneinander linear unabhängig sind. Wir werden noch näher darauf eingehen. Das Gleichungssystem

(6) $\sum_{j=1}^{n} p_j D_j = D$

ist dann nicht singulär und besitzt also genau eine Lösung $(\rho_1, \ldots, \rho_m)$.

Wie man leicht nachrechnet, ist dann

(7) $H := \sum_{j=1}^{m} \rho_j H_j$

die gesuchte Lösung von (4).

Ist $\Delta(M) = 0$, so können zwei Fälle auftreten:

1) $\nu(M) \neq 0$. Dann sind die erzeugten $D_1, \ldots, D_m$ linear abhängig, obwohl die zugehörigen $H_1, \ldots, H_m$ linear unabhängig sind. Das folgt unmittelbar aus Satz 2: Wären die $D_1, \ldots, D_m$ linear unabhängig, könnte man (6) für $D = I$ lösen und hätte mit (7) im Widerspruch zu Satz 2 eine Lösung von (4).

2) $\nu(M) = 0$. In diesem Fall kann es vorkommen, daß Gleichung (6) lösbar ist. Dann kann man nach Satz 2 so weiterrechnen, wie wenn $\Delta(M) \neq 0$ ist. Andernfalls kann man eine Translation ausführen wie im Fall 1.

Hat man nun eine Lösung H von (4), so bleibt die Aufgabe, In H zu bestimmen. Zunächst sieht man leicht ein, daß H nicht singulär ist, also $\nu(H) = 0$.

Sei nämlich u ein Eigenvektor von H zum Eigenwert λ, $HM + M^*H = 2D$, D positiv definit. Dann ist

$$\begin{aligned} u^*(HM + M^*H)u &= \lambda u^*Mu + \lambda u^*M^*u \\ &= \lambda(u^*Mu + u^*M^*u) \\ &= 2\,u^*Du \\ &> 0, \end{aligned}$$

also ist $\lambda \neq 0$.

l (H) und r (H) kann man nach Satz 4 bestimmen, wenn man H durch eine Ähnlichkeitstransformation tridiagonal macht.

Satz 4:

Sei H eine hermitesche, tridiagonale Matrix,

$$H = \begin{pmatrix} a_1 & b_1 & & 0 \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ 0 & & b_{n-1} & a_n \end{pmatrix}.$$

Definiert man Zahlenfolgen $\{f_i\}, \{v_i\}, \{\rho_i\}$,

$i = 0, \ldots, n$, durch

$$f_0 := 1, \quad f_1 := a_1$$

$$f_i := a_i f_{i-1} - |b_{i-1}|^2 f_{i-2} \qquad i = 2, \ldots, n$$

$$v_0 := 0$$

$$v_i := \begin{cases} v_{i-1}+1 & \text{falls } f_i f_{i-1} < 0 \text{ oder } (f_i f_{i-1} = 0 \text{ und } f_i f_{i-2} < 0) \\ v_{i-1} & \text{sonst} \end{cases}$$

$$\rho_0 := 0$$

$$\rho_i := \begin{cases} \rho_{i-1}+1 & \text{falls } f_i f_{i-1} \neq 0 \\ \rho_{i-1}+2 & \text{falls } f_i \neq 0, f_{i-1} = 0, b_{i-1} \neq 0 \\ \rho_{i-1} & \text{sonst} \end{cases}$$

$$i = 1, \ldots, n$$

und ändert man f_i, $i=1, \ldots, n-1$, nach Berechnung

von ϱ_i und ν_i um in f_i : =1, falls $b_i = 0$, so ist

$$l(H) = \nu_n \qquad \text{und} \qquad r(H) = \varrho_n - \nu_n .$$

Zum Beweis braucht man einige Hilfssätze, die wir zunächst behandeln wollen.

Hilfssatz 1: Wenn alle b_i, $i = 1, \ldots, n-1$, ungleich Null sind, können keine zwei aufeinanderfolgenden f_i's verschwinden..

Beweis: Angenommen, $f_i = f_{i-1} = 0$ für das kleinstmögliche $i > 1$. Dann ist $f_{i-2} \neq 0$, also $f_i = a_i f_{i-1} - b_{i-1}\bar{b}_{i-1} f_{i-2} = -|b_{i-1}|^2 f_{i-2} \neq 0$ im Widerspruch zur Annahme.

Hilfssatz 2: Sind alle $b_i \neq 0$, $i=1, \ldots, n-1$, so ist

$$f_i = \det\begin{pmatrix} a_1 & \bar{b}_1 & & 0 \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & \bar{b}_{i-1} \\ 0 & & b_{i-1} & a_i \end{pmatrix} =: D_i , \quad i = 1, \ldots, n .$$

Das beweist man sofort durch Entwicklung der Determinante nach der letzten Zeile (Induktionsbeweis).

Hilfssatz 3: Ist r der Rang von H, $D_r \neq 0$, und verschwindet kein Glied der Folge

$$(*) \quad D_0 := 1, D_1, \ldots, D_r ,$$

so ist die Anzahl der positiven bzw. negativen Eigenwerte von H gleich der Anzahl P der Zeichenfolgen bzw. gleich der Anzahl V der Zeichenwechsel in der Zahlenfolge (*).

Zusatz: Ist $D_i = 0$, aber $D_{i-1} D_{i+1} \neq 0$, so kann Hilfssatz 3 ebenfalls angewandt werden. D_i ist dann in der Folge (*) einfach zu streichen.

Diesen Hilfssatz samt Zusatz findet man in [2], Kap. 10, § 3.

Beweis von Satz 4:

1) $b_i \neq 0$, $i = 1, \ldots, n-1$.

Dann ist $\operatorname{rg} H \geq n-1$.

Hilfssatz 1 und 2 sind dann auf H anwendbar. Hilfssatz 3 auch. Dazu ist nur zu zeigen, daß $f_{n-1} \neq 0$ ist, falls $\operatorname{rg} H = n-1$. Angenommen, $f_{n-1} = 0$. Dann ist nach Hilfssatz 1 $f_n \neq 0$, im Widerspruch zu $\operatorname{rg} H < n$. Nun überzeugt man sich leicht davon, daß $\varrho_n = \operatorname{rg} H$ und ν_n gleich der Anzahl der Zeichenwechsel in der Folge (*) ist, also nach Hilfssatz 3 gleich der Anzahl der negativen Eigenwerte von H. Dann ist aber $\varrho_n - \nu_n$ gleich der Anzahl der positiven Eigenwerte von H.

Damit ist Satz 4 für diesen Spezialfall bewiesen.

2) Ist genau ein $b_i = o$, so läßt sich H in zwei Blöcke aufteilen:

$$H = \left(\begin{array}{c|c} H_1 & O \\ \hline O & H_2 \end{array}\right),$$

wobei H_1 eine i x i - Matrix und H_2 eine $(n-i)x(n-i)$-Matrix ist. Das unter 1) Bewiesene läßt sich nun auf H_1 und H_2 getrennt anwenden, und anschließend kann man dann die Anzahlen der Eigenwerte addieren.

Man hat in diesem Fall also zwei Folgen von Determinanten

$$1, f_1, \ldots, f_i$$

$$1, f_{i+1}, \ldots, f_{rgH} \cdot$$

Genau dies wird erreicht durch die Vorschrift:

"ist $b_i = o$, so setze man nach Berechnung von g_i und v_i $f_i := 1$."

3) Sind mehrere $b_i = o$, so kann man diesen Vorgang natürlich iterieren.

Damit ist Satz 4 bewiesen.

<u>Bemerkung:</u> Ähnliche Sätze befinden sich in [11], § 1.5, S.16 und [4], § 3, S.4 für reelle Matrizen. Die Beweisidee von [11] wurde hier auch weitgehend übernommen. Beide Sätze sind aber falsch, wenn $b_i = o$ wird, da dies von den genannten Autoren nur bei der Berechnung von f_{i+1}, nicht aber bei der Berechnung von f_{i+2} berücksichtigt wurde.

3. Die Methode und ihre Realisierung im Programm

In diesem Paragraphen wollen wir die in § 2 grob skizzierte Methode näher erläutern und auch auf einige programmtechnische Einzelheiten eingehen.

In das Programm einzugeben sind die zu untersuchende Matrix M, ihre Ordnung N und Größen EPS 1, EPS 2 und EPS 3, deren Bedeutung im Verlauf des Textes angegeben wird. Als Output erhält man außer evtl. Zwischenergebnissen:

AL, AR, AM - falls AL + AR + AM = N, so ist In M = (AL,AR,AM)

In einigen Fällen wird das Programm unterbrochen, wenn sichergestellt ist, daß AR >0 ist.

Wir gehen in folgenden Schritten vor:

1) M wird durch eine unitäre Ähnlichkeitstransformation in obere Hessenbergform gebracht. Der Einfachheit halber wird die transformierte Matrix ebenfalls mit M bezeichnet.

2) Falls in der unteren Nebendiagonalen von M Nullen auftreten, wird M in kleinere Hessenbergmatrizen $M^{(1)}, \ldots, M^{(m)}$ zerlegt, die dann einzeln behandelt werden.

   Für j = 1, ..., m werden folgende Schritte ausgeführt:

3) Falls Re (Sp $M^{(j)}$) > 0, so besitzt $M^{(j)}$ mindestens einen Eigenwert λ mit Re $\lambda > 0$. Dann wird AR um 1 erhöht und das Programm beendet.

4) Für $M^{(j)}$ wird die Ljapunowgleichung

$$HM^{(j)} + M^{(j)*} H = 2I$$

gelöst.

Dieser Schritt ist in zwei Fällen nicht voll ausführbar:

a) $D_1, \ldots, D_m$ sind linear abhängig.
Dann Sprung zu 6), falls noch keine Translation stattgefunden hat, andernfalls AR: = AR + 1 und Beendigung des Programms.

b) Die Matrix ist für dieses Verfahren zu schlecht konditioniert. Abbruch des Verfahrens mit entsprechender Fehlermeldung.

