Computing 53, 323–335 (1994)

Computing

Safe Starting Regions by Fixed Points and Tightening*

H. Hong and **V. Stahl,** Linz

Received November 23, 1993; revised April 27, 1994

Abstract — Zusammenfassung

Safe Starting Regions by Fixed Points and Tightening. In this paper, we present a method for finding safe starting regions for a given system of non-linear equations. The method is an improvement of the usual method which is based on the fixed point theorem. The improvement is obtained by enclosing the components of the equation system by univariate interval polynomials whose zero sets are found. This operation is called "tightening". Preliminary experiments show that the tightening operation usually reduces the number of bisections, and thus the computing time. The reduction seems to become more dramatic when the number of variables increases.

Key words: Nonlinear equation systems, interval arithmetic, safe starting regions, tightening.

Startintervalle mit garantierter Konvergenz durch Fixpunktiteration und Einengung. In dieser Arbeit wird eine Methode zur Bestimmung von Startintervallen mit garantierter Konvergenz für ein gegebenes nichtlineares Gleichungssystem vorgestellt. Die Methode ist eine Verbesserung der gebräuchlichen, auf dem Fixpunkt Theorem basierenden Methode. Die Verbesserung wird durch Einschließen der Komponenten des Gleichungssystems durch univariate Intervallpolynome, deren Lösungsmengen berechnet werden, erzielt. Diese Operation wird "Einengung" genannt. Erste experimentelle Untersuchungen zeigen, daß Einengung im allgemeinen die Anzahl der Intervallhalbierungen und somit die Rechenzeit reduziert. Die Reduktion scheint umso signifikanter, je höher die Anzahl der Variablen ist.

1. Introduction

Finding safe starting regions for all the real solutions of a given system of equations is a very important problem in scientific computing, geometric modelling, constraint logic programming, etc. In [3, 7–10, 14–18] various methods for finding safe starting regions have been described, which are based on the Krawczyk operator [4] and various improvements.

In this paper, we provide another improvement by adding an operation called "tightening". Roughly put, tightening is an operation which takes an equation and a box and produces (sufficiently) small boxes such that they together still contain

* The research was done within the framework of the ACCLAIM project sponsored by European Community Basic Research Action (ESPRIT 7195) and Austrian Science Foundation (P9374-PHY).

all the zeros of the equations in the original box. This operation has been investigated in the AI community [2, 5, 13] for solving some simple equations and inequalities such $xy = z$ and $x > 0$. The Hansen-Sengupta operator [3] applies a special form of tightening to preconditioned linearized equation systems. Our contribution lies in that we allow arbitrary equations and in that tightening is combined with the fixed point methods developed in the interval mathematics community.

Preliminary experiments show that the tightening operation usually reduces the number of bisections, and thus the computing time. The reduction seems to become more dramatic when the number of variables increases.

In Section 2, we fix the notational conventions and state precisely the problem of safe starting regions. In Section 3, we give a brief description of the known methods based on the fixed point theorem. In Section 4, we give a precise definition of the tightening operation. In Section 5, we describe an algorithm for finding safe starting regions, which uses the tightening operation. In Section 6, we illustrate the algorithm described in the previous section on a simple example. In Section 7, we give a general description of a tightening procedure, and a detailed one for multivariate polynomials in particular. In Section 8, we report some experimental results.

2. Notations and Problem Statement

The notational and typographical conventions used in this paper are as follows:

$\mathbb{R}$	the set of all real numbers
$\mathbb{IR}$	the set of all closed intervals over $\mathbb{R}$
lower-case	real number (vector, matrix, function)
upper-case	interval (vector, matrix, function)
italic	scalar
bold	vector
roman	matrix

For instance, an interval vector will be written such as $\boldsymbol{X}$, and a real matrix will be written (though against tradition) such as $\mathbf{m}$, for consistency.

The alphabets f, g will stand for functions, the alphabets x, y for variables, the alphabets i, j, k, l, m, n for indices, and the others for constants. In particular, throughout this paper, let $\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^n$ be a differentiable function, and let $\boldsymbol{X} \in \mathbb{IR}^n$. We are interested in the solutions of $\boldsymbol{f} = 0$ in $\boldsymbol{X}$.

