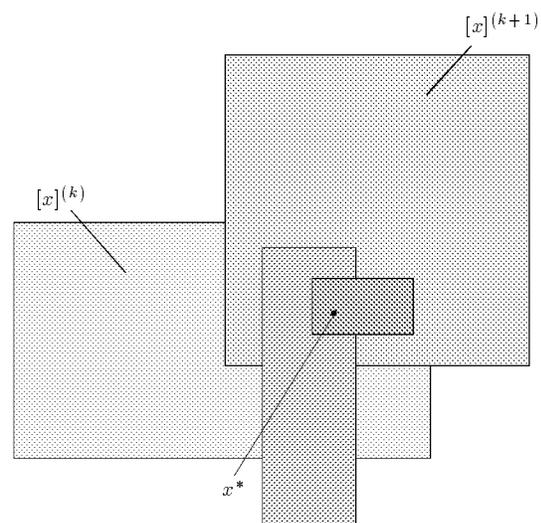


**Praktische Realisierung und Test eines
Verifikationsverfahrens zur Lösung globaler
Optimierungsprobleme mit
Ungleichungsnebenbedingungen**

Astrid Goos, Dietmar Ratz

Forschungsschwerpunkt
Computerarithmetik,
Intervallrechnung und
Numerische Algorithmen mit
Ergebnisverifikation



Impressum

Herausgeber:	Institut für Angewandte Mathematik Lehrstuhl Prof. Dr. Ulrich Kulisch Universität Karlsruhe (TH) D-76128 Karlsruhe
--------------	---

Redaktion:	Dr. Dietmar Ratz
------------	------------------

Internet-Zugriff

Die Berichte sind in elektronischer Form erhältlich über

`ftp://iamk4515.mathematik.uni-karlsruhe.de`
im Verzeichnis: `/pub/documents/reports`

oder über die World Wide Web Seiten des Instituts

`http://www.uni-karlsruhe.de/~iam`

Autoren-Kontaktadresse

Rückfragen zum Inhalt dieses Berichts bitte an

Astrid Goos, Dietmar Ratz
Institut für Angewandte Mathematik
Universität Karlsruhe (TH)
D-76128 Karlsruhe

E-Mail: `Dietmar.Ratz@math.uni-karlsruhe.de`

Praktische Realisierung und Test eines Verifikationsverfahrens zur Lösung globaler Optimierungsprobleme mit Ungleichungsnebenbedingungen

Astrid Goos, Dietmar Ratz

Inhaltsverzeichnis

1	Einleitung, Problemstellung, Motivation	4
2	Grundlegendes Verfahren	6
3	Zulässige Bereiche und Punkte	9
4	Boxreduktion und -elimination mit \bar{f}	10
5	Behandlung der Nebenbedingungen	13
6	Boxreduktion und -elimination mit Hilfe der John-Bedingungen	17
6.1	Intervall-Newton-Schritt	17
6.2	John-Bedingungen	19
6.3	Lokaler Existenznachweis	22
7	Heuristiken für Listenordnung, Splittingtechniken und Abbruchkriterium	23
7.1	Listenordnung	23
7.2	Splittingtechniken	24
7.3	Abbruchbedingung	26
8	Der vollständige Algorithmus	27
8.1	Der Algorithmus MAIN	27
8.2	Der Algorithmus ConstrainedGlobalOptimize	29
8.3	Abschließende Bemerkungen zum Verfahren	30
9	Numerische Tests	32
	Literatur	59

Zusammenfassung

Globale Optimierung mit Ungleichungsnebenbedingungen: Das hier behandelte Verfahren dient der verifizierten Lösung globaler Optimierungsprobleme mit Ungleichungsnebenbedingungen. Ziel hierbei ist es, eine Funktion mehrerer Variablen zu minimieren, deren zulässiger Bereich durch Ungleichungsnebenbedingungen gekennzeichnet ist. Das Verfahren berechnet garantierte Schranken für das globale Minimum und die globalen Minimalstellen. Außerdem kann das Verfahren, das zur Klasse der Branch-and-Bound-Methoden gehört, die lokale Existenz einer Minimalstelle innerhalb einer berechneten Einschließung automatisch nachweisen.

Es wird zunächst ein Überblick über das Verfahren und seine praktische Realisierung gegeben. Anschließend dokumentiert eine umfangreiche Reihe von Testergebnissen die Einsatzfähigkeit des Verfahrens.

Abstract

Global optimization with inequality constraints: Our method leads to the verified solution of global optimization problems with inequality constraints. To this end a function of several variables is minimized. The feasible area is characterized by inequalities. The method computes guaranteed bounds for the global minimum and the global minimizers. The method belongs to the class of interval-branch-and-bound-methods. The local existence of a minimizer within the computed enclosure can also automatically be proved.

First we give an overview about the method and its implementation. The applicability of the method is shown by a large number of elaborate test results.

1 Einleitung, Problemstellung, Motivation

Seien $f : D \rightarrow \mathbb{R}$, $p_i : D \rightarrow \mathbb{R}$, $i = 1, \dots, m$ zweimal stetig differenzierbare Funktionen, $[x] \subseteq D \subseteq \mathbb{R}^n$ und

$[x] \in I\mathbb{R}^n$ ein Intervallvektor bzw. eine n -dimensionale Box. Es soll das spezielle globale Optimierungsproblem

$$\begin{aligned} & \min_{x \in [x]} f(x) & (1) \\ & \text{unter den Bedingungen } p_i(x) \leq 0, & i = 1, \dots, m, \end{aligned}$$

gelöst werden. Es wird ein Verifikationsverfahren vorgestellt, mit Hilfe dessen garantierte Schranken für die globalen Minimalstellen und das globale Minimum des Optimierungsproblems berechnet werden können.

Durch die Nebenbedingungen sind die möglichen Lösungen auf einen Teilbereich, den zulässigen Bereich, eingeschränkt. Die Zielfunktion kann in diesem Bereich mehrere lokale Minima haben. Das Ziel der globalen Optimierung ist nun, unter diesen lokalen Minima die Stelle zu finden, an der die Zielfunktion den kleinsten Wert annimmt.

Aufgrund der Rechnerarithmetik treten bei der Lösung dieses Problems gewöhnlich Rundungsfehler zusätzlich zu den evtl. fehlerbehafteten Eingangsdaten eines praktischen Problems auf. Die Genauigkeit bzw. Zuverlässigkeit der berechneten Ergebnisse läßt sich mit Hilfe der in einem Intervallverfahren eingesetzten Intervallarithmetik überprüfen. Diese ermöglicht eine Vorwärtsfehleranalyse für viele Probleme, d.h. die

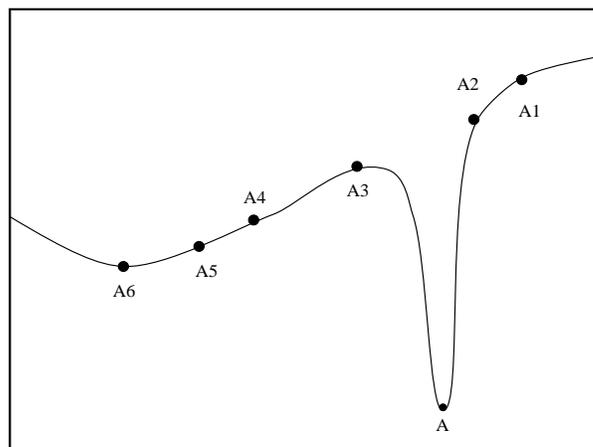


Abbildung 1: Funktion, deren Minimum in einem „Tal“ liegt.

Angabe von expliziten Fehlerschranken der berechneten Lösung unter Berücksichtigung aller möglichen Fehlerquellen und eine automatische Kontrolle dieser Fehler. Zusätzlich kann der Beweis der Existenz und Eindeutigkeit von Lösungen erbracht werden. Man spricht daher in diesem Zusammenhang von Methoden mit automatischer Ergebnisverifikation, Einschließungsmethoden oder selbstvalidierenden numerischen Verfahren.

Wichtigster Vorteil der Intervallrechnung, die somit geradezu prädestiniert für die Behandlung globaler Optimierungsprobleme ist, ist die Möglichkeit globale Aussagen über Teilbereiche des Optimierungsgebiets zu machen.

Klassische Verfahren für globale Problemstellungen starten gewöhnlich einen Iterationsprozeß, der von einzelnen Approximationen ausgeht. Funktionen werden somit nur an endlich vielen Punkten ausgewertet, im Gegensatz zu Funktionen in Intervallverfahren, deren Argumente Intervalle sind. Diese repräsentieren das Kontinuum zwischen ihren Grenzen und ermöglichen damit garantierte Aussagen über die Lage aller Funktionswerte in diesem Intervall. Über die bekannte Rundungsfehlerproblematik hinaus gibt es bei klassischen Verfahren keine Garantie, daß kein Minimum übersprungen wurde. Abbildung 1 verdeutlicht ein solches Problem. Unter der Voraussetzung, daß die Punkte A, A_1, \dots, A_6 zulässig sind, liegt das globale Minimum hier in einem „schmalen, tiefen Tal“. Durch die Iterationsfolge A_1, A_2, \dots eines Iterationsverfahrens wurde die globale Minimalstelle A übersprungen und das lokale Minimum in A_6 als globale Lösung angegeben. Durch wenige Intervallauswertungen aber könnte dieser approximierte Wert in Frage gestellt werden. Eine Auswertung über dem Intervall, das dieses „Tal“ enthält, würde anzeigen, daß es noch kleinere Funktionswerte als den in A_6 geben kann.

Für das hier behandelte Optimierungsverfahren muß für das Optimierungsproblem (1) vorausgesetzt werden, daß sowohl die Zielfunktion f als auch die Nebenbedingungen $p_i, i = 1, \dots, m$, zweimal stetig differenzierbar sind. Die Lösung wird in einer Startbox $[x]$ gesucht. Wenn eine oder mehrere Nebenbedingungen des Problems (1) die Form $x_i \geq a_i$ oder $x_i \leq b_i$ mit $a_i, b_i \in \mathbb{R}$ und $i = 1, \dots, n$ haben, werden diese verwendet, um die Grenzen der Startbox $[x]$ festzulegen. Wenn nicht alle $2n$ Seiten der Startbox durch solche Nebenbedingungen spezifiziert sind, muß der Benutzer des Verfahrens die übrige

gen Seiten festlegen. Diese speziellen $2n$ Nebenbedingungen, auch Randbedingungen genannt, werden im folgenden mit p'_i bezeichnet, um die anderen Nebenbedingungen gesondert betrachten zu können. Durch diese Festlegung der Ränder wird natürlich das Gebiet, auf dem die globale Lösung gesucht wird, eingeschränkt, und die ermittelten Einschließungen enthalten Kandidaten für die globalen Minimalstellen in diesem Gebiet. Das berechnete Minimum muß nicht unbedingt auch ein lokales Minimum sein, da es auch auf dem Rand des zulässigen Bereichs angenommen werden kann.

Um garantierte Schranken für die globalen Minimalstellen zu erhalten, ist eine sichere Aussage über die Zulässigkeit von Punkten notwendig. Dies ist mit der üblichen approximativen Auswertung einer Funktion auf dem Rechner nicht möglich, aber mit der Intervallauswertung der entsprechenden Intervallfunktion. Sei $p_i(x) \leq 0$, $i = 1, \dots, m$, eine Menge von linearen oder nichtlinearen Ungleichungen mit einem reellen Vektor x der Dimension n . In der Optimierungstheorie heißt ein Punkt $x \in \mathbb{R}^n$ zulässig, wenn $p_i(x) \leq 0$ für alle $i = 1, \dots, m$ gilt, ansonsten ist x unzulässig. Wenn in der Praxis bei der Auswertung von $p_i(x)$ Rundungsfehler gemacht wurden, kann nicht mit Sicherheit bestimmt werden, ob ein Punkt wirklich zulässig ist oder nicht. Wenn die Funktion p_i eine Intervallfunktion ist, erfordert dies eine Erweiterung des Begriffs *zulässig*.

Definition 1.1 : Seien P_i die Einschließungsfunktionen von p_i ($i = 1, \dots, m$). Wenn man $P_i(x)$ mit Rundung nach außen auswertet, erhält man das Intervall $[\underline{P}_i(x), \overline{P}_i(x)]$. Ein Punkt x heißt sicher zulässig bzw. sicher unzulässig, wenn gilt

$$\overline{P}_i(x) \leq 0, \quad \forall i = 1, \dots, m$$

bzw.

$$\underline{P}_i(x) > 0, \quad \text{für mind. ein } i = 1, \dots, m.$$

Analog gilt dies auch für Intervallargumente $[x] \in I\mathbb{R}^n$ anstelle der reellen Vektoren x . Für ein Intervall, das weder sicher zulässig noch sicher unzulässig ist, kann nicht sofort entschieden werden, ob es zulässige Punkte enthalten kann oder nicht, hier müssen also komplexere Verfahren zum Einsatz kommen. Im folgenden wird nun die Realisierung eines Intervall-Branch-and-Bound-Verfahrens beschrieben, dessen Grundlage auf ein Verfahren von Hansen [5] zurückgeht. Eine ausführliche Beschreibung des Verfahrens findet sich in [3].

Im weiteren wird die Menge der reellen Intervalle mit $I\mathbb{R}$ und die Menge der reellen n -dimensionalen Intervallvektoren (oder auch Boxen) mit $I\mathbb{R}^n$ bezeichnet. Die Intervallerweiterung einer Funktion $f : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ wird mit dem zugehörigen Großbuchstaben F bezeichnet, das globale Minimum von f mit f^* und die Menge der globalen Minimalstellen mit X^* .

2 Grundlegendes Verfahren

Das Ziel, das durch den Einsatz von Intervallverfahren erreicht werden soll, ist die Berechnung garantierter Einschließungen für das globale Minimum f^* und die globalen Minimalstellen x^* von (1) innerhalb der Startbox.

Algorithmen zur Lösung dieser Aufgabenstellung arbeiten meist nach dem Intervall-Branch-and-Bound-Prinzip:

1. Zerlege den Startbereich $[x]$ in immer kleinere Teilbereiche $[y] \subseteq [x]$.
2. Bestimme garantierte Schranken für f auf dem Teilbereich $[y]$ (unter Beachtung der Nebenbedingungen).
3. Eliminiere die Teilbereiche, die aufgrund dieser Schranken keine globalen Minimalstellen enthalten können.

Neben der eingesetzten Intervallarithmetik muß ein solches Verfahren mit einer möglichst optimalen Aufteilungsstrategie und Listenverwaltung für die Teilbereiche arbeiten. Für die verwendeten Listen L und die Listenelemente E seien die folgenden Operationen definiert

- $L := \{\}$, Initialisierung mit der leeren Liste;
- $L := E$, Initialisierung mit nur einem Element;
- $L := L + E$, Einhängen eines Elements E in die Liste L ;
- $L := L - E$, Aushängen eines Elements E aus L ;
- $E := \text{Head}(L)$ Zuweisung des ersten Elements von L an E .

Als Elemente der Liste werden Paare $E = ([y], [r])$ verwendet, die aus einem Intervallvektor $[y] \in I\mathbb{R}^n$ und einem Intervall $[r] \in I\mathbb{R}$ bestehen. Dabei wird im folgenden stets

$$[r] = F([y])$$

verwendet.

Nun soll die grundlegende Vorgehensweise des Verfahrens verdeutlicht werden. Es wird mit einer Liste L_{start} gearbeitet, in der als Elemente jeweils die noch zur Bearbeitung anstehenden Teilbereiche $[y]$ des Optimierungsbereichs $[x]$ zusammen mit dem Intervall $F_{[y]} = F([y])$ abgespeichert werden.

Startphase:

- Die Liste L_{start} wird mit dem Startbereich $[x]$ initialisiert:

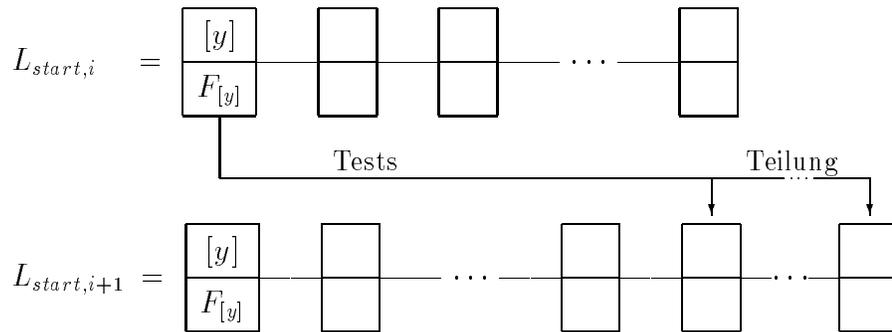
$$L_{start,0} := \frac{\boxed{[x]}}{\boxed{F_{[x]}}}, \text{ mit } F_{[x]} = F([x]).$$

- Bestimmung einer garantierten Oberschranke \bar{f} für f^* an einer zulässigen Stelle $\bar{x} \in [x]$.

Iteration:

- In der $(i + 1)$ -ten Iteration wird der erste Bereich $[y]$ aus der Liste $L_{start,i}$ (aus dem i -ten Iterationsschritt) entfernt. Falls durch entsprechende Tests die Existenz einer globalen Minimalstelle in diesem Teilbereich nicht ausgeschlossen werden kann, so wird die Box unter bestimmten Bedingungen in maximal acht Teilboxen aufgeteilt.

- Für die neu entstandenen Elemente $[y_i]$ wird zunächst geprüft, ob $F_{[y_i]} \leq \bar{f}$ gilt. Die Elemente, die dieser Bedingung genügen, werden in die Liste $L_{start,i+1}$ aufgenommen, nachdem versucht wurde, mit ihnen eine neue, verbesserte garantierte Oberschranke \bar{f} zu erhalten.



Terminierung:

- Durch eine Genauigkeitsforderung (z. B. in Form einer Bedingung für die Durchmesser der Intervalle $[y]$ und $F([y])$) kann der Algorithmus abgebrochen werden.

Ergebnis: Resultatsliste L_{res} und Oberschranke \bar{f} .

Satz 2.1 Nach Durchführung des grundlegenden Verfahrens gilt, wenn ein sicher zulässiger Punkt gefunden wurde

- $X^* \subseteq \bigcup_{[y] \in L_{res}} [y]$,
- $f^* \in [\min\{\underline{f}_y \in L_{res}\}, \bar{f}]$.

Wenn kein sicher zulässiger Punkt gefunden wurde, gilt, falls die Existenz eines solchen nicht sicher ausgeschlossen werden kann¹

- $X^* \subseteq \bigcup_{[y] \in L_{res}} [y]$,
- $f^* \in [\min\{\underline{f}_y \in L_{res}\}, \max\{\bar{f}_y \in L_{res}\}]$.

Ist die Liste L_{res} nach Durchführung des grundlegenden Verfahrens leer, so existiert nachweislich kein zulässiger Punkt im Minimierungsbereich.

Die in den folgenden Abschnitten beschriebenen Verfahren stellen Hilfsmittel zur Beschleunigung eines solchen Intervall-Branch-and-Bound-Verfahrens bereit, wozu u.a.

¹Aufgrund von Überschätzungen bei der Auswertung der Nebenbedingungen kann eine Box weder als sicher zulässig noch sicher unzulässig angesehen werden. Hier sind „genauere“ Auswertungen notwendig, um dies zu entscheiden. Boxen, für die das gilt und die auch nicht durch einen anderen Test eliminiert werden konnten, können in diesem Fall vom Algorithmus nicht verworfen werden, auch wenn nicht sicher ist, ob sie überhaupt einen zulässigen Punkt enthalten.

1. die Suche nach zulässigen Punkten, um die Oberschranke \bar{f} festzulegen,
2. die Tests für zulässige Bereiche (Monotonie-, Konkavitätstest),
3. die Reduktion oder Elimination mit Hilfe von \bar{f} ,
4. die Einschränkung des Suchbereichs auf den zulässigen Bereich und
5. der Intervall-Newton-Schritt für die Johnbedingungen

zählen. Die einzelnen Testverfahren werden im Abschnitt 8 in dieses grundlegende Verfahren eingesetzt.

Bemerkung: Im folgenden wird die garantierte Oberschranke für f^* immer mit \bar{f} und der zulässige Punkt x , für den $\bar{f} = f(x)$ gilt, mit \bar{x} bezeichnet. Ist noch kein zulässiger Punkt gefunden worden, so gilt $\bar{f} = \infty$.

3 Zulässige Bereiche und Punkte

In einer sicher zulässigen Teilbox kann das Optimierungsproblem (1) wie ein Optimierungsproblem ohne Nebenbedingungen behandelt werden. Sei also $[y]$ eine sicher zulässige Teilbox der Startbox $[x]$. Um festzustellen, ob $[y]$ eine Minimalstelle enthält, kann man zwei Testverfahren (den Monotonie- und den Konkavitätstest) anwenden und die Box entsprechend reduzieren oder eliminieren (für eine nähere Beschreibung dieser Tests vgl. [3] und [13]).

Aus einer Box können alle Punkte x eliminiert werden, für die $f(x) > \bar{f}$ und damit auch $f(x) > f^*$ gilt. Um einen ersten Wert für \bar{f} zu bestimmen, muß also möglichst schnell ein sicher zulässiger Punkt mit möglichst kleinem Funktionswert gefunden werden. Dazu wird zunächst für alle Elemente in der Startliste geprüft, ob ihr Mittelpunkt sicher zulässig ist. Wenn dies der Fall ist, wird das Supremum der Intervallfunktionsauswertung an diesem Punkt mit \bar{f} verglichen und, wenn möglich, \bar{f} damit verbessert. Für \bar{f} gilt damit

$$\bar{f} \geq f^*$$

Dies wird durch den Algorithmus **update** realisiert, der für alle Elemente der Liste durchgeführt werden muß.

