Über die Verbesserung von Schranken für die Lösung bei linearen Gleichungssystemen

G. Alefeld, J. Herzberger, Karlsruhe

Zusammenfassung: *Lassen sich von einem linearen Gleichungssystem $x = C \cdot x + b$ die konstanten Koeffizienten nur in Intervalle einschließen, dann sucht man möglichst gute Schranken für die Lösungsmenge dieser Schar von Gleichungen. Zu deren Berechnung gibt es verschiedene Iterationsverfahren. Die vorliegende Arbeit behandelt ein Verfahren, welches es gestattet, derart berechnete Schranken in vielen Fällen beträchtlich zu verbessern. Hierfür wird ein Algol-60-Programm angegeben. Die Wirksamkeit dieser Methode wird an einigen numerischen Beispielen diskutiert.*

Summary: *The coefficients of a system of linear algebraic equations $x = Cx + d$ are only known to be in the range of some given intervals. There exist iterative methods for calculating bounds for the set of all possible solutions. This paper is concerned with a method to improve considerably such bounds in many cases. A suitable algorithm is formulated in Algol-60 and some of the numerical results are discussed.*

0. Einleitung

Wir betrachten die Aufgabe, die Lösungsmenge Ω einer Schar von linearen Gleichungssystemen der Gestalt

$$\underline{x} = \underline{C}\underline{x} + \underline{b};\ \underline{x} = (x_i),\ \underline{C} = (c_{ij}),\ \underline{b} = (b_i) \qquad (1)$$

in Schranken einzuschließen. Die Schar der Gleichungssysteme sei dabei gegeben durch Intervalle $C_{ij} = [c_{ij}^1, c_{ij}^2]$ bzw. $B_i = [b_i^1, b_i^2]$ für die möglichen Werte der Koeffizienten der Matrix $\underline{C}$ bzw. des Vektors $\underline{b}$. Solche Intervallmatrizen $\mathbf{C} = (C_{ij})$ bzw. Intervallvektoren $\mathbf{b} = (B_i)$ treten schon dann auf, wenn man beim Einlesen von auf der Rechenanlage nicht darstellbaren Konstanten die Konvertierungsfelder erfassen will. Mit Hilfe der Eigenschaften der bekannten Verknüpfungen in der Intervallrechnung (siehe hierzu [7], [9]) kann man die obige Aufgabenstellung mit dem intervallmäßigen Gleichungssystem

$$\mathbf{x} = \mathbf{C}\mathbf{x} + \mathbf{b};\ \mathbf{x} = (X_i),\ \mathbf{C} = (C_{ij}),\ \mathbf{b} = (B_i) \qquad (2)$$

in Verbindung bringen. Unter bestimmten Voraussetzungen über die Matrix $\mathbf{C}$ besitzt die Gleichung (2) einen eindeutig bestimmten Fixpunkt x^*. Für diesen besteht dann die Beziehung $\Omega \subset x^*$. Verschiedene Iterationsverfahren zur Bestimmung von x^* sind in [7] und [8] untersucht worden. Entstehen die Koeffizientenintervalle C_{ij} und B_i nicht nur durch das Erfassen von Einlesefehlern, sondern sind sie aufgrund der vorliegenden Problemstellung als Intervalle von nichtverschwindendem Durchmesser gegeben, dann ist x^* in vielen Fällen eine relativ grobe Einschließung für Ω. Die bestmögliche Einschließung der Lösungsmenge Ω ist die Intervallhülle v von Ω, d. h. der Durchschnitt aller Intervallvektoren y mit der Eigenschaft $\Omega \subset y$. Zur Illustration dieses Sachverhalts sei das folgende Beispiel angegeben (siehe Beispiel 1).

In der vorliegenden Arbeit wird nun eine Methode beschrieben, die es gestattet, die durch den Intervallvektor x^* gegebenen Schranken für die Lösungsmenge Ω in vielen Fällen beträchtlich zu verbessern. Es wird also ein Intervallvektor $\hat{x}$ berechnet, für den stets gilt: $v \subset \hat{x} \subset x^*$. Bei der Lösung derartiger intervallmäßiger Aufgaben handelt es sich um nichtlineare Probleme. Die angegebene Methode kann man auch als ein Verfahren in einer verallgemeinerten Intervallarithmetik deuten, deren Beschreibung etwa in [6] gegeben wurde. Das verwendete Steuerungsprinzip wurde von Hansen in [5] zur Lösung von $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ und bei [12] zur intervallarithmetischen Behandlung des Gauß' schen Algorithmus angewendet.