Definition 1 (Safe Starting Region). *A box $\boldsymbol{X}$ is called a* safe starting region *for a function $\boldsymbol{f}$ iff it contains exactly one solution of $\boldsymbol{f} = 0$ and a Newton-like method converges to the solution from anywhere in the box.* □

Now the problem of safe starting regions can be stated as follows:

Given: a function $\boldsymbol{f}$ and a box $\boldsymbol{X}$,
Task: find a set of disjoint safe starting regions of $\boldsymbol{f}$ such that each solution of $\boldsymbol{f}$ within $\boldsymbol{X}$ is contained in one of them.

3. Review of Fixed-point Methods

The usual method is based on the following operator or its improvements.

Definition 2 (Krawczyk). *Let $\boldsymbol{p}$ be a point in $\boldsymbol{X}$. Let $\mathbf{m}$ be any non-singular matrix over $\mathbb{R}$. Let $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x} - \mathbf{m}\boldsymbol{f}(\boldsymbol{x})$. Let $\boldsymbol{G}'$ be an interval extension of the Jacobian of $\boldsymbol{g}$. Then the Krawczyk operator $\boldsymbol{K}$: $\mathbb{IR}^n \to \mathbb{IR}^n$ is defined by*

$$\boldsymbol{K}(\boldsymbol{X}) = \boldsymbol{g}(\boldsymbol{p}) + \boldsymbol{G}'(\boldsymbol{X})(\boldsymbol{X} - \boldsymbol{p}). \quad \square$$

The usual methods essentially repeat the following operations until safe starting regions for all the real solutions of $\boldsymbol{f}$ are found.

1. *Range test:* If $0 \notin \boldsymbol{F}(\boldsymbol{X})$, then $\boldsymbol{X}$ does not contain a solution.
2. *Intersection test:* If $\boldsymbol{K}(\boldsymbol{X}) \cap \boldsymbol{X} = \varnothing$, then $\boldsymbol{X}$ does not contain a solution.
3. *Inclusion test:* If $\varnothing \neq \boldsymbol{K}(\boldsymbol{X}) \subset \operatorname{int}(\boldsymbol{X})$ then $\boldsymbol{X}$ is a safe starting region.
4. *Bisection:* Replace a box $\boldsymbol{X}$ by two sub-boxes $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$ of equal size such that $\boldsymbol{X}^{(1)} \uplus \boldsymbol{X}^{(2)} = \boldsymbol{X}$.
5. *Intersection:* Replace a box $\boldsymbol{X}$ by $\boldsymbol{X} \cap \boldsymbol{K}(\boldsymbol{X})$.

It is important to add that if the Hansen-Sengupta operator $\boldsymbol{H}$ [3] is used instead of $\boldsymbol{K}$, the *Intersection* and *Intersection test* usually lead to better results. For details, see the excellent monographs [6, 1, 12].

4. Tightening

In this section, we define the notion "tightening". Throughout this section, we assume that f is a function from $\mathbb{R}^n$ to $\mathbb{R}$, and $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$ is a box $\mathbb{IR}^n$.

Definition 3 (Variety). *The variety of f, written as $V(f)$, is defined by*

$$V(f) = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) = 0\} \quad \square$$

Definition 4 (Projection). *Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. The i-th projection of $\boldsymbol{x}$, written as $\pi_i(\boldsymbol{x})$, is defined by*

$$\pi_i(\boldsymbol{x}) = x_i.$$

Let $\boldsymbol{S} \subseteq \mathbb{R}^n$. Then the i-th projection of $\boldsymbol{S}$, written also as $\pi_i(\boldsymbol{S})$, is defined by

$$\pi_i(\boldsymbol{S}) = \{\pi_i(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{S}\}. \quad \square$$

Definition 5 (Optimal Tightening). *The optimal tightening of $\boldsymbol{X}$ on x_i by f is defined as the set*

$$\pi_i(V(f) \cap \boldsymbol{X}). \quad \square$$

Proposition 1 (Solution Preservation). *Let $\boldsymbol{X}_i'$ be the optimal tightening of $\boldsymbol{X}$ on x_i by f, and let $\boldsymbol{X}' = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}, \boldsymbol{X}_i', \boldsymbol{X}_{i+1}, \ldots, \boldsymbol{X}_n)$. Then we have*

$$V(f) \cap \boldsymbol{X} = V(f) \cap \boldsymbol{X}'. \quad \square$$

Thus, we can always safely replace $\boldsymbol{X}$ by $\boldsymbol{X}'$ when we are interested only in the solution of f in $\boldsymbol{X}$. But, it is expensive to compute the optimal tightening, since it in general requires exact computations with real algebraic numbers. Thus we relax the definition as follows:

Definition 6 (Tightening). *A tightening of $\boldsymbol{X}$ on x_i by f is a finite set of disjoint subintervals of X_i whose union contains the optimal tightening. More precisely, it is a set*

$$\{X^1, \ldots, X^l\}$$

such that $X^k \in \mathbb{IR}$, $X^k \subseteq X_i$, $X^k \cap X^j = \varnothing$ for $k \neq j$, and $\biguplus_k X^k \supseteq \pi_i(V(f) \cap \boldsymbol{X})$. □

Definition 7 (Tightening operator). *A tightening operator is a procedure that, given f, $\boldsymbol{X}$ and i, produces a finite set of boxes $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^l$ such that for every $k = 1, \ldots, l$*

$$\boldsymbol{X}^k = (X_1, \ldots, X_{i-1}, X_i^k, X_{i+1}, \ldots, X_n)$$

where the set $\{X_i^1, \ldots, X_i^l\}$ forms a tightening of $\boldsymbol{X}$ on x_i by f. □

One extreme tightening operator is the most expensive one which computes the optimal tightening and the other extreme is the cheapest one which trivially returns $\{\boldsymbol{X}\}$. Obviously, we are interested in one between these two extremes, which strikes a "good" compromise between accuracy and computational cost. One such tightening operator for multivariate polynomial functions will be described in Section 7. In fact, we have developed a tightening operator for arbitrary elementary functions (involving exponentials, trigonometric functions, etc.), and it will be reported elsewhere.

5. Algorithm for Finding Safe Starting Regions

In this section, we describe an algorithm for the problem posed in Section 2, namely that of finding safe starting regions for all the zeros of a function within a box. The general structure of the algorithm is similar to the usual ones. The difference is that we allow during the execution the tightening operation defined in the last section.

Algorithm 1 (Safe Start Region).

In: $\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^n$
$\boldsymbol{X} \in \mathbb{IR}^n$.

Out: $\boldsymbol{S}_{\text{safe}}$, a finite set of safe starting regions of $\boldsymbol{f}$ in $\boldsymbol{X}$.
$\boldsymbol{S}_{\text{small}}$, a finite set of sub-boxes of $\boldsymbol{X}$ that are too small to work on.

Local: $\boldsymbol{S}_{\text{work}}$, a finite set of sub-boxes of $\boldsymbol{X}$ that need to be worked on.

(1) [Initialization.] $\boldsymbol{S}_{\text{work}} \leftarrow \{\boldsymbol{X}\}$. $\boldsymbol{S}_{\text{safe}} \leftarrow \{\ \}$. $\boldsymbol{S}_{\text{small}} \leftarrow \{\ \}$.
(2) [Choice.] Choose (and remove) a box from $\boldsymbol{S}_{\text{work}}$. Choose one of the following operations: range test, intersection test, inclusion test, bisection, intersection, and tightening.

(3) [Process.] Apply the operation on the chosen box, obtaining possibly one ore more sub-boxes. Insert the resulting boxes into the proper sets: $\boldsymbol{S}_{\text{safe}}$, $\boldsymbol{S}_{\text{small}}$, or $\boldsymbol{S}_{\text{work}}$.
(4) [Loop.] If there is a box in $\boldsymbol{S}_{\text{work}}$ then go to Step (2). Otherwise, we are done. □

The algorithm is correct no matter which boxes and operations are chosen in Step (2). Since we are interested in finding all safe starting regions (not just one), the efficiency of the algorithm does not depend on which box we choose, because we need to analyze every box eventually. Thus, it is fine to choose the first one in the data structure of $\boldsymbol{S}_{\text{work}}$.

But the efficiency of the algorithm heavily depends on which operation is chosen. Based on preliminary experimental study, we found that the following strategy seems to work well (at least for the examples we have tested):

Strategy.