Dies ist nur eine „grobe“ Suche nach einem zulässigen Punkt. Eine umfassendere Suche nach dem „besten“ zulässigen Punkt wird mit **LineSearch** durchgeführt. Dieses Verfahren sucht auf einer Geraden ausgehend vom Mittelpunkt einer Box einen Punkt, der sicher zulässig ist und eine bessere Oberschranke liefert. Für eine genauere Beschreibung dieses Verfahrens siehe [3].

Die beiden Teilalgorithmen **update** und **LineSearch** werden zu dem Algorithmus **Reduce \bar{f}** zusammengefaßt. Dieser versucht mit allen Elementen der Liste L , \bar{f} zu verbessern und führt für alle Elemente der Liste, für die es möglich ist, **LineSearch** durch.

Algorithmus 3.1: Reduce $\bar{f}(L, \bar{f}, \bar{x})$

1. **while** $L \neq \{\}$ **do**
 - (a) $([z], F_{[z]}) := \text{Head}(L); L := L - ([z], F_{[z]});$

(b) $m := m([z]); F_m := F(m);$

(c) **update**(m, F_m, \bar{x}, \bar{f})

2. **while** $L \neq \{\}$ **do**

(a) $([z], F_{[z]}) := \text{Head}(L); L := L - ([z], F_{[z]});$

(b) $m := m([z]); F_m := F(m);$

(c) **if** $\bar{f} < \infty$ **and** $\overline{F_m} < \bar{f}$ **then**

$\left\{ \begin{array}{l} \text{Voraussetzungen für die Durch-} \\ \text{führung von LineSearch} \end{array} \right\}$

$\text{LineSearch}([z], m, \bar{x}, \bar{f});$

3. **return** $\bar{f}, \bar{x};$

4 Boxreduktion und -elimination mit \bar{f}

Für die kleinste obere Schranke \bar{f} für das globale Minimum f^* gilt sicher: $\bar{f} \geq f^*$. Deshalb kann man \bar{f} verwenden, um eine Box zu eliminieren oder zu reduzieren. Es werden nun zwei verschiedene Möglichkeiten vorgestellt, mit denen eine Box oder eine Teilbox mit Hilfe von \bar{f} aus der Startliste gelöscht werden kann. Die einfachste Möglichkeit, ganze Boxen aus der Liste zu entfernen, wird mit dem **CutOffTest** (oder auch **Mittelpunktstest**) bereitgestellt. Alle Boxen $[x]$ in der Liste, für die gilt: $\underline{F}([x]) > \bar{f}$, werden gelöscht. Wenn dies für eine Box $[x]$ gilt, dann gilt mit Sicherheit auch

$$f(x) > f^*, \forall x \in [x].$$

Mit diesem Test kann man entweder nur die ganze Box löschen oder aber keinen einzigen Punkt von ihr. Deshalb wird dieser Test noch verfeinert.

Für eine Verbesserung dieses Tests, muß nun $f \in C^2([x])$ gefordert werden, um mit der Ungleichung $\underline{F}(x) > \bar{f}$ Punkte aus $[x]$ zu löschen. Für $x, y \in [x]$ gilt (vgl. [3]):

$$f(y) \in f(x) + (y - x)^T g(x) + \frac{(y - x)^T [H(x, [x])](y - x)}{2},$$

wobei die Elemente $[h(x, [x])]_{ij}$ der Hessematrix $[H(x, [x])]$ mit

$$h_{ij} = \begin{cases} \partial^2 f / \partial x_i^2 & \text{für } j = i (i = 1, \dots, n), \\ 2\partial^2 f / \partial x_i \partial x_j & \text{für } j < i (i = 2, \dots, n; j = 1, \dots, i - 1), \\ 0 & \text{sonst.} \end{cases}$$

von f die Argumente $([x]_1, \dots, [x]_j, x_{j+1}, \dots, x_n)$ haben. Hierbei ist zu beachten, daß $[H(x, [x])]$ (oder auch kurz $[H]$) hier eine untere Dreiecksmatrix ist. Meist wählt man $x = m([x])$.

Sei $z := y - x$. Aus der Box $[x]$ können die Punkte eliminiert werden, für die gilt

$$f(x) + z^T g(x) + \frac{z^T [H] z}{2} > \bar{f}.$$

Hierzu bestimmt man die Komplementärmenge der Lösung, d.h. es werden die Punkte bestimmt, für die gilt

$$f(x) + z^T g(x) + \frac{z^T [H] z}{2} \leq \bar{f}. \quad (2)$$

Diese quadratische Ungleichung soll nun für z_1, \dots, z_n gelöst werden.

Um (2) für z_i zu lösen, wird $z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_n$ durch die Einschließungen $[x]_1 - x_1, \dots, [x]_{i-1} - x_{i-1}, [x]_{i+1} - x_{i+1}, \dots, [x]_n - x_n$ ersetzt, d.h. (2) wird umgeformt zu

$$[a] + [b]z_i + [c]z_i^2 \leq 0$$

mit

$$\begin{aligned} [a] &:= f(x) + \sum_{\substack{j=1 \\ j \neq i}}^n z_j g_j(x) + \frac{1}{2} \left(\sum_{\substack{k=1 \\ k \neq i}}^n [z]_k \sum_{\substack{j=1 \\ j \neq i}}^k [h]_{kj} [z]_j \right) - \bar{f}, \\ [b] &:= g_i(x) + \frac{1}{2} \left(\sum_{j=1}^{i-1} [h]_{ij} [z]_j + \sum_{j=i+1}^n [h]_{ij} [z]_j \right), \\ [c] &:= \frac{[h]_{ii}}{2}. \end{aligned} \quad (3)$$

Die Lösung dieser Ungleichung ist durch eine Menge T (vgl. Abschnitt 1.8 in [3]) bestimmt, die mit erweiterter Intervallarithmetik berechnet wird und folgende Form hat. Entweder T ist

- ein Intervall oder
- die Vereinigung von zwei Intervallen oder
- leer.

Wenn diese Menge oder der Schnitt $[x]_i \cap (T + x_i)$ leer ist, kann $[x]$ aus der Liste gelöscht werden, ansonsten ersetzt man die Komponente $[x]_i$ durch $[x]_i \cap (T + x_i)$ und bestimmt die nächste Komponente $[x]_{i+1}$. Wenn die Menge T und damit der Schnitt $[x]_i \cap (T + x_i)$ die Vereinigung zweier Mengen $[w]^1, [w]^2$ ist, dann ist die neue Komponente $[x]_i$ folgendermaßen zu bestimmen

$$[x]_i := [\underline{w}^1, \overline{w}^2].$$

Die entstandene Lücke $(\overline{w}^1, \underline{w}^2)$, die aus der Komponente $[x]_i$ gelöscht werden kann, wird in einem getrennten Vektor gehalten und für eine spätere Verwendung gespeichert.

Der Algorithmus 4.1 beschreibt unter Verwendung der erweiterten Intervallarithmetik diese Reduktion einer Box $[x]$. Dazu muß ihm die kleinste bisher gefundene obere Schranke \bar{f} für das globale Minimum f^* und der Intervallvektor $[x]$ übergeben werden. Als Ergebnis liefert er die erneuerte Box $[x]$, den Intervallvektor $[gap]$, in dem die evtl. in $[x]$ entstandenen Lücken gespeichert sind, und den Wahrheitswert *IsEmpty*, der anzeigt, ob $[x]$ aus der Startliste gelöscht werden kann.

Algorithmus 4.1: QuadraticMethod($[x]$, \bar{f} , $[gap]$, $IsEmpty$)

1. $x := m([x]); IsEmpty := false;$
2. $[H] := [H(x, [x]);$ { Berechnung der speziellen Hesse-
matrix. }
3. $[z] := [x] - x; i := 1;$
4. **while not** $IsEmpty$ **and** $i \leq n$ **do**
 if $\underline{h_{ii}} \geq 0$ **then**
 (a) $[a_1, a_2] := f(x) + \sum_{\substack{j=1 \\ j \neq i}}^n z_j g_j(x) + \frac{1}{2} \left(\sum_{\substack{k=1 \\ k \neq i}}^n [z]_k \sum_{\substack{j=1 \\ j \neq i}}^k [h]_{kj} [z]_j \right) - \bar{f};$
 $[b_1, b_2] := g_i(x) + \frac{1}{2} \left(\sum_{j=1}^{i-1} [h]_{ij} [z]_j + \sum_{j=i+1}^n [h]_{ij} [z]_j \right);$
 $[c_1, c_2] := \frac{[h]_{ii}}{2};$
 (b) $[D_1] := [b_1, b_1]^2 - 4[a_1, a_1] \cdot [c_1, c_1]; [D_2] := [b_2, b_2]^2 - 4[a_1, a_1] \cdot [c_1, c_1];$
 (c) $ComputeSolutionSet(a_1, [b_1, b_2], c_1, [D_1], [D_2], [t]);$
 (d) $[w]^1 \cup [w]^2 := [t] \cap [z]_i;$
 if $[w]^1 \neq \emptyset$ **then**
 if $[w]^2 = \emptyset$ **then** $[z]_i := [w]^1; [gap]_i := \emptyset$
 else $[z]_i := [\underline{w}^1, \overline{w}^2], [gap]_i := [\underline{w}^1, \overline{w}^2] + x$
 else $[z]_i = \emptyset; IsEmpty := true; \mathbf{exit}_{i-loop};$
 (e) $i := i + 1;$
5. $i := 1;$
6. **while not** $IsEmpty$ **and** $i \leq n$ **do**
7. **if** $\underline{h_{ii}} < 0$ **then**

- (a) $[a_1, a_2] := f(x) + \sum_{\substack{j=1 \\ j \neq i}}^n z_j g_j(x) + \frac{1}{2} \left(\sum_{\substack{k=1 \\ k \neq i}}^n [z]_k \sum_{\substack{j=1 \\ j \neq i}}^k [h]_{kj} [z]_j \right) - \bar{f};$
 $[b_1, b_2] := g_i(x) + \frac{1}{2} \left(\sum_{j=1}^{i-1} [h]_{ij} [z]_j + \sum_{j=i+1}^n [h]_{ij} [z]_j \right);$
 $[c_1, c_2] := \frac{[h]_{ii}}{2};$
- (b) $[D_1] := [b_1, b_1]^2 - 4[a_1, a_1] \cdot [c_1, c_1]; [D_2] := [b_2, b_2]^2 - 4[a_1, a_1] \cdot [c_1, c_1];$
- (c) $ComputeSolutionSet(a_1, [b_1, b_2], c_1, [D_1], [D_2], [t]);$
- (d) $[w]^1 \cup [w]^2 := [t] \cap [z]_i;$
 if $[w]^1 \neq \emptyset$ **then**
 if $[w]^2 = \emptyset$ **then** $[z]_i := [w]^1; [gap]_i := \emptyset$
 else $[z]_i := [\underline{w}^1, \overline{w}^2], [gap]_i := [\underline{w}^1, \overline{w}^2] + x$

- else** $[z]_i = \emptyset$; $IsEmpty := true$; **exit** _{i -loop};
- (e) $i := i + 1$;
8. $[x] := [z] + x$;
9. **return** $[x]$, $[gap]$, $IsEmpty$;

In Schritt 4 des Algorithmus 4.1 werden nur die Komponenten $[z]_i$ behandelt, für die $h_{ii} \geq 0$ gilt. Die anderen werden erst in Schritt 6 behandelt, damit zunächst die Komponenten bearbeitet werden, die keine Lücke enthalten. So haben die neuen Komponenten genauere Grenzen, da entstandene Lücken hier zwar gespeichert werden, aber in der Berechnung der neuen Komponente $[z]_i$ ignoriert werden. `ComputeSolutionSet` liefert als Ergebnis eine Einschließung $[t]$ der Lösungsmenge T für die behandelte Ungleichung, für eine genauere Beschreibung dieser Berechnung vgl. Algorithmus 2.10 in [3].

Mit dem folgenden Satz wird gezeigt, daß durch die Ausführung des Algorithmus 4.1 keine Teilbox einer Box $[x]$ gelöscht wird, die eine globale Minimalstelle enthalten kann.

Satz 4.1 : *Sei $f : \mathbb{R}^n \rightarrow \mathbb{R}$ zweimal stetig differenzierbar auf $[x]$. Für jede Box $[y] \subseteq [x]$ für die $x^* \in [y]$ für eine globale Minimalstelle $x^* \in [x]$ des Optimierungsproblems (1) gilt, gilt auch nach Ausführung des Algorithmus `QuadraticMethod` die Beziehung $x^* \in [y]$.*

Beweis: Der Beweis ergibt sich direkt aus der Konstruktion der Lösungsmenge T einer quadratischen Intervallungleichung (vgl. Abschnitt 1.8 in [3]) und den vorhergehenden Überlegungen. □

5 Behandlung der Nebenbedingungen

Das Ziel eines approximativen Verfahrens ist, einen einzelnen zulässigen Punkt zu finden. Beim Einsatz von Intervallverfahren versucht man dagegen, sicher unzulässige Teilboxen zu eliminieren. Die zulässigen Punkte bleiben erhalten, auch wenn Rundungsfehler auftreten. Es wird also die größte Teilbox $[x']$ der aktuellen Box $[x]$ gesucht, so daß jeder Punkt von $[x] \setminus [x']$ sicher zulässig ist. Mit Hilfe der nichtlinearen Intervallungleichung $P_i([x]) \leq 0$ sollen nun sicher unzulässige Punkte aus der Box $[x]$ entfernt werden. Dazu wird neben der gewöhnlichen Intervallauswertung die Taylorentwicklung einer Funktion verwendet. Man entwickelt p_i um x und erhält

$$p(y) \in p(x) + (y - x)^T [G_{p_i}(x, [x])], \quad (4)$$

wobei G_{p_i} ein spezieller Intervallgradient von p_i ist. Die j -te Komponente des Gradienten hat das Argument $([x]_1, \dots, [x]_j, x_{j+1}, \dots, x_n)$, wobei $x = m([x])$ ist. Eine genaue Beschreibung dieser Form der Taylorentwicklung findet man in [3] und [5]. Gleichung (4) gilt für alle $y \in [x]$. Ein Punkt y ist dann sicher unzulässig, wenn gilt:

$$\inf(p_i(x) + (y - x)^T [G_{p_i}(x, [x])]) > 0. \quad (5)$$

Zunächst soll kurz erläutert werden, welche der Nebenbedingungen dafür geeignet sind, unzulässige Punkte zu löschen. Eine Nebenbedingung, die für alle $x \in [x]$ erfüllt ist, kann verständlicherweise nichts dazu beitragen.

Auswahltest für die Nebenbedingungen

Gegeben seien m Ungleichungsbedingungen $p_i(x) \leq 0$, $i = 1, \dots, m$.

$$s_i := \begin{cases} 0 & \text{falls } \underline{P_i([x])} = \overline{P_i([x])}, \\ \frac{\overline{P_i([x])}}{\overline{P_i([x])} - \underline{P_i([x])}} & \text{sonst,} \end{cases}$$

für $i = 1, \dots, m$. Für s_i gilt damit

$$s_i = 0 \iff \underline{P_i([x])} = 0 \vee \underline{P_i([x])} = \overline{P_i([x])},$$

d.h die Auswertung von $P_i([x])$ ist ein Punktintervall, und p_i ist keine Hilfe, um sicher unzulässige Punkte zu eliminieren. Gilt einerseits

$$s_i = 1 \iff \underline{P_i([x])} = 0,$$

so kann $[x]$ keine inneren Punkte des zulässigen Bereichs enthalten. Gilt andererseits

$$s_i < 0 \iff \overline{P_i([x])} \leq 0,$$

dann enthält $[x]$ nur zulässige Punkte. Interessant sind also nur die Nebenbedingungen, für die gilt

$$0 < s_i \leq 1.$$

Es werden die Nebenbedingungen ausgewählt, für die gilt: $s_i > 0.25$. Der Wert 0.25 wurde aufgrund von Erfahrungen festgelegt. Ist dieser Wert kleiner, so werden Nebenbedingungen mit ausgewählt, die so gut wie nichts zum Eliminieren von unzulässigen Punkten beitragen.

Außerdem beschränkt man sich nicht nur auf die Nebenbedingungsungleichungen, die auf diese Weise ausgewählt wurden, sondern es kann noch die Ungleichung $f(x) - \overline{f} \leq 0$ hinzugefügt werden (falls $\overline{f} < \infty$ und $\frac{\overline{F([x])} - \overline{f}}{\overline{F([x])} - \underline{F([x])}} > 0.25$ gilt). Diese Auswahl wird durch den Algorithmus **Select** durchgeführt. Dieser ermittelt eine Indexmenge \mathcal{I} , welche die Indizes der ausgewählten Nebenbedingungen p_i enthält. Wenn die Ungleichung $f(x) - \overline{f} \leq 0$ ebenfalls mitausgewählt wurde, wird der Index $m + 1$ in die Menge \mathcal{I} aufgenommen.

Linearisieren der Nebenbedingungen

Seien p_1^*, \dots, p_s^* die mit dem Auswahltest **Select** gewählten Ungleichungen. Jede Ungleichung wird mit einer speziellen Taylorentwicklung (vgl. Abschnitt 1.9 in [3]) linearisiert.

$$p_k^*(y) \in p_k^*(m([x])) + \sum_{i=1}^n (y_i - m([x])_i) \nabla P_k^*([x]_1, \dots, [x]_i, m([x])_{i+1}, \dots, m([x])_n)_i,$$

$k = 1, \dots, s$ und, falls $f - \overline{f}$ auch ausgewählt wurde

$$f(y) - \overline{f} \in f(m([x])) - \overline{f} + \sum_{i=1}^n (y_i - m([x])_i) \nabla F([x]_1, \dots, [x]_i, m([x])_{i+1}, \dots, m([x])_n)_i$$

Die aus den ausgewählten Ungleichungen entstehende Koeffizientenmatrix $[A]$ ist eine Intervallmatrix und die rechte Seite b (theoretisch) ein Punktvektor, der jedoch später auch intervallarithmetic ausgewertet wird.

Lösen der Nebenbedingungen

Sei $[A]$ die entstandene Intervallkoeffizientenmatrix²

$$[A] := \begin{pmatrix} \nabla P_1^*([x]_1, x_2, \dots, x_n)_1 & \cdots & \nabla P_1^*([x]_1, \dots, [x]_n)_n \\ \nabla P_2^*([x]_1, x_2, \dots, x_n)_1 & \cdots & \nabla P_2^*([x]_1, \dots, [x]_n)_n \\ \vdots & & \vdots \\ \nabla P_s^*([x]_1, x_2, \dots, x_n)_1 & \cdots & \nabla P_s^*([x]_1, \dots, [x]_n)_n \\ \nabla F([x]_1, x_2, \dots, x_n)_1 & \cdots & \nabla F([x]_1, \dots, [x]_n)_n \end{pmatrix}$$

und

$$b := \begin{pmatrix} p_1^*(m([x])) \\ \vdots \\ p_s^*(m([x])) \\ (f - \bar{f})(m([x])) \end{pmatrix}$$

die rechte Seite. Das Ungleichungssystem

$$[A](y - m([x])) + b \leq 0 \quad (6)$$

wird nun in jeder Zeile für jedes $y_i, i = 1, \dots, n$ gelöst. D.h.

$$[a]_{k1}[x]_1 + \cdots + [a]_{ki}y_i + \cdots + [a]_{kn}[x]_n + b_k \leq 0$$

muß für alle $y_i, i = 1, \dots, n$ gelöst werden. Die unbekanntenen Komponenten von y werden jeweils durch die zugehörigen Intervallkomponenten von $[x]$ ersetzt, da $y_i \in [x]_i$ gilt. So erhält man die Menge T_i als Lösungsmenge der entstandenen linearen Intervallungleichung (vgl. Abschnitt 1.7 in [3]) und damit das Intervall

$$[z]_i := ([x]_i - m([x])_i) \cap T_i$$

für $y_i - m([x])_i$. $[z]_i$ ist auch dann endlich, wenn T_i es nicht ist. Wenn T_i die Vereinigung zweier Intervalle ist, dann erhält man zwei Lösungsintervalle

$$[z]_{i1} = ([x]_i - m([x])_i) \cap T_{i1} \text{ und } [z]_{i2} = ([x]_i - m([x])_i) \cap T_{i2}.$$

Die neue Komponente von $[x]$ ist dann durch das Intervall

$$[x]_i := [\underline{z}_{i1} + m([x])_i, \overline{z}_{i2} + m([x])_i]$$

gegeben. Das Intervall $[\overline{z}_{i1} + m([x])_i, \underline{z}_{i2} + m([x])_i]$ kann keine zulässigen Punkte enthalten und wird hier für eine spätere Verwendung gespeichert. Dieser Prozeß wird wiederholt für alle $i = 1, \dots, n$ und für alle Zeilen der Matrix $[A]$. Jedes $[x]_i$ wird gleich durch das neu berechnete $[x]_i$ ersetzt.

Der folgende Algorithmus **LinearizeSolve** löst das System der linearisierten Nebenbedingungen. Für die Matrix $[A]$ wird zunächst eine Vorkonditionierungsmatrix mit dem Algorithmus **PreCondMat** berechnet. **PreCondMat** führt eine Gaußelimination mit

²Es wird vorausgesetzt, daß die Ungleichung $f(x) - \bar{f} \leq 0$ auch ausgewählt wurde.

erstem und zweiten Pivotelement durch und liefert eine Matrix R , die mehr Zeilen als $[A]$ haben kann (vgl. Abschnitt 1.3 in [3]). In Algorithmus **LinearizeSolve** gehen der Intervallvektor $[x]$ und die Indexmenge \mathcal{I} der ausgewählten Nebenbedingungen ein. Die Ausgabe besteht aus dem verbesserten Vektor $[x]$, einer Intervallmatrix $[gap]$, welche die durch die Prozedur entstandenen Lücken speichert und dem Parameter $IsEmpty$, der anzeigt, ob $[x]$ aus der Startliste gelöscht werden kann.