1. Hilfsmittel

Für die späteren Überlegungen sollen hier kurz einige Begriffe aus der Intervallarithmetik aufgeführt werden. Unter dem Betrag eines Intervalles $X = [x^1, x^2]$ versteht man die Größe $|X| := \max\{|x^1|, |x^2|\}$. Entsprechend definiert man $|\mathbf{X}| := |(X_{ij})| = (|X_{ij}|)$ bzw. $|x| := |(X_i)| = (|X_i|)$. Daneben definiert man den Durchmesser eines Intervalles X als $d(X) = d([x^1, x^2]) = x^2 - x^1$ und bildet dann entsprechend die Größen $d(\mathbf{X}) = d((X_{ij})) = (d(X_{ij}))$ bzw. $d(x) = d((X_i)) = (d(X_i))$. Gilt für zwei Intervallmatrizen $\mathbf{X}$ und $\mathbf{Y}$ die Beziehung $\mathbf{X} \subset \mathbf{Y}$, d. h. $X_{ij} \subset Y_{ij}$ für $1 \leqslant i, j \leqslant n$, dann folgt daraus stets $|\mathbf{X}| \leqslant |\mathbf{Y}|$ und $d(\mathbf{X}) \leqslant d(\mathbf{Y})$. Entsprechendes gilt für Intervallvektoren. Bezüglich weiterer Rechenregeln im Zusammenhang mit Beträgen und Durchmessern vergleiche man etwa [7] und [3]. Zu je zwei Intervallen X und Y ist der Abstand erklärt als

$$q(X, Y) = q([x^1, x^2], [y^1, y^2]) := \max\{|x^1 - y^1|, |x^2 - y^2|\};$$

Beispiel 1:

$$\mathbf{C} = \begin{bmatrix} [-\frac{1}{2}, -\frac{1}{2}] & [-\frac{1}{4}, -\frac{1}{8}] \\ [-\frac{1}{2}, -\frac{1}{2}] & [-\frac{1}{4}, -\frac{1}{8}] \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} [\frac{3}{4}, 1] \\ [\frac{3}{4}, 1] \end{bmatrix} \quad v = \begin{bmatrix} [\frac{19}{50}, \frac{37}{58}] \\ [\frac{10}{29}, \frac{18}{25}] \end{bmatrix} \quad x^* = \begin{bmatrix} [0,1] \\ [0,1] \end{bmatrix}$$

entsprechend definiert man $q(x, y) = q((X_i), (Y_i)) = (q(X_i, Y_i))$. Ferner ordnet man jedem Intervall X die Größe sign (X) zu, welche sich aus folgender Tabelle berechnet:

$$\text{sign}(X) = \text{sign}([x^1, x^2]) = \begin{cases} 1 \text{ falls } x^1 > 0 \\ 0 \text{ falls } x^1 \leqslant 0, x^2 \geqslant 0 \\ -1 \text{ falls } x^2 < 0 \end{cases}$$

Es gilt dann: $\text{sign}(X \cdot Y) = \text{sign}(X) \cdot \text{sign}(Y)$.

Nach dieser kurzen Aufzählung von Begriffen kehren wir nun zu Aufgabenstellung (2) zurück. Für das Iterationsverfahren

$$x'_{n+1} = \{ Cx'_n + b \} \cap x'_n \qquad (2)^*$$

lassen sich dabei die folgenden Aussagen beweisen:

Aus $v \subset x'_0$ folgt $v \subset x'_n$ für alle $n \geqslant 1$, und die Iterationsfolge $x'_0 \supset x'_1 \supset x'_2 \supset \ldots$ konvergiert genau dann gegen den eindeutigen Fixpunkt x^*, wenn für den Spektralradius $\rho(|C|) < 1$ gilt.

In der Praxis wird man als hinreichendes Kriterium für die Konvergenz des Iterationsverfahrens $(2)^*$ die Bedingung $\| |C| \| < 1$ für eine Matrixnorm $\| \cdot \|$ verwenden. Neben diesem Vorgehen (dem sog. Gesamtschrittverfahren) kann man noch ein weiteres Iterationsverfahren zur Berechnung von x^* angeben. Dazu sei die Matrix **C** zerlegt in die Summe $\mathbf{C} = \mathbf{L} + \mathbf{D} + \mathbf{R}$, wobei **L** bzw. **R** eine strenge untere bzw. obere Dreiecksmatrix und **D** eine Diagonalmatrix bezeichne.

Dann lautet die Vorschrift:

$$x_{n+1} = \{ (\mathbf{D} + \mathbf{R}) x_n + \mathbf{L} x_{n+1} + b \} \cap x_n \; . \qquad (2)^{**}$$

Die Iterationsfolge $x_0 \supset x_1 \supset x_2 \supset \ldots$ konvergiert genau dann gegen den eindeutigen Fixpunkt x^*, wenn $\rho((\underline{E} - |L|)^{-1}(|D| + |R|)) < 1$ gilt. Mit Hilfe der Theorie der nichtnegativen Matrizen (siehe [10]) folgt, daß Verfahren $(2)^*$ genau dann gegen x^* konvergiert, wenn dies für Verfahren $(2)^{**}$ der Fall ist. Da sich beweisen läßt, daß bei $x'_0 = x_0$ stets $x_n \subset x'_n$ gilt, wird man Verfahren $(2)^{**}$ den Vorzug geben. Für den Fixpunkt x^* läßt sich allgemein nur die Beziehung

$$q(v, x^*) \leqslant |C| \cdot (\underline{E} - |C|)^{-1} \cdot d(x^*)$$

beweisen. Ist jedoch $\text{sign}(C_{ij}) = 1$ für alle $1 \leqslant i, j \leqslant n$, dann gilt stets $v = x^*$. Da das jedoch allgemein nicht der Fall ist, wie auch Beispiel 1 zeigt, soll nun eine Methode beschrieben werden, die es gestattet, von x^* ausgehend einen verbesserten Einschließungsvektor $\hat{x}$ mit $v \subset \hat{x} \subset x^*$ zu berechnen. In einer Reihe von Fällen läßt sich dabei sogar $v = \hat{x}$ nachweisen.