1. First, apply tightening with respect to every equation and variable until the state does not change any more (or changes very slowly). During the tightening, carry out range check, since it can be done cheaply using the intermediate results of tightening.
2. Next, apply Krawczyk intersection and inclusion test.
3. Then carry out intersection.
4. If the state has been "significantly" changed, then go back to tightening, else bisect and go back to tightening. □

In order to apply this heuristics, one should decide how much change is "significant" to avoid bisection. In our current implementation, we consider a change significant if it is greater than 10% in size, where the size of a box is the sum of the widths of its components.

For purposes of comparison, we have implemented the method described in [3], and we applied a similar strategy.

6. Illustration

Before going into the details of tightening operation, we illustrate the main algorithm described in the previous section on a simple example. The purpose of such exercise is to strengthen the intuitive understanding of the method, which will be helpful in understanding and motivating the detailed discussions in the next section. We will use the following example taken from [7].

$$x^2 + y^2 - 1$$

$$x^2 - y$$

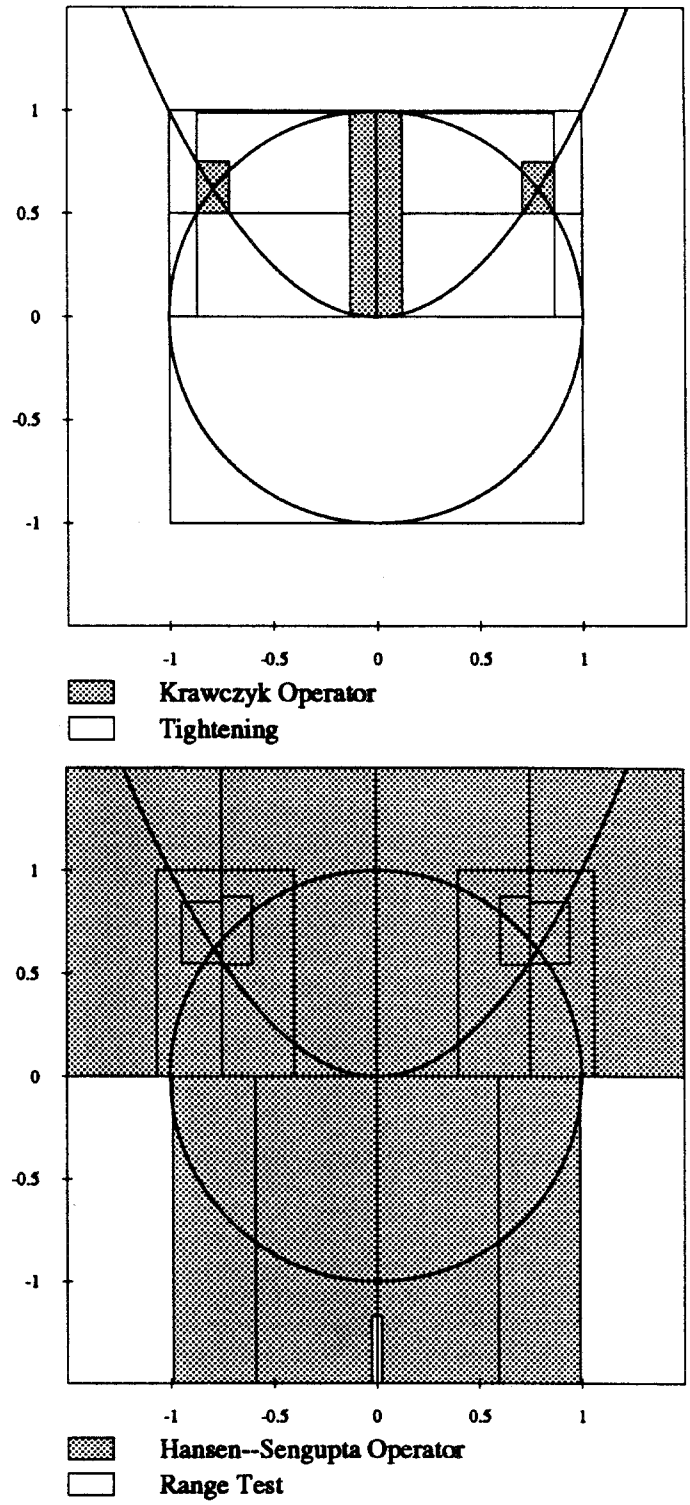

Figure 1. Illustration of the main algorithm at a simple example

See the two pictures in Fig. 1. The upper part traces the boxes produced during the execution of the algorithm of the last section. The lower part provides the same information produced by the algorithm of [3] which is basically run by Hansen-Sengupta operator, range test, intersection, and bisection.