Algorithmus 5.1: LinearizeSolve($[x]$, \mathcal{I} , $[gap]$, $IsEmpty$)

1. $x := m([x]);$
2. **for all** $i \in \mathcal{I}$ **do**
 - if** $i = m + 1$ **then** $[a]_{i,*} := \nabla F(x, [x]); b_i := f(x) - \bar{f};$
 - else** $[a]_{i,*} := \nabla P_i(x, [x]); b_i := p_i(x);$ { Initialisieren von $[A]$ und $b.$ }
3. **PreCondMat** ($m([A])$, R); { Berechnung der Vorkonditionierungsmatrix $R \in \mathbb{R}^{l \times n}$ für $[A].$ }
4. $[M] := R \cdot [A]; c := R \cdot b; [z] := [x] - x;$
5. **for** $i := 1$ **to** l **do** { $[M]$ hat jetzt l Zeilen. }

 - (a) $[u] := \sum_{j=1}^n [m]_{ij}[z]_j + c_i$
 - (b) **for** $k := 1$ **to** n **do**
 - i. $a := \underline{u} - \underline{[m]_{ik}[z]_k}; b := \bar{u} - \overline{[m]_{ik}[z]_k};$ { Spezielle Intervallsubtraktion. }
 - ii. $[c, d] := [m]_{ik};$
 - iii. **ComputeSolutionSet**($[a, b], [c, d], [t]$); { Berechnung der Einschließung $[t]$ der Lösungsmenge $T.$ }
 - iv. $([w]^1 \cup [w]^2) := [t] \cap [z]_k;$
 - v. **if** $[w]^1 \neq \emptyset$ **then**
 - if** $[w]^2 = \emptyset$ **then** $[z]_k := [w]^1; [gap]_{ik} := \emptyset$
 - else** $[z]_k := [\underline{w}^1, \underline{w}^2];$

 - $[gap]_{ik} := [\overline{w}^1, \overline{w}^2] + x;$ { In der k -ten Spalte von $[gap]$ stehen die Lücken der Komponente $[x]_k.$ }
 - else** $[z]_k = \emptyset; IsEmpty := true; \mathbf{exit}_{i-loop};$

6. $[x] := [z] + x;$
7. **return** $[x], [gap], IsEmpty;$

Auch hier kann ein Satz formuliert werden, der aussagt, daß durch die Anwendung des Algorithmus **LinearizeSolve** keine Boxen oder Teilboxen eliminiert werden, die eine globale Minimalstelle des Optimierungsproblems enthalten können. Der Beweis ergibt sich auch hier direkt aus der Konstruktion der Lösungsmenge T einer linearen Intervallungleichung, wie sie in [3], 1.7 beschrieben ist, und den vorhergehenden Überlegungen.

Satz 5.1 : Seien $f : \mathbb{R}^n \rightarrow \mathbb{R}$ und $p_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m$ aus $C^1([x])$. Für jede Box $[y] \subseteq [x]$ für die $x^* \in [y]$ für eine globale Minimalstelle $x^* \in [x]$ des Optimierungsproblems (1) gilt, gilt auch nach Ausführung des Algorithmus **LinearizeSolve** die Beziehung $x^* \in [y]$.

6 Boxreduktion und -elimination mit Hilfe der John-Bedingungen

In diesem Abschnitt wird mit den John-Bedingungen ein wesentliches Hilfsmittel zur Beschleunigung des Verfahrens vorgestellt. Mit ihnen kann man, ähnlich wie im Abschnitt 4 mit \bar{f} , einerseits bestimmte Boxen frühzeitig aus der Liste entfernen, wenn garantiert werden kann, daß diese keine globale Minimalstelle enthalten können, und andererseits ermöglichen sie eine Verkleinerung der jeweils aktuellen Box.

Zunächst wird ein modifiziertes Newton-Verfahren (nach Hansen [5]) eingeführt. Es wird angewendet auf die durch die John-Bedingungen gegebene Funktion f_{john} , auf die in Abschnitt 6.2 näher eingegangen wird.

6.1 Intervall-Newton-Schritt

Zum Intervall-Newton-Verfahren kommt man wie beim klassischen Newton-Verfahren, durch eine Linearisierung der Funktion $g : [x] \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^n$. Mit $c, x^* \in [x]$, $J \in \mathbb{R}^{n \times n}$ und x^* Nullstelle von g gilt

$$0 = g(x^*) = g(c) + J \cdot (x^* - c).$$

Ausgehend von einer Einschließung $[y]$ der Nullstelle x^* und $[J]$ von J wird in jedem Intervall-Newton-Schritt das entsprechende Intervallgleichungssystem

$$[J] \cdot (c - [y]^N) = g(c) \tag{7}$$

gelöst. Dabei ist $[J]$ die Jacobimatrix der Funktion G , ausgewertet über dem Intervallvektor $[y]$, und $c \in [y]$ beliebig. Die Einschließung $[y]$ kann dann durch $[y] := [y]^N \cap [y]$ verbessert werden.

Bevor das Verfahren hier näher spezifiziert wird, muß geklärt werden, was mit der „Lösung“ eines Intervallgleichungssystems gemeint ist. Dazu dient die folgende Definition.

Definition 6.1 Sei $[A] \in I\mathbb{R}^{n \times n}$ eine Intervallmatrix und $[b] \in I\mathbb{R}^n$ ein Intervallvektor. Die Lösung des Intervallgleichungssystems

$$[A]x = [b] \quad \text{oder auch} \quad [A][x] = [b]$$

ist definiert durch die Menge

$$L := \{x \in \mathbb{R} \mid Ax = b, \text{ für ein } A \in [A] \text{ und ein } b \in [b]\}$$

aller Vektoren x , die Lösungen der reellen Gleichung $Ax = b$ mit einem $A \in [A]$ und einem $b \in [b]$ sind.

Im allgemeinen ist es schwierig, die Menge L zu beschreiben, da sie nicht einfach ein Intervallvektor ist. Deshalb ist es sinnvoller, einen Intervallvektor $[x]$ zu suchen, der

die Menge L enthält und der möglichst kleine Intervallkomponenten hat. Man sagt, man löst das Intervallgleichungssystem, wenn man $[x]$ sucht.

Um die oben genannte Lösung zu ermitteln, soll keine vollständige Newton-Iteration durchgeführt werden, sondern nur ein Newton-Schritt, denn diese Iteration kann gegen einen Punkt konvergieren, der nicht die globale Lösung des Optimierungsproblems ist. Ein anderer Teilschritt des Optimierungsverfahrens hätte die Box ev. früher eliminieren können.

Gauß-Seidel-Schritt

Die Intervall-Gauß-Seidel-Iteration wurde bereits von Alefeld in [1] beschrieben und in [6] in Verbindung mit erweiterter Intervallarithmetik behandelt. Der Intervall-Gauß-Seidel-Schritt entsteht durch die Aufspaltung der Matrix $[A] = R \cdot [J]$ in ihren Diagonaleil und den Rest und die nachfolgende Auflösung des entstandenen Systems nach dem Einzelschrittverfahren. Mit $c = m([x])$ und $b = R \cdot g(c)$ folgt

$$\left. \begin{aligned} [z]_i &:= c_i - \left(b_i + \sum_{\substack{j=1 \\ j \neq i}}^n [a]_{ij}([x]_j - c_j) \right) / [a]_{ii} \\ [x]_i &:= [z]_i \cap [x]_i \end{aligned} \right\} i = 1, \dots, n. \quad (8)$$

Falls nach der Schnittbildung $[x]_i = \emptyset$ gilt, kann die Berechnung der $[x]_i$ abgebrochen werden, da $[x]$ keine Lösung des Intervallgleichungssystems enthalten kann. Es wird also $[x] := \emptyset$ gesetzt. Die Division durch das Diagonalelement von $[A]$ wird in erweiterter Intervallarithmetik durchgeführt, um den Fall $0 \in [a]_{ii}$ nicht ausschließen zu müssen. Hierbei wird $[z]_i$ bzw. $[x]_i$ als abkürzende Schreibweise für die Vereinigung zweier Intervalle $[z]_i^1 \cup [z]_i^2$ bzw. $[x]_i^1 \cup [x]_i^2$ verwendet. Wenn dieser Fall auftritt, wird die Lücke zwischen den beiden Intervallen gespeichert und mit der Intervallhülle der beiden weitergearbeitet. Der Gauß-Seidel-Schritt bricht die Berechnung der $[x]_i$ beim Auftreten eines leeren Schnitts vorzeitig ab und bietet damit einen großen Vorteil gegenüber anderen Verfahren, z. B. gegenüber der Intervall-Gauß-Elimination.

Spezieller Intervall-Newton-Schritt

Nun kann der spezielle Intervall-Newton-Schritt nach Hansen formuliert werden.

Der Algorithmus **SpNewton** benötigt als Eingangsdaten die Box $[x]$, die Matrix $[J]$ und den Vektor f_c . Er liefert die veränderte Box $[x]$, die Information, ob die Box gelöscht werden kann (*IsEmpty*) und den Vektor $[gap]$, in dem die Lücken, die durch die erweiterte Intervalldivision entstanden sind, gespeichert werden.

Algorithmus 6.1: **SpNewton**($[x]$, $[J]$, f_c , *IsEmpty*, $[gap]$)

1. $R := m([J])^{-1}$; $[x]_{old} := [x]$; $[M] := R \cdot [J]$; $b := R \cdot f_c$;
2. **if** *DiagDominant*($[M]$) **then** $\left\{ \begin{array}{l} \textit{DiagDominant} \text{ getestet, ob } [M] \\ \textit{diagonaldominant} \text{ ist.} \end{array} \right\}$

```

IGA([x] - m([x], [M], -b, err);
    { Durchführung der Gauß-Elimination, wenn ein Fehler aufgetreten ist, wird err = 1 gesetzt. }
if err = 1 then
    GS_Step([x], [M], b, [gap], IsEmpty);    { Gauß-Seidel-Schritt }
else
    if [x] ∩ [x]old = ∅ then IsEmpty:=true
    else [x] := [x] ∩ [x]old; IsEmpty:=false;
3. else GS_Step([x], [M], b, [gap], IsEmpty);
4. return [x], IsEmpty, [gap]
    
```

Wenn ein Gauß-Seidel-Schritt durchgeführt wurde, können Lücken in den Komponenten von $[x]$ entstanden sein, die im Vektor $[gap]$ für eine spätere Verwendung gespeichert werden. Das Ergebnis kann also eine der vier Formen haben:

1. Die Box $[x]$ ist verkleinert worden.
2. Die Box $[x]$ kann gelöscht werden.
3. Die Box $[x]$ konnte nicht verkleinert werden, aber es gibt Teilboxen in $[x]$, die keine Nullstelle enthalten können. Diese sind in $[gap]$ gespeichert.
4. Die Box $[x]$ wurde verkleinert, und es wurden Lücken in $[gap]$ gespeichert.

6.2 John-Bedingungen

Eine notwendige Bedingung für ein (lokales oder globales) Minimum \bar{x} des Optimierungsproblems

$$\min f(x)$$

$$x \in G := \{x \in [x] : p_i(x) \leq 0, i = 1 \dots m\}$$

mit stetig differenzierbaren Funktionen f und p_i , $i = 1 \dots m$ liefern die John-Bedingungen (vgl. [4]). Hierzu definiert man die Menge

$$\mathcal{I}_0(\bar{x}) := \{i \in \{1 \dots m\} : p_i(\bar{x}) = 0\} \cup \{i \in \{1 \dots 2n\} : p'_i(\bar{x}) = 0\}$$

der aktiven Nebenbedingungen und der aktiven Randbedingungen im Punkt \bar{x} , wobei man die Randbedingungen auch als Nebenbedingungen ansehen kann. Nun definiert man analog

$$\mathcal{I}_0([x]) := \{i \in \{1 \dots m\} : 0 \in P_i([x])\} \cup \{i \in \{1 \dots 2n\} : 0 \in P'_i([x])\}$$

Die Randbedingungen müssen hier mit einbezogen werden, da der Fall, daß die globale Minimalstelle auf dem Rand des Suchgebietes liegt, nicht vernachlässigt werden darf. Seien p_1^*, \dots, p_r^* die aktiven Nebenbedingungen und aktiven Randbedingungen. Damit gilt

$$u_0 \nabla f(\bar{x}) + \sum_{j=1}^r u_j \nabla p_j^*(\bar{x}) = 0,$$

$$u_j p_j^*(\bar{x}) = 0, j = 1, \dots, r,$$

$$u_0 + \sum_{j=1}^r u_j = 1,$$

$$u_0 \geq 0, u_j \geq 0, j = 1, \dots, r,$$

mit den Lagrangemultiplikatoren $u_j, j = 0, \dots, r$.

Der Vektor $[t] \in I\mathbb{R}^N, N := n + r + 1$ setzt sich aus den beiden Vektoren $[x] = ([x]_1, \dots, [x]_n)^T$ und $[u] = ([u]_0, \dots, [u]_r)^T$ zusammen

$$[t] := \begin{pmatrix} [x] \\ [u] \end{pmatrix},$$

wobei r die Anzahl der aktiven Restriktionen ist und $x \in [x]$ und $u \in [u]$ gilt.

Ausgehend von einer Einschließung $[x]$ der Minimalstelle \bar{x} soll nun mit dem speziellen Intervall-Newton-Schritt die Intervallgleichung

$$F_{john, \mathcal{I}_0}([t]) = \begin{pmatrix} [u]_0 + \sum_{j=1}^r [u]_j - 1 \\ [u]_0 \nabla F([x]) + \sum_{j=1}^r [u]_j \nabla P_j^*([x]) \\ [u]_1 P_1^*([x]) \\ \vdots \\ [u]_r P_r^*([x]) \end{pmatrix} = 0$$

gelöst werden. Hierzu werden für die Lagrangemultiplikatoren die Einschließungen $u_j \in [u]_j = [0, 1], j = 1, \dots, r$, die durch die Bedingungen gegeben sind, verwendet und die Gleichung linearisiert.

Die dafür notwendige Jacobimatrix $[J]$ von $F_{john, \mathcal{I}_0}([t])$ erhält man wieder mit Hilfe einer speziellen Taylorentwicklung. Hierbei sind die einzelnen Elemente der Matrix definiert durch

$$[J]_{i,j} = \frac{\partial}{\partial t_j} F_{john, \mathcal{I}_0, i}([t]_1, \dots, [t]_j, t_{j+1}, \dots, t_N),$$

wobei $t := m([t])$ und $F_{john, \mathcal{I}_0, i}$ die i -te Komponente von F_{john, \mathcal{I}_0} ist und $i, j = 1, \dots, N$. Die Gleichung $F_{john, \mathcal{I}_0}([t]) = 0$ kann man folgendermaßen linearisieren

$$F_{john, \mathcal{I}_0}(t') + [J]([t] - t') = 0,$$

wobei $t' \in [t]$ ein reeller Vektor ist, üblicherweise wird $t' = m([t])$ gewählt.

Nun wird dieses Gleichungssystem mit dem speziellen Newton-Schritt gelöst. Die für die Aufstellung des Gleichungssystems benötigte Jacobimatrix sieht folgendermaßen aus:

$$\left(\begin{array}{cccccccc} 0 & \cdots & 0 & 1 & 1 & \cdots & 1 \\ u_0 \frac{\partial^2 f}{\partial x_1^2} + \sum_{i=1}^m u_i \frac{\partial^2 p_i}{\partial x_1^2} & \cdots & u_0 \frac{\partial^2 f}{\partial x_1 \partial x_n} + \sum_{i=1}^m u_i \frac{\partial^2 p_i}{\partial x_1 \partial x_n} & \frac{\partial f}{\partial x_1} & \frac{\partial p_1}{\partial x_1} & \cdots & \frac{\partial p_m}{\partial x_1} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ u_0 \frac{\partial^2 f}{\partial x_1 \partial x_n} + \sum_{i=1}^m u_i \frac{\partial^2 p_i}{\partial x_1 \partial x_n} & \cdots & u_0 \frac{\partial^2 f}{\partial x_n^2} + \sum_{i=1}^m u_i \frac{\partial^2 p_i}{\partial x_n^2} & \frac{\partial f}{\partial x_n} & \frac{\partial p_1}{\partial x_n} & \cdots & \frac{\partial p_m}{\partial x_n} \\ u_1 \frac{\partial p_1}{\partial x_1} & \cdots & u_1 \frac{\partial p_1}{\partial x_n} & 0 & p_1(x) & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ u_m \frac{\partial p_m}{\partial x_1} & \cdots & u_m \frac{\partial p_m}{\partial x_n} & 0 & 0 & \cdots & p_m(x) \end{array} \right)$$

Wenn weiter gezeigt werden kann, daß für die Lösungsmenge $[t']$ gilt: $[t'] \overset{\circ}{\subset} [t]$, dann ist die Existenz einer Lösung in $[t]$ gezeigt (vgl. Abschnitt 6.3).

Außerdem ist es möglich, mit Hilfe dieses Verfahrens die Nicht-Existenz einer Lösung zu zeigen, d.h. $[t] \cap [t'] = \emptyset$. Entsprechend gilt auch: Wenn für ein $i \in \{1, \dots, m\}$ $[u']_i < 0$ gilt, dann kann $[t']$ auch keine Lösung enthalten, da eine der John-Bedingungen verletzt wurde.

Aufgrund dieser Überlegungen kann nun der Algorithmus **John** formuliert werden. Die Menge \mathcal{I}_0 ist die Menge der aktiven Restriktionen, also der aktiven Nebenbedingungen und aktiven Randbedingungen, P_1^*, \dots, P_r^* von $[x]$. Von der Anzahl ihrer Elemente ist die Größe des zu lösenden Gleichungssystems abhängig.

Algorithmus 6.2: John ($[x]$, \mathcal{I}_0 , $IsEmpty$, $[gap]$)

1. $r := \# \mathcal{I}_0$;
2. **for** $i := 1$ **to** n **do** $[t]_i := [x]_i$; { Initialisierung von $[t]$ mit $[x]$ und $[u]_i = [0, 1]$. }
3. **for** $i := n + 1$ **to** $n + r + 1$ **do** $[t]_i := [0, 1]$;
4. $[J] := \nabla F_{john, \mathcal{I}_0}(m([t]), [t])$; { $[J]$ wird mit spezieller Taylor-entw. berechnet. }
 $f_c := f_{john, \mathcal{I}_0}(m([t]))$;
5. **SpNewton**($[t]$, $[J]$, f_c , $IsEmpty$, $[gap']$);
6. **for** $i := 1$ **to** n **do** $[x]_i := [t]_i$; $[gap]_i := [gap']_i$;
7. **return** $[x]$, $[gap]$, $IsEmpty$;

Von den Vektoren $[t]$ und $[gap']$ werden für das weitere Verfahren nur Teilvektoren benötigt, da auf die Lagrangemultiplikatoren nicht weiter eingegangen wird. Nur der benötigte Teil dieser Vektoren wird aus dem Algorithmus zurückgeliefert.

Aufgrund des im nächsten Abschnitt folgenden Satzes 6.1 werden durch diese Prozedur keine Teilboxen eliminiert, die eine Nullstelle der John-Funktion und damit eine Minimalstelle des Optimierungsproblems (1) enthalten.

6.3 Lokaler Existenznachweis

Für den speziellen Newton-Schritt gilt folgender Satz

Satz 6.1 : *Sei $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$ zweimal stetig differenzierbar und $[J(x, X)]$ die Jacobimatrix von g (in spezieller Form berechnet). Dann gilt für das Ergebnis $[x']$ eines speziellen Intervall-Newton-Schrittes*

- (a) *Ist x^* Nullstelle von g mit $x^* \in [x]$, dann gilt $x^* \in [x']$.*
- (b) *Ist $[x'] = \emptyset$, dann enthält $[x]$ keine Nullstellen von g .*
- (c) *Gilt $[x'] \overset{\circ}{\subset} [x]$, dann besitzt g in $[x]$ und damit auch in $[x']$ eine Nullstelle (Existenz).*

Bemerkung: Da $[J]$ mit Punkt- und Intervallargumenten ausgewertet wird, handelt es sich hier um eine Intervall-Steigungsmatrix, d.h. es kann nur die Existenz einer Minimalstelle in der Box nachgewiesen werden (vgl. hierzu [8]).

Der Intervall-Newton-Schritt für die John-Bedingungen liefert aufgrund dieses Satzes automatisch die Verifikation für die Existenz einer lokalen Minimalstelle. Dieser Nachweis kann auf dem Rechner erbracht werden, in dem lediglich gezeigt wird, daß die neu berechnete Box im Inneren der alten liegt. Allerdings kann es unter bestimmten Voraussetzungen vorkommen, daß diese Inklusion nicht erfüllt werden kann. In der behandelten Box können noch mehrere Minimalstellen enthalten sein, die so dicht zusammenliegen, daß für sie die verwendete Toleranzangabe keine Trennung durch Splittings mehr bewirkt. Andererseits kann die Bedingung auch nicht erfüllt sein, obwohl bereits eine einzelne Minimalstelle eingeschlossen wurde. Mögliche Gründe dafür sind:

1. Die Minimalstelle ist singulär.
2. Durch die Randbedingungen in der John-Funktion enthält die Intervall-Jacobimatrix eventuell eine singuläre Matrix.
3. Bei der intervallarithmetischen Auswertung treten zu große Überschätzungen auf.
4. Die Minimalstelle liegt auf dem Rand der Box.