2. Verfahren

Wir betrachten ein Gleichungssystem der Gestalt (1) und untersuchen, wie sich dessen Lösung $\underline{x} = \underline{x}(\underline{C}, \underline{b})$ in Abhängigkeit von den Konstanten c_{ij} und b_i verhält. Dafür maßgebend ist das Vorzeichenverhalten der einzelnen partiellen Ableitungen

$\frac{\partial \underline{x}}{\partial c_{ij}}$ und $\frac{\partial \underline{x}}{\partial b_i}$. Gilt etwa für alle $\underline{C} \in \mathbf{C}$ und alle $\underline{b} \in \mathbf{b}$ stets

$\frac{\partial x_k}{\partial c_{ij}} \geqslant 0$, dann durchläuft die Lösungskomponente x_k von $\underline{x}$ monoton alle Werte von w_k^1 bis w_k^2. Dabei bestimmt c_{ij}^1 bzw. c_{ij}^2 den Wert von w_k^1 bzw. w_k^2. Eine analoge Überlegung gilt im Falle von $\frac{\partial x_k}{\partial c_{ij}} \leqslant 0$, wo w_k^1 bzw. w_k^2 durch die Schranken c_{ij}^2 bzw. c_{ij}^1 bestimmt werden. Wechselt dagegen $\frac{\partial x_k}{\partial c_{ij}}$ mindestens einmal sein Vorzeichen, dann werden w_k^1 und w_k^2 durch zwei i. a. nicht näher bekannte Werte aus C_{ij} bestimmt.

Die gleichen Überlegungen wie für die Abhängigkeit der Lösung von c_{ij} gelten auch für ihre Abhängigkeit von b_i. Um die oben aufgezählten Fallunterscheidungen treffen zu können, müssen noch die pariellen Ableitungen

$\frac{\partial x_k}{\partial c_{ij}}$ und $\frac{\partial x_k}{\partial b_i}$ berechnet werden. Wir gehen dabei wie in [5] vor, wo dies im Falle $\underline{A}\,\underline{x} = \underline{b}$ durchgeführt wird.

Aus (1) folgen durch partielle Differation nach c_{ij} die folgenden Beziehungen:

$$\frac{\partial \underline{x}}{\partial c_{ij}} = (\delta_{ik}\, \delta_{lj})\; \underline{x} + \underline{C}\, \frac{\partial \underline{x}}{\partial c_{ij}} \; ,$$

$$\mathbf{O} = (\delta_{ik}\, \delta_{lj})\; \underline{x} - (\underline{E} - \underline{C})\, \frac{\partial \underline{x}}{\partial c_{ij}} \; ,$$

$$\frac{\partial \underline{x}}{\partial c_{ij}} = x_j\, (\underline{E} - \underline{C})^{-1}\, \underline{e}_i \quad (\underline{e}_i \text{ ist der i-te Einheitsvektor}).$$

Damit erhalten wir:

$$\frac{\partial x_k}{\partial c_{ij}} = x_j \cdot (\underline{E} - \underline{C})^{-1}_{ki} \; . \qquad (3a)$$

Bei partieller Differentiation nach b_i folgt entsprechend:

$$\frac{\partial \underline{x}}{\partial b_i} = \underline{C}\, \frac{\partial \underline{x}}{\partial b_i} + \underline{e}_i,$$

$$\frac{\partial \underline{x}}{\partial b_i} = (\underline{E} - \underline{C})^{-1} \cdot \underline{e}_i,$$

und damit dann

$$\frac{\partial x_k}{\partial b_i} = (\underline{E} - \underline{C})^{-1}_{ki} \; . \qquad (3b)$$

Aufgrund der eben hergeleiteten Formeln (3a), (3b) hat man sich je eine Einschließung für die Mengen $\{(\underline{E} - \underline{C})^{-1} : \underline{C} \in \mathbf{C}\} = \tilde{\Omega}$ und v zu verschaffen, um Aussagen über das Vorzeichenverhalten der partiellen Ableitungen machen zu können. Mit Hilfe von Verfahren $(2)^{**}$ erhält man etwa die Einschließung $x^* = (X_i^*)$ für v, und das analog aufgebaute Verfahren

$$X_{n+1} = \{ (\mathbf{D} + \mathbf{R}) X_n + \mathbf{L} X_{n+1} + \underline{E} \} \cap X_n \; , \qquad (4)$$

für das die gleiche Konvergenzbedingung gilt wie für $(2)^{**}$, liefert als Fixpunkt $X^* = (X_{ij}^*)$ eine Einschließung für $\tilde{\Omega}$. Nun läßt sich die vorgeschlagene Methode wie folgt beschreiben:

Man bilde für $1 \leqslant k \leqslant n$ jeweils die Gleichungssysteme

$$^{(k)}y = {}^{(k)}\mathbf{A}\; {}^{(k)}y + {}^{(k)}f \qquad (5a)$$

$$^{(k)}\underline{z} = {}^{(k)}\underline{H}\; {}^{(k)}\underline{z} + {}^{(k)}g \qquad (5b)$$

mit Hilfe der Vorschriften:

$$({}^{(k)}A_{ij}) = \begin{cases} [c_{ij}^1, c_{ij}^1] & \text{für sign } (X_{ki}^*) \cdot \text{sign } (X_j^*) = 1 \\ [c_{ij}^2, c_{ij}^2] & \text{für sign } (X_{ki}^*) \cdot \text{sign } (X_j^*) = -1 \\ C_{ij} & \text{sonst} \end{cases}$$

$$({}^{(k)}F_i) = \begin{cases} [b_i^1, b_i^1] & \text{für sign } (X_{ki}^*) = 1 \\ [b_i^2, b_i^2] & \text{für sign } (X_{ki}^*) = -1 \\ B_i & \text{sonst} \end{cases}$$

$$({}^{(k)}H_{ij}) = \begin{cases} [c_{ij}^2, c_{ij}^2] & \text{für sign } (X_{ki}^*) \cdot \text{sign } (X_j^*) = 1 \\ [c_{ij}^1, c_{ij}^1] & \text{für sign } (X_{ki}^*) \cdot \text{sign } (X_j^*) = -1 \\ C_{ij} & \text{sonst} \end{cases}$$

$$({}^{(k)}G_i) = \begin{cases} [b_i^2, b_i^2] & \text{für sign } (X_{ki}^*) = 1 \\ [b_i^1, b_i^1] & \text{für sign } (X_{ki}^*) = -1 \\ B_i & \text{sonst} \end{cases}$$

Man berechne die Fixpunkte ${}^{(k)}y^*$ und ${}^{(k)}z^*$ dieser beiden Systeme, etwa nach dem Verfahren $(2)^{**}$, und betrachte davon jeweils die k-te Komponente ${}^{(k)}Y_k^*$ bzw. ${}^{(k)}Z_k^*$. Daraus bildet man die Komponente $\hat{X}_k = [{}^{(k)}y_k^{*1}, {}^{(k)}z_k^{*2}]$ des Intervallvektors $\hat{x}$. Für das eben beschriebene Verfahren gilt der folgende

Satz: Vorgelegt sei die Gleichung (2) $x = C x + b$ mit $\| |C| \| < 1$ für eine Matrixnorm $\| \cdot \|$. Dann gelten die Aussagen:

I. Für $v \subset x_0$ und $\{ (\underset{\cdot}{E} - \underset{\cdot}{C})^{-1} : \underset{\cdot}{C} \in C \} \subset X_0$ konvergieren die Verfahren $(2)^{**}$ bzw. (4) monoton gegen den eindeutigen Fixpunkt x^* bzw. X^*.

II. Mit ${}^{(k)}y_0 = {}^{(k)}z_0 = x^*$ konvergiert Verfahren $(2)^{**}$ für die Gleichungen (5a), (5b) gegen den eindeutigen Fixpunkt ${}^{(k)}y^*$ bzw. ${}^{(k)}z^*$. Der Intervallvektor $\hat{x}$ erfüllt die Beziehung $v \subset \hat{x} \subset x^*$.

Beweis:
Zu I. Dies folgt aus den eingangs erwähnten Aussagen.
Zu II. Es gilt ${}^{(k)}A \subset C$, ${}^{(k)}H \subset C$ und ${}^{(k)}f \subset b$, ${}^{(k)}g \subset b$. Die Inklusionsmonotonie der Intervallverknüpfungen stellt sicher, daß x^* die Lösungsmengen ${}^{(k)}\Omega_1 = \{ (\underset{\cdot}{E} - \underset{\cdot}{A})^{-1} \underset{\cdot}{f} : \underset{\cdot}{A} \in {}^{(k)}A, \underset{\cdot}{f} \in {}^{(k)}f \}$ und ${}^{(k)}\Omega_2 = \{ (\underset{\cdot}{E} - \underset{\cdot}{H})^{-1} \underset{\cdot}{g} : \underset{\cdot}{H} \in {}^{(k)}H, \underset{\cdot}{g} \in {}^{(k)}g \}$ umfaßt. Außerdem gilt wegen ${}^{(k)}A \subset C$, ${}^{(k)}H \subset C$ auch $|{}^{(k)}A| \leqslant |C|$ und $|{}^{(k)}H| \leqslant |C|$. Wegen $\| |C| \| < 1$ ist dann auch $\| |{}^{(k)}A| \|' < 1$ und $\| |{}^{(k)}H| \|'' < 1$ für zwei Matrixnormen $\| \cdot \|'$, $\| \cdot \|''$. Somit konvergiert Verfahren $(2)^{**}$ für die Gleichungen (5a) bzw. (5b) gegen ${}^{(k)}y^* \subset x^*$ bzw. ${}^{(k)}z^* \subset x^*$. Damit ist $\hat{x} \subset x^*$ nachgewiesen. $v \subset \hat{x}$ folgt schließlich aus der Konstruktion der Systeme (5a) bzw. (5b).