From the upper part of Fig. 1, we see that the initial box $[-1.5, 1.5]^2$, that is, $[-1.5, 1.5] \times [-1.5, 1.5]$, is first tightened by the circle and then by the parabola. The white patch is the portion that has been tightened out. The remaining box can neither be reduced by the Krawczyk operator nor by tightening, so it is bisected vertically. In each sub-box tightening is not successful, but Krawczyk intersection leads to some reduction, pruning out the gray long strip. The remaining box is bisected horizontally. The lower half box is tightened by the circle and then ruled out during tightening by the parabola. The upper half box is tightened by both curves and the Krawczyk operator detects that the remaining box is a safe starting region.

See the lower part of Fig. 1; we do not go into details, but note that it produced more intermediate boxes. It is partly due to the increased number of bisections (the upper picture has 3 bisection, while the lower one has 5 bisections). The preliminary experimental results in Section 8 show that this effect seems to become greater for higher dimensional problems.

7. Algorithm for Tightening

In this section, we describe a general scheme for tightening continuous functions, and in particular give a detailed procedure for the case of multi-variate polynomials.[1] Let us begin by recalling the problem of tightening.

In: $f: \mathbb{R}^n \to \mathbb{R}$, a continuous function, $\boldsymbol{X} \in \mathbb{IR}^n$, and $i \in \{1, \ldots, n\}$.
Out: a tightening of $\boldsymbol{X}$ on x_i by f.

In the following, we will reduce this problem to two subproblems. First observe the following straightforward rewriting of the definition of the optimal tightening:

$$\begin{aligned}
&\pi_i(V(f) \cap \boldsymbol{X}) \\
&\quad = \{x_i \in X_i | (\exists x_1 \in X_1) \cdots (\exists x_{i-1} \in X_{i-1})(\exists x_{i+1} \in X_{i+1}) \cdots (\exists x_n \in X_n) f(\boldsymbol{x}) = 0\} \\
&\quad = \{x_i \in X_i | 0 \in \{f(\boldsymbol{x}) | x_1 \in X_1, \ldots, x_{i-1} \in X_{i-1}, x_{i+1} \in X_{i+1}, \ldots, x_n \in X_n\}\}.
\end{aligned}$$

The last expression can be simplified by defining the function $F(x_i)$: $\mathbb{R} \to 2^{\mathbb{R}}$ such that

$$F(x_i) = \{f(\boldsymbol{x}) | x_1 \in X_1, \ldots, x_{i-1} \in X_{i-1}, x_{i+1} \in X_{i+1}, \ldots, x_n \in X_n\}.$$

Using this function, we can rewrite the above formulas as

$$\pi_i(V(f) \cap \boldsymbol{X}) = \{x_i \in X_i | 0 \in F(x_i)\}.$$

---

[1] We have also developed a more general tightening procedure that allows elementary transcendental functions and which will be reported elsewhere.

Since f is continuous, $F(x_i)$ is an interval for every x_i, and this motivates to defin the two functions $\underline{F}, \overline{F}: \mathbb{R} \to \mathbb{R}$ such that

$$F(x_i) = [\underline{F}(x_i), \overline{F}(x_i)].$$

Continuing using these two functions, we have

$$\begin{aligned}\pi_i(V(f) \cap X) &= \{x_i \in X_i | 0 \in [\underline{F}(x_i), \overline{F}(x_i)]\} \\ &= \{x_i \in X_i | \underline{F}(x_i) \leq 0 \wedge \overline{F}(x_i) \geq 0\} \\ &= \{x_i \in X_i | \underline{F}(x_i) \leq 0\} \cap \{x_i \in X_i | \overline{F}(x_i) \geq 0\}\end{aligned}$$

Thus, we have reduced the problem of optimal tightening into two sub-problems: (1) finding the functions $\underline{F}$ and $\overline{F}$, (2) solving the inequalities $\underline{F}(x_i) \leq 0$ and $\overline{F}(x_i) \geq 0$. Obviously for the purpose of just tightening (not necessary optimal), it will be sufficient to over-estimate them. Thus we have the following main algorithm:

Algorithm 2 (Tightening).