Um das Erfüllen der Inklusionsbedingung in einigen Fällen doch möglich zu machen, kann man die sogenannte Epsilonaufblähung einsetzen (vgl. z.B. [15]). Wird für eine auf diese Weise vergrößerte Box $[z]$ ein Newton-Schritt durchgeführt und ist die Inklusionsbedingung für sie erfüllt, so enthält die Box eine lokale Minimalstelle. Problematisch ist hierbei, daß durch die Aufblähung der Box $[z]$ Punkte hinzukommen können, die nicht in der Startbox $[x]$ liegen. Daher muß zusätzlich die Bedingung $[z] \subseteq [x]$ erfüllt sein.

Zusammenfassend läßt sich jetzt der eigentliche Verifikationsschritt formulieren. Dieser wird separat für alle Boxen in der Resultatsliste durchgeführt, unabhängig davon, ob vorher bereits durch die Prozedur **John** die Existenz einer Minimalstelle in einer Box nachgewiesen werden konnte. Dies erhöht zwar den Aufwand des Verfahrens, hat sich aber in der Praxis als sinnvoll erwiesen.

Algorithmus 6.3: Verification_Step(L_{res} , $OptiVector$, $InfoVector$, s)

1. $s := 0$;
2. **for all** $([x], F_{[x]}) \in L_{res}$ **do**
 - (a) $[y] := \mathbf{Blow}([x], \varepsilon)$; $[y_{old}] := [y]$; { Epsilonaufblähung }
 - (b) **John** $([y], \mathcal{I}_0, IsEmpty, [gap])$
 - (c) **if not** $IsEmpty$ **then**
 - $s := s + 1$; $OptiVector_s := [y]$; $InfoVector_s := [y] \overset{\circ}{\subset} [y_{old}]$;
3. **return** $Optivector$, $InfoVector$, s ;

Bemerkung: Der Nachweis der Existenz hat sich in der Praxis weniger einfach erwiesen als in der Literatur oft angegeben wird. Grund dafür ist zum einen die evtl. lineare Abhängigkeit der Nebenbedingungen und Randbedingungen. Zum anderen treten Überschätzungen bei der Intervallauswertung und bei der Summierung innerhalb der expliziten Matrixberechnung auf. Oft liegt die globale Minimalstelle auch auf dem durch die Bedingungen spezifizierten Rand des zulässigen Bereichs und damit, da immer versucht wird, die Boxen auf zulässige Boxen zu reduzieren, auf dem Rand der Box in der Resultatsliste. Dann läßt sich, wie oben bereits erwähnt, die Existenz mit der Inklusionsbedingung nicht oder schwer nachweisen, selbst wenn die Box durch die Epsilonaufblähung vergrößert wurde.

7 Heuristiken für Listenordnung, Splittingtechniken und Abbruchkriterium

Nachdem nun die einzelnen Testmethoden zur Elimination und Reduktion von Teilboxen behandelt worden sind, beschäftigt sich dieser Abschnitt mit der Anordnung der Boxen in der Startliste und der Aufteilung der Boxen nach einem Iterationsschritt.

7.1 Listenordnung

Zur Anordnung der entstehenden Boxen wird eine geordnete Liste verwendet. Die notwendigen Operationen für die Listen und die Listenelemente wurden bereits in Abschnitt 2 definiert. In dem hier betrachteten Algorithmus wird die Listenordnung aus [13] verwendet. Die Boxen sollen bezüglich der Untergrenze ihrer Funktionsauswertung und ihres Alters in der Liste geordnet werden. Auf diese Weise wird die Möglichkeit

ausgeschlossen, daß immer wieder die gleiche Box zur Bearbeitung aus der Liste genommen wird, obwohl dort auch andere Boxen mit gleicher Untergrenze der Funktionsauswertung zu finden sind. Außerdem können so die Boxen zuerst bearbeitet werden, die die kleinsten Funktionswerte enthalten (eine exakte Beschreibung der verwendeten Listenordnung findet man in [3] und [13]).

7.2 Splittingtechniken

Bei der Anwendung des Algorithmus ist ein Kriterium notwendig, mit Hilfe dessen entschieden wird, ob eine Box durch einen Schritt des Algorithmus „genügend“ reduziert worden ist. Andernfalls muß die Box in mehrere Teilboxen aufgesplittet werden, für die der Algorithmus dann separat durchgeführt wird.

Sei $[x]$ die Box, für die ein Schritt des Algorithmus durchgeführt wird. Unter der Voraussetzung, daß $[x]$ nicht gelöscht wurde, sei $[x']$ die Teilbox von $[x]$, die durch den Algorithmusschritt entstanden ist. Es muß entschieden werden, ob $[x']$ wieder in die Startliste L , zur weiteren Behandlung mit dem Algorithmus kommt oder ob $[x']$ gesplittet werden soll.

Dazu muß festgelegt werden, was es bedeutet, daß eine Box „genügend“ reduziert worden ist. Um zu garantieren, daß wenigstens eine Komponente von $[x]$ um einen Faktor reduziert wird, der von dem Durchmesser der größten Komponente von $[x]$ abhängt, fordert man für ein $i = 1, \dots, n$

$$d([x]_i) - d([x']_i) > \gamma d([x])$$

für eine Konstante γ , $0 < \gamma < 1$.

In dem hier behandelten Algorithmus soll eine etwas schwächere Form dieses Kriteriums eingesetzt werden. Dazu sei γ abhängig von n , $\gamma(n) := \frac{n+47}{5n+88}$. Mit dieser Wahl ist $\gamma = 0.5$ für $n = 2$, $\gamma = 0.25$ für $n = 100$ und $0.25 > \gamma > 0.2$ für $n > 100$. Man definiert also

$$D := \frac{n+47}{5n+88} d([x]) - \max_{1 \leq i \leq n} \{d([x]_i) - d([x']_i)\}$$

Die Box $[x]$ wird als „genügend“ reduziert angesehen, wenn $D \leq 0$ gilt.

Verschiedene Splittingregeln

Wenn durch die verschiedenen auf eine Box $[x]$ angewendeten Verfahren Lücken in der Box produziert wurden, legen diese natürlich fest, an welchen Stellen die Box gesplittet werden soll. Dieser Fall soll am Ende dieses Abschnitts erläutert werden.

Sei $[x]$ eine Box, die nicht „genügend“ reduziert wurde. Diese Box soll nun in drei (oder n , falls $n < 3$) Komponenten (nacheinander) in der Mitte gesplittet werden. Auf diese Weise entstehen maximal acht neue Teilboxen, die in die Startliste übernommen und vom Algorithmus separat behandelt werden müssen. Mit Hilfe der verschiedenen Splittingregeln (vgl. dazu auch [14]) wird entschieden, in welchen Komponenten $[x]$ aufgesplittet wird. Dazu wird jeweils der Vektor T definiert. Die drei (oder n , falls $n < 3$) größten Komponenten von T legen die Komponenten fest, in denen $[x]$ gesplittet wird.

Mit Hilfe des Algorithmus **Split**, wird die Auswahl der drei größten Komponenten und das Splitten durchgeführt. Dazu werden der Vektor $[x]$, der Vektor T , der durch die Splittingregeln festgelegt wird, der Intervallvektor $[m]$, der die Stelle, an der gesplittet wird, festlegt und \bar{f} und ε_x für die Übernahme in L übergeben. Die neue Liste L mit den gesplitteten Teilboxen wird zurückgegeben.

Damit kann man nun den Algorithmus **SplitX** formulieren. Für das Splitting wird in diesem Algorithmus der Mittelpunkt von $[x]$ gewählt, der hier als Punktintervallvektor behandelt wird. Die Regel, nach der gesplittet werden soll, wird durch den Parameter *rule* festgelegt.

Algorithmus 7.1: **SplitX**($[x]$, G , \bar{f} , ε_x , L , *rule*)

1. $c := m([x])$;
2. **if** *rule* = A **then**
 - for** $i := 1$ **to** n **do** $T_i := d([x]_i)$;
3. **else if** *rule* = B **then**
 - for** $i := 1$ **to** n **do** $T_i := d(G_i) \cdot d([x]_i)$;
4. **else if** *rule* = C **then**
 - for** $i := 1$ **to** n **do** $T_i := d(G_i \cdot ([x]_i - m([x]_i)))$;
5. **else if** *rule* = D **then**
 - for** $i := 1$ **to** n **do**
 - if** $0 \in [x]_i$ **then** $T_i := d([x]_i)$
 - else** $T_i := d([x]_i) / \min\{|x_i| \mid x_i \in [x]_i\}$;
6. **Split**($[x]$, T , $[c, c]$, \bar{f} , ε_x , L);
7. **return** L ;

Nun zu der oben bereits erwähnten anderen Variante des Splittens. Wurde eine Lücke in einer oder mehreren Komponenten gefunden, so soll der Vektor $[x']$ in diesen Komponenten gesplittet werden, da dies den Suchbereich verkleinert. Außerdem ist die Chance groß, daß durch diese Lücke zwei Minimalstellen voneinander getrennt werden, d.h. daß sie sich danach in zwei verschiedenen „Suchboxen“ befinden. Auch hier ist die Anzahl der Komponenten, in denen gesplittet wird, auf maximal drei (oder n , falls $n < 3$) festgelegt, damit nur höchstens acht neue Teilboxen entstehen können. Wenn eine Lücke sehr klein ist und noch dazu am Rand der zu splittenden Komponente liegt, soll mit ihr $[x]$ nicht gesplittet werden. Auch hier soll nur in den größten Komponenten gesplittet werden. Sei also $[x']_i := [\underline{x}'_i, \underline{gap}] \cup [\overline{gap}, \overline{x}'_i]$ die Komponente, in der eine Lücke entstanden ist. Die Lücke $(\underline{gap}, \overline{gap})$ soll zum Splitten von $[x']_i$ verwendet werden, wenn gilt

$$\min\{\overline{gap} - \underline{x}_i, \overline{x}_i - \underline{gap}\} \geq 0.25 \cdot d([x']). \quad (9)$$

Wenn also der Durchmesser $d_{\sqcup}([x']) = \overline{gap} - \underline{gap}$ der Lücke größer als $0.25 \cdot d([x'])$ ist, dann ist die Bedingung (9) sicherlich erfüllt. Wenn es mehr als drei Lücken gibt, die (9) genügen, dann werden die drei größten verwendet.

Der Algorithmus **SplitGap** beschreibt dieses Splitten des Vektors $[x]$. Ihm wird ein Vektor $[gap]$ übergeben. Seine Komponenten $[gap]_i$ enthalten die durch den Algorithmus in den Komponenten $[x]_i$ entstandenen Lücken. Wenn mehrere Lücken in einer Komponente entstanden sind, dann werden diese mit dem ALgorithmus **Merge** vereinigt. Dieser arbeitet nach dem Scan-line Prinzip aus der algorithmischen Geometrie und vereinigt nur die Intervalle, die sich überschneiden oder berühren. Dieses Verfahren läßt sich hier hervorragend einsetzen, da es wesentlich geringeren Aufwand benötigt als der paarweise Vergleich der Intervalle (vgl. [3] und [9]). In der Komponente $[gap]_i$ steht die Lücke von $[x]_i$, die den größten Durchmesser hat. Wenn keine Lücken gefunden worden sind, die sich zum Splitten eignen, wird dies mit $NoGaps=false$ angezeigt. Ansonsten werden die neuen Teilboxen in die Liste L geschrieben.

Algorithmus 7.2: SplitGap($[x]$, $[gap]$, \bar{f} , ε_x , L , $NoGaps$)

1. $NoGaps:=true$;
2. **for** $i := 1$ **to** n **do**
 - if** $\overline{gap}_i - \underline{x} < \bar{x} - \underline{gap}_i$ **then** $Min := \overline{gap}_i - \underline{x}$ $\left\{ \begin{array}{l} \text{Auswahl der Komponenten, die} \\ \text{für ein Splitting in Frage kom-} \\ \text{men.} \end{array} \right.$
 - else** $Min := \bar{x} - \underline{gap}_i$;
 - if** $Min \geq 0.25 \cdot d([x])$ **then** $T_i := d([gap]_i)$; $NoGaps:=false$
 - else** $T_i := -1$; $\left\{ \begin{array}{l} \text{Wenn eine Komponente nicht} \\ \text{zum Splitten geeignet ist, wird } T_i \\ \text{auf einen für einen Durchmesser} \\ \text{unzulässigen Wert gesetzt.} \end{array} \right.$
3. Split($[x]$, T , $[gap]$, \bar{f} , ε_x , L);
4. **return** L , $NoGaps$;

7.3 Abbruchbedingung

Die Bedingung für das Umspeichern einer Box von der Startliste in die Resultatsliste erhält man aus den Genauigkeitsanforderungen für die Boxen in der Resultatsliste. Im Gegensatz zu der von Hansen in [5] verwendeten absoluten Genauigkeitsanforderung wird in dem hier behandelten Verfahren eine relative Toleranzangabe verwendet, die sich sowohl auf den Durchmesser der Box selbst, wie auch auf den Durchmesser der entsprechenden intervallarithmetischen Funktionsauswertung bezieht. Dazu können zwei verschiedene Toleranzangaben ε_x und ε_F definiert werden. Eine Box $[x]$ wird umgespeichert, wenn sie der folgenden Bedingung genügt

$$d_{\text{rel}}([x]) < \varepsilon_x \vee d_{\text{rel}}(F([x])) < \varepsilon_F.$$

Dieses Umspeichern wird durch den Algorithmus **Update_Terminate** durchgeführt, nachdem noch einmal mit jedem Element der Liste L versucht wurde, einen besseren Wert für \bar{f} zu ermitteln.

Algorithmus 7.3: Update_Terminate($L, L_{start}, L_{res}, \bar{f}, \bar{x}, \varepsilon_x, \varepsilon_F$)

1. Reduce \bar{f} (L, \bar{f}, \bar{x}); { Verbessern von \bar{f} . }
2. **if** $\bar{f} < \infty$ **then** CutOffTest(L, \bar{f}) { Löschen aller Elemente, die kein
Minimum enthalten können. }
3. **while** $L \neq \{\}$ **do**
 - (a) $([x], F_{[x]}) := \text{Head}(L)$;
 - (b) $L := L - ([x], F_{[x]})$;
 - (c) **if** $d_{\text{rel}}([x]) < \varepsilon_x$ **or** $d_{\text{rel}}(F([x])) < \varepsilon_F$ **then** $L_{res} := L_{res} + ([x], F_{[x]})$
else $L_{start} := L_{start} + ([x], F_{[x]})$;
4. **return** $L_{start}, L_{res}, \bar{f}, \bar{x}$;

Bemerkung: Es hat sich in der Praxis als sinnvoll erwiesen, die Bedingung in Schritt 3(c) mit „oder“ zu formulieren, im Gegensatz zu dem von Hansen vorgeschlagenen „und“. Bei größeren Funktionen können die Überschätzungen bei der Intervallauswertung so groß sein, daß der Intervallvektor $[x]$ zwar der Genauigkeitsanforderung genügt, die Funktionsauswertung aber auch z.B. durch mehrfaches Splitten von $[x]$ nicht mehr kleiner wird.

8 Der vollständige Algorithmus

Bevor der vollständige Algorithmus `ConstrainedGlobalOptimize` angegeben wird, soll ein weiterer Teilalgorithmus eingeführt werden, der die bisher beschriebenen Algorithmen verwendet, um die eigentliche Iteration durchzuführen.

8.1 Der Algorithmus MAIN

Für den Algorithmus MAIN sind als Eingangsdaten folgende Parameter notwendig:

- Die Startliste L_{start} , die die Startbox enthält,
- \bar{f} , falls durch die Benutzereingabe oder eine vorhergehende Prozedur bereits ein \bar{f} bestimmt wurde, ansonsten gilt $\bar{f} = \infty$,
- der Vektor \bar{x} , entsprechend \bar{f} ,
- der Steuerparameter für die Splittingregel *rule* und
- die Genauigkeitsforderungen $\varepsilon_x, \varepsilon_F$.

Als Operator $+$ für das Einhängen eines Paares $([x], F_{[x]})$ in die Liste wird der mit der Listenordnung aus Abschnitt 7.1 definierte Operator verwendet.

Algorithmus 8.1: MAIN(L_{start} , L_{res} , \bar{f} , \bar{x} , $rule$, ε_x , ε_F)

1. $([x]_{start}, F_{[x]_{start}}) := \text{Head}(L_{start});$
2. **while** $L_{start} \neq \{\}$ **do**
begin
 - (a) $([x], F_{[x]}) := \text{Head}(L_{start});$
 - (b) $L_{start} := L_{start} - ([x], F_{[x]})$
 - (c) $[x]_{anf} := [x]; \mathcal{I}_0 := \emptyset; \mathcal{I}'_0 := \emptyset$
 $\left\{ \begin{array}{l} [x]_{anf} \text{ wird für die die späte-} \\ \text{ren Vergleiche des Ausgangsin-} \\ \text{tervallvektors und des bearbeite-} \\ \text{ten Intervallvektors benötigt.} \end{array} \right\}$
 - (d) **if** $\text{infeasible}([x])$ **then goto** 2.;
 $\left\{ [x] \text{ ist sicher unzulässig.} \right\}$
 - (e) **for** $i := 1$ **to** m **do**
 - if** $0 \in P_i([x])$ **then** $\mathcal{I}_0 := \mathcal{I}_0 \cup \{i\};$
 $\left\{ \begin{array}{l} \mathcal{I}_0 \text{ ist die Indexmenge der akti-} \\ \text{ven Nebenbedingungen für } [x]. \end{array} \right\}$
 - (f) **if** $\mathcal{I}_0 = \emptyset$ **then**
 $\left\{ [x] \text{ ist sicher zulässig.} \right\}$
 - $\text{MonotonicityTest}([x]_{start}, [x], \text{delete})$
 - if** delete **then goto** 2.
 - $\text{ConcavityTest}([x]_{start}, [x], L_{start}, \bar{f}, \text{delete})$
 - if** delete **then goto** 2.
 - (g) **if** $\bar{f} < \infty$ **then**
 $\left\{ \begin{array}{l} \text{Die Voraussetzung für die Durch-} \\ \text{führung von } \text{QuadraticMethod} \\ \text{ist, daß bereits mind. ein zulässi-} \\ \text{ger Punkt gefunden wurde.} \end{array} \right\}$
 - $c := m([x]);$
 - if** $\overline{F(c)} \geq \bar{f}$ **then**
 $\left\{ \begin{array}{l} \text{Unter dieser Bedingung kann} \\ \text{man erwarten, daß Teile von } [x] \\ \text{gelöscht werden können.} \end{array} \right\}$
 - $\text{QuadraticMethod}([x], \bar{f}, [gap_1], \text{IsEmpty});$
 - if** IsEmpty **then goto** 2.;
 - (h) **if not** $\text{infeasible}([x])$ **and not** $\text{feasible}([x])$ **then**
 $\left\{ \begin{array}{l} \text{Die Voraussetzung für die Durch-} \\ \text{führung von } \text{LinearizeSolve} \text{ ist,} \\ \text{daß für mindestens ein } i \text{ gilt: } 0 \in \\ P_i([x]). \end{array} \right\}$
 - $\text{Select}([x], \mathcal{I});$
 - $\text{LinearizeSolve}([x], \mathcal{I}, [gap_2], \text{IsEmpty});$
 - if** IsEmpty **then goto** 2.;
 - (i) **if** $0 \in P'_i([x])$ **then** $\mathcal{I}'_0 := \mathcal{I}'_0 \cup \{i\};$
 $\left\{ \begin{array}{l} \mathcal{I}'_0 \text{ ist die Indexmenge der akti-} \\ \text{ven Randbedingungen für } [x]. \end{array} \right\}$
 - $\text{John}([x], \mathcal{I}_0 \cup \mathcal{I}'_0, \text{IsEmpty}, [gap_3]);$
 - if** IsEmpty **then goto** 2.;
 - (j) $L' := \{\};$
 - if** $d_{\text{rel}}([x]) < \varepsilon_x$ **then**

```

    L' := L' + ([x], F[x]);
    Update_Terminate(L', Lstart, Lres,  $\bar{f}$ ,  $\bar{x}$ ,  $\varepsilon_x$ ,  $\varepsilon_F$ ); goto 2.;
(k) else Merge([gap1], [gap2], [gap3], [Gap]);  $\left\{ \begin{array}{l} \text{Mit Merge werden die durch} \\ \text{die Teilverfahren entstandenen} \\ \text{Lücken [gap}_1\text{], [gap}_2\text{] und [gap}_3\text{]} \\ \text{vereinigt.} \end{array} \right.$ 
        SplitGap([x], [Gap],  $\varepsilon_x$ ,  $\bar{f}$ , L', NoGaps);
        if not NoGaps then
            Update_Terminate(L', Lstart, Lres,  $\bar{f}$ ,  $\bar{x}$ ,  $\varepsilon_x$ ,  $\varepsilon_F$ ); goto 2.;
        (l) D :=  $\gamma(n)d([x]_{anf}) - \max_{1 \leq i \leq n} \{d([x]_{anf,i}) - d([x]_i)\}$ ;
        (m) if D ≤ 0 then Lstart := Lstart + ([x], F[x]);
        (n) else SplitX([x], G,  $\bar{f}$ ,  $\varepsilon_x$ , L', rule);
            Update_Terminate(L', Lstart, Lres,  $\bar{f}$ ,  $\bar{x}$ ,  $\varepsilon_x$ ,  $\varepsilon_F$ ); goto 2.;
    end{while};
3. return Lres,  $\bar{f}$ ,  $\bar{x}$ ;

```

8.2 Der Algorithmus ConstrainedGlobalOptimize

Dieser Algorithmus faßt nun letztendlich alle für das Verfahren notwendigen Teile zusammen. Als Eingangsdaten benötigt er die Funktion f , die Nebenbedingungen p_1, \dots, p_m , die Startbox $[x]$, die Toleranzangaben ε_x und ε_F und den Steuerparameter $rule$ für die Auswahl der Splittingregel. Wenn ein erster Wert für \bar{f} bereits bekannt ist, kann \bar{f} gesetzt werden, d.h. es gilt schon zu Beginn $\bar{f} < \infty$. **Compress**(L_{res}) reduziert die Anzahl der Boxen in der Resultatsliste L_{res} , in dem die sich schneidenden Boxen vereinigt werden. Dies wird durch Vergleichen der Boxen wie in [13], 2.7.3 realisiert.