Bemerkungen:
Im Falle von sign $(X_{ki}^*) = 0, 1 \leqslant i \leqslant n$, folgt wegen ${}^{(k)}A = {}^{(k)}H = C$ und ${}^{(k)}f = {}^{(k)}g = b$ die Gleichung ${}^{(k)}y^* = {}^{(k)}z^* = x^*$. Es gilt somit $\hat{X}_k = X_k^*$, d.h. das Verfahren bringt in der k-ten Komponente keine Verbesserung. Damit ergibt sich für sign $(X_{ij}^*) = 0$, $1 \leqslant i, j \leqslant n$ die Aussage $\hat{x} = x^*$, d.h. das Verfahren geht in Verfahren $(2)^{**}$ über. Ist hingegen $\prod_{1 \leqslant i,\, j \leqslant n} \text{sign } (X_{ij}^*) \text{ sign } (X_i^*) \neq 0$ erfüllt, dann folgt aus der Konstruktion dieser Methode $\hat{x} = v$. Somit liefert das Verfahren in diesem Falle die Intervallhülle von Ω. Tritt für mindestens einen Index l sign $(X_l^*) \neq$ sign $(\hat{X}_l)$ ein, dann läßt sich das Verfahren mit $\hat{x}$ an Stelle von x^* nochmals wiederholen. Ein solcher Fall kann höchstens n-mal auftreten, bis sign $(\hat{X}_i) \neq 0$, $1 \leqslant i \leqslant n$ eingetreten ist.

Es sei noch bemerkt, daß die Einschließungsmengen x^* und X^* bei diesem Verfahren nicht notwendig in der geschilderten Weise bestimmt werden müssen. Daneben gibt es noch die Möglichkeit, durch einfache Normabschätzungen solche Einschließungen zu berechnen (vergleiche etwa [11]), auf die hier nicht eingegangen wird.

3. Algorithmus

In diesem Abschnitt geben wir nun eine Algol-Prozedur an, welche für ein System der Form (2) nach dem in Abschnitt 2 beschriebenen Vorgehen die Lösungsmenge v unter Berücksichtigung von Rundungsfehlern einschließt. Im Rumpf dieser Prozedur werden mit Hilfe der Prozedur iterat die Iterationsverfahren $(2)^{**}$ und (4) durchgeführt. Einschließungsmengen für die jeweiligen Fixpunkte findet man bei erfülltem Zeilensummenkriterium mit Hilfe der Prozedur einschl1, bei erfülltem Spaltensummenkriterium mit Hilfe der Prozedur einschl2 (siehe [1]). Aus der Vorzeichenverteilung dieser Fixpunkte können die neuen Systeme zur Verbesserung der Einschließung von v aufgebaut werden. Diese Systeme werden wiederum mit Hilfe der Prozedur iterat gelöst. Die Rechnung wird beendet, falls bei diesem Vorgehen keine Verbesserung mehr möglich ist. Die Prozedur lingl setzt als globale Größe die Vereinbarung der Algol-Prozedur low voraus (siehe [4]), mit deren Hilfe auf beliebigen Algol-60-Compilern die Intervallverknüpfungen unter Einschluß der Rundungsfehler realisiert wurden. Da die Intervalladdition bzw. Intervallmultiplikation nur an einer Stelle des Programms vorkommen, wurde auf die Verwendung der dafür vorgesehenen Prozeduren verzichtet. Die dadurch gewonnene Rechenzeitersparnis ist erheblich.

Die Prozedur lingl

Liste der formalen Parameter:

integer n	n ist die Ordnung des linearen Gleichungssystems.
array mat	mat steht für eine nxn-Intervallmatrix mat = ([mat_{ij}^1, mat_{ij}^2]); die Feldkomponenten mat [i, j, 1] bzw. mat [i, j, 2] entsprechen dabei den Intervallschranken mat_{ij}^1 bzw. mat_{ij}^2.
array vek	steht für einen Vektor vek = ([vek_i^1, vek_i^2]), die Feldkomponenten vek [i, 1] bzw. vek [i, 2] entsprechen dabei den Intervallschranken vek_i^1 bzw. vek_i^2.

array x — steht für einen Vektor $x = ([x_i^1, x_i^2])$; die Feldkomponenten x [i, 1] bzw. x [i, 2] entsprechen dabei den Intervallschranken x_i^1 bzw. x_i^2.

label exit1 — steht für eine Marke, welche angesprungen wird, falls bei den Ausgangsdaten eine untere Intervallschranke größer ist als die entsprechende obere.