(1) Compute the functions $\underline{F}$ and $\overline{F}$ (suitably over-estimate).
(2) Solve the two inequalities $\underline{F} \leq 0$ and $\overline{F} \geq 0$ (over-estimate again).
(3) Return the intersection of the two solution sets. □

In the following two subsections, we show the details of the first two steps. For the first step, we will present an algorithm for polynomials only, while for the second step, we will give a general algorithm. Thus we will present the second step first in the order of generality.

7.1 Solving Inequalities

We begin with the second step, namely that of solving the inequalities. The two inequalities are solved in the same way, and thus we describe how to solve only one of them, $\overline{F} \geq 0$. So here is the sub-problem statement:

In: $f: \mathbb{R} \to \mathbb{R}$ and $X \in \mathbb{IR}$.
Out: disjoint intervals $P_1, \ldots, P_k$ such that $\biguplus_j P_j \supseteq \{x \in X | f(x) \geq 0\}$.

The basic idea is to compute all the real roots of f within X, which induces a finite set of intervals on which the sign of f is constant. From these, we only need to select the ones with non-negative signs. In doing this, we face the following technical problems: due to finite precision arithmetic and multiple roots, we do not get the exact roots, but intervals which contain them. Moreover such an interval may contain more than one root. The following algorithm handles such difficulties.

Algorithm 3 (Solving Inequality).

(1) Compute disjoint (root) intervals $R_1, \ldots, R_r$ such that $\biguplus_j R_j \supseteq \{x \in X | f(x) = 0\}$. (This can be done for instance by the Interval Newton Method.)

(2) The following code extracts intervals where f is non-negative. It essentially scans the root intervals from left to right while picking up solution intervals. In the code, R_i is the current root interval being checked, k keeps track of the number of the solution intervals extracted so far, b holds the upper end point of the previous root interval, and Y is the interval evaluation of f on the middle of the gap between two consecutive root intervals.

Initialize $b \leftarrow \underline{X}$ and $k \leftarrow 0$.
for $i = 1, \ldots, r$
  $Y \leftarrow f(1/2(b + \underline{R}_i))$ using interval arithmetic.
  case: $\overline{Y} < 0$: $k \leftarrow k + 1$. $P_k \leftarrow R_i$.
  case: $\overline{Y} \geq 0$ and $k > 0$: $P_k = \mathrm{Hull}(P_k, [b, \overline{R}_i])$.
  case: $\overline{Y} \geq 0$ and $k = 0$: $k \leftarrow k + 1$. $P_k \leftarrow [b, \overline{R}_i]$.
  $b \leftarrow \overline{R}_i$.
if $b \neq \overline{X}$ or ($\underline{X} = \overline{X}$ and $r = 0$)
  $Y \leftarrow f(1/2(b + \overline{X}))$ using interval arithmetic.
  case: $\underline{Y} \geq 0$ and $k > 0$: $P_k = \mathrm{Hull}(P_k, [b, \overline{X}])$.
  case: $\overline{Y} \geq 0$ and $k = 0$: $k \leftarrow 1$. $P_k \leftarrow X$. □

7.2 *Finding Bounding Functions*

Now we tackle the problem of finding the bounding functions for the case of polynomial functions. This can be generalized to allow elementary transcendental functions, but due to page limit, it will be reported elsewhere. Here is the problem statement.

In: $f \in \mathbb{R}[x_1, \ldots, x_n]$, $\boldsymbol{X} \in \mathbb{IR}^n$, and $i \in \{1, \ldots, n\}$.
Out: $\underline{F}$ and $\overline{F}$ as defined above (over-estimated).

For this, let us recall that $\underline{F}$ and $\overline{F}$ are defined by

$$F(x_i) = \{f(\boldsymbol{x}) | x_1 \in X_1, \ldots, x_{i-1} \in X_{i-1}, x_{i+1} \in X_{i+1}, \ldots, x_n \in X_n\}.$$
$$= [\underline{F}(x_i), \overline{F}(x_i)].$$

The following algorithm solves the problem.