5) Ermittlung der Trägheit von H:
Das in 4) ermittelte H wird durch eine unitäre Ähnlichkeitstransformation tridiagonal gemacht.
Für H werden $l^{(j)}$ und $r^{(j)}$ ermittelt und AL und AR entsprechend verändert. Ist nun AR > o, oder $j = m$, wird das Verfahren beendet. Andernfalls Rücksprung zu 3) mit nächstem Index j.

6) Translation: $M^{(j)}$ wird durch $M^{(j)} - \mu I$ ersetzt.
Rücksprung zu 3). Falls auch ein weiterer Durchlauf kein Ergebnis bringt, werden mit immer kleiner werdenden μ weitere Translationen ausgeführt, bis schließlich μ so klein ist, daß man auf AM $\neq$ o schließen kann.

3.1 Transformation einer Matrix in obere Hessenbergform.

Zunächst ist klar, daß das Stabilitätsproblem gegen Ähnlichkeitstransformationen invariant ist. Außerdem ist bekannt, daß sich die Konditionen einer Matrix bzgl. des Eigenwertproblemes nicht verschlechtert, wenn man bei der Transformation nur unitäre Matrizen verwendet (siehe [9], Abschnitt III, 46). Denn die Spektralkonditionszahl $k(M) := \|M\|_2 \cdot \|M^{-1}\|_2$ wird durch eine unitäre Transformation von M nicht geändert, da die spektralnorm von unitären Matrizen 1 ist.
Im Programm geschieht die Transformation in obere Hessenbergform durch das Householder'sche Verfahren, bei dem man N-2 unitäre Ähnlichkeitstransformationen ausführt. Nach Wilkinson [8], Chap. 5, § 29) ist dies so ziemlich das beste verfügbare Verfahren.
Beschreibungen für die reelle Fassung dieses Verfahrens findet man in [8], S.347 f und [5], S. 166 f. In [8], S. 342 ist außerdem angegeben, wie man die Formeln für komplexe Matrizen abändern kann. Im r-ten Schritt, $r = 1, \ldots, n-2$, wird die zu transformierende Matrix M beidseitig mit einer Matrix $P_r = P_r^{-1}$ multipliziert, die so gewählt ist, daß alle Elemente verschwinden, die in der r-ten Spalte unter der Subdiagonalen stehen.

Man hat

$$P_\tau = I - u_\tau u_\tau^* / (2K_\tau^2)$$

$$(8)\quad u_{i\tau} = \begin{cases} 0 & \text{für } i = 1, 2, \dots, \tau \\ m_{\tau+1,\tau} + \dfrac{m_{\tau+1,\tau}}{|m_{\tau+1,\tau}|} \cdot S_\tau & \text{für } i = \tau+1 \\ m_{i\tau} & \text{für } i = \tau+2, \dots, n \end{cases}$$

falls $m_{\tau+1,\tau} \neq 0$. Andernfalls wählen wir $u_{\tau+1,\tau} = S_\tau$ (wird durch die Bedingung $u_\tau^* u_\tau = 4K_\tau^2$ nahegelegt).

Dabei ist

$$S_\tau^2 = \sum_{i=\tau+1}^{n} |m_{i\tau}|^2 \,, \quad S_\tau = +\sqrt{S_\tau^2}$$

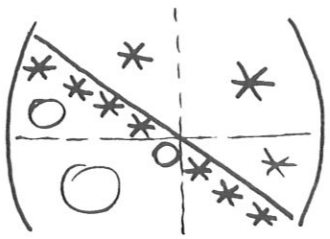
$$2K_\tau^2 = S_\tau^2 + |m_{\tau+1,\tau}| S_\tau$$

Wenn $S_\tau^2 = 0$ ist, wird man natürlich nichts rechnen, denn dann hat die τ-te Spalte ja schon die gewünschte Form. Wie man leicht nachrechnet, erhält man für das neue Element mit Index $(\tau+1, \tau)$ den Wert $-\dfrac{m_{\tau+1,\tau}}{|m_{\tau+1,\tau}|} S_\tau$ bzw. $-S_\tau$. Diesen Wert und die nachfolgenden Nullen setzen wir direkt.

Da die P_τ unitäre Matrizen sind, ist die Ergebnismatrix hermitesch, wenn die Ausgangsmatrix es war. Insbesondere wird eine hermitesche Matrix durch dieses Verfahren also tridiagonal gemacht. Das ist für den 5. Schritt des Verfahrens wichtig.

3.2 Zerlegung von M in Untermatrizen.

Tritt an einer Stelle in der unteren Nebendiagonalen von M eine Null auf, so läßt sich M aufteilen, wie nebenstehende Skizze zeigt. Da das Rechnen mit Übermatrizen



denselben Regeln folgt wie das Rechnen mit normalen Matrizen (siehe [2], S.38 ff), braucht man zur Ermittlung der Eigenwerte von M nur noch die beiden Hessenbergmatrizen oben links und unten rechts zu berücksichtigen. Ist das erste oder letzte Element der unteren Nebendiagonale Null, so ist das nebenstehende Diagonalelement jeweils ein Eigenwert.
Treten in der unteren Nebendiagonalen von M mehrere Nullen direkt hintereinander auf, so sind die neben den Nullen stehenden Diagonalelemente von M Eigenwerte, ausgenommen das letzte davon. Von den Realteilen der auf diese Weise ermittelten Eigenwerte wird sofort das Vorzeichen ermittelt. Die Größen AL, AM und AR werden entsprechend verändert. Ist nun AR > 0 , so könnte das Verfahren beendet werden. Andernfalls werden die folgenden Schritte für die ermittelten Untermatrizen $M^{(1)}$, ... , $M^{(m)}$ ausgeführt. Hat ein Subdiagonalelement von M einen sehr kleinen Absolutbetrag,

so empfiehlt es sich, dieses Element als Null anzusehen, denn 1. bedeutet das nach Wilkinson i.a. eine nur geringfügige Änderung der Eigenwerte, 2. müßte man sonst später durch dieses Element an entscheidenden Stellen dividieren, so daß der Schaden dann sicherlich größer wäre. Die Größe EPS 1 bestimmt, welche Werte noch vernachlässigt werden können. Es ist in der gegenwärtigen Fassung vom Benutzer zu setzen.

EPS 1 sollte wohl mindestens so groß sein wie MACHEPS, dh die größte Zahl, für die die Maschine 1 + MACHEPS = 1 und 1 - MACHEPS = 1 berechnet. Denn Zahlen dieser Größenordnung können bei der vorhergehenden Transformation in Hessenbergform an die Stelle von Nullen treten.

Besser ist es natürlich, EPS 1 in Abhängigkeit der Elemente der aufzuspaltenden Matrix zu setzen. Überlegungen, wie das zu machen ist, findet man in Beschreibungen des LR- und des QR-Algorithmus, z.B. in [10] , § 1, S.370. Für das Problem gibt es aber meines Wissens noch keine befriedigende Lösung.

3.3 Berechnung der Spur von M

Es wird nun die Spur der (Unter-) Matrix $M^{(j)}$ ermittelt. Ist der Realteil der Spur größer als $(EPS\ 3) \cdot N$, so wird AR verändert und das Verfahren beendet.
Wegen der möglicherweise auftretenden Rundungsfehler sollte EPS 3 wirklich größer Null gewählt werden. Wir haben es in unseren Testläufen großzügig 10^{-3} gesetzt.

3.4 Die Lösung der Ljapunowgleichung $HM + M^{*}H = 2D$.

Wie die Lösung dieser Gleichung zu geschehen hat, wurde in Abschnitt 2 schon kurz angedeutet. Es wurde auch schon gesagt, daß die Lösung dieser Gleichung mit Hilfe mehrmaliger Lösung der Gleichung (5) geschieht. Wir wollen uns deshalb zunächst ansehen, wie Gleichung

(5) $HM = T + D$

gelöst wird.

3.41 Die Lösung vom $HM=T + D$, $T^{*}= - T$, $D^{*}= D$

Sei M in oberer Hessenbergform gegeben, bei der alle Elemente in der unteren Nebendiagonale ungleich Null sind.

Dann wird die erste Zeile von H beliebig vorgegeben, s.d. das Element h_{11} reell ist. Daraus berechnet man die erste Zeile von H•M Durch die Forderungen: H soll sich höchstens in der Diagonalen von einer hermiteschen Matrix unterscheiden und HM soll sich höchstens in der Diagonalen von einer schiefhermiteschen Matrix unterscheiden, sind dann auch die 1. Spalte von H und von HM bestimmt.

Nun läuft für $j = 2, \ldots, \text{ord}(M) =: N$ folgende Schleife ab:

1) die j-te Spalte von H wird nach der Formel

(9) $$h_{\ell j} = -\frac{1}{m_{j,j-1}} \left(\sum_{k=1}^{j-1} h_{\ell k} m_{k,j-1} + \overline{hm}_{j-1,\ell} \right)$$

$\ell = j, \ldots, N$

berechnet.

2.) die j-te Zeile von HM wird durch Matrizenmultiplikation berechnet.

3.) Es wird geprüft, ob der Imaginärteil von h_{jj} größer als 10 * MACHEPS ist, denn dann kann H nicht als hermitesch angesehen werden.

Später gebraucht werden H und D, die deshalb gespeichert werden. Ist das berechnete H nicht hermitesch, braucht man später insbesondere die Imaginärteile der Diagonalelemente von H, diese werden deshalb in diesem Fall extra gespeichert.

Wir wollen an dieser Stelle auch gleich die wichtigeren anderen später gebrauchten Unterprozeduren besprechen.