Algorithmus 8.2: ConstrainedGlobalOptimize

```

([x],  $\bar{x}$ ,  $\bar{f}$ ,  $\varepsilon_x$ ,  $\varepsilon_F$ , rule, OptiVector, InfoVector, Minimum)
1. Lstart :=  $\{([x], F_{[x]})\}$ ; Lres :=  $\{\}$ ;  $\left\{ \begin{array}{l} \text{In der Startliste ist nur das Paar} \\ \text{([x], F}_{[x]}\text{) enthalten.} \end{array} \right.$ 
2. reduce $\bar{f}$ (Lstart,  $\bar{f}$ ,  $\bar{x}$ );
3. MAIN(Lstart, Lres,  $\bar{f}$ ,  $\bar{x}$ , rule,  $\varepsilon_x$ ,  $\varepsilon_F$ );
4. if  $\bar{f} < \infty$  then CutOffTest(Lres,  $\bar{f}$ );
5. Compress(Lres);
6. Verification_Step(Lres, OptiVector, InfoVector, s);
7. if  $\bar{f} < \infty$  then  $\underline{F} := \min_{1 \leq i \leq s} \underline{F}(\text{OptiVector}_i)$ ; Minimum :=  $[\underline{F}, \bar{f}]$ 
    else  $\underline{F} := \min_{1 \leq i \leq s} \underline{F}(\text{OptiVector}_i)$ ;  $\overline{F} := \max_{1 \leq i \leq s} \overline{F}(\text{OptiVector}_i)$ ;
        Minimum :=  $[\underline{F}, \overline{F}]$ ;
8. return OptiVector, InfoVector, Minimum;

```

Die Einschließungen aller Kandidaten für globale Minimalstellen sind am Ende dieses Verfahrens im Vektor *OptiVector* abgespeichert und die Einschließung des globalen Minimums in *Minimum*. Der dazugehörige Vektor *InfoVector* gibt an, ob die Existenz einer lokalen Minimalstelle in der berechneten Einschließung gezeigt werden konnte. Wenn die Listen L_{start} und L_{res} beliebig viele Elemente aufnehmen können, terminiert das Verfahren nach endlich vielen Schritten. Nach Beendigung des Verfahrens gilt für die s ermittelten Einschließungen in *OptiVector* und das globale Minimum f^* :

- Wenn $\bar{f} < \infty$ gilt, dann wurde mindestens ein sicher zulässiger Punkt gefunden. Deshalb gilt

$$\underline{E} \leq f^* \leq \bar{f}$$

mit

$$\underline{E} := \min_{1 \leq i \leq s} \underline{F}(\text{OptiVector}_i).$$

- Wenn kein sicher zulässiger Punkt gefunden wurde ($\bar{f} = \infty$), dann gilt entweder

$$\underline{E} \leq f^* \leq \bar{F}$$

mit

$$\underline{E} := \min_{1 \leq i \leq s} \underline{F}(\text{OptiVector}_i) \quad \text{und} \quad \bar{F} := \max_{1 \leq i \leq s} \bar{F}(\text{OptiVector}_i),$$

d.h. es konnte nicht nachgewiesen werden, daß kein zulässiger Punkt existiert, oder der Algorithmus terminiert mit einer leeren Resultatsliste, d.h. es existiert nachweislich kein zulässiger Punkt.

Dies ergibt sich aus den zuvor beschriebenen Algorithmen, mit denen Boxen, die globale Minimalstellen enthalten, nicht verworfen werden und dem Aufbau des Verfahrens.

8.3 Abschließende Bemerkungen zum Verfahren

Die Grundlage für das behandelte Verfahren bildet ein Verfahren von Hansen [5]. Zu einigen der zahlreichen Modifikationen, die im Rahmen der praktischen Realisierung an dem in [5] informell beschriebenen Verfahren vorgenommen wurden, sollen nun noch Anmerkungen gemacht werden.

In [5] wird vorausgesetzt, daß die globale Minimalstelle im Inneren des Startbereichs liegt. Wenn dieser aber nicht groß genug gewählt wurde, kann über die erhaltene Lösung nichts ausgesagt werden, da diese nur eine lokale Lösung sein könnte.

Abbildung 2 demonstriert dieses Problem mit Hilfe der Testfunktion $f(x) = \frac{1}{4}x^3 - \frac{1}{4}x^2 - 2x + 6$. Der Einfachheit wegen wird ein Optimierungsproblem nur mit Randbedingungen betrachtet. f hat an der Stelle $x = 2$ ein lokales Minimum. Wählt man den Startbereich $[-4, 4]$, und legt keine Randbedingungen (oder vielleicht nur die rechte Randbedingung) fest, wie dies in [5] beschrieben ist, so wird die Stelle $x = 2$ als globale Minimalstelle der Funktion f ausgegeben. Wählt man den Startbereich etwas größer, so wird $x = 2$ als Minimalstelle verworfen, und der Algorithmus gibt den linken Randpunkt als Lösung an, von dem er jedoch lediglich aussagen kann, daß er zulässig

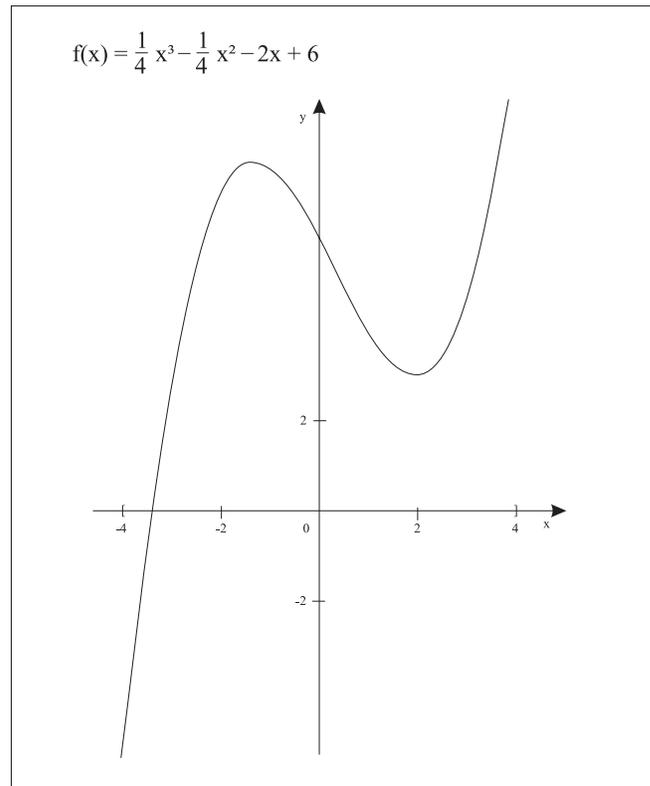


Abbildung 2: f^* wird auf dem linken Rand des Startintervalls $[-4,4]$ angenommen.

ist und den kleinsten gefundenen Funktionswert hat. Die Box, die diesen Punkt enthält, wurde von dem Intervall-Newton-Schritt, der für die Funktion der John-Bedingungen durchgeführt wurde, verworfen. Zurecht, da der Punkt unter den gegebenen Bedingungen kein Minimum ist und in [5] auch keine getrennte Randbehandlung, wie z.B. in [13], vorgenommen wird.

Aufgrund dessen muß eine Randbehandlung durchgeführt werden, da sonst im Grunde keine Aussagen über die erhaltene Lösung gemacht werden können. Die Randbedingungen können entweder als Nebenbedingungen formuliert werden, was den Aufwand des Verfahrens enorm erhöht, oder in die Teile des Verfahrens, in denen sie gebraucht werden, getrennt von den Nebenbedingungen eingehen. In dem in dieser Arbeit beschriebenen Verfahren ergeben sich die Randbedingungen aus der Wahl der Startbox. Die aktiven Randbedingungen werden in die Johnbedingungen mit aufgenommen. Ansonsten müssen sie nicht weiter betrachtet werden, da Monotonie- und Konkavitätstest mit Randbehandlung ausgeführt werden, und in das Teilverfahren `QuadraticMethod` die Restriktionen nicht eingehen. Für das Teilverfahren `LinearizeSolve` werden die Randbedingungen p'_i mit dem Selektionstest nicht ausgewählt, da sie durch die Wahl der Startbox festgelegt werden und damit für alle p'_i gilt: $s_i \leq 0$.

Die berechneten Boxen enthalten dann alle Kandidaten für mögliche globale Minimalstellen der Funktion f unter den Nebenbedingungen p_i , $i = 1, \dots, m$ und den $2n$ Randbedingungen, und die ermittelte Einschließung enthält das globale Minimum im Startbereich.

Ein weiteres Problem ergab sich mit der Wahl der Abbruchbedingung. Formuliert man die Bedingung, wie in [5] genannt, mit „und“, so wird eine Box, die durch einen Iterationsschritt nicht verkleinert werden konnte, und die einen Durchmesser kleiner als ε_x hat, deren Funktionsauswertung aber noch zu großen Durchmesser hat, immer wieder in die Startliste geschrieben. Entweder muß in einem solchen Fall die Box gesplittet werden oder die Bedingung, wie das hier geschehen ist, mit „oder“ formuliert werden.

9 Numerische Tests

Das Verfahren ist in *Pascal-XSC* 2.03 für den Gnu C++ Compiler programmiert. Alle Tests wurden auf dem gleichen Rechner (Intel Pentium 100 MHz, 8 MB RAM) durchgeführt.

Da der Algorithmus in der Implementierung von der Differentiationsarithmetik Gebrauch macht, wird bei jeder Gradientenkomponentenberechnung auch ein Funktionswert und bei jeder Hessematrixauswertung auch ein Gradient (bzw. ein Teil des Gradienten) und ein Funktionswert berechnet. Es werden dabei jedoch nicht alle Zähler erhöht, sondern nur jeweils der Zähler für die höchste Ableitung. Für eine Gradientenkomponentenberechnung erhöht sich also nur GC und nicht auch FA. Die Differentiationsarithmetik ist so konstruiert, daß eine Komponente des Gradienten, eine Spalte oder die Diagonale der Hessematrix einzeln berechnet werden können. Diese Differentiationsarithmetik stand für [3] noch nicht zur Verfügung. Die dort angegebenen Meßdaten konnten mit dieser Arithmetik wesentlich verbessert werden.

Für jeden Test werden die Zielfunktion f , die Nebenbedingungen p_1, \dots, p_m , der Startbereich und die mit einer relativen Toleranzangabe von $\varepsilon_x = \varepsilon_F = 10^{-12}$ berechneten Einschließungen, die die Kandidaten für die globalen Minimalstellen enthalten, und die Einschließung des globalen Minimums angegeben. Die gegebenen Randbedingungen legen jeweils die Lage der Startbox fest.

Für die Aufwandsangaben der Testfunktionen wurde mit einer relativen Toleranzangabe von $\varepsilon_x = \varepsilon_F = 10^{-2}$ gearbeitet. Nur für die hier dokumentierten unrestringierten Probleme wurde die schärfere Genauigkeitsforderung $\varepsilon_x = \varepsilon_F = 10^{-8}$ gewählt.

Die Aufwandsübersichten der einzelnen Probleme werden in tabellarischer Form angegeben. Es werden die folgenden Abkürzungen verwendet

n	Dimension des Problems,
FA_f	Anzahl der Funktionsauswertungen der Funktion f ,
GC_f	Anzahl der Gradientenkomponentenauswertungen der Funktion f ,
HC_f	Anzahl der Hessematrixkomponentenauswertungen der Funktion f ,
FA_{p_i}	Anzahl der Funktionsauswertungen der Funktion p_i ,
GC_{p_i}	Anzahl der Gradientenkomponentenauswertungen der Funktion p_i ,
HC_{p_i}	Anzahl der Hessematrixkomponentenauswertungen der Funktion p_i ,
MaxL	Maximale Anzahl von Elementen in der Startliste,
STU-time	Laufzeit in STU-Zeiteinheiten,
time	absolute Zeitangabe.

Eine STU-Zeiteinheit ist die sogenannte Standard-Time-Unit, die der Berechnungszeit für 1000 reelle Auswertungen der Shekel-5-Funktion entspricht. Diese wird als eine

nahezu von Compiler und Rechner unabhängige Zeitmeßkonstante verwendet. Auf dem für die Tests eingesetzten Rechner liegt eine STU-Zeiteinheit zwischen 1.45 und 1.55s.

Für einige Testprobleme konnte gezeigt werden, daß die in der jeweiligen Literatur angegebenen Ergebnisse, die mit approximativen Verfahren berechnet wurden, nicht in den verifizierten mit CGOP berechneten Einschließungen enthalten sind. Dies gilt z.B. für **f6**, **f9**, **f19**, **f24**, **f54**, **f55**, **f64**, **f66**, **f67**.

Testgruppe 1: Unrestringierte Probleme

Beispiel 1 : (Shekel-5-Funktion, $x \in \mathbb{R}^4$)

$$f_{S_5} = - \sum_{i=1}^5 \frac{1}{(x - A_i)(x - A_i)^T + c_i}$$

Dabei ist

$$A = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix} \quad \text{und} \quad c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}$$

Startbox: $[x]_i = [0, 10]$, für $i = 1, \dots, 4$.

Minimalstelle: [4.000037152819157E+000, 4.000037152820193E+000]
 [4.000133276591086E+000, 4.000133276592036E+000]
 [4.000037152803115E+000, 4.000037152836235E+000]
 [4.000133276591083E+000, 4.000133276592035E+000]

Minimum: [-1.015319967905887E+001, -1.015319967905822E+001]

FA f_{S_5}	GC f_{S_5}	HC f_{S_5}	MaxL	STU-time
383	176	604	13	6.34

Beispiel 2 : (Six-Hump-Camel-Back-Function, $x \in \mathbb{R}^2$)

$$f_{SHCB}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4.$$

Startbox: $[x]_i = [-5, 5]$, für $i = 1, 2$.

Minimalstellen: 1. [-8.984201310039366E-002, -8.984201310027967E-002]
 [7.126564030205180E-001, 7.126564030212119E-001]
 2. [8.984201310025305E-002, 8.984201310040561E-002]
 [-7.126564030210457E-001, -7.126564030204394E-001]

Minimum: [-1.031628453493978E+000, -1.031628453489877E+000]

FA f_{SHCB}	GC f_{SHCB}	HC f_{SHCB}	MaxL	STU-time
910	538	1280	42	7.66

Beispiel 3 : (Levy No.5, $x \in \mathbb{R}^2$)

$$f_{W_5}(x) = \sum_{i=1}^5 i \cos((i-1)x_1 + i) \sum_{j=1}^5 j \cos((j+1)x_2 + j) + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2.$$

Startbox: $[x]_i = [-10, 10]$, für $i = 1, 2$.

Minimalstelle: $[-1.306853009753776\text{E}+000, -1.306853009753453\text{E}+000]$
 $[-1.424845041560694\text{E}+000, -1.424845041560673\text{E}+000]$

Minimum: $[-1.761375780016517\text{E}+002, -1.761375780016293\text{E}+002]$

Diese Funktion hat 760 lokale Minimalstellen im Startbereich, aber nur eine globale Minimalstelle.

FA $_{f_{W5}}$	GC $_{f_{W5}}$	HC $_{f_{W5}}$	MaxL	STU-time
806	426	962	53	41.97

Testgruppe 2

Bei den Testproblemen **f8**, **f31**, **f39**, **f57**, **f69**, **f76**, **f77**, **f80**, **f81** konnte die Existenz einer Minimalstelle in den ermittelten Einschließungen nachgewiesen werden.

Beispiel 4 : (**f8**, $x \in \mathbb{R}^2$) (Nummer 16 in [7])

$$f_8(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$p_1(x) = -x_1 - x_2^2 \quad p_2(x) = -x_1^2 - x_2$$

Startbox: $[x] = ([-0.5, 0.5], [-0.5, 1])^T$.

Minimalstelle: $[5.000000000000000\text{E}-001, 5.000000000000000\text{E}-001]$
 $[2.500000000000000\text{E}-001, 2.500000000000001\text{E}-001]$

Minimum: $[2.500000000000000\text{E}-001, 2.500000000000000\text{E}-001]$

Beispiel 5 : (**f39**, $x \in \mathbb{R}^3$) Rosenbrock's-Post-Office-Problem (Nummer 250 in [7])

$$f_{39}(x) = -x_1 x_2 x_3$$

$$p_1(x) = -x_1 - 2x_2 - 2x_3 \quad p_2(x) = x_1 + 2x_2 + 2x_3 - 72$$

Startbox: $[x] = ([0, 20], [0, 11], [0, 42])^T$.

Minimalstelle: $[1.999999999999459\text{E}+001, 2.000000000000000\text{E}+001]$
 $[1.099999999999984\text{E}+001, 1.100000000000000\text{E}+001]$
 $[1.499999999999878\text{E}+001, 1.500000000000368\text{E}+001]$

Minimum: $[-3.300000000000809\text{E}+003, -3.29999999999720\text{E}+003]$

Beispiel 6 : (**f76**, $x \in \mathbb{R}^2$) (Nummer 21 in [7])

$$f_{76}(x) = 0.01x_1^2 + x_2^2 - 100$$

$$p_1(x) = x_2 - 10x_1 + 10$$

Startbox: $[x] = ([2, 50], [-50, 50])^T$.

Minimalstelle: $[2.000000000000000\text{E}+000, 2.000000000000000\text{E}+000]$
 $[-1.015747792622045\text{E}-007, 1.015747792622045\text{E}-007]$

Minimum: $[-9.996000000000001\text{E}+001, -9.995999999999999\text{E}+001]$

Beispiel 7 : (**f77**, $x \in \mathbb{R}^2$) (Nummer 20 in [7])

$$f_{77}(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 = f_8$$

$$p_1(x) = -x_1 - x_2^2 \quad p_2(x) = -x_1^2 - x_2$$

$$p_3(x) = -x_1^2 - x_2^2 + 1$$

Startbox: $[x] = ([-0.5, 0.5], [0, 1])^T$.

Minimalstelle: [4.999999999999999E-001, 5.000000000000000E-001]
 [8.660254037844137E-001, 8.660254037847470E-001]

Minimum: [3.819872981077499E+001, 3.819872981080866E+001]

Beispiel 8 : (f81, $x \in \mathbb{R}^2$)

(Nummer 12 in [7])

$$f_{81}(x) = 0.5x_1^2 + x_2^2 - x_1(x_2 + 7) - 7x_2$$

$$p_1(x) = 4x_1^2 + x_2^2 - 25$$

Startbox: $[x] = ([-20, 10], [-20, 10])^T$.

Bei diesem Test wurde für die Ermittlung der Minimalstelle mit einer Genauigkeit von $\varepsilon_x = \varepsilon_F = 10^{-15}$ gearbeitet.

Minimalstelle: [1.999999999999998E+000, 2.000000000000001E+000]
 [2.999999999999998E+000, 3.000000000000002E+000]

Minimum: [-3.000000000000005E+001, -2.999999999999999E+001]

	f8	f39	f76	f77	f81
n	2	3	2	2	2
FA _f	101	745	10	106	173
GC _f	54	345	8	46	112
HC _f	96	432	11	48	114
FA _{p1}	34	610	8	60	180
GC _{p1}	2	0	2	2	74
HC _{p1}	4	0	4	4	84
FA _{p2}	37	740	–	55	–
GC _{p2}	8	177	–	0	–
HC _{p2}	8	234	–	0	–
FA _{p3}	–	–	–	77	–
GC _{p3}	–	–	–	34	–
HC _{p3}	–	–	–	36	–
MaxL	4	34	1	5	7
STU-time	0.53	2.71	0.07	0,53	1.0
time	0:00:00,820	0:00:04,180	0:00:00,110	0:00:00,830	0:00:01,540

Beispiel 9 : (f9, $x \in \mathbb{R}^2$)

(Nummer 19 in [7])

$$f_9(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

$$p_1(x) = 100 - (x_1 - 5)^2 - (x_2 - 5)^2 \quad p_2(x) = (x_2 - 5)^2 + (x_1 - 6)^2 - 82.81$$

Startbox: $[x] = ([13, 100], [0, 100])^T$.

Minimalstelle: [1.409499999999999E+001, 1.409500000000053E+001]
 [8.429607892154527E-001, 8.429607892259997E-001]

Minimum: [-6.961813875580172E+003, -6.961813875567571E+003]

Beispiel 10 : (f10, $x \in \mathbb{R}^2$)

(Nummer 24 in [7])

$$f_{10}(x) = \frac{1}{27\sqrt{3}}((x_1 - 3)^2 - 9)x_2^3$$

$$p_1(x) = x_2 - \frac{1}{\sqrt{3}}x_1 \quad p_2(x) = -x_1 - \frac{1}{\sqrt{3}}x_2$$

$$p_3(x) = x_1 + \sqrt{3}x_2 - 6$$

Startbox: $[x] = ([0, 5], [0, 5])^T$.