Algorithm 4 (Bounding Functions for Polynomials).

(1) Obtain the coefficients $A_j \in \mathbb{IR}$ of an interval polynomial $F(x_i) = \sum_{j=0}^{d} A_j x^j$ by evaluating f on $x_j = X_j$ for every $j \neq i$, using interval arithmetic.
(2) Compute the bounding functions of $F(x)$: (We drop the subscript i for simplicity.)

$$\underline{F}(x) = \begin{cases} \sum_{j=0}^{d} \underline{A}_j x^j & \text{if} \quad x \geq 0 \\ \sum_{j=0}^{d} A_j^{\inf} x^j & \text{else} \end{cases}$$

$$\overline{F}(x) = \begin{cases} \sum_{j=0}^{d} \overline{A}_j x^j & \text{if} \quad x \geq 0 \\ \sum_{j=0}^{d} A_j^{\sup} x^j & \text{else} \end{cases}$$

where

$$A_j^{\inf} = \begin{cases} \underline{A}_j & \text{if } j \text{ is even} \\ \overline{A}_j & \text{if } j \text{ is odd} \end{cases} \qquad A_j^{\sup} = \begin{cases} \overline{A}_j & \text{if } j \text{ is even} \\ \underline{A}_j & \text{if } j \text{ is odd} \end{cases} \quad \square$$

The proof of Step (2) is straightforward, thus is omitted. One remark is needed here. The resulting bounding functions are piecewise differentiable but not differentiable at 0. This does not cause difficulty in solving the corresponding inequalities since one only needs to apply the method of the previous subsection on each piece separately and merge the resulting intervals. While merging, it might be necessary to concatenate two intervals containing 0.

8. Experimental Results

The algorithms described in this paper are implemented in the C++ language on a Silicon Graphics workstation with a 100 MHz processor MIPS 4000/4010. In the sequel **TKIB** denotes the method of this paper (Tightening, Krawczyk operator, Intersection, Bisection) and **HRIB** stands for the method described in [3] (Hansen-Sengupta operator, Range test, Intersection, Bisection). We have tested these programs on the following 10 examples.

A: This example is taken from [7].

$$x_1^2 + x_2^2 = 1$$
$$x_1^2 - x_2 = 0$$

Starting box: $[-1.5, 1.5]^2$.

B: This example is taken from [11].

$$x_1 + x_2 + x_3 + x_4 - 1 = 0$$
$$x_1 + x_2 - x_3 + x_4 - 3 = 0$$
$$x_1^2 + x_2^2 + x_3^2 + x_4^2 - 4 = 0$$
$$x_1^2 + x_2^2 + x_3^2 + x_4^2 - 2x_1 - 3 = 0$$

Starting box: $[-10, 10]^4$.

C1: This example is taken from [9], where the specific values of the coefficients a_i, b_i and the indices i_1, i_2, i_3 are given.

$$x_i - a_i - b_i x_{i_1} x_{i_2} x_{i_3} = 0, \qquad i = 1, \ldots, 10$$

Starting box: $[-2, 2]^{10}$.

C2: This example is also taken from [9]. The difference from **C1** is that it has 20 variables.

$$x_i - a_i - b_i x_{i_1} x_{i_2} x_{i_3} = 0, \qquad i = 1, \ldots, 20$$

Starting box: $[-1, 2]^{20}$.

C3: This example is also taken from [9] with 20 variables. The difference from **C2** is that its starting box is bigger.

$$x_i - a_i - b_i x_{i_1} x_{i_2} x_{i_3} = 0, \qquad i = 1, \ldots, 20$$

Starting box $[-2, 2]^{20}$.

C4: This example is a modification of **C1** in that each variable x_i is replaced by x_i^2. Note that if $(z_1, \ldots, z_{10})$ is a solution of **C4** then $(\pm z_1, \ldots, \pm z_{10})$ is also a solution, hence the number of solutions is a multiple of 1024. In fact, both methods found exactly 1024 safe starting regions.

$$x_i^2 - a_i - b_i x_{i_1}^2 x_{i_2}^2 x_{i_3}^2 = 0, \qquad i = 1, \ldots, 10$$

Starting box: $[-1, 1]^{10}$.