3.42 Die Lösung eines inhomogenen linearen Gleichungssystems (LSYS)

Hierbei handelt es sich um das übliche Gauss'sche Eliminationsverfahren zur Lösung eines nicht-singulären Gleichungssystems. Es braucht deshalb nicht näher darauf eingegangen zu werden. Es sei nur noch darauf hingewiesen, daß die Wahl der unteren Grenze für die Pivotelemente,durch die noch dividiert werden darf, eng mit der Wahl von EPS 2 (wann ist der Realteil eines Eigenwertes noch als ungleich Null anzusehen?) zusammenhängt.

3.43 Die Prozedur SINGSYS zur nicht-trivialen Lösung eines homogenen linearen Gleichungssystems

Im Programmdurchlauf ist sichergestellt, daß keine Spalte der Eingangsmatrix der Nullvektor ist. Auf den Fall braucht also keine Rücksicht genommen werden.
Zunächst wird durch elementare Spaltenumformungen festgestellt, welcher Spaltenvektor als erster von den vorhergehenden linear abhängig ist. Dieser bekommt den Index

R. Durch gleichzeitige Zeilenumordnungen wird sichergestellt, daß anschließend die (R-1)- dimensionale Hauptuntermatrix nicht-singulär ist. Dann wird mit LSYS das zugehörige inhomogene Gleichungssystem gelöst, s.d. die R-te Spalte als Linearkombination der vorhergehenden R-1 Spalten dargestellt ist. Der Lösungsvektor X enthält also i.a. in den ersten R-1 Komponenten Zahlen ungleich Null, während die R-te Komponente -1 ist und die nachfolgenden Null.
Die festimplementierte untere Grenze für ein Pivotelement, durch das während der elementaren Umformungen noch dividiert werden darf, hängt eng zusammen mit der entsprechenden unteren Grenze in LSYS. Es darf also keines von beiden unabhängig vom anderen verändert werden.

3.44 Die Struktur der Prozedur CLYAPUNV

Zunächst wird mit dem ersten Einheitsvektor als Anfangsbedingung die Gleichung (5) gelöst. Ist die resultierende Matrix H hermitesch, wird sie bzw. ihre erste Zeile als erste Spalte von H 1 und die zugehörige Diagonale als erste Spalte von D 1 gespeichert. Dann wird solange der nächste

Einheitsvektor als Anfangsbedingung genommen, bis entweder N hermitesche Matrizen gefunden sind oder eine nicht-hermitesche Matrix entstanden ist. Im ersten Fall kann man zur Lösung des Gleichungssystems (6) übergehen (~~Sprung zu LINCOMB~~), im zweiten Fall merkt man sich außer H und der zugehörigen Diagonalmatrix (als Spalte der Matrix D2) noch die Imaginärteile der Diagonalen von H (Speicherung als Spalte von C). Sind alle Einheitsvektoren aufgebraucht und noch keine N hermiteschen Matrizen gefunden, geht man dazu über, die mit i multiplizierten Einheitsvektoren als Anfangsbedingung zu benutzen. Wenn man Glück hat, findet man auf diese Weise direkt die noch fehlenden hermiteschen Matrizen. Andernfalls kann man sich aus je N nicht-hermiteschen Lösungen von (5) durch reelle Linearkombinationen eine hermitesche Lösung von (5) konstruieren (Anwendung von SINGSYS auf die (N-1)-dim. Hauptuntermatrix von C bzw. Anwendung von LSYS auf die aus C unter weglassen der letzten Zeile[+] entstehende Matrix wenn die ersten N - 1 (N - 1)-dim. Spalten linear unabhängig sind).

Wie in [6] bewiesen ist, kann man auf diese Art immer N linear unabhängige Hermitesche Matrizen $H_1, \ldots, H_N$ finden, die eine Gleichung vom Typ (5) lösen. Wird ein H durch Linearkombination von nichthermiteschen Matrizen erzeugt, erhält man die zugehörige Diagonalmatrix durch

[+] siehe 3.45

Überlagerung der zugehörigen, in D 2 gespeicherten, Diagonalmatrizen mit denselben Koeffizienten.
Hat man nun also N linear unabhängige Ljapunowpaare $\{H_j, D_j\}_{j=1,\dots,N}$ konstruiert, kann man mit LSYS die Gleichung (6) lösen, wobei auf der rechten Seite die ursprünglich vorgegebene Diagonalmatrix D steht. (Aus Bequemlichkeitsgründen haben wir D = I gewählt). Duch Gleichung (7) ist dann die Lösung H von der Ausgangsgleichung (4) gegeben.

3.45 Möglichkeiten zur Speicherung der Zwischenlösungen von (5).

Bei der Speicherung der Zwischenlösungen von (5) hat man im wesentlichen zwei Möglichkeiten:
1) Man speichert jeweils praktisch das ganze H, dh alle im oberen Dreieck befindlichen Elemente, die man in einen Vektor gesteckt hat, und rechnet immer mit allen Elementen. Das bedeutet Reservierung von Speicherplatz für

$$(10) \quad 2 \cdot (2N - 1)N(N+1)/2 = (2N^3 + N^2 - N)$$

komplexe Zahlen.

2) Man speichert von jedem hermiteschen H nur die erste Zeile (in H1), von jedem nicht-hermiteschen H nur den Index des Einheitsvektors, mit dem als Anfangsbedingung man diese Matrix aus (5) gewonnen hat. Diesen Index kann man bequem als letztes Element der zugehörigen Spalte von C unterbringen und sich den ganzen Einheitsvektor mit Hilfe von DELTA wieder rekonstruieren, wenn er gebraucht wird. Man braucht dann immer nur Vektoren der Länge N linear zu kombinieren, bis man Gleichung (7) gelöst hat. Aus der ersten Zeile der Lösungsmatrix erhält man dann durch nochmalige Lösung von (5) die gesamte Lösungsmatrix. In diesem Fall braucht man statt der in (10) angegebenen Zahl nur N^2 Speicherplätze für komplexe Zahlen, N Speicherplätze für reelle Zahlen und den Platz zur Speicherung von DELTA, der gering ist.

Im 1. Fall braucht man also erheblich mehr Speicherplatz als im 2. Fall. Im 2. Fall braucht man einerseits weniger Rechenzeit zur Linearkombination der Matrizen (da tatsächlich nur ihre ersten Zeilen linearkombiniert werden), dafür braucht man die Zeit zur nochmaligen Lösung von (5).

Außerdem riskiert man im 2. Fall etwas mehr Rundungsfehler in der Darstellung der Lösungsmatrix H. Diese beiden Nach-

teile im 2. Fall waren aber im Test praktisch nicht spürbar. Stattdessen weiß man im 2. Fall mit Sicherheit, daß im Endergebnis auf der rechten Seite außerhalb der Diagonalen nur Nullen stehen. Das ist ein nicht zu vernachlässigender Vorteil, da bekanntlich auch kleine außerhalb der Diagonale stehende Zahlen einen großen Einfluß auf die Eigenwerte einer Matrix haben können, so daß im Fall 1) die positive Definitheit der Ergebnismatrix nicht so bequem gesichert ist wie im 2. Fall. Das ist dann besonders interessant, wenn die tatsächliche rechte Seite des Ergebnisses von der gewünschten merkbar abweicht (schlechte Kondition).
Diese Überlegungen haben uns bewogen, uns für die 2. Version zu entscheiden.

3.5 Ermittlung der Trägheit von H

Um die Trägheit von H zu ermitteln, möchten wir Satz 4 anwenden. Dazu muß H in Tridiagonalgestalt gebracht werden:

3.51 Transformation von H in Tridiagonalgestalt

Wie in 3.1 schon erläutert wurde, bleibt eine hermitesche Matrix hermitesch, wenn man sie mit CHOUSEHL in obere

Hessenbergform transformiert. Eine hermitesche Hessenbergmatrix ist aber tridiagonal. Wir können hier also wiederum CHOUSEHL benutzen. Zwar könnte man die Prozedur effektiver gestalten, wenn man sie speziell für hermitesche Matrizen noch einmal programmierte, aber der Vorteil, für zwei Zwecke dieselbe Prozedur anwenden zu können, schien uns größer.

3.52 Bestimmung von r (H) und l (H) einer tridiagonalen hermiteschen Matrix (CJACOBI).

Hierbei handelt es sich um die Programmierung der in Satz 4 angegebenen Formeln, was keine Schwierigkeiten aufwirft. Es empfiehlt sich jedoch, statt des Vorzeichens vom Produkt aufeinanderfolgender Unterdeterminanten das Produkt der Vorzeichen dieser Determinanten zu berechnen, um einen überflüssigen Overflow zu vermeiden. Es bietet sich an dieser Stelle eine weitere Gelegenheit, die Richtigkwit der erhaltenen Ergebnisse zu überprüfen: der Rang von H darf nicht kleiner sein als die Ordnung von H, wie in § 2 gezeigt wurde.
Hat man nun r (H) und l (H) bestimmt, so werden AL und AR bestimmt: Hat keine Translation stattgefunden so ist AL:= AL + l (H), AR: = AR + r (H). Ist nun AR > 0 , so

sind wir fertig, andernfalls Rücksprung zu 3), wenn noch weitere Untermatrizen von M zu behandeln sind. Was getan wird, wenn eine Translation stattgefunden hatte, wird in 3.6 beschrieben.

3.6 Translation

3.61 allgemeine Überlegungen

Hat das Gleichungssystem

(6) $\sum_{i=1}^{N} \rho_i D_i = I$ keine Lösung, dh sind die D_i linear abhängig, so ist $\Delta(M) = o$. Dann ist einer der beiden Fälle eingetreten:

1) $\nu(M) \neq 0$, dh M hat Eigenwerte mit Realteil o.

2) M hat außer $a + ib$ auch $-a + ib$ als Eigenwert.