label exit2 — steht für eine Marke, welche angesprungen wird, falls für die eingelesene Matrix mat keines der beiden untersuchten Konvergenzkriterien erfüllt ist.

```
procedure lingl (mat, vek, n, exit1, exit2, x);
          value n;
          integer n;
          array mat, vek, x;
          label exit1, exit2;
begin  integer g, i, j, k, l;
       real norm;
       boolean b1, b2;
       integer array sa [1 : n, 1 : n], sx [1 : n];
       array z [1 : n], b [1 : n, 1 : 2], xr [1 : n, 1 : n, 1 : 2, 1 : 2],
             a [1 : n, 1 : n, 1 : 2];
procedure iterat (mat, vek, x);
          array mat, vek, x;
begin  real a1, a2, b1, b2, c1, c2, ik1, ik2, hx1, hx2, p;
       integer i, j;
       boolean ba, bb, bv;
m: bv:=true;
for i:=1 step 1 until n do
begin ik1:=vek[i,1]; ik2:=vek[i,2];
      for j:=1 step 1 until n do
      begin a2:=-mat[i, j, 2]; b2:=-x[j, 2];
            ba:=a2 less 0; bb:=b2 less 0;
            if ba then a1:=mat[i, j, 1] else
            begin a1:=a2; a2:=mat[i, j, 1] end;
            if bb then b1:=x[j, 1] else
            begin b1:=b2; b2:=x[j, 1] end;
            c2:=low (- a2×b2);
            if b1 less 0 then
            begin c1:=low (-a2×b1);
                  if a1 less 0 then
                  begin p:=low (-a1×b2);
                        if p less c1 then c1:=p;
                        p:=low (-a1×b1);
                        if p less c2 then c2:=p
                  end
            end
            else
            c1:=if a1 less 0 then low(-a1×b2)else low(a1×b1);
            if ba equiv bb then
            begin hx1:=c1; hx2:=-c2 end
            else
            begin hx1:=c2; hx2:=-c1 end;
            ik1:=low(hx1+ik1);
            ik2:=-low(-hx2-ik2)
      end j;
      hx1:=x[i, 1]; hx2:=x[i, 2];
      if ik1 less hx1 then ik1:=hx1;
      if ik2 greater hx2 then ik2:=hx2;
      if bv then
      begin if ik1 notequal x[i, 1]or
            ik2 notequal x[i, 2]then
            bv:=false
      end;
      x[i, 1]:=ik1; x[i, 2]:=ik2
end i;
if not bv then goto m
end iterat;

procedure zeilennorm(y,b);
          array y;
          boolean b;
begin  integer i, j;
       real h,ugr,ogr;
b:=true;
for i:=1 step 1 until n do
begin  h:=0;
       for j:=1 step 1 until n do
       begin ugr:=abs (mat[i, j, 1]); ogr:=abs (mat[i, j, 2]);
             ogr:=if ugr greater ogr then ugr else ogr;
             h:=-low(-h-ogr)
       end j;
       if h notless 1 then
       begin b:=false;
             goto m
       end;
       y[i]:=h
end i;
m:
end zeilennorm;

procedure spaltennorm(norm,b);
          real norm;
          boolean b;
begin  integer i, j;
       real hnorm, ugr, ogr;
norm:=0;
b:=true;
for j:=1 step 1 until n do
begin  hnorm:=0;
       for i:=1 step 1 until n do
       begin ugr:=abs (mat[i, j, 1]); ogr:=abs (mat[i, j, 2]);
             ogr:=if ugr greater ogr then ugr else ogr;
             hnorm:=-low(-hnorm-ogr)
       end i;
       if hnorm greater norm then norm:=hnorm;
       if norm notless 1 then
       begin b:=false;
             goto m
       end
end j;
m:
end spaltennorm;

procedure einschl1 (y, vek, x);
          array y, vek, x;
begin  integer i, j;
       real ugr,ogr,max,hmax;
       array z[1 : n];
for i:=1 step 1 until n do
begin  ugr:=abs (vek[i, 1]); ogr:=abs (vek[i, 2]);
       z[i]:=if ugr greater ogr then ugr else ogr
end i;
max:=0;
for i:=1 step 1 until n do
begin  hmax:=0;
       for j:=1 step 1 until n do
       begin  ugr:=abs(mat[i,j,1]); ogr:=abs(mat[i,j,2]);
              ogr:=if ugr greater ogr then ugr else ogr;
              hmax:=-low(-hmax+low(-ogr×z[j]))
       end j;
       hmax:=-low(-hmax/low(1-y[i]));
       if max less hmax then max:=hmax
end i;
for i:=1 step 1 until n do
begin  x[i,1]:=low(vek[i,1]-max);
       x[i,2]:=-low(-vek[i,2]-max)
end i
end einschl1;
```

```
procedure einschl2(norm,vek,x);
          real norm;
          array vek,x;
begin  integer i,j;
       real max,ugr,ogr;
       array z[1:n];
max:=0;
for i:= 1 step 1 until n do
begin ugr:=abs(vek[i,1]); ogr:=abs(vek[i,2]);
      z[i]:=if ugr greater ogr then ugr else ogr
end i;
for i:=1 step 1 until n do
for j:=1 step 1 until n do
begin  ugr:=abs(mat[i,j,1]); ogr:=abs(mat[i,j,2]);
       ogr:=if ugr greater ogr then ugr else ogr;
       max:=-low(-max+low(-ogr X z[j]))
end i,j;
max:=-low(-max/low(1-norm));
for i:=1 step 1 until n do
begin  x[i,1]:=low(vek[i,1]-max);
       x[i,2]:=-low(-vek[i,2]-max)
end i
end einschl2;


for i:=1 step 1 until n do
begin  if vek[i,1] greater vek[i,2] then goto exit1;
       for j:=1 step 1 until n do
       if mat[i,j,1] greater mat[i,j,2] then goto exit1
end i;
zeilennorm(z,b1);
if b1 then einschl1(z,vek,x)
else
begin  spaltennorm(norm,b2);
       if b2 then einschl2(norm,vek,x)
       else
       goto exit2
end;
iterat(mat,vek,x);
for i:=1 step 1 until n do
begin  sx[i]:=if sign(x[i,1])X sign(x[i,2]) greater 0 then
              sign(x[i,1]) else 0;
       for j:=1 step 1 until n do
       begin  xr[j,i,1,1]:=xr[j,i,1,2]:=x[j,1];
              xr[j,i,2,1]:=xr[j,i,2,2]:=x[j,2]
       end j
end i;
for k:=1 step 1 until n do
begin  for i:=1 step 1 until n do
       b[i,1]:=b[i,2]:=if i equal k then 1 else 0;
       if b1 then einschl1(z,b,x) else einschl2(norm,b,x);
       iterat(mat,b,x);
       for i:=1 step 1 until n do
       sa[i,k]:=if sign(x[i,1])X sing(x[i,2]) greater 0
                then sign(x[i,1]) else 0
end k;
m:
for k:=1 step 1 until n do
begin  for g:=1,2 do
       begin  for i:=1 step 1 until n do
              begin  if sa[k,i] equal 0 then
                     begin  b[i,1]:=vek[i,1];
                            b[i,2]:=vek[i,2]
                     end
                     else
                     begin  l:=if sa[k,i] equal 1 then 1 else 2;
                            if g equal 2 then l:=3-l;
                            b[i,1]:=b[i,2]:=vek[i,l]
                     end;
                     for j:=1 step 1 until n do
                     begin  if sa[k,i]X sx[j] equal 0 then
                            begin  a[i,j,1]:=mat[i,j,1];
                                   a[i,j,2]:=mat[i,j,2]
                            end
                            else
                            begin  l:=if sa[k,i]X sx[j] equal
                                      1 then 1 else 2;
                                   if g equal 2 then l:=3-l;
                                   a[i,j,1]:=a[i,j,2]:=mat[i,j,l]
                            end
                     end j
              end i;
              for j:=1 step 1 until n do
              begin  x[j,1]:=xr[j,k,1,g];
                     x[j,2]:=xr[j,k,2,g]
              end j;
              iterat(a,b,x);
              l:=sign(xr[k,k,g,g]);
              for i:=1 step 1 until n do
              begin  xr[i,k,1,g]:=x[i,1];
                     xr[i,k,2,g]:=x[i,2]
              end i;
              if l notequal sign(x[k,g]) and
                sign(x[k,g]) notequal 0 then
              begin  sx[k]:=sign(x[k,g]);
                     goto m
              end
       end g
end k;
for i:=1 step 1 until n do
begin  x[i,1]:=xr[1,i,1,1];
       x[i,2]:=xr[i,i,2,2]
end i
end prozedur lingl
```