Minimalstelle: [2.999999999997865E+000, 3.000000000001614E+000]
 [1.732050807567645E+000, 1.732050807568878E+000]

Minimum: [-1.000000000000001E+000, -9.99999999970282E-001]

Beispiel 11 : (f18, $x \in \mathbb{R}^4$)

(Nummer 44 in [7])

$$f_{18}(x) = (x_1 - x_2)(x_4 - x_3 + 1) - x_3$$

$$p_1(x) = x_1 + 2x_2 - 8 \quad p_2(x) = 4x_1 + x_2 - 12$$

$$p_3(x) = 3x_1 + 4x_2 - 12 \quad p_4(x) = 2x_3 + x_4 - 8$$

$$p_5(x) = x_3 + 2x_4 - 8 \quad p_6(x) = x_3 + x_4 - 5$$

Startbox: $[x] = ([0, 1], [0, 4], [0, 1], [0, 5])^T$.

Minimalstelle: [0.000000000000000E+000, 8.273562986336874E-028]
 [2.999999999998079E+000, 3.000000000000000E+000]
 [0.000000000000000E+000, 1.116750827024463E-014]
 [3.999999999994074E+000, 4.000000000000000E+000]

Minimum: [-1.500000000000001E+001, -1.49999999999073E+001]

Beispiel 12 : (f19, $x \in \mathbb{R}^2$)

(Nummer 57 in [7])

$$f_{19}(x) = \sum_{i=1}^{44} f_i(x)^2 \quad \text{mit}$$

$$f_i(x) = b_i - x_1 - (0.49 - x_1)e^{-x_2(a_i - 8)}$$

$$p_1(x) = -0.49x_2 + x_1x_2 + 0.09$$

 a_i, b_i : vgl. Anhang 9Startbox: $[x] = ([0.4, 1], [-4, 2])^T$.

Minimalstelle: [4.199526507577211E-001, 4.199526507578713E-001]
 [1.284845193623297E+000, 1.284845193626128E+000]

Minimum: [2.845966972283974E-002, 2.845966972298684E-002]

Beispiel 13 : (f24, $x \in \mathbb{R}^4$)

(Nummer 76 in [7])

$$f_{24}(x) = x_1^2 + 0.5x_2^2 + x_3^2 + 0.5x_4^2 + x_3(x_4 - x_1 + 1) - x_1 - 3x_2 - x_4$$

$$p_1(x) = x_1 + 2x_2 + x_3 + x_4 - 5 \quad p_2(x) = 3x_1 + x_2 + 2x_3 - x_4 - 4$$

$$p_3(x) = 1.5 - x_2 - 4x_3$$

Startbox: $[x] = ([0, 1], [0, 2.5], [0, 1], [0, 1])^T$.

Minimalstelle: [2.727272727268042E-001, 2.727272727276062E-001]
 [2.090909090908369E+000, 2.090909090909450E+000]
 [0.000000000000000E+000, 6.887388297406407E-029]
 [5.454545454542156E-001, 5.454545454551903E-001]

Minimum: [-4.681818181822184E+000, -4.6818181818179E+000]

Beispiel 14 : (f41, $x \in \mathbb{R}^4$) Modifiziertes Rosen-Suzuki-Problem (Nummer 264 in [16])

$$f_{41}(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$$

$$p_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 - x_3 - x_4 - 8$$

$$p_2(x) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 + x_1 - x_4 - 9$$

$$p_3(x) = 2x_1^2 + x_2^2 + x_3^2 + 2x_4^2 - x_2 - x_4 - 5$$

Die Zielfunktion und die Nebenbedingungen wurden umgeformt, damit die Intervallauswertungen kleineren Durchmesser haben. Mit der ursprünglichen Form benötigte der Algorithmus wesentlich mehr Funktionsauswertungen und damit eine längere Laufzeit.

$$f_{41}(x) = \left(x_1 - \frac{5}{2}\right)^2 + \left(x_2 - \frac{5}{2}\right)^2 + 2\left(x_3 - \frac{21}{4}\right)^2 + \left(x_4 + \frac{7}{2}\right)^2 - \frac{639}{8}$$

$$p_1(x) = \left(x_1 + \frac{1}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2 + \left(x_3 - \frac{1}{2}\right)^2 + \left(x_4 - \frac{1}{2}\right)^2 - 9$$

$$p_2(x) = \left(x_1 - \frac{1}{2}\right)^2 + 2x_2^2 + x_3^2 + 2\left(x_4 - \frac{1}{4}\right)^2 - \frac{75}{8}$$

$$p_3(x) = 2\left(x_1 + \frac{1}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2 + x_3^2 - x_4 - \frac{23}{4}$$

Startbox: $[x] = ([0, 1], [0, 1], [0, 2], [-1, 0])^T$.

Minimalstelle: [0.000000000000000E+000, 7.275957614183426E-012]
 [9.999999999967449E-001, 1.000000000000000E+000]
 [1.999999999998646E+000, 2.000000000000000E+000]
 [-1.000000000000000E+000, -9.99999999981810E-001]

Minimum: [-4.40000000003638E+001, -4.3999999999710E+001]

Beispiel 15 : (f45, $x \in \mathbb{R}^2$), Around-the-World Problem (Nummer 315 in [16])

$$f_{45}(x) = -x_2$$

$$p_1(x) = 1 - 2x_2 + x_1 \quad p_2(x) = x_1^2 + x_2^2$$

$$p_3(x) = 1 - x_1^2 - x_2^2$$

Startbox: $[x] = ([0, 1], [0, 1])^T$.

Minimalstelle: [5.99999999995276E-001, 6.00000000008621E-001]
 [7.99999999993536E-001, 8.00000000000000E-001]

Minimum: [-8.00000000000000E-001, -7.9999999999192E-001]

Beispiel 16 : $(f_{74}, x \in \mathbb{R}^2)$

(Nummer 23 in [7])

$$f_{74}(x) = x_1^2 + x_2^2$$

$$p_1(x) = 1 - x_1 - x_2 \quad p_2(x) = 1 - x_1^2 - x_2^2$$

$$p_3(x) = 9 - 9x_1^2 - x_2^2 \quad p_4(x) = x_2 - x_1^2$$

$$p_5(x) = x_1 - x_2^2$$

Startbox: $[x] = ([-50, 50], [-50, 50])^T$.

Minimalstelle: [9.999999999999997E-001, 1.0000000000000234E+000]

[9.999999999999997E-001, 1.0000000000000234E+000]

Minimum: [1.999999999999999E+000, 2.000000000000850E+000]

	f9	f10	f19	f24
n	2	2	2	4
FA_f	166	69	340	1621
GC_f	44	32	252	1996
HC_f	84	37	271	3564
FA_{p_1}	121	51	183	1664
GC_{p_1}	74	22	152	1132
HC_{p_1}	84	28	148	2048
FA_{p_2}	107	30	–	215
GC_{p_2}	82	0	–	112
HC_{p_2}	84	0	–	288
FA_{p_3}	–	46	–	128
GC_{p_3}	–	26	–	12
HC_{p_3}	–	28	–	48
MaxL	24	3	13	79
STU-time	1.24	0.46	47.89	21.9
time	0:00:01,920	0:00:00,720	0:01:13,760	0:00:33,730

	f18	f41	f45	f74
n	4	4	2	2
FA_f	212	737	63	448
GC_f	160	876	34	144
HC_f	312	1486	31	141
FA_{p_1}	62	455	56	517
GC_{p_1}	4	0	24	46
HC_{p_1}	16	0	28	56
FA_{p_2}	61	455	26	137
GC_{p_2}	0	4	0	40
HC_{p_2}	0	16	0	76
FA_{p_3}	83	680	37	143
GC_{p_3}	64	504	22	50
HC_{p_3}	176	816	24	88
FA_{p_4}	59	–	–	135
GC_{p_4}	0	–	–	62
HC_{p_4}	0	–	–	96
FA_{p_5}	84	–	–	129
GC_{p_5}	76	–	–	64
HC_{p_5}	192	–	–	96
FA_{p_6}	56	–	–	–
GC_{p_6}	4	–	–	–
HC_{p_6}	16	–	–	–
MaxL	10	33	2	19
STU-time	1.92	9.44	0.35	2.03
time	0:00:02,970	0:00:14,550	0:00:00,550	0:00:03,130

Beispiel 17 : (f16, $x \in \mathbb{R}^4$) Rosen-Suzuki-Problem

(Nummer 43 in [7])

$$f_{16}(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5(x_1 + x_2) - 21x_3 + 7x_4$$

$$p_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 - x_1 + x_2 - x_3 + x_4 - 8$$

$$p_2(x) = x_1^2 + 2x_2^2 + x_3^2 + x_4^2 - x_1 - x_4 - 10$$

$$p_3(x) = 2x_1^2 + x_2^2 + x_3^2 + 2x_4 - x_2 - x_4 - 5$$

Startbox: $[x] = ([-0.3, 0.2], [0.7, 1.3], [1.6, 2.3], [-1.4, -0.8])^T$.

Minimalstelle: [-1.830184051430775E-013, 5.472840844472763E-013]
 [9.99999999997574E-001, 1.000000000000326E+000]
 [1.99999999999826E+000, 2.00000000000195E+000]
 [-1.000000000000623E+000, -9.99999999986054E-001]

Minimum: [-4.40000000001751E+001, -4.39999999999969E+001]

Beispiel 18 : (f44, $x \in \mathbb{R}^4$)

(Nummer 277 in [16])

$$f_{44}(x) = b^T x$$

$$p_i(x) = b_i - \sum_{j=1}^4 a_{ij}x_j, \quad i = 1, \dots, 4$$

mit $b_i = -\sum_{j=1}^4 \frac{1}{i+j-1}$, $a_{ij} = \frac{1}{i+j-1}$ für $i, j = 1, \dots, 4$.

Startbox: $[x] = ([0, 1], [0, 1], [0, 1], [0, 1])^T$.

Minimalstelle: [9.9999999999999996E-001, 1.0000000000000000E+000]
 [9.9999999999999991E-001, 1.0000000000000000E+000]
 [9.9999999999999988E-001, 1.0000000000000000E+000]
 [9.9999999999999984E-001, 1.0000000000000000E+000]
 Minimum: [5.076190476190470E+000, 5.076190476190478E+000]

Beispiel 19 : (f35, $x \in \mathbb{R}^2$)

(Nummer 224 in [16])

$$f_{35}(x) = 2x_1^2 + x_2^2 - 48x_1 - 40x_2$$

$$p_1(x) = -x_1 - 3x_2 \quad p_2(x) = x_1 + 3x_2 - 18$$

$$p_3(x) = -x_1 - x_2 \quad p_4(x) = x_1 + x_2 - 8$$

Startbox: $[x] = ([0, 6], [0, 6])^T$.

Bei diesem Testproblem terminiert der Algorithmus bei den Genauigkeitsforderungen $\varepsilon_x = \varepsilon_F = 10^{-12}$ mit 21 disjunkten Boxen in der Resultatsliste. Hier wird nur die Box angegeben, die die exakt bekannte Lösung dieses theoretischen Problems enthält. Dies ist die erste Box in der Resultatsliste.

Minimalstelle: [3.999999999991209E+000, 4.000000000004725E+000]
 [3.99999999995250E+000, 4.000000000008766E+000]
 Minimum: [-3.040000000007561E+002, -3.0399999999997E+002]

Beispiel 20 : (f85, $x \in \mathbb{R}^5$)

(Nummer 359 in [16])

$$f_{85}(x) = -\left(\sum_{i=1}^5 A_i x_i - 24345\right)$$

$$p_1(x) = x_2 - 2.4x_1 \quad p_2(x) = 1.2x_1 - x_2$$

$$p_3(x) = x_3 - 60x_1 \quad p_4(x) = 20x_1 - x_3$$

$$p_5(x) = x_4 - 9.3x_1 \quad p_6(x) = 9x_1 - x_4$$

$$p_7(x) = x_5 - 7x_1 \quad p_8(x) = 6.5x_1 - x_5$$

$$p_9(x) = -\sum_{i=1}^5 B_i x_i \quad p_{10}(x) = -\sum_{i=1}^5 C_i x_i$$

$$p_{11}(x) = -\sum_{i=1}^5 D_i x_i \quad p_{12}(x) = \sum_{i=1}^5 B_i x_i - 294000$$

$$p_{13}(x) = \sum_{i=1}^5 C_i x_i - 294000 \quad p_{14}(x) = \sum_{i=1}^5 D_i x_i - 294000$$

A, B, C, D : vgl. Anhang Testdaten

Startbox: $[x] = ([4, 5], [10, 11], [272, 273], [42, 43], [31, 32])^T$.

Minimalstelle: [4.571428571428380E+000, 4.571428571428616E+000]
 [1.097142857142577E+001, 1.097142857142925E+001]
 [2.720000000000000E+002, 2.720000000004699E+002]
 [4.251428571428353E+001, 4.251428571428681E+001]
 [3.19999999999902E+001, 3.200000000000000E+001]
 Minimum: [-5.320825515722319E+006, -5.320825515719634E+006]

Beispiel 21 : (f87, $x \in \mathbb{R}^5$)

(Nummer 361 in [16])

$$f_{87}(x) = -(x_1(A_1 + \sum_{i=1}^5 A_i x_i) - 24345)$$

$$p_1(x) = -(x_1(B_1 + \sum_{i=2}^5 B_i x_i)) \quad p_2(x) = -(x_1(C_1 + \sum_{i=2}^5 C_i x_i))$$

$$p_3(x) = -(x_1(D_1 + \sum_{i=2}^5 D_i x_i)) \quad p_4(x) = p_1(x) - 29400$$

$$p_5(x) = p_2(x) - 29400 \quad p_6(x) = p_3(x) - 277200$$

A, B, C, D : gleiche Testdaten wie bei f_{85} .

Startbox: $[x] = ([0, 1], [1.2, 2.4], [20, 60], [9, 9.3], [6.5, 7])^T$.

Minimalstelle: [6.739403050471650E-002, 6.739403050487106E-002]
 [2.400000000000000E+000, 2.400000000000000E+000]
 [2.000000000000000E+001, 2.000000000000000E+001]
 [9.300000000000000E+000, 9.300000000000000E+000]
 [7.000000000000000E+000, 7.000000000000000E+000]

Minimum: [-1.526215485888275E+004, -1.526215485879129E+004]

Beispiel 22 : (f82, $x \in \mathbb{R}^6$)

(Nummer 2.4 in [2])

$$f_{82}(x) = 6.5x_1 - 0.5x_1^2 - x_2 - 2x_3 - 3x_4 - 2x_5 - x_6$$

$$p_1(x) = \sum_{j=1}^5 a_{1j}x_j - 16 \quad p_2(x) = \sum_{j=1}^5 a_{2j}x_j + 1$$

$$p_3(x) = \sum_{j=1}^5 a_{3j}x_j - 24 \quad p_4(x) = \sum_{j=1}^5 a_{4j}x_j - 12$$

$$p_5(x) = \sum_{j=1}^5 a_{5j}x_j - 3$$

$$\text{mit } A = \begin{pmatrix} 1 & 2 & 8 & 1 & 3 & 5 \\ -8 & -4 & -2 & 2 & 4 & -1 \\ 2 & 0.5 & 0.2 & -3 & -1 & -4 \\ 0.2 & 2 & 0.1 & -4 & 2 & 2 \\ -0.1 & -0.5 & 2 & 5 & -5 & 3 \end{pmatrix}$$

Startbox: $([0, 1], [0, 10], [0, 1], [0, 1], [0, 1], [0, 2])^T$.

Minimalstelle: [0.000000000000000E+000, 2.705607603806137E-031]
 [5.999999999994091E+000, 6.00000000002842E+000]
 [0.000000000000000E+000, 1.051679796793147E-012]
 [9.99999999994712E-001, 1.000000000000000E+000]
 [9.99999999981992E-001, 1.000000000000000E+000]
 [0.000000000000000E+000, 5.820800133327761E-014]

Minimum: [-1.100000000000501E+001, -1.09999999999708E+001]

	f16	f44	f35	f85	f87	f82
n	4	4	2	5	5	6
FA_f	6763	5	355	583	78	2123
GC_f	7772	8	196	685	145	3216
HC_f	12036	32	191	1540	491	8607
FA_{p_1}	6605	6	241	398	27	2170
GC_{p_1}	3832	12	0	0	0	1992
HC_{p_1}	5216	32	0	0	0	5436
FA_{p_2}	2864	4	269	407	27	99
GC_{p_2}	1156	12	30	5	0	30
HC_{p_2}	2032	32	28	25	0	180
FA_{p_3}	4071	4	229	398	28	80
GC_{p_3}	4124	12	0	0	5	0
HC_{p_3}	5216	32	0	0	25	0
FA_{p_4}	–	4	319	398	27	339
GC_{p_4}	–	12	128	0	0	756
HC_{p_4}	–	32	132	0	0	3204
FA_{p_5}	–	–	–	476	27	295
GC_{p_5}	–	–	–	130	0	726
HC_{p_5}	–	–	–	350	0	2628
FA_{p_6}	–	–	–	366	27	–
GC_{p_6}	–	–	–	0	0	–
HC_{p_6}	–	–	–	0	0	–
FA_{p_7}	–	–	–	422	–	–
GC_{p_7}	–	–	–	110	–	–
HC_{p_7}	–	–	–	350	–	–
FA_{p_8}	–	–	–	347	–	–
GC_{p_8}	–	–	–	0	–	–
HC_{p_8}	–	–	–	0	–	–
FA_{p_9}	–	–	–	451	–	–
GC_{p_9}	–	–	–	265	–	–
HC_{p_9}	–	–	–	700	–	–
$FA_{p_{10}}$	–	–	–	244	–	–
$GC_{p_{10}}$	–	–	–	0	–	–
$HC_{p_{10}}$	–	–	–	0	–	–
$FA_{p_{11}}$	–	–	–	377	–	–
$GC_{p_{11}}$	–	–	–	445	–	–
$HC_{p_{11}}$	–	–	–	1000	–	–
$FA_{p_{12}}$	–	–	–	75	–	–
$GC_{p_{12}}$	–	–	–	20	–	–
$HC_{p_{12}}$	–	–	–	75	–	–
$FA_{p_{13}}$	–	–	–	171	–	–
$GC_{p_{13}}$	–	–	–	360	–	–
$HC_{p_{13}}$	–	–	–	925	–	–
$FA_{p_{14}}$	–	–	–	55	–	–
$GC_{p_{14}}$	–	–	–	5	–	–
$HC_{p_{14}}$	–	–	–	25	–	–
MaxL	498	1	15	35	9	128
STU-time	99.14	0.42	1.74	11.01	1.67	45.36
time	0:02:32,690	0:00:00,660	0:00:02,690	0:00:16,970	0:00:02,580	0:01:09,860

Beispiel 23 : (f66, $x \in \mathbb{R}^2$)

(Nummer 4.6 in [2])

$$f_{66}(x) = -x_1 - x_2$$

$$p_1(x) = x_2 - 2x_1^4 + 8x_1^3 - 8x_1^2 - 2 \quad p_2(x) = x_2 - 4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 - 36$$

Startbox: $[x] = ([0, 3], [0, 4])^T$.

Minimalstelle: [2.329520197477190E+000, 2.329520197478016E+000]
 [3.178493074116024E+000, 3.178493074117737E+000]

Minimum: [-5.508013271595753E+000, -5.508013271594483E+000]

Beispiel 24 : (f67, $x \in \mathbb{R}^2$)

(Nummer 59 in [7])

$$\begin{aligned} f_{67}(x) = & -75.196 + x_1(3.8112 - 0.030234x_2 - 3.405 \cdot 10^{-4}x_2^2 + 1.6638 \cdot 10^{-6}x_2^3) \\ & + x_1^3(0.0020567 + 6.3 \cdot 10^{-8}x_2^2 - 7 \cdot 10^{-10}x_2^3 - 3.5256 \cdot 10^{-5}x_2) \\ & + x_1^4(-1.0345 \cdot 10^{-5} + 2.266 \cdot 10^{-7}x_2) \\ & + x_2(6.8306 + 1.28134 \cdot 10^{-3}x_1^2) \\ & + x_2^2(-0.25645 + 5.2375 \cdot 10^{-6}x_1^2) \\ & + 0.0034604x_2^3 - 1.3514 \cdot 10^{-5}x_2^4 + 28.106/(x_2 + 1) + 2.8673e^{0.0005x_1x_2} \end{aligned}$$

$$p_1(x) = 700 - x_1x_2 \quad p_2(x) = 0.0008x_1^2 - x_2$$

$$p_3(x) = 5(x_1 - 55) - (x_2 - 50)^2$$

Startbox: $[x] = ([0, 25], [40, 65])^T$.

Minimalstelle: [1.240621328548685E+001, 1.240621328559942E+001]
 [5.642334077979176E+001, 5.642334078027904E+001]

Minimum: [1.336911453471838E+001, 1.336911455437590E+001]

Beispiel 25 : f48(, $x \in \mathbb{R}^2$)

(Nummer 326 in [16])

$$f_{48}(x) = x_1^2 + x_2^2 - 16x_1 - 10x_2$$

$$p_1(x) = x_1^2 - 6x_1 + 4x_2 + 11 \quad p_2(x) = 3x_2 + e^{x_1-3} - x_1x_2 - 1$$

Startbox: $[x] = ([0, 10], [0, 10])^T$.