Natürlich können auch beide Fälle gleichzeitig auftreten. Man bestimmt nun eine reelle Zahl $\mu > 0$ und untersucht die Matrix $M - \mu I$. Für $M - \mu I$ wird nun die Ljapunow-gleichung gelöst usw. Stellt sich raus, daß auch $\Delta(M - \mu I) = o$, so liegt mindestens ein Eigenwert von M in der rechten Halbebene, man setzt AR: = 1 und beendet das Programm.

Ist $r(M - \mu I) > 0$, so gilt dies erst recht für M, man setzt AR: = $r(M - \mu I)$ und beendet das Programm.

Liegen alle Eigenwerte von $M - \mu I$ in der linken Halbebene,

so kann man daraus noch nichts schließen. Es wird dann eine weitere Translation ausgeführt mit $\mu := \mu / k$. Dieser Vorgang wiederholt sich solange, bis entweder Eigenwerte von $M - \mu I$ in der rechten Halbebene auftauchen, oder $\mu < EPS2$ so klein geworden ist, daß die Eigenwerte von $M - \mu I$ praktisch mit denen von M übereinstimmen. Dann wird AM: = 1 gesetzt und zu Behandlung der nächsten Untermatrix übergegangen bzw. aufgehört. Es ist nicht zulässig, außer AM noch AL zu verändern, da man nur $AM > 0$ weiß, aber nicht den Wert kennt.

3.62 Zur Wahl der vorkommenden Parameter

μ : Natürlich sollte man es vermeiden, eine Translation mit einem μ auszuführen, das größer ist als der Spektralradius $\varrho(M)$. Nun ist der nicht bekannt. Wegen der Beziehung

(11) $$\varrho(M) \leq \|M\|,$$

kann man sich jedoch mit einer der üblichen Normen behelfen. Wir haben die Frobeniusnorm

(12) $$\|M\|_F := \left(\sum_{i,j=1}^{N} |m_{ij}|^2 \right)^{1/2}$$

gewählt, weil sie einerseits leicht zu berechnen ist und andererseits nach [9], S.103, oft eine gute Näherung für die Spektralnorm $\|M\|_2$ darstellt.

Wegen der Gültigkeit der Abschätzung

$$\|M\|_2 \leq \|M\|_F \leq N^{1/2}\|M\|_2 \tag{13}$$

ist es naheliegend, $\|M\|_F$ noch durch N zu dividieren, um wirklich unter $\|M\|_2$ zu kommen.
Die Brauchbarkeit des Wertes $\mu = \|M\|_F/N$ haben wir experimentell geprüft. Er lag praktisch in allen Fällen noch zu hoch. Mit den Ergebnissen für

$$\mu = \frac{\|M\|_F}{10 \cdot N} \tag{14}$$

waren wir dann zufrieden.

K: Für K haben wir den Wert 10^2 gewählt. Für K = 10 wurden zuviele Iterationen benötigt, wenn der am weitesten rechts liegende Eigenwert imaginär war. Für K = 10^3 scheint uns das Auflösungsvermögen zu gering.

EPS2: Für unsere Testmatrizen war EPS2 = 10^{-13} ein ganz vernünftiger Wert. Wenn EPS2 "zu klein" gewählt wird, können auf der imaginären Achse liegende Eigenwerte in der rechten Halbebene auftauchen (das μ angucken, für das AR > 0 wurde!)

3.7 Vereinfachungen, wenn die zu untersuchende Marix reell ist

Ist die zu untersuchende Matrix reell, ergeben sich außer bei den Transformationen in Hessenbergform vor allem bei der Lösung der Ljapunowgleichung wesentliche Vereinfachungen. Diese Gleichung heißt dann

(4a) $$SM + M^tS = 2I,$$

wobei S eine symetrische Matrix ist. Die Gleichung

(5a) $SM = T + D,\ T^t = -T,\ D$ diag.

braucht dann nur mit N Einheitsvektoren als Anfangsbedingung gelöst zu werden, da jede berechnete Matrix dann symmetrisch ist. Damit entfallen die im komplexen Fall die Sache so komplizierenden Verzweigungen in CLYAPUNV.

4. Ergebnisse

Wie die im Anhang aufgeführten Beispiele zeigen, sind die Ergebnisse im allgemeinen sehr zufriedenstellend. Die von uns verwendeten Testmatrizen (entnommen aus [3]) zerfielen sehr häufig, s.d. oft ein viel geringerer Aufwand nötig war, als es auf den ersten Blick schien. Über die gebrauchte Rechenzeit läßt sich allgemein wenig sagen, da sie auch für gleich große Matrizen sehr unterschiedlich sein kann. Es soll noch geprüft werden, ob dieses Verfahren der direkten Bestimmung des am weitesten rechts liegenden Eigenwertes überlegen ist.

4.1 Schlecht konditionierte Matrizen

Schwierig zu behandeln sind mit diesem Verfahren vor allem solche Matrizen, die rein imaginäre Eigenwerte haben oder Eigenwerte mit sehr kleinem Realteil, aber keinen Eigenwert mit deutlich positivem Realteil. Denn in diesem Fall sind wiederholte Translationen mit immer kleinerem μ notwendig. Dabei wird das durch Gleichung (6) gegebene Gleichungssystem immer schlechter konditioniert (für $\mu = 0$ ist die

Matrix des Gleichungssystems singulär).
Das kann sich bei der Lösung der Ljapunowgleichung vor allem in zwei Weisen äußern:

1) Die zur Lösungsmatrix gehörige Diagonalmatrix unterscheidet sich beträchtlich von der ursprünglich vorgegebenen (2I).

2) Berechnete Matrizen sich nicht hermitesch, obwohl sie es theoretisch sein sollten.

Wie wir gleich sehen werden, ist jeder dieser beiden Fehler allein gar nicht so schlimm, wie es scheinen mag. Das macht das Verfahren recht robust.

4.2 Unempfindlichkeit des Ergebnisses gegen bestimmte Änderungen von M, H und D

4.21 Änderungen von H und D

Tritt der eben genannte erste Fehler auf, so macht das nach Satz 1 nichts, wenn nur H hermitesch ist und $HM + M^*H$ positiv definit ist. Wie groß die Abweichung sein darf, wenn der eben genannte 2. Fehler auftritt, sagt folgender Satz:

Satz 5: Seien $M, B \in L(\mathbb{C}^n)$; M in oberer Hessenbergform, $B = H + iA$ mit $H^* = H$ und $A = \operatorname{diag}(a_1, \ldots, a_n) \in L(\mathbb{R}^n)$
Weiter sei

$$BM + M^* B^* = 2D = 2\operatorname{diag}(d_1, \ldots, d_n).$$

Gilt dann

$$(15)\quad \max_{1\le j,k\le n} |m_{jk} a_j| < \frac{2}{n+3} \min_{1\le k\le n} d_k ,$$

so ist In H = In M.

Bem.: Diese Abschätzung ließe sich noch verschärfen. Sie ist aber auch so schon recht weittragend, wie folgendes Beispiel zeigt:

Sei $M \in L(\mathbb{C}^{50})$, $|m_{ij}| \le 10$, $i,j = 1,\dots,50$, $\min_k d_k = 10^{-1}$

Dann ist immer noch In H = In M, wenn

$$\max_j |a_j| < 3 \cdot 10^{-4}$$ gilt.

Beweis:

Es ist

$$(16)\quad HM + M^* H = 2D - i(AM - M^* A).$$

Nach dem Satz von Gerschgorin (seihe [5], Kap. 4.1) ist die Matrix auf der rechten Seite von (16) positiv definit, wenn

$$(17)\quad |2d_k - i(a_k m_{kk} - \overline{m_{kk}}\, a_k)| - \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_k m_{kj} - \overline{m_{jk}}\, a_j| > 0$$

gilt. Hierfür wollen wir jetzt eine hinreichende Bedingung herleiten.

Die linke Seite von (17) ist

$$\ge 2d_k - \sum_{j=1}^{n} |a_k m_{kj} - \overline{m_{jk}}\, a_j| \qquad (j = k \text{ zugelassen})$$

Weiter gilt

$$\sum_{j=1}^{\tilde{n}} |a_k m_{kj} - \overline{m_{jk}}\, a_j| \le \sum_{j=1}^{n} |a_k m_{kj}| + \sum_{j=1}^{\tilde{n}} |a_j m_{jk}|$$

$$= \sum_{j=\ell}^{\tilde{n}} |a_k m_{kj}| + \sum_{j=1}^{p} |a_j m_{jk}|,$$

wobei

$$\ell := \begin{cases} k-1 & \text{falls } k>1 \\ 1 & k=1 \end{cases} \quad \text{und} \quad p := \begin{cases} k+1 & \text{falls } k<n \\ n & k=n \end{cases}$$

zu setzen ist. Hierbei wurde ausgenützt, daß $m_{ij} = 0$ für $i > j+1$.

Nun folgt

$$\sum_{j=1}^{\tilde{n}} |a_k m_{kj} - \overline{m_{jk}}\, a_j| \le (n-\ell+1) \max_{1\le j\le n} |a_k m_{kj}| +$$

$$+ p \cdot \max_{1\le j\le n} |a_j m_{jk}|$$

$$\le (n-\ell+1+p) \max_{1\le k,j\le n} |a_j m_{jk}|$$

$$\le (n+3) \max_{1\le k,j\le n} |a_j m_{jk}|.$$

(17) ist also sicher erfüllt, wenn

(18) $\min\limits_{1\le k\le n} 2d_k - (n+3) \max\limits_{1\le j,k\le n} |a_j m_{jk}| > 0$

gilt, also (15) erfüllt ist.

4.22 Änderung von M

Es ist In M = In (M + tI), wenn $t \in i\mathbb{R}$ ist.

Wir wollen nun kurz die Frage untersuchen, wie das zu berechnende H sich ändert, wenn M so einer Translation unterworfen wird.