4. Numerische Beispiele

In diesem Abschnitt geben wir einige der gerechneten Beispiele an. Es werden jeweils zunächst für ein System der Form (2) die Matrix **C** und der Vektor **b** angegeben. Daneben wird unter Verwendung der Vektornorm

$$\| \underset{\cdot}{x} \| = \| (x_i) \| = \sum_{i=1}^{n} | x_i | \quad \text{das Verhältnis} \quad \sigma = \frac{\| d(x^*) \|}{\| d(\hat{x}) \|}$$

der Normen des Durchmessers von x^* (Fixpunkt von (2)) und $\hat{x}$ („verbesserte" Einschließung von **v** mit Hilfe der Prozedur lingl) angegeben.

a)
$$\mathbf{C} = \begin{pmatrix} [-0.70000, & -0.69998] & [-0.20000, & -0.19998] \\ [-0.20000, & -0.19998] & [-0.70000, & -0.69998] \end{pmatrix};$$

$$\mathbf{b} = \begin{pmatrix} [0.9, 1] \\ [0.9, 1] \end{pmatrix}; \quad x^* \approx \begin{pmatrix} [-0.33_{10}-10, 1.01] \\ [-0.34_{10}-10, 1.01] \end{pmatrix};$$

$$\hat{x} \approx \begin{pmatrix} [0.46666, & 0.53335] \\ [0.46666, & 0.53335] \end{pmatrix}; \quad d(x^*) \approx \begin{pmatrix} 1 \\ 1 \end{pmatrix}; \quad d(\hat{x}) \approx \begin{pmatrix} 0.067 \\ 0.067 \end{pmatrix};$$

daraus ergibt sich $\sigma = \dfrac{\| d(x^*) \|}{\| d(\hat{x}) \|} \approx \dfrac{2}{0.134} \approx 16;$