Minimalstelle: [5.239609115500730E+000, 5.239609115503008E+000]
 [3.746037752437124E+000, 3.746037752441509E+000]

Minimum: [-7.980782084654071E+001, -7.980782084647142E+001]

Beispiel 26 : (f52, $x \in \mathbb{R}^2$) Journal bearing design, case 2

(Nummer 330 in [16])

$$f_{52}(x) = 0.1 \left[\frac{x_1}{x_2} (0.44x_1^2 + 0.592 \frac{1}{x_2}) + \frac{10}{x_1} \right]$$

$$p_1(x) = 8.62 \frac{x_2^3}{x_1} - 1$$

Startbox: $[x] = ([0.1, 5], [0.1, 5])^T$.

Minimalstelle: [1.286677505152305E+000, 1.286677505154838E+000]
[5.304618401334966E-001, 5.304618401339437E-001]

Minimum: [1.620583322398463E+000, 1.620583322401459E+000]

Beispiel 27 : (**f62**, $x \in \mathbb{R}^2$)

(Levy-Gomez, 1985 aus [12], S.195)

$$f_{62}(x) = 0.1(x_1^2 + x_2^2)$$

$$p_1(x) = 2 \sin(2\pi x_2) - \sin(4\pi x_1)$$

Startbox: $[x] = ([-10, 20], [-10, 20])^T$.

Minimalstelle: [-5.421010862429942E-020, 4.680566537021410E-007]
[-4.442110655169645E-007, 4.336808689942018E-019]

Minimum: [0.000000000000000E+000, 5.877471754115372E-040]

Das Problem hat 24 lokale, aber nur eine globale Minimalstelle. Der zulässige Bereich ist nicht zusammenhängend.

	f66	f67	f48	f52	f62
n	2	2	2	2	2
FA _f	1454	204	220	467	109
GC _f	454	150	148	260	30
HC _f	276	161	145	269	34
FA _{p₁}	1947	188	214	480	85
GC _{p₁}	330	102	98	172	26
HC _{p₁}	216	104	88	172	28
FA _{p₂}	1637	30	124	–	–
GC _{p₂}	382	0	88	–	–
HC _{p₂}	216	0	88	–	–
FA _{p₃}	–	30	–	–	–
GC _{p₃}	–	0	–	–	–
HC _{p₃}	–	0	–	–	–
MaxL	24	9	13	21	7
STU-time	7.31	4.74	1.77	3.13	0.81
time	0:00:11,260	0:00:07,310	0:00:02,740	0:00:04,830	0:00:01,260

Beispiel 28 : (**f7**, $x \in \mathbb{R}^2$)

(Nummer 13 in [7])

$$f_7(x) = (x_1 - 2)^2 + x_2^2$$

$$p_1(x) = (1 - x_1)^3 + x_2$$

Startbox: $[x] = ([0, 3], [0, 1])^T$.

Minimalstelle: [9.999999999969355E-001, 1.00000000000364E+000]
[0.000000000000000E+000, 3.289948170347719E-035]

Minimum: [9.99999999992726E-001, 1.000000000005433E+000]

Beispiel 29 : (f11, $x \in \mathbb{R}^3$)

(Nummer 29 in [7])

$$f_3(x) = -x_1x_2x_3$$

$$p_1(x) = x_1^2 + 2x_2^2 + 4x_3^2 - 48$$

Startbox: $[x] = ([-4, 4], [-4, 4], [-4, 4])^T$.

Minimalstellen: 1. [-4.000000000000000E+000, -3.9999999999995215E+000]
 [2.828427124745543E+000, 2.828427124748568E+000]
 [-2.000000000001502E+000, -1.99999999999386E+000]
 2. [3.999999999995210E+000, 4.000000000000000E+000]
 [-2.828427124748571E+000, -2.828427124745519E+000]
 [-2.000000000001458E+000, -1.99999999999386E+000]
 3. [3.999999999995210E+000, 4.000000000000000E+000]
 [2.828427124745519E+000, 2.828427124748571E+000]
 [1.99999999999386E+000, 2.000000000001458E+000]
 4. [-4.000000000000000E+000, -3.999999999995407E+000]
 [-2.828427124748473E+000, -2.828427124745631E+000]
 [1.99999999999411E+000, 2.000000000001133E+000]

Minimum: [-2.262741699800554E+001, -2.262741699796951E+001]

Dieses theoretische Problem hat vier globale Minimalstellen.

Beispiel 30 : (f13, $x \in \mathbb{R}^3$)

(Nummer 56 in [7])

$$f_{13}(x) = (x_1 - 1)(x_1 - 2)(x_1 - 3) + x_3$$

$$p_1(x) = x_1^2 + x_2^2 - x_3^2 \quad p_2(x) = -x_1^2 - x_2^2 - x_3^2 + 4$$

Startbox: $[x] = ([0, 5], [0, 5], [0, 5])^T$.

Minimalstelle: [0.000000000000000E+000, 8.702650494982118E-027]
 [1.414213562370515E+000, 1.414213562382256E+000]
 [1.414213562373095E+000, 1.414213562377675E+000]

Minimum: [-4.585786437626906E+000, -4.585786437624968E+000]

Beispiel 31 : (f30, $x \in \mathbb{R}^5$)

(Nummer 2.1 in [2])

$$f_{30}(x) = c^T x - 0.5x^T Q x$$

$$p_1(x) = 20x_1 + 12x_2 + 11x_3 + 7x_4 + 4x_5 - 40$$

mit $Q = 100E_5$ und $c = (42, 44, 45, 47, 47.5)^T$.

Startbox: $[0, 1,], i = 1, \dots, 5$.

Minimalstelle: [1.000000000000000E+000, 1.000000000000000E+000]
 [1.000000000000000E+000, 1.000000000000000E+000]
 [0.000000000000000E+000, 0.000000000000000E+000]
 [1.000000000000000E+000, 1.000000000000000E+000]
 [0.000000000000000E+000, 0.000000000000000E+000]

Minimum: [-1.700000000000000E+001, -1.700000000000000E+001]

Beispiel 32 : (**f33**, $x \in \mathbb{R}^2$)

(Nummer 221 in [16])

$$f_{33}(x) = -x_1$$

$$p_1(x) = x_2 - (1 - x_1)^3$$

Startbox: $[x] = ([0, 5], [0, 5])^T$.

Minimalstelle: [9.999999999999778E-001, 1.000000000000231E+000]
 [0.000000000000000E+000, 5.044156728932130E-036]

Minimum: [-1.000000000000231E+000, -9.602909959809863E-001]

	f7	f11	f13	f30	f33
n	2	3	3	5	2
FA_f	70	3067	561	1605	59
GC_f	26	2862	168	2195	28
HC_f	48	3904	267	6774	52
FA_{p_1}	63	2351	542	863	60
GC_{p_1}	46	1878	90	590	46
HC_{p_1}	48	2484	180	2025	48
FA_{p_2}	–	–	203	–	–
GC_{p_2}	–	–	84	–	–
HC_{p_2}	–	–	162	–	–
MaxL	7	138	41	315	6
STU-time	0.5	20.72	2.31	32.03	0.42
time	0:00:00,770	0:00:31,910	0:00:03,570	0:00:49,330	0:00:00,660

Beispiel 33 : (**f34**, $x \in \mathbb{R}^2$)

(Nummer 223 in [16])

$$f_{34}(x) = -x_1$$

$$p_1(x) = -e^{e^{x_1}} \quad p_2(x) = e^{e^{x_1}} - x_2$$

Startbox: $[x] = ([0, 5], [0, 10])^T$.

Minimalstelle: [8.340324452475039E-001, 8.340324452479559E-001]
 [9.999999999990422E+000, 1.000000000000000E+001]

Minimum: [-8.340324452479559E-001, -8.340324452477299E-001]

Beispiel 34 : (**f38**, $x \in \mathbb{R}^3$)

(Nummer 249 in [16])

$$f_{38}(x) = x_1^2 + x_2^2 + x_3^2$$

$$p_1(x) = 1 - x_1^2 - x_2^2$$

Startbox: $[x] = ([1, 2], [0, 1], [0, 1])^T$.

Minimalstelle: [1.000000000000000E+000, 1.000000000000340E+000]
 [0.000000000000000E+000, 2.380624967122432E-007]
 [0.000000000000000E+000, 2.380624442165046E-007]

Minimum: [1.000000000000000E+000, 1.00000000000028E+000]

Beispiel 35 : (f46, $x \in \mathbb{R}^2$)

(Nummer 323 in [16])

$$f_{46}(x) = x_1^2 + x_2^2 - 4x_1 + 4$$

$$p_1(x) = x_2 - x_1 - 2p_2(x) = x_1^2 - x_2 + 1$$

Startbox: $[x] = ([0, 2], [0, 2])^T$.

Minimalstelle: [5.535737822174610E-001, 5.535737822179146E-001]
 [1.306443932358545E+000, 1.306443932359117E+000]

Minimum: [3.798944551883344E+000, 3.798944551885167E+000]

Beispiel 36 : (f54, $x \in \mathbb{R}^3$)

(Nummer 337 in [16])

$$f_{54}(x) = 9x_1^2 + x_2^2 + 9x_3^2$$

$$p_1(x) = 1 - x_1x_2$$

Startbox: $[x] = ([0, 1], [1, 2], [-1, 1])^T$.

Minimalstelle: [5.773502691874385E-001, 5.773502691906883E-001]
 [1.732050807565689E+000, 1.732050807575439E+000]
 [-3.390266144573530E-014, 2.600631080253432E-014]

Minimum: [5.99999999966224E+000, 6.000000000000002E+000]

Beispiel 37 : (f56, $x \in \mathbb{R}^3$) Pascal-Problem

(Nummer 340 in [16])

$$f_{56}(x) = -x_1x_2x_3$$

$$p_1(x) = x_1 + 2x_2 + 2x_3 - 1.8$$

Startbox: $[x] = ([0, 1], [0, 1], [0, 1])^T$.

Minimalstelle: [5.99999999994726E-001, 6.000000000010119E-001]
 [2.99999999995899E-001, 3.00000000002342E-001]
 [2.99999999996447E-001, 3.00000000004144E-001]

Minimum: [-5.40000000020783E-002, -5.39999999999996E-002]

Beispiel 38 : (f57, $x \in \mathbb{R}^3$)

(Nummer 341 in [16])

$$f_{57}(x) = -x_1x_2x_3 (= f56)$$

$$p_1(x) = x_1^2 + 2x_2^2 + 4x_3^2 - 48$$

Startbox: $[x] = ([0, 5], [0, 3], [0, 3])^T$.

Minimalstelle: [3.99999999998701E+000, 4.00000000001299E+000]
 [2.828427124745272E+000, 2.828427124747108E+000]
 [1.99999999999351E+000, 2.00000000000649E+000]

Minimum: [-2.262741699799156E+001, -2.260157754064949E+001]

	f34	f38	f46	f54	f56	f57
n	2	3	2	3	3	3
FA_f	150	11	262	1445	1818	718
GC_f	32	12	114	921	1308	603
HC_f	38	21	108	1326	1686	747
FA_{p_1}	127	9	249	1523	1843	725
GC_{p_1}	0	3	0	549	858	387
HC_{p_1}	0	9	0	864	1080	459
FA_{p_2}	150	–	306	–	–	–
GC_{p_2}	30	–	70	–	–	–
HC_{p_2}	32	–	80	–	–	–
MaxL	4	1	8	63	53	37
STU-time	1.0	0.1	1.03	6.81	8.38	4.1
time	0:00:01,540	0:00:00,160	0:00:01,590	0:00:10,490	0:00:12,910	0:00:06,320

Beispiel 39 : (**f63**, $x \in \mathbb{R}^2$)

(Zowe, 1985, aus [12], S.197)

$$f_{63}(x) = (x_1 - 2)^2 + (x_2 - 1)^2$$

$$p_1(x) = x_1^2 - x_2 \quad p_2(x) = x_1 + x_2 - 2$$

Startbox: $([-10, 20], [-10, 20])^T$.

Minimalstelle: [9.9999999999995833E-001, 1.0000000000000000E+000]
 [9.9999999999994331E-001, 1.0000000000000167E+000]

Minimum: [9.99999999999996E-001, 1.0000000000000443E+000]

Beispiel 40 : (**f69**, $x \in \mathbb{R}^3$)

(Nummer 36 in [7])

$$f_{69}(x) = -x_1 x_2 x_3$$

$$p_1(x) = x_1 + 2x_2 + 2x_3 - 72$$

Startbox: $([0, 20], [0, 11], [0, 42])^T$.

Minimalstelle: [1.999999999999459E+001, 2.0000000000000000E+001]
 [1.099999999999984E+001, 1.1000000000000000E+001]
 [1.499999999999878E+001, 1.5000000000000368E+001]

Minimum: [-3.3000000000000809E+003, -3.29999999999720E+003]

Beispiel 41 : (**f70**, $x \in \mathbb{R}^3$)

(Nummer 34 in [7])

$$f_{70}(x) = -x_1$$

$$p_1(x) = e^{x_1} - x_2 \quad p_2(x) = e^{x_2} - x_3$$

Startbox: $([0, 100], [0, 100], [0, 10])^T$.

Minimalstelle: [8.340324452472930E-001, 8.340324452479559E-001]
 [2.302585092990993E+000, 2.302585092994046E+000]
 [9.99999999972857E+000, 1.0000000000000000E+001]

Minimum: [-8.340324452479559E-001, -8.340324452476244E-001]

Beispiel 42 : (f78, $x \in \mathbb{R}^2$)

(Nummer 18 in [7])

$$f_{78}(x) = 0.01x_1^2 + x_2^2$$

$$p_1(x) = 25 - x_1x_2 \quad p_2(x) = 25 - x_1^2 - x_2^2$$

Startbox: $([2, 50], [0, 50])^T$.

Folgende Einschließungen enthalten Kandidaten für die globale Minimalstelle:

```
[ 1.581138830084188E+001, 1.581138830084192E+001]
[ 1.581138830084187E+000, 1.581138830084192E+000]
[ 1.581138830084186E+001, 1.581138830084187E+001]
[ 1.581138830084193E+000, 1.581138830084194E+000]
[ 1.581138830084193E+001, 1.581138830084194E+001]
[ 1.581138830084186E+000, 1.581138830084186E+000]
```

Minimum: [4.99999999999983E+000, 5.000000000000001E+000]

Beispiel 43 : (f79, $x \in \mathbb{R}^2$)

(Nummer 17 in [7])

$$f_{79}(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$p_1(x) = x_1 - x_2^2 \quad p_2(x) = x_2 - x_1^2$$

Startbox: $([-0.5, 0.5], [-2, 1])^T$.

Minimalstelle: [-6.049752990635100E-015, 6.906121893384664E-015]
 [-2.895065827955220E-008, 1.393215370664152E-008]

Minimum: [9.99999999999860E-001, 1.000000000000007E+000]

Beispiel 44 : (f80, $x \in \mathbb{R}^2$)

(Nummer 15 in [7])

$$f_{80}(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 = f_{79}$$

$$p_1(x) = 1 - x_1x_2 \quad p_2(x) = -x_1 - x_2^2$$

Startbox: $([-2, 0.5], [-2, 4])^T$.

Minimalstelle: [4.999999999999999E-001, 5.000000000000000E-001]
 [2.000000000000000E+000, 2.00000000000585E+000]

Minimum: [3.065000000000000E+002, 3.06500000001903E+002]

	f63	f69	f70	f78	f79	f80
n	2	3	3	2	2	2
FA_f	100	745	361	234	104	184
GC_f	50	345	84	140	48	96
HC_f	49	432	162	134	41	93
FA_{p_1}	93	740	368	206	81	163
GC_{p_1}	34	177	108	92	30	70
HC_{p_1}	0	234	162	92	32	72
FA_{p_2}	58	–	119	43	40	37
GC_{p_2}	34	–	102	8	24	16
HC_{p_2}	36	–	153	16	32	24
MaxL	3	34	18	8	4	10
STU-time	0.53	2.53	2.20	1.07	0.46	1.0
time	0:00:00,820	0:00:03,900	0:00:03,400	0:00:01,650	0:00:00,710	0:00:01,540

Beispiel 45 : (**f5**, $x \in \mathbb{R}^2$)

(Nummer 10 in [7])

$$f_5(x) = x_1 - x_2$$

$$p_1(x) = 3x_1^2 - 2x_1x_2 + x_2^2 - 1$$

Startbox: $([-2, 1], [0, 3])^T$.

Minimalstelle: [-5.405663492274195E-013, 3.657455326388037E-013]
 [9.9999999999993643E-001, 1.000000000000366E+000]

Minimum: [-1.000000000000907E+000, -9.99999999999893E-001]

Beispiel 46 : (**f6**, $x \in \mathbb{R}^2$)

(Nummer 11 in [7])

$$f_6(x) = (x_1 - 5)^2 + x_2^2 - 25$$

$$p_1(x) = x_1^2 - x_2$$

Startbox: $([-5, 5], [-5, 5])^T$.

Minimalstelle: [1.234772825052364E+000, 1.234772825053767E+000]
 [1.524663929488586E+000, 1.524663929491187E+000]

Minimum: [-8.498464223162838E+000, -8.498464223154560E+000]

Beispiel 47 : (**f12**, $x \in \mathbb{R}^3$)

(Nummer 30 in [7])

$$f_{12}(x) = x_1^2 + x_2^2 + x_3^2$$

$$p_1(x) = 1 - x_1^2 - x_2^2$$

Startbox: $([1, 10], [-10, 10], [-10, 10])^T$.

Minimalstelle: [1.000000000000000E+000, 1.000000000000000E+000]
 [0.000000000000000E+000, 0.000000000000000E+000]
 [0.000000000000000E+000, 0.000000000000000E+000]

Minimum: [1.000000000000000E+000, 1.000000000000000E+000]

Beispiel 48 : (**f32**, $x \in \mathbb{R}^2$)

(Nummer 215 in [16])

$$f_{32}(x) = x_2$$

$$p_1(x) = x_1^2 - x_2$$

Startbox: $([0, 5], [-5, 5])^T$.

Minimalstelle: [0.000000000000000E+000, 1.334038669272764E-006]
 [-6.673754536309278E-013, 7.824851877558103E-013]

Minimum: [-6.673754536309278E-013, 3.912425938779052E-013]

Beispiel 49 : (f59, $x \in \mathbb{R}^4$)

(Nummer 354 in [16])

$$f_{59}(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 10(x_1 - x_4)^4$$

$$p_1(x) = 1 - x_1 - x_2 - x_3 - x_4$$

Startbox: $([0, 1], [-1, 0], [0, 1], [0, 1])^T$.

Minimalstelle: [5.033179336656615E-001, 5.033179336661978E-001]
 [-4.556116733569570E-002, -4.556116733561033E-002]
 [2.358189842057176E-001, 2.358189842060575E-001]
 [3.064242494636646E-001, 3.064242494639429E-001]
 Minimum: [1.137838489908149E-001, 1.137838489914485E-001]

Beispiel 50 : (f64, $x \in \mathbb{R}^1$)

(Nummer 4.1 aus [2])

$$f_{64}(x) = x_1^6 - \frac{52}{25}x_1^5 + \frac{39}{80}x_1^4 + \frac{71}{10}x_1^3 - \frac{79}{20}x_1^2 - x_1 + \frac{1}{10}$$

Startbox: $([-2, 11])$.

Minimalstelle: [-1.191299814188635E+000, -1.191299814187387E+000]
 Minimum: [-7.487312364951846E+000, -7.487312364902355E+000]

	f5	f6	f12	f32	f59	f64
n	2	2	3	2	4	1
FA_f	492	209	10	205	5419	69
GC_f	232	134	12	44	5604	32
HC_f	228	134	27	50	11088	43
FA_{p_1}	542	215	8	265	4751	–
GC_{p_1}	172	92	3	32	2972	–
HC_{p_1}	156	92	9	44	5984	–
MaxL	13	7	1	7	291	4
STU-time	1.81	0.96	0.11	0.5	63.33	0.5
time	0:00:02,800	0:00:01,480	0:00:00,170	0:00:00,770	0:01:37,540	0:00:00,770

Beispiel 51 : (f14, $x \in \mathbb{R}^3$) Beales Problem

(Nummer 35 in [7])

$$f_{14}(x) = 9 - 6x_2 - 4x_3 + 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1(x_2 + x_3 - 4)$$

$$p_1(x) = x_1 + x_2 + 2x_3 - 3$$

Startbox: $([0, 2], [0, 2], [0, 2])^T$.

Minimalstelle: [1.33333333333332E+000, 1.33333333333337E+000]
 [7.77777777777758E-001, 7.77777777777787E-001]
 [4.44444444444423E-001, 4.44444444444453E-001]
 Minimum: [1.111111111110425E-001, 1.11111111111125E-001]

Um diese Einschließung zu berechnen, wurde mit den Genauigkeitsforderungen $\varepsilon_x = \varepsilon_F = 10^{-15}$ gearbeitet. Bei den Forderungen $\varepsilon_x = \varepsilon_F = 10^{-12}$ sind noch zwei Elemente in der Resultatsliste enthalten, die dicht zusammenliegen.

Beispiel 52 : (**f21**, $x \in \mathbb{R}^3$)

(Nummer 65 in [7])

$$f_{21}(x) = (x_1 - x_2)^2 + \frac{(x_1 + x_2 - 10)^2}{9} + (x_3 - 5)^2$$

$$p_1(x) = x_1^2 + x_2^2 + x_3^2 - 48$$

Startbox: $([-4.5, 4.5], [-4.5, 4.5], [-5, 5])^T$.

Minimalstelle: [3.650461725213026E+000, 3.650461725213045E+000]
 [3.650461725213027E+000, 3.650461725213042E+000]
 [4.620417555320001E+000, 4.620417555320019E+000]

Minimum: [9.535288568047657E-001, 9.535288568047832E-001]

Auch für dieses Testbeispiel sind Genauigkeitsforderungen $\varepsilon_x = \varepsilon_F = 10^{-15}$ notwendig, damit der Algorithmus mit nur einem Element in der Resultatsliste terminiert.