Lemma: Sei M': = M + tI, $t \in i\mathbb{R}$. . Ist H eine hermitesche Matrix, so gilt

$$HM + M^{*}H = HM' + M'^{*}H.$$

Beweis:

$$\begin{aligned} HM' + M'^{*}H &= H\,(M+ tI) + (M+ tI)^{*}\,H \\ &= H\,(M+ tI) + (M^{*}-tI)\;\;H \\ &= HM + M^{*}H + tH - tH. \end{aligned}$$

Die Kondition einer Matrix bzgl. des eben beschriebenen Verfahrens ist also völlig unabhängig vom Verhältnis Realteil-Imaginärteil eines einzelnen Eigenwertes, denn das kann ja nach dem Lemma beliebig abgeändert werden, ohne daß sich die zu berechnenden Matrizen ändern. Wesentlich ist aber die Differenz der Imaginärteile zweier Eigenwerte, wenn ihre Realteile sich nur im Vorzeichen unterscheiden. Je weiter sie auseinanderliegen, desto größer wird der Betrag vom $\Delta(M)$, das nicht verschwinden darf.

5. Beispiele

5.1 Eine komplexe 62 x 62 - Matrix

Definiert man $M^{(n)} = (m_{k\ell})_{k,\ell = 1, \ldots, n}$ mit

$$m_{k\ell} := \begin{cases} -k-\ell+k\cdot i & \text{falls } k \geq \ell \\ 0 & \text{sonst} \end{cases},$$

so ist $M^{(n)}$ eine komplexe untere Dreiecksmatrix, von der man sämtliche Eigenwerte kennt, deren Behandlung mit dem eben beschriebenen Verfahren jedoch nicht trivial ist (wie bei oberen Dreiecksmatrizen).

Die größte Ordnung n, für die das Verfahren noch richtige Ergebnisse liefert, ist n = 62. Dafür wurden (ohne check option) ca. 12 Minuten gebraucht.

Für n = 63 erhält man dann eine Fehlermeldung (die Ljapunowgleichung kann nicht mehr so gelöst werden, daß das in Satz 5, (15) gegebene Kriterium erfüllt ist).

5.2 Die Hilbertmatrix

Die endlichen Abschnitte der Hilbertmatrix

$$A^{(n)} = \left(\frac{1}{i+j-1}\right)_{i,j = 1, \ldots, n}.$$

sind als sehr schlecht konditionierte Matrizen bekannt (siehe [3], Beispiel 3.8). Es interessiert uns deshalb besonders, bei welcher Ordnung der Matrix das Verfahren versagt. Wie zu vermuten war, hängt das von der Wahl von EPS1 und EPS 2 ab.

Die $A^{(n)}$ sind alle positiv definit. Deshalb haben wir mit $-A^{(n)}$ gearbeitet (sonst Abbruch des Verfahrens, da die Spur positiv ist). Wir haben drei Durchläufe gemacht:

1) EPS1=EPS2 = 10^{-13}
2) EPS1=EPS2 = o

} EPS3 = 10^{-3},

3) Transformation von $-A^{(n)}$ in Tridiagonalgestalt und Bestimmung der Trägheit dieser Matrix. (Ist möglich, da $A^{(n)}$ symmetrisch).

Im Fall 1) wurden die Matrizen schließlich aufgespalten in eine definite und mehrere der Ordnung 1, deren Elemente als o gerechnet wurden.
Im Fall 2) trat keine Aufspaltung auf. Einen genaueren Überblick gibt die folgende Tabelle:

	1)		2)		3)		1)-3)
N	AL	AM	AL	AM	AL	AM	AR
$\leq$ 10	N	o	N	o	N	o	o
11	10	1	11	o	-	-	
12	11	1	-	-			
13	11	2					
14	11	3					
15	12	3					
16	12	4					
17	12	5					
18	12	6					
19	12	7					
20	-	-					o

Jeder der drei Durchgänge wurde durch einen Overflow in CJACOBI beendet, weil in dieser Prozedur sehr groß bzw. klein werdende Determinanten berechnet werden müssen: Bei 2) ist es die Determinante der Inversen der Hilbertmatrix, bei 3) die Determinante der Hilbertmatrix selbst (det $A^{(10)} \approx 10^{-53}$). Theoretisch sollten 2)und 3) deshalb natürlich bei demselben N abbrechen.

Bei 1) und 2) ist in CLYAPUNV die Abweichung der berechneten Diagonalmatrix von der eingangs vorgegebenen elementweise kleiner als 10^{-13}. Deshalb läßt sich vermuten, daß man

für höhere Ordnungen noch richtige Ergebnisse erhielte, wenn man die Berechnung der Determinante umginge.

5.3 Die Matrix $B^{(n)} = (|i-j|)_{i,j=1,\ldots,n}$

Diese Matrizen sind wieder symmetrisch und haben für jedes n $(n-1)$ Eigenwerte mit negativem Realteil und einen dominierenden mit positivem Realteil (Spur $B^{(n)} = 0$), (siehe [3] , Beispiel 4.22). Wieder interessierte uns, bis zu welchem n wir kommen würden.

Mit EPS1 = EPS2 = 10^{-13}, EPS3 = 10^{-3}, haben wir In ($B^{(n)}$) für n = 5 (5) 40, 41,42, 43, richtig berechnen können. Keine der vorkommenden Matrizen wurde aufgespalten, s.d. man für andere Werte von EPS1, EPS2 dasselbe Ergebnis bekäme. Diese Matrizen sind ein schönes Beispiel dafür, daß die Lösung der Ljapunowgleichung ziemlich falsch ausfallen darf, wenn nur gesichert bleibt, daß die Matrix auf der rechten Seite positiv definit ist. Für n = 44 wurde dann ein Element der Diagonalmatrix negativ.

Auf der folgenden Seite geben wir die berechneten rechten Seiten für n = 43 und n = 44 an (jede Zahl sollte gleich 1 sein).

$n = 43$

COMPUTED DIAGONAL D
1.000000C02861111'+00
9.999999714258280'-01
1.000000009835789'+00
1.000C000011935609'+00
9.999999901269989'-01
1.000000002982648'+00
1.000000002287525'+00
9.999999983478531'-01
9.999999988835624'-01
1.000C000008266504'+00
1.000000004435300'+00
1.000C0000C0954421'+00
9.999999892876467'-01
1.000000006236949'+00
1.000C000004185117'+00
1.000C000007907876'+00
9.999999990228933'-01
1.000000029931422'+00
9.999999983547041'-01
9.999999579291755'-01
1.000000036349754'+00
9.999999310900025'-01
1.000C000049764729'+00
9.999999354673862'-01
1.000000072832947'+00
9.999999464215281'-01
1.000C000031228444'+00
9.999999740841619'-01
1.000C000048969012'+00
9.999999567104075'-01
1.C0C000238150219'+00
9.999996831521421'-01
1.000000789581562'+00
9.999980743001401'-01
9.999999410471803'-01
1.000000802965431'+00
1.000012822232033'+00
9.999603264620224'-01
9.999997794855910'-01
1.000808971223820'+00
9.919532051230246'-01
3.930313844608841'-01
1.667669209781671'+02

$n = 44$

COMPUTED DIAGONAL D
9.999999978148753'-01
1.000000333438064'+00
9.999999478364320'-01
1.000000004238880'+00
1.000000000524710'+00
1.000C000009451364'+00
1.000000002836479'+00
9.999999975348672'-01
1.000000007639460'+00
1.000C000013953356'+00
9.999999901035144'-01
1.000C00C003713468'+00
1.000000054001910'+00
1.000000075887437'+00
9.999998316785339'-01
1.000000129137534'+00
9.999998847818376'-01
1.000C000058820100'+00
9.999999729644339'-01
9.999999998084739'-01
1.000000054016707'+00
1.000000050390697'+00
9.999999722031248'-01
1.000000031465905'+00
9.999999958921019'-01
1.000000023120285'+00
1.000C000069283050'+00
1.000000C00077571'+00
9.999999589828944'-01
9.999999761178238'-01
1.000000058325711'+00
9.999995497347885'-01
1.000000844993356'+00
9.999980434377296'-01
1.000001331973225'+00
1.000001607507080'+00
1.000000C275507517'+00
9.999702499826189'-01
1.000059059910389'+00
9.990445119682358'-01
9.769390925257278'-01
1.240366376171746'+00
-2.537217290118420'+01
3.942106573853675'+02