(Bei den nachfolgenden Beispielen wurde σ analog berechnet.)

b)
$$\mathbf{C} = \begin{pmatrix} [0.155,\ 0.156] & [-0.156,\ -0.154] & [0.142,\ 0.143] \\ [0.125,\ 0.126] & [-0.123,\ -0.122] & [0.70,\ 0.7001] \\ [0.007,\ 0.008] & [\ 0.009,\ 0.009] & [-0.9,\ -0.89] \end{pmatrix};$$

$$\mathbf{b} = \begin{pmatrix} [0.99, 1] \\ [0.99, 1] \\ [0.99, 1] \end{pmatrix};$$

$\sigma \approx 6$;

c)
$$\mathbf{C} = \begin{pmatrix} -0.3 & -0.2 & -0.1 & -0.3 \\ -0.3 & -0.3 & -0.2 & -0.1 \\ -0.1 & -0.3 & -0.3 & -0.2 \\ -0.2 & -0.1 & -0.3 & -0.3 \end{pmatrix} + \mathbf{H}$$

mit $\mathbf{H} = (H_{ij})$; $H_{ij} = [-_{10}{}^{-6}, {}_{10}{}^{+6}]$, i, j = 1 (1) n.

$$\mathbf{b} = \begin{pmatrix} [0.89, 1.01] \\ [0.90, 1.02] \\ [0.88, 1{,}01] \\ [0.90, 1{,}02] \end{pmatrix};$$

$\sigma \approx 9.7$;

d)
$$\mathbf{C} = \begin{pmatrix} [-0.2,\ -0.1] & [-0.1,\ 0] & [0.1\ 0.2] & [0,\ 0.1] & [0.1,\ 0.11] \\ [-0.1,\ -0.1] & [-0.2,\ -0.19] & [0.19,\ 0.19] & [0,\ 0] & [0.1,\ 0.11] \\ [-0.1,\ -0.09] & [0.1,\ 0.15] & [0.01,\ 0.02] & [0.1,\ 0.1] & [0.2,\ 0.21] \\ [0.18,\ 0.19] & [-0.1,\ -0.1] & [-0.19,\ -0.18] & [-0.2,\ -0.19] & [0,\ 0] \\ [0.1,\ 0.11] & [0.09,\ 0.1] & [-0.09,\ -0.09] & [0,\ 0] & [-0.02,\ 0] \end{pmatrix};$$

$$\mathbf{b} = \begin{pmatrix} [1,\ 1.1] \\ [0,\ 0] \\ [0.9,\ 1] \\ [-5,\ -4.9] \\ [-16,\ -15.9] \end{pmatrix};$$

$\sigma \approx 1.65$;

Die Beispiele wurden auf der Rechenanlage ELECTROLOGICA X8 am Rechenzentrum der Universität Karlsruhe gerechnet.

Literatur

[1] *G. Alefeld:* Über die aus monoton zerlegbaren Operatoren gebildeten Iterationsverfahren. Comp. 6, 161–172 (1970).

[2] *G. Alefeld u. J. Herzberger:* Algol-60 Algorithmen zur Auflösung linearer Gleichungssysteme mit Fehlererfassung. Comp. 6, 28–34 (1970).

[3] *G. Alefeld u. J. Herzberger:* Matrizeninvertierung mit Fehlererfassung. elektron. datenverarbeitung. (1970), H.9, S.410-416.

[4] *H. Christ:* Realisierung einer Maschinenintervallarithmetik mit beliebigen Algol-60-Compilern. Elektron. Rechenanlagen, **10** (1968); 217-222.

[5] *E. Hansen:* On the Solution of Linear Algebraic Equations with Interval Coefficients. Linear Algebra and Its Applications, **2** (1969), 153-165.

[6] *J. Herzberger:* Definition und Eigenschaften allgemeiner Intervallräume. ZAMM **50**, T50-51, 1970.

[7] *U. Kulisch:* Grundzüge der Intervallrechnung, In 'Überblicke Mathematik 2', 1969, Bibliographisches Institut Mannheim.

[8] *O. Mayer:* Über die Bestimmung von Einschließungsmengen für die Lösungen linearer Gleichungssysteme mit fehlerbehafteten Koeffizienten. elektron. datenverarbeitung, **4** (1970), 164-167.

[9] *R. E. Moore:* Interval Analysis. Prentice-Hall 1966.

[10] *R. S. Varga:* Matrix Iterative Analysis. Prentice-Hall 1962.

[11] *J. H. Wilkinson:* Rundungsfehler. Springer Verlag 1969.

[12] *P. Wißkirchen:* Ein Steuerungsprinzip in der Intervallrechnung und dessen Anwendung auf den Gauß'schen Algorithmus. Bericht der Gesellsch. f. Math. u. Datenverarbeitung Bonn Nr. **20** (1969).

[13] *G. Zielke:* Numerische Berechnung von benachbarten Matrizen und linearen Gleichungssystemen. Schriften zur Datenverarbeitung, Band 2, Friedr. Vieweg + Sohn, Braunschweig 1970.