Beispiel 53 : (**f29**, $x \in \mathbb{R}^2$)

(aus [10])

$$f_{29}(x) = 0.5x_1^2 - 0.5x_2^2 - 0.1x_1$$

$$p_1(x) = -x_1 + 2x_2 \quad p_2(x) = -x_1 - 2x_2$$

Startbox: $([-50, 50], [-50, 50])^T$.

Minimalstellen: 1. [1.33333333332880E-001, 1.33333333334496E-001]
 [-6.666666666672480E-002, -6.666666666664398E-002]

2. [1.33333333332264E-001, 1.33333333333641E-001]
 [6.666666666660859E-002, 6.666666666671869E-002]

Minimum: [-6.666666666688224E-003, -6.66666666666664E-003]

Das Testproblem hat 2 globale Minimalstellen im Startbereich.

Beispiel 54 : (**f47**, $x \in \mathbb{R}^2$)

(aus [10])

$$f_{47}(x) = -(x_1 - 20)^2 - (x_2 - 10)^2$$

$$p_1(x) = 2x_2 - x_1 - 20 \quad p_2(x) = x_1^2 - (x_2 - 10)^2 - 500$$

Startbox: $([0, 20], [0, 20])^T$.

Minimalstelle: [0.000000000000000E+000, 0.000000000000000E+000]
 [0.000000000000000E+000, 0.000000000000000E+000]

Minimum: [-5.000000000000000E+002, -5.000000000000000E+002]

Beispiel 55 : (**f1**, $x \in \mathbb{R}^2$)

(aus [5])

$$f_1(x) = x_1$$

$$p_1(x) = x_1^2 + x_2^2 - 1 \quad p_2(x) = x_1^2 - x_2$$

Startbox: $([-100, 100], [-100, 100])^T$.

Minimalstelle: [-7.861513777574234E-001, -7.861513777569793E-001]
 [6.180339887487073E-001, 6.180339887508985E-001]

Minimum: [-7.861513777574234E-001, -7.861513777571597E-001]

Aufwandsübersicht

	f14	f21	f29	f47	f1
n	3	3	2	2	2
FA_f	2370	1065	750	33	324
GC_f	1767	810	414	22	96
HC_f	2482	1086	409	24	87
FA_{p_1}	2265	987	496	20	412
GC_{p_1}	933	498	116	6	80
HC_{p_1}	1305	684	116	8	80
FA_{p_2}	–	–	371	16	156
GC_{p_2}	–	–	126	0	68
HC_{p_2}	–	–	168	0	84
MaxL	95	60	15	1	7
STU-time	16.16	6.66	3.20	0.18	1.0
time	0:00:24,890	0:00:10,270	0:00:04,940	0:00:00,280	0:00:01,540

Beispiel 56 : (**f49**, $x \in \mathbb{R}^2$)

(aus [10])

$$f_{49}(x) = -x_1 + x_2(x_1 - 1)$$

$$p_1(x) = -6x_1 + 8x_2 - 3 \quad p_2(x) = 3x_1 - x_2 - 3$$

Startbox: $([0, 20], [-10, 5])^T$.

Minimalstelle: [1.1666666666666666E+000, 1.1666666666666667E+000]
 [4.9999999999999992E-001, 5.00000000000000013E-001]

Minimum: [-1.0833333333333334E+000, -1.0833333333333333E+000]

Bei Genauigkeitsforderungen $\varepsilon_x = \varepsilon_F = 10^{-12}$ terminiert das Verfahren mit zwei Elementen in der Resultatsliste. Deshalb wurde für die Berechnung der exakten Einschließung $\varepsilon_x = \varepsilon_F = 10^{-15}$ gefordert.

Beispiel 57 :(**f31**, $x \in \mathbb{R}^3$)

(aus [10])

$$f_{31}(x) = -x_1^2 - x_2^2 - (x_3 - 1)^2$$

$$p_1(x) = x_1 + x_2 - x_3$$

$$p_2(x) = -x_1 + x_2 - x_3$$

$$p_3(x) = -6x_1 + x_2 + x_3 - 1.9$$

$$p_4(x) = 12x_1 + 5x_2 + 12x_3 - 22.8$$

$$p_5(x) = 12x_1 + 12x_2 + 7x_3 - 17.1$$

Startbox: $([-20, 20], [0, 20], [-20, 20])^T$.

Minimalstelle: [-2.0000000000000000E+001, -2.0000000000000000E+001]
 [1.6666666662669387E-001, 1.6666666666666667E-001]
 [2.0000000000000000E+001, 2.0000000000000000E+001]

Minimum: [-7.6102777777777778E+002, -7.61027777777777711E+002]

Beispiel 58 : (**f15**, $x \in \mathbb{R}^2$)

(aus [10])

$$\begin{aligned}f_{15}(x) &= -x_1^2 - x_2^2 \\p_1(x) &= x_1 + 4x_2 - 5\end{aligned}$$

Startbox: $([0, 1], [0, 10])^T$.

Minimalstelle: [9.999999999999999E-001, 1.000000000000000E+000]
 [9.99999999996121E-001, 1.000000000000052E+000]
 Minimum: [-2.000000000000104E+000, -1.99999999999597E+000]

Beispiel 59 : (f37, $x \in \mathbb{R}^3$)

(aus [11])

$$f_{37}(x) = -(x_1 - 1)^2 - x_2^2 - (x_3 - 1)^2$$

$$\begin{aligned}p_1(x) &= x_1 + x_2 - x_3 - 1 & p_2(x) &= -x_1 + x_2 - x_3 + 1 \\p_3(x) &= 12x_1 + 5x_2 + 12x_3 - 34.8 & p_4(x) &= 12x_1 + 12x_2 + 7x_3 - 29.1 \\p_5(x) &= -6x_1 + x_2 + x_3 + 4.1\end{aligned}$$

Startbox: $([0, 10], [0, 10], [0, 10])^T$.

Bei Genauigkeitsforderungen $\varepsilon_x = \varepsilon_F = 10^{-15}$ terminiert das Verfahren mit 7 Elementen in der Resultatsliste. Für keine Einschließung kann nachgewiesen werden, daß sie eine Minimalstelle enthält. Die bekannte Lösung $(1, 0, 0)$ dieses theoretischen Problems ist in der letzten Einschließung enthalten, diese wird hier angegeben.

Minimalstelle: [9.999999999999977E-001, 1.000000000000002E+000]
 [0.000000000000000E+000, 2.270484605928812E-015]
 [0.000000000000000E+000, 9.256605382617034E-016]
 Minimum: [-1.000000000000000E+000, -9.9999999999989E-001]

Beispiel 60 : (f40, $x \in \mathbb{R}^3$)

(aus [11])

$$f_{40}(x) = x_2^2 - x_1^2 + 6x_1 - 4x_2$$

$$\begin{aligned}p_1(x) &= x_2 - 2x_1 - 2 & p_2(x) &= 2x_1 - x_2 - 8 \\p_3(x) &= -x_1 - 2x_2 - x_3 + 4\end{aligned}$$

Startbox: $([0, 10], [0, 4], [-10, 10])^T$.

Minimalstelle: [0.000000000000000E+000, 0.000000000000000E+000]
 [2.000000000000000E+000, 2.000000000000000E+000]
 [0.000000000000000E+000, 1.000000000000000E+001]
 Minimum: [-4.000000000000000E+000, -4.000000000000000E+000]

	f49	f31	f15	f37	f40
n	2	3	2	3	3
FA_f	335	183	45	747	25
GC_f	132	63	30	270	33
HC_f	133	140	34	454	63
FA_{p_1}	335	88	33	635	19
GC_{p_1}	16	24	22	207	9
HC_{p_1}	20	54	28	324	27
FA_{p_2}	386	85	–	372	20
GC_{p_2}	76	39	–	168	6
HC_{p_2}	88	81	–	279	9
FA_{p_3}	–	56	–	242	14
GC_{p_3}	–	57	–	90	15
HC_{p_3}	–	117	–	135	27
FA_{p_4}	–	52	–	167	–
GC_{p_4}	–	60	–	111	–
HC_{p_4}	–	108	–	180	–
FA_{p_5}	–	45	–	105	–
GC_{p_5}	–	51	–	93	–
HC_{p_5}	–	90	–	171	–
MaxL	9	29	2	52	2
STU-time	1.1	1.85	0.28	4.77	0.39
time	0:00:01,700	0:00:02,850	0:00:00,440	0:00:07,360	0:00:00,610

Beispiel 61 : (**f55**, $x \in \mathbb{R}^3$) Geometric Container Problem (Nummer 339 in [16])

$$f_{55}(x) = \frac{0.2}{x_1 x_2 x_3} + \frac{4}{x_1} + \frac{3}{x_3}$$

$$p_1(x) = 2x_1 x_3 + x_1 x_2 - 10$$

Startbox: $([0.1, 3], [0.1, 1], [0.1, 2])^T$.

Bei diesem Problem terminiert das Verfahren bei den Genauigkeitsforderungen $\varepsilon_x = \varepsilon_F = 10^{-15}$ mit 7 Elementen in der Resultatsliste. Die 7 Einschließungen liegen sehr dicht zusammen, überschneiden sich aber nicht. Hier wird nur die erste dieser Boxen angegeben.

Minimalstelle: [2.379762719685490E+000 , 2.379762719685512E+000]
 [3.162277660168347E-001 , 3.162277660168415E-001]
 [1.942935922772538E+000 , 1.942935922772555E+000]

Minimum: [3.361679689249522E+000 , 3.361679689249542E+000]

Beispiel 62 : (**f71**, $x \in \mathbb{R}^2$)

(aus [11])

$$f_{71}(x) = x_1 + (x_1 - x_2 + 5)(x_1 + x_2 - 1)$$

$$p_1(x) = 9 - 2x_1 - 3x_2 \quad p_2(x) = 3x_1 - x_2 - 8$$

$$p_3(x) = -x_1 + 2x_2 - 8 \quad p_4(x) = x_1 + 2x_2 - 12$$

Startbox: $([0, 10], [0, 10])^T$.

Minimalstelle: [0.000000000000000E+000, 1.209538521459342E-028]
 [2.999999999999999E+000, 3.000000000001740E+000]

Minimum: [-2.0000000000000001E+000, -1.99999999999130E+000]

Beispiel 63 : (**f73**, $x \in \mathbb{R}^2$)

(aus [11])

$$f_{73}(x) = x_1^2 + 4x_1x_2 + x_2^2$$

$$p_1(x) = 6 - 3x_1 - 2x_2$$

Startbox: $([0, 4], [0, 4])^T$.

Minimalstelle: [1.999999999999999E+000, 2.000000000001313E+000]
 [0.000000000000000E+000, 3.085308034422964E-028]

Minimum: [3.999999999999997E+000, 4.000000000003220E+000]

Beispiel 64 : (**f42**, $x \in \mathbb{R}^5$)

(Nummer 270 in [16])

$$f_{42}(x) = x_1(x_2(x_3(x_4 - 4) - 3x_4 + 12) + x_3(-2x_4 + 8) + 6x_4 + 8x_3 - 24) \\ + x_2(x_4(-x_3 + 3) + 4 * x_3 - 12) + x_4(2x_3 - 6) - 8x_3 + 24 \\ + 1.5x_5^4 - 5.75x_5^3 + 5.25x_5^2$$

$$p_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 34$$

Startbox: $([1, 2], [2, 3], [3, 4], [4, 5], [2, 3])^T$.

Minimalstelle: [1.000000000000000E+000, 1.000000000001819E+000]
 [2.000000000000000E+000, 2.000000000001819E+000]
 [3.000000000000000E+000, 3.000000000001327E+000]
 [4.000000000000000E+000, 4.000000000003638E+000]
 [2.000000000000000E+000, 2.000000000001819E+000]

Minimum: [2.299999999980956E+001, 2.300000000024473E+001]

	f55	f71	f73	f42
n	3	2	2	5
FA_f	2170	74	124	243
GC_f	1416	44	56	55
HC_f	1857	48	54	275
FA_{p_1}	2291	56	113	165
GC_{p_1}	936	24	32	55
HC_{p_1}	1179	28	36	275
FA_{p_2}	–	38	–	–
GC_{p_2}	–	6	–	–
HC_{p_2}	–	8	–	–
FA_{p_3}	–	40	–	–
GC_{p_3}	–	12	–	–
HC_{p_3}	–	16	–	–
FA_{p_4}	–	31	–	–
GC_{p_4}	–	10	–	–
HC_{p_4}	–	12	–	–
MaxL	81	4	7	14
STU-time	12.77	0.57	0.5	3.46
time	0:00:19,470	0:00:00,880	0:00:00,770	0:00:05,330

Beispiel 65 : ($f90, x \in \mathbb{R}^6$)

(Nummer 95 in [7])

$$f_{90}(x) = 4.3x_1 + 31.8x_2 + 63.3x_3 + 15.8x_4 + 68.5x_5 + 4.7x_6$$

$$p_1(x) = x_1(-17.1 + 169x_3) - 38.2x_2 + x_3(3580x_5 - 204.2) + x_4(3810x_5 - 212.3 + 18500x_6) + x_5(24300x_6 - 623.4) - 1495.5x_6 + 4.97$$

$$p_2(x) = x_1(139x_3 - 17.9) - 36.8x_2 - 113.9x_3 + x_4(2450x_5 - 169.7 + 16600x_6) + x_5(17200x_6 - 337.8) - 1385x_6 - 1.88$$

$$p_3(x) = 273x_2 + 70x_4 + x_5(819 - 26000x_4) - 29.08$$

$$p_4(x) = -159.9x_1 + 311x_2 - 587x_4 - 391x_5 + x_6(14000x_1 - 2198) - 78.02$$

Startbox: $([0, 0.31], [0, 0.046], [0, 0.068], [0, 0.042], [0, 0.028], [0, 0.0134])^T$.

Minimalstelle: [0.000000000000000E+000, 5.786245613083282E-013]
 [0.000000000000000E+000, 4.187373547839012E-014]
 [0.000000000000000E+000, 1.577175414777481E-020]
 [0.000000000000000E+000, 1.988505176263941E-013]
 [0.000000000000000E+000, 1.690178786818900E-022]
 [3.323303243052894E-003, 3.323303243062617E-003]

Minimum: [1.561952524234860E-002, 1.561952524594185E-002]

In der nachfolgenden Tabelle ist das Verhalten der Funktion bei Testläufen mit den Splittingregeln A, B, C und D und den Genauigkeitsforderungen $\varepsilon_x = \varepsilon_F = 10^{-12}$ dokumentiert.

<i>rule</i>	A	B	C	D
FA_f	12685	9233	22258	12685
GC_f	6432	6468	10530	6432
HC_f	16392	16818	25887	16392
FA_{p_1}	10297	4168	19231	10297
GC_{p_1}	3906	3576	6810	3906
HC_{p_1}	12132	11376	20844	12132
FA_{p_2}	1436	1999	1667	1436
GC_{p_2}	6	6	6	6
HC_{p_2}	36	36	36	36
FA_{p_3}	1439	2001	1667	1439
GC_{p_3}	18	18	6	18
HC_{p_3}	108	108	36	108
FA_{p_3}	1434	1997	1665	1434
GC_{p_3}	0	0	0	0
HC_{p_3}	0	0	0	0
MaxL	442	359	758	442
STU-time	122.83	108.31	205.07	123.01
time	0:03:09,160	0:02:46,810	0:05:15,820	0:03:09,440

Testdaten

Testproblem 12:

<i>i</i>	a_i	<i>i</i>	a_i	<i>i</i>	a_i	<i>i</i>	a_i
1	8	12	14	23	22	34	30
2	8	13	14	24	22	35	30
3	10	14	16	25	24	36	32
4	10	15	16	26	24	37	32
5	10	16	16	27	24	38	34
6	10	17	18	28	26	39	36
7	12	18	18	29	26	40	36
8	12	19	20	30	26	41	38
9	12	20	20	31	28	42	38
10	12	21	20	32	28	43	40
11	14	22	22	33	30	44	42

<i>i</i>	b_i	<i>i</i>	b_i	<i>i</i>	b_i	<i>i</i>	b_i
1	0.49	12	0.43	23	0.41	34	0.4
2	0.49	13	0.43	24	0.4	35	0.38
3	0.48	14	0.44	25	0.42	36	0.41
4	0.47	15	0.43	26	0.4	37	0.4
5	0.48	16	0.43	27	0.4	38	0.4
6	0.47	17	0.46	28	0.41	39	0.41
7	0.46	18	0.45	29	0.4	40	0.38
8	0.46	19	0.42	30	0.41	41	0.4
9	0.45	20	0.42	31	0.41	42	0.4
10	0.43	21	0.43	32	0.4	43	0.39
11	0.45	22	0.41	33	0.4	44	0.39

Testproblem 20:

$$A = \begin{pmatrix} -872028.8849 \\ 150512.5253 \\ -156.6950325 \\ 476470.3222 \\ 729482.8271 \end{pmatrix}, \quad B = \begin{pmatrix} -145421.402 \\ 2931.1506 \\ -40.427932 \\ 5106.192 \\ 15711.36 \end{pmatrix}$$

$$C = \begin{pmatrix} -155011.1084 \\ 4360.53352 \\ 12.9492344 \\ 10236.884 \\ 13176.786 \end{pmatrix}, \quad D = \begin{pmatrix} -326669.5104 \\ 7390.68412 \\ -27.8986976 \\ 16643.76 \\ 30988.146 \end{pmatrix}$$

Literatur

- [1] Alefeld, G. und Herzberger, J.: *Introduction to Interval Computations*. Academic Press, New York, 1983.
- [2] Floudas, C. A. und Pardalos, P. M.: *A Collection of Test Problems for Constrained Global Optimization*. Lecture Notes in Computer Science, Nr. 455, Springer Verlag, Berlin, Heidelberg, New York, 1990.
- [3] Goos, A.: *Verifikationsverfahren zur globalen Optimierung mit Ungleichungsnebenbedingungen*. Diplomarbeit, Institut für angewandte Mathematik, Universität Karlsruhe, 1996.
- [4] Großmann, Ch. und Terno, J.: *Numerik der Optimierung*. Teubner, 1993.
- [5] Hansen, E.: *Global Optimization Using Interval Analysis*. Marcel Dekker, 1992.
- [6] Hansen, E. und Sengupta, S.: *Bounding Solutions of Systems of Equations Using Interval Analysis*. BIT **21**, 203–211, 1981.
- [7] Hock, W. und Schittkowski, K.: *Test Examples for nonlinear Programming Code*. Lecture Notes in Economics and Mathematical Systems, Nr. 187, Springer Verlag, Berlin, Heidelberg, New York, 1981.
- [8] Neumaier, A.: *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.
- [9] Ottmann, T. und Widmayer, P.: *Algorithmen und Datenstrukturen*. Reihe Informatik, Bd. 70, 2. Auflage, BI-Wiss.-Verlag, Mannheim, 1993.
- [10] Pardalos, P. M. und Rosen, J. B.: *Constrained Global Optimization: Algorithms and Applications*. Lecture Notes in Computer Science, Nr. 268, Springer Verlag, Berlin, Heidelberg, New York, 1987.
- [11] Pardalos, P. M. und Rosen, J. B. (ed.): *Computational Methods in Global Optimization*. Annals of Operation Research, Vol. 25, J.C. Baltzer AG, 1990.
- [12] Ratschek, H. und Rokne, J.: *New Computer Methods for Global Optimization*. Ellis Horwood Limited, 1988.

- [13] Ratz, D.: *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*. Dissertation, Universität Karlsruhe, 1992.
- [14] Ratz, D.: *On the Selection of Subdivision Directions in Interval Branch-and-Bound Methods for Global Optimization*. *Journal of Global Optimization* **7**, 183–207, 1995.
- [15] Rump, S. M.: *Kleine Fehlerschranken bei Matrixproblemen*. Dissertation, Universität Karlsruhe, 1980.
- [16] Schittkowski, K.: *More Test Examples for Nonlinear Programming Code*. *Lecture Notes in Economics and Mathematical Systems*, Nr. 282, Springer Verlag, Berlin, Heidelberg, New York, 1987.

In dieser Reihe sind bisher die folgenden Arbeiten erschienen:

- 1/1996 Ulrich Kulisch: *Memorandum über Computer, Arithmetik und Numerik.*
- 2/1996 Andreas Wiethoff: *C-XSC — A C++ Class Library for Extended Scientific Computing.*
- 3/1996 Walter Krämer: *Sichere und genaue Abschätzung des Approximationsfehlers bei rationalen Approximationen.*
- 4/1996 Dietmar Ratz: *An Optimized Interval Slope Arithmetic and its Application.*
- 5/1996 Dietmar Ratz: *Inclusion Isotone Extended Interval Arithmetic.*
- 1/1997 Astrid Goos, Dietmar Ratz: *Praktische Realisierung und Test eines Verifikationsverfahrens zur Lösung globaler Optimierungsprobleme mit Ungleichungsnebenbedingungen.*
- 2/1997 Stefan Herbort, Dietmar Ratz: *Improving the Efficiency of a Nonlinear-System-Solver Using a Componentwise Newton Method.*
- 3/1997 Ulrich Kulisch: *Die fünfte Gleitkommaoperation für top-performance Computer — oder — Akkumulation von Gleitkommazahlen und -produkten in Festkommaarithmetik.*
- 4/1997 Ulrich Kulisch: *The Fifth Floating-Point Operation for Top-Performance Computers — or — Accumulation of Floating-Point Numbers and Products in Fixed-Point Arithmetic.*
- 5/1997 Walter Krämer: *Eine Fehlerfaktorarithmetik für zuverlässige a priori Fehlerabschätzungen.*