6. Anhang

Liste der Prozeduren in Algol W:

```
PROCEDURE INERTIA(LONG COMPLEX ARRAY M1(*,*); INTEGER VALUE N1;
                  LONG REAL VALUE EPS1,EPS2,EPS3;
                  INTEGER RESULT AL,AR,AM,IER);


BEGIN
COMMENT
PURPOSE:
   TO COMPUTE THE INERTIA OF A GENERAL COMPLEX MATRIX.

DESCRIPTION OF PARAMETERS:
   INPUT:
   M1  -MATRIX TO BE TREATED,M1(1::N1,1::N1).
   N1  -ORDER OF M1.
   EPS1-IF A SUBDIAGONAL ELEMENT OF THE HESSENBERG TRANSFORMED OF M1
        IS ABSOLUTELY LESS THAN EPS1 ,IT IS TREATED AS ZERO.
   EPS2-IF A REALPART OF AN EIGENVALUE OF M1 IS ABSOLUTELY LESS
        THAN EPS2,IT IS TREATED AS ZERO.
   EPS3-IF THE REALPART OF THE TRACE OF MATRIX M IS >EPS3 THEN M HAS
        AT LEAST ONE EIGENVALUE WITH POSITIVE REALPART.
   OUTPUT:
  AL,AR-M1 HAS AT LEAST AL (AR) EIGENVALUES WITH
        NEGATIVE (POSITIVE) REALPART <-EPS2 (>EPS2).
   AM  -M1 HAS AT LEAST AM EIGENVALUES WHOSE REALPARTS ARE
        ABSOLUTELY LESS EPS2.
   IER -INDEX OF ERROR:
        IER=0: ALL RIGHT.
        IER¬=0:COMPUTATION HAS FAILED ;

LONG REAL MUE;  INTEGER ITERATION;

IER:=0;

COMMENT
TRANSFORMATION OF M1 INTO UPPER HESSENBERGFORM;

CHOUSEHL(N1,M1);
```

```
COMMENT
SPLITTING OF M1;

BEGIN
   INTEGER N,F,R;
   INTEGER ARRAY E(1::N1);


   AL:=AR:=AM:=0 ;
   IF ABS M1(N1,N1-1)<EPS1 THEN BEGIN
      IF LONGREALPART(M1(N1,N1)) >EPS2 THEN AR:=AR+1
         ELSE IF LONGREALPART(M1(N1,N1)) <-EPS2 THEN AL:=AL+1
                                                ELSE AM:=AM+1;
   END;
   F:=1;  R:=1;
   FOR I:=1 UNTIL N1-1 DO
   IF ABS M1(I+1,I)>EPS1
   THEN  F:=F+1
   ELSE BEGIN
      IF F=1 THEN BEGIN
         IF LONGREALPART(M1(I,I))>EPS2 THEN AR:=AR+1
            ELSE IF LONGREALPART(M1(I,I))<-EPS2 THEN AL:=AL+1
                                                ELSE AM:=AM+1;
      END;
      E(R):=F;
      F:=1;  R:=R+1;
   END;
   E(R):=F;

COMMENT
HERE  R-THE NUMBER OF BLOCKS OF M1 -AND E -ARRAY OF ORDERS OF THE
BLOCKS MAY BE PRINTED OUT;

   R:=0;  F:=0;

AUFSPALTUNG:
   IF F=N1 THEN  GOTO AUSGANG;
   R:=R+1; N:=E(R);
   IF N=1 THEN BEGIN
      F:=F+1;
      GOTO AUFSPALTUNG;
   END;

COMMENT: TREATMENT OF SUBMATRICES OF M1;

      BEGIN
      INTEGER SIGMA1,L,RR,RHO1;
      LONG COMPLEX ARRAY H,M(1::N,1::N );  LONG REAL ARRAY D(1::N);
      LONG COMPLEX ARRAY MORIG(1::N);

   FOR I:=1 UNTIL N DO
      FOR J:=1 UNTIL N DO  M(I,J):=M1(I+F,J+F);
   ITERATION:=0;  MUE:=0.L;
```

```
  IF LONGREALPART(CSPUR(N,M))>EPS3*N THEN AR:=AR+1;
  IF AR>0 THEN GOTO AUSGANG;

  FOR I:=1 UNTIL N DO  D(I):=1;
LIAP:

  CLYAPUNV(M,H,N,D,IER);

  IF IER<2 THEN GOTO AEHNTRANSF;
  IF IER=2 THEN  BEGIN
     IF ITERATION >0 THEN BEGIN
        AR:=AR+1;  IER:=0;  GOTO AUSGANG;
     END;
     FOR I:=1 UNTIL N DO  MORIG(I):=M(I,I);
     MUE:=CFNORM(N,M)/(10*N);
  END
  ELSE BEGIN
     F:=F+N;  GOTO AUFSPALTUNG;   END;

TRANSLATION:
  ITERATION:=ITERATION+1;
  FOR I:=1 UNTIL N DO  M(I,I):=MORIG(I)-MUE;
  GOTO LIAP;

AEHNTRANSF:
     CHOUSEHL(N,H);
     IER:=0;


     CJACOBI(H,N,L,RR);
     IF L+RR <N THEN BEGIN  IER:=3; GOTO AUSGANG;  END;



COMMENT
 HERE MUE -PARAMETER OF THE LAST TRANSLATION -AND RR -NUMBER OF
EIGENVALUES OF THIS TRANSLATED BLOCK WITH POSITIVE REALPART -
MAY BE PRINTED OUT;

  IF ITERATION=0  THEN AL:=AL+L;
  IF RR>0 THEN  BEGIN
                AR:=AR+RR;  GOTO AUSGANG;
                END
          ELSE BEGIN
                IF ITERATION >0  THEN BEGIN
                                      MUE:=MUE/100.L;
                                      IF MUE >EPS2 THEN GOTO TRANSLATION;
                                      AM:=AM+1;
                                      END;
                END;

   END;
   F:=F+N;
   GOTO AUFSPALTUNG;
   END;
```

```
AUSGANG:
END INERTIA;


PROCEDURE CLYAPUNV(LONG COMPLEX ARRAY M,H(*,*); INTEGER VALUE N;
                   LONG REAL ARRAY D(*); INTEGER RESULT IER);

BEGIN


COMMENT
PURPOSE:
   TO FIND A HERMITIAN H WITH H*M+M(*)*H=2*D,WHERE M AND D ARE GIVEN
   MATRICES.

DESCRIPTION OF PARAMETERS:
   INPUT:
   M   -GIVEN MATRIX.M HAS TO BE IN UPPER HESSENBERG FORM WITH ALL ITS
        SUBDIAGONAL ELEMENTS UNEQUAL TO ZERO ,M(1::N,1::N).
   N   -ORDER OF M.
   D   -VECTOR OF PRESCRIBED DIAGONAL ELEMENTS OF THE RIGHT HAND SIDE
        OF THE EQUATION,MUST BE POSITIVE ,D(1::N).
   OUTPUT:
   D   -COMPUTED DIAGONAL OF THE RIGHT HAND SIDE MATRIX.ITS DEVIATION
        FROM THE INPUT VECTOR IS CAUSED BY ROUNDING ERRORS.
   H   -HERMITIAN MATRIX,H(1::N,1::N).
   IER -INDEX OF ERROR:
    IER=0: ALL RIGHT.
    IER=1: IN(H)=IN(M),BUT HM+M(*)H¬=2DIAG(D).
    IER=2: M HAS SOME EIGENVALUES L1,L2 WITH L1+CONJ(L2)=0.
    IER=3: COMPUTATION HAS FAILED TOTALLY.(ILL CONDITION).
    IER=4: ERROR IN SINGSYS;


   LONG COMPLEX ARRAY H1(1::N,1::N); LONG COMPLEX ARRAY DD(1::N);
   LONG REAL ARRAY D1(1::N,1::N+1); LONG REAL ARRAY X,C1,C2(1::N);
   LONG REAL ARRAY C,D2(1::N,1::N);
   LOGICAL HERMITIAN;  INTEGER I,J,K,P,R;
   LONG REAL MAX1,MAX2;


LONG COMPLEX PROCEDURE DELTA(INTEGER VALUE L; LONG REAL VALUE S ;
INTEGER N);
   IF L=S  THEN 1
           ELSE IF (L=S-N+1) AND (L¬=1) THEN IMAG(1)
                             ELSE 0;



PROCEDURE CGLT(LONG COMPLEX ARRAY M,H(*,*); INTEGER VALUE N,INH;
               LOGICAL RESULT HERMITIAN; LONG REAL ARRAY C1,C2(*));

BEGIN

COMMENT
PURPOSE:
   TO FIND H WITH HM=T+D: M IN UPPER HESSENBERGFORM,T ANTI-HERMITIAN,
   D DIAGONAL. THE FIRST ROW OF H HAS GIVEN VALUES.
```

```
DESCRIPTION OF PARAMETERS:
   INPUT:
   M          -MATRIX WITH M(I,J)=0 IF I>(J+1). M(1::N,1::N).
   N          -ORDER OF M.
   INH        -GIVEN INFORMATION ABOUT THE FIRST ROW OF H.  0<=INH<=2N-1.
               INH=0: THE FIRST ROW OF H IS NOT AFFECTED IN THIS
               PROCEDURE.
               1<=INH<=N: H(1,*) HAS TO BE THE INH-TH UNITARY VECTOR.
               N<INH<=2N-1: H(1,*) HAS TO BE THE (INH-N+1)ST UNITARY
               VECTOR,MULTIPLIED BY I=SQRT(-1).
   H          -ITS FIRST ROW MUST CONTAIN SOME VALUES ONLY IF INH=0.
   OUTPUT:
   HERMITIAN-TRUE: H IS HERMITIAN,I.E. THE IMAGINARY PARTS OF ITS
               DIAGONAL ELEMNTS ARE ABSOLUTELY <=10*LONGEPSILON.
               FALSE: OTHERWISE.
   C1         -IMAGINARY PARTS OF THE DIAGONAL ELEMENTS OF H,IF
               HERMITIAN=FALSE,EMPTY OTHERWISE.
   C2         -DIAGONAL OF MATRIX D ;

   LONG COMPLEX ARRAY HM(1::N,1::N);  LONG COMPLEX SUM;
   LOGICAL IMAGINARY;

   HERMITIAN:=TRUE;  IMAGINARY:=FALSE;
COMMENT  INITIALISING THE FIRST ROWS OF H AND HM;
   IF INH=0 THEN  BEGIN
      FOR K:=2 UNTIL N DO
         H(K,1):=CONJ(H(1,K));
      FOR K:=1 UNTIL N DO  HM(1,K):=CVECVEC(N,H(1,*),M(*,K));
   END ZERO  ELSE BEGIN
      FOR K:=1 UNTIL N DO  BEGIN
         H(1,K):=0L;  H(K,1):=0L;  END K;
      IF INH<=N THEN  BEGIN
         FOR K:=1 UNTIL N DO  HM(1,K):=M(INH,K);
         H(INH,1):=1L;  H(1,INH):=1L;
         END LESS   ELSE BEGIN
         IMAGINARY:=TRUE;  INH:=INH-N+1;
         FOR K:=1 UNTIL N DO  HM(1,K):=-LONGIMAGPART(M(INH,K))+
         LONGIMAG(LONGREALPART(M(INH,K)));
         H(INH,1):=-1IL;  H(1,INH):=1IL;
         END IF2;
      END IF1;
   FOR J:=2 UNTIL N DO  BEGIN
COMMENT  COMPUTATION OF THE J-TH COLUMN AND ROW OF H ...;
      FOR L:=J UNTIL N DO  BEGIN
         SUM:=-CONJ(HM(J-1,L));
         FOR K:=1 UNTIL J-1 DO  SUM:=SUM-H(L,K)*M(K,J-1);

         SUM:=SUM/M(J,J-1);
         H(J,L):=CONJ(SUM);  H(L,J):=SUM;
         END L;
COMMENT  ...AND OF THE RELATED ROW OF HM;
      FOR L:=J UNTIL N DO  HM(J,L):=CVECVEC(N,H(J,*),M(*,L));
COMMENT  IS H HERMITIAN ? ;
      IF ABS(LONGIMAGPART(H(J,J))) >10*LONGEPSILON THEN
         HERMITIAN:=FALSE;
      END J;
```

```
COMMENT  SAVING RESULTS;
   IF IMAGINARY THEN INH:=INH+N-1;
   C1(N):=INH;
      FOR L:=1 UNTIL N DO  C2(L):=LONGREALPART(HM(L,L));
   IF ¬HERMITIAN THEN FOR L:=1 UNTIL N-1 DO C1(L):=LONGIMAGPART(
 H(L+1,L+1));
   END CGLT;

   I:=J:=K:=1;  P:=N+1; IER:=0;
ISCHLEIFE:
   IF J>N THEN  GOTO LINCOMB;

COMMENT: SOLUTION OF H*M=T+DIAG(C2). THE FIRST ROW OF H IS THE I-TH
UNITARY VECTOR;

   CGLT(M,H,N,I,HERMITIAN,C1,C2);

COMMENT:  SAVE RESULTS AND DECIDE WHAT TO DO NEXT;

   IF HERMITIAN THEN BEGIN
      FOR L:=1 UNTIL N DO  BEGIN
         H1(L,J):=H(1,L);  D1(L,J):=C2(L);
         END L;
      J:=J+1;  I:=I+1;
      GOTO ISCHLEIFE;
      END HERMITIAN  ELSE BEGIN
   FOR L:=1 UNTIL N DO  BEGIN
      C(L,K):=C1(L);  D2(L,K):=C2(L);
   END L;
   I:=I+1;  K:=K+1;
   IF K=2 THEN GOTO PSCHLEIFE;
   GOTO SING;
   END ;
PSCHLEIFE:
   IF P=2*N THEN  GOTO SING;

COMMENT: SOLUTION OF H*M=T+DIAG(C2).THE FIRST ROW OF H IS THE
(P-N+1) ST UNITARY VECTOR,MULTIPLIED BY I=SQRT(-1);

   CGLT(M,H,N,P,HERMITIAN,C1,C2);

COMMENT:  SAVE RESULTS AND DECIDE WHAT DO DO NEXT;

   IF HERMITIAN THEN BEGIN
      FOR L:=1 UNTIL N DO  BEGIN

         H1(L,J):=H(1,L);  D1(L,J):=C2(L);
         END L;
      J:=J+1;  P:=P+1;
      IF J<=N THEN GOTO PSCHLEIFE   ELSE  GOTO LINCOMB;
      END HERMITIAN  ELSE  BEGIN
      FOR L:=1 UNTIL N DO  BEGIN
         C(L,K):=C1(L);  D2(L,K):=C2(L);
      END L;
```

```
        K:=K+1;  P:=P+1;
        GOTO PSCHLEIFE;
        END ;
SING:

COMMENT:  FIND THE COEFFICIENTS FOR COMBINING THE NON-HERMITIAN
MATRICES TO A HERMITIAN ONE ...;

    IF K<=N THEN  GOTO ISCHLEIFE;
    SINGSYS(C,N-1,N-1,X,R,IER);
    IF IER¬=0 THEN BEGIN  IER:=4;  GOTO ENDE;  END;
    IF R=0 THEN BEGIN
       LSYS(N-1,C,X,IER);  X(N):=-1L;
       IF IER=-1 THEN BEGIN  IER:=4;  GOTO ENDE;  END;
       R:=N;
       END;

COMMENT:  ...AND COMBINE THEM,COMBINE ALSO THE DIAGONALS;

    FOR L:=1 UNTIL N DO  BEGIN
       H1(L,J):=DELTA(L,C(N,R),N);
       D1(L,J):=D2(L,R);
       FOR Q:=1 UNTIL R-1 DO  BEGIN
          H1(L,J):=H1(L,J)-X(Q)*DELTA(L,C(N,Q),N);
          D1(L,J):=D1(L,J)-X(Q)*D2(L,Q);
       END Q;
       H(1,L):=H1(L,J);
    END;

    J:=J+1;  K:=R;
    GOTO ISCHLEIFE;


LINCOMB:

COMMENT: FIND THE COEFFICIENTS FOR COMBINING N THEORETICALLY
HERMITIAN MATRICES TO THE REQUIRED ONE...;

    FOR L:=1 UNTIL N DO  D1(L,N+1):=D(L);
    LSYS(N,D1,X,IER);
    IF IER=-1  THEN BEGIN  IER:=2;  GOTO ENDE;  END;

COMMENT  ...AND COMPUTE H;

    FOR L:=1 UNTIL N DO  DD(L):=X(L);
    FOR L:=1 UNTIL N DO  H(1,L):=CVECVEC(N,H1(L,*),DD);

    CGLT(M,H,N,0,HERMITIAN,C1,D);

COMMENT  IS THE RIGHT HAND SIDE OF THE EQUATION POSITIVE DEFINITE,WHEN

H IS REPLACED BY ITS HERMITIAN PART ? ;
```

```
   FOR J:=2 UNTIL N DO
      IF D(J)<D(1) THEN D(1):=D(J);
   IF D(1)<1'-3 THEN BEGIN  IER:=3;  GOTO ENDE;  END;
   IF ¬HERMITIAN THEN BEGIN
      IER:=1;  MAX2:=0;
      FOR J:=2 UNTIL N DO BEGIN
         MAX1:=ABS M(J,J-1);
         FOR K:=J UNTIL N DO
            IF ABS M(J,K)>MAX1 THEN MAX1:=ABS M(J,K);
         IF MAX1*ABS C1(J-1)>MAX2 THEN   MAX2:=MAX1*ABS C1(J-1);
      END J;
      IF MAX2< D(1)*2/(N+3)  THEN BEGIN FOR J:=2 UNTIL N DO
         H(J,J):=LONGREALPART(H(J,J));  END
                              ELSE IER:=3;
   END;


   ENDE:
END CLYAPUNV;


PROCEDURE CHOUSEHL(INTEGER VALUE N; LONG COMPLEX ARRAY A(*,*));

COMMENT
PURPOSE:
   TO TRANSFORM A GENERAL COMPLEX   MATRIX INTO UPPER HESSENBERG FORM.
DESCRIPTION OF PARAMETERS:
   N-ORDER OF MATRIX A.
   A-INPUT :MATRIX TO BE TRANSFORMED,
     OUTPUT:MATRIX IN UPPER HESSENBERG FORM;

BEGIN
LONG REAL S,T;
LONG COMPLEX B,SZ,C;
LONG COMPLEX ARRAY V(2::N);  LONG COMPLEX ARRAY H(2::N,2::N);

FOR K:=1 UNTIL N-2 DO
BEGIN
   SZ:=0L;
   FOR I:=2+K UNTIL N DO
      SZ:=SZ+A(I,K)*CONJ(A(I,K));
   IF SZ=0.L THEN GOTO RETURN;
   B:=A(K+1,K);
   SZ:=SZ+B*CONJ(B);
   S:=LONGSQRT(LONGREALPART(SZ));

   T:=S*ABS B;
   IF ABS B=0.L THEN BEGIN V(K+1):=S; C:=-S;  END
                ELSE BEGIN V(K+1):=B*(1.L+SZ/T); C:=-B*SZ/T;  END;
   FOR I:=K+2 UNTIL N DO
      V(I):=A(I,K);

   B:=1.L/(SZ+T);
   FOR I:=K+1 UNTIL N DO
   BEGIN
```

```
      FOR J:=I UNTIL N DO
      BEGIN
         SZ:=-V(I)*CONJ(V(J))*B;
       H(I,J):=SZ;  H(J,I):=CONJ(SZ);
      END J;
      H(I,I):=H(I,I)+1L;
   END I;

   FOR I:=1 UNTIL N DO
   BEGIN
      FOR J:=K+1 UNTIL N DO
      BEGIN
         B:=0L;
         FOR T:=K+1 UNTIL N DO
            B:=B+A(I,T)*H(T,J);
         V(J):=B;
      END J;
      FOR J:=K+1 UNTIL N DO
         A(I,J):=V(J);
   END I;

   FOR J:=K+1 UNTIL  N DO
   BEGIN
      FOR I:=K+1 UNTIL N DO
      BEGIN
         B:=0L;
         FOR T:=K+1 UNTIL N DO
            B:=B+H(I,T)*A(T,J);
         V(I):=B;
      END I;
      FOR I:=K+1 UNTIL N DO
         A(I,J):=V(I);
   END J;

   A(K+1,K):= C;
   FOR J:=K+2 UNTIL N DO  A(J,K):=0L;

   RETURN:
END K;
END CHOUSEHL;


PROCEDURE CJACOBI(LONG COMPLEX ARRAY H(*,*); INTEGER VALUE N;
                  INTEGER RESULT L,R);

COMMENT
PURPOSE:
   TO COMPUTE THE NUMBER OF POSITIVE AND  THE NUMBER OF NEGATIVE
   EIGENVALUES OF A TRIDIAGONAL HERMITIAN MATRIX.

DESCRIPTION OF PARAMETERS:
INPUT PARAMETERS:
   H-TRIDIAGONAL HERMITIAN MATRIX,H(1::N,1::N),
   N-ORDER OF H.

OUTPUT PARAMETERS:
   L-NUMBER OF NEGATIVE EIGENVALUES OF H.
   R-NUMBER OF POSITIVE EIGENVALUES OF H;
```

```
BEGIN
   INTEGER RHO;  LONG REAL D0,D1,D2;

   L:=RHO:=0;
   D0:=1L;
   D1:=LONGREALPART(H(1,1));
   IF D1¬=0
   THEN BEGIN
      RHO:=1;
      IF D1<0 THEN L:=1;
   END;

   FOR I:=2 UNTIL N DO
   BEGIN
      IF H(I-1,I)=0 THEN D1:=1;
      D2:=D1*LONGREALPART(H(I,I))-D0*(ABS H(I,I-1))**2;
      IF D2¬=0
      THEN BEGIN
         IF D1¬=0
         THEN BEGIN
            RHO:=RHO+1;
            IF SIGN(D2,0L)*SIGN(D1,0L) <0 THEN L:=L+1;
         END
         ELSE BEGIN
            IF H(I-1,I)¬=0 THEN RHO:=RHO+2;
            L:=L+1;
         END;
      END;
      D0:=D1;  D1:=D2;
   END I;
   R:=RHO-L;

END CJACOBI;


PROCEDURE SINGSYS(LONG REAL ARRAY B(*,*); INTEGER NR,NC;
                  LONG REAL ARRAY X(*);INTEGER RESULT R,IER);
BEGIN


COMMENT

PURPOSE:
   TO FIND A NONTRIVIAL SOLUTION OF B*X=0.
DESCRIPTION OF PARAMETERS:
   INPUT:
   B    -MATRIX,B(1::NR,1::NC)
   NR,NC-NUMBER OF ROWS AND COLUMNS OF B,RESPECTIVELY,NR>=NC.
   OUTPUT:
   R    -SUBSCRIPT OF THE FIRST COLUMN OF B WHICH IS LINEARLY
         DEPENDENT OF THE PRECEDING ONES.  R=0:B IS NON SINGULAR.
   X    -SOLUTION OF B*X=0, X(1::NC),   X(R+1)=...=X(NC)=0.
   IER  -IER=0: NURMAL EXIT,  IER=1:  NC>NR,NO EXECUTION OF PROGRAMM.
REMARK:
   THE PROCEDURE FAILS,IF SOME COLUMN OF B IS IDENTICALLY EQUAL TO
   ZERO;
```

```
LONG REAL ARRAY A,C(1::NR,1::NC);
LONG REAL PIVOT,TEMP;  LONG REAL ARRAY MULT(1::NR);  INTEGER IZ;


PROCEDURE EQLBR(LONG REAL ARRAY A(*,*);INTEGER VALUE NR,NC;
                LONG REAL ARRAY MULT(*);INTEGER RESULT IER);
  BEGIN   LONG REAL MX;   REAL C;
  IER:=0;
  FOR I:=1 UNTIL NR DO
     BEGIN  MX:=0.L;
     FOR J:=1 UNTIL NC DO
       IF ABS A(I,J)>MX THEN MX:=ABS A(I,J);
     IF MX=0 THEN BEGIN
                  IER:=1;  MULT(I):=1L;  GOTO ENDE;
                  END;
     MULT(I):=MX:=16**(-ENTIER  (LONGLN(MX)/LONGLN(16)));
     IF MX¬=1. THEN  FOR J:=1 UNTIL NC DO  A(I,J):=A(I,J)*MX;
  ENDE:
     END I;
  END EQLBR;

IF NC>NR THEN BEGIN  IER:=1;  GOTO ENDE;  END;
FOR I:=1 UNTIL NR DO
   FOR J:=1 UNTIL NC DO  C(I,J):=A(I,J):=B(I,J);

EQLBR(A,NR,NC,MULT,IER);
R:=0;  IER:=0;
FOR K:=1 UNTIL NC-1 DO  BEGIN
   PIVOTSUCHE:
   PIVOT:=ABS A(K,K);   IZ:=K;
   FOR I:=K+1 UNTIL NR DO
      IF A(I,K)>PIVOT THEN BEGIN
         PIVOT:=ABS A(I,K);   IZ:=I;
         END;
   IF IZ¬=K THEN  BEGIN
      FOR J:=K UNTIL NC DO  BEGIN
         TEMP:=A(IZ,J);  A(IZ,J):=A(K,J);  A(K,J):=TEMP;
         END J;
      FOR J:=1 UNTIL NC DO  BEGIN
         TEMP:=C(IZ,J);  C(IZ,J):=C(K,J);  C(K,J):=TEMP;
         END J;   END;
   IF ABS A(K,K)<10*LONGEPSILON THEN  BEGIN
      R:=K;  GOTO SOLUTION;
      END;
   ELIMINATION:
   FOR J:=K+1 UNTIL NC DO  BEGIN
      A(K,J):=A(K,J)/A(K,K);
      FOR I:=K+1 UNTIL NR DO  A(I,J):=A(I,J)-A(I,K)*A(K,J);
      END J;
   END K;
IF ABS A(NC,NC)<10*LONGEPSILON THEN   R:=NC;
SOLUTION:
IF R¬=0 THEN BEGIN
   LSYS(R-1,C,X,IER);
   X(R):=-1L;
   END;
FOR I:=R+1 UNTIL NC DO  X(I):=0L;
ENDE:
END SINGSYS;
```

Außerdem werden folgende Standardprozeduren gebraucht,

die hier nur kurz aufgelistet werden sollen:

1. Zur Lösung eines nicht-singulären Gleichungssystems:

```
PROCEDURE LSYS (INTEGER VALUE N;LONG REAL ARRAY B(*,*);
                LONG REAL ARRAY X(*); INTEGER RESULT CONTROL);

DESCRIPTION OF PARAMETERS:
   N - NUMBER OF UNKNOWNS.
   B - MATRIX WITH N ROWS AND N+1 COLUMNS, THE
       FIRST N COLUMNS CONTAIN THE COEFFICIENTS OF THE JNKNOWNS,
       THE N+1-TH COLUMN THE CONSTANT TERMS.   THE COEFFICIENTS
       ARE NOT DESTROYED.
   X - SOLUTIONVECTOR OF DIMENSION N.
    CONTROL- IS NORMALLY 0. IT BECOMES -1 IF THE SYSTEM HAS NO UNIQUE
             SOLUTION. THE CALCULATION IS THEN INTERRUPTED;
```

2. Zur Vorzeichenbestimmung einer reellen Zahl:

```
INTEGER PROCEDURE SIGN(LONG REAL VALUE X,EPS);

DESCRIPTION OF PARAMETERS:
   X  -NUMBER OF WHICH THE SIGN IS TO BE COMPUTED.
   EPS-THE SIGN OF X WITH ABS X <=EPS IS PUT TO ZERO;
```

3. Zur Berechnung des konjungiert komplexen:

```
LONG COMPLEX PROCEDURE CONJ(LONG COMPLEX VALUE Z);
```

4. Zur Berechnung der Spur einer Matrix:

```
LONG COMPLEX PROCEDURE CSPUR(INTEGER VALUE N;
                             LONG COMPLEX ARRAY A(*,*));

      DESCRIPTION OF PARAMETERS:
         N-ORDER OF MATRIX A.
         A-INPUT MATRIX,A(1::N,1::N) ;
```

5. Zur Berechnung der Frobeniusnorm einer Matrix:

```
LONG REAL PROCEDURE CFNORM(INTEGER VALUE N;
                           LONG COMPLEX ARRAY A(*,*));

        DESCRIPTION OF PARAMETERS:
           N-ORDER OF MATRIX A.
           A-INPUT MATRIX,A(1::N,1::N)
```

6. Zur Berechnung von $X^T \cdot Y$ für komplexe Vektoren X, Y :

```
LONG COMPLEX PROCEDURE CVECVEC(INTEGER VALUE L;
                               LONG COMPLEX ARRAY X,Y(*));

     DESCRIPTION OF PARAMETERS:
        L  -LENGTH OF X AND Y.
        X,Y-VECTORS,X(1::N),Y(1::N);
```

Dank

Herrn Peter Martin möchte ich sehr herzlich für die Programmierungsarbeiten und einige wertvolle Hinweise danken.

Literaturverzeichnis

[1] Bellmann, Richard: Introduction to Marix Analysis, McGraw-Hill, New York, Toronto, London, 1960

[2] Gantmacher, F.R.: Matrizenrechnung I, VEB Deutscher Verlag der Wissenschaften, Berlin 1958

[3] Gregory, R.T., Karney, D.L.: A Collection of Matrices for Testing Computational Algorithms. John Wiley and Sons, Inc. 1969

[4] Howland, J.L., Senez, J.A.: A Constructive Method for the Solution of the Stability Problem. Numer. Math. 16, 1-7, 1970

[5] Isaacson, E., Keller, H.B.: Analysis of Numerical Methods, John Wiley and Sons, New York 1966

[6] Meyer-Spasche, R.: A Constructive Method of the Solution of the Liapunov Equation for Complex Matrices, to be published

[7] Ostrowski, A., Schneider, H.: Some Theorems on the Inertia of General Matrices, J. Math. Anal. Appl. 4, 72-84, 1962

[8] Wilkinson, J.H.: The Algebraic Eigenvalue Problem. Clarendon Press, Oxford 1965

[9] Wilkinson, J.H.: Rundungsfehler, Springerverlag Berlin, Heidelberg, New York 1969

[10] Martin, R.S. und Wilkinson, J.H.: The modified LR-Algorithm for Complex Hessenberg Matrices. Numer.Math. 12, 369-376 (1968)

[11] Givens, W.: Numerical Computation of the characteristic values of a real symmetric matrix. ORNL - 1574, 1954