C5: This example is another modification of **C1**, where we use the same coefficients but increase the degrees of some variables and add one more term.

$$x_i - a_i - b_i x_{i_1}^3 x_{i_2}^3 x_{i_3}^3 + x_{i_2}^4 x_{i_3}^7 = 0, \qquad i = 1, \ldots, 10.$$

Starting box: $[-1, 1]^{10}$.

D1: This is a sparse system with 12 variables and low degree.

$$\begin{aligned}
-x_3 x_{10} x_{11} - x_5 x_{10} x_{11} - x_7 x_{10} x_{11} + x_4 x_{12} + x_6 x_{12} + x_8 x_{12} &= 0.4077 \\
x_2 x_4 x_9 + x_2 x_6 x_9 + x_2 x_8 x_9 + x_1 x_{10} &= 1.9115 \\
x_3 x_9 + x_5 x_9 + x_7 x_9 &= 1.9791 \\
3x_2 x_4 + 2x_2 x_6 + x_2 x_8 &= 4.0616 \\
3x_1 x_4 + 2x_1 x_6 + x_1 x_8 &= 1.7172 \\
3x_3 + 2x_5 + x_7 &= 3.9701 \\
x_i^2 + x_{i+1}^2 &= 1, \; i \text{ odd}
\end{aligned}$$

Starting box: $[0.38, 0.40] \times [0.92, 0.93] \times [0.56, 0.57] \times [0.82, 0.83] \times [-1, 1]^8$.

D2: This example is the same as **D1**. The only difference is that its starting box is larger. Starting box: $[0.38, 0.40] \times [0.92, 0.93] \times [-1, 1]^{10}$.

D3: This example is again the same as **D1**. The only difference is that its starting box is even larger. Starting box: $[-1, 1]^{12}$.

In Table 1, we report various statistics. At the top, we report the computing times. It seems that the **TKIB** method is faster, in particular for the problems with many variables. However, as is well-known, one should not trust timings too much, because the values depend on the implementation of the algorithms and data types, memory management, clock resolution, etc.

More reliable parameters are statistics such as the number of explicit bisections. The second part of Table 1 shows that the number of bisections is much smaller for **TKIB**, which explains why **TKIB** is usually faster. The reduction in the number of bisections is mainly due to the tightening operation.

Table 1. Experimental comparison of **TKIB** and **HRIB**

	A	B	C1	C2	C3	C4	C5	D1	D2	D3
Total Time (sec)										
TKIB	0.03	0.78	0.13	0.88	1.04	101.18	0.80	4.03	48.38	117.10
HRIB	0.05	0.82	16.76	1487.50	4699.85	537.34	457.96	64.64	672.09	2395.40
Bisections										
TKIB	3	10	0	0	0	0	2	7	108	257
HRIB	5	87	495	6407	28588	9215	12862	880	9423	29189
TKIB										
Tightening										
calls	16	200	20	60	60	11520	80	480	5632	12798
avg (ms)	1.88	2.85	4.50	8.50	10.17	6.20	8.25	5.38	5.48	5.97
percent	100.00	73.08	69.23	57.95	58.65	70.58	82.50	64.02	63.83	65.22
Krawczyk Operator										
calls	6	47	2	3	3	1152	5	40	464	1034
avg (ms)	0.00	4.26	20.00	123.33	143.33	25.46	28.00	36.00	37.28	38.93
percent	0.00	25.64	30.77	42.05	41.35	28.99	17.50	35.73	35.76	34.37
HRIB										
Range Test										
calls	25	249	1086	16271	59577	26623	25796	2241	23450	71297
avg (ms)	0.40	0.68	2.87	11.16	10.61	3.39	3.74	5.07	4.99	9.36
percent	20.00	20.73	18.62	12.20	13.45	16.82	21.07	17.59	17.39	27.85
Hansen-Sengupta Operator										
calls	21	169	605	10078	31400	17407	12911	1471	15231	47305
avg (ms)	1.90	3.85	22.40	129.47	129.35	25.56	27.85	36.13	36.35	36.44
percent	80.00	79.27	80.85	87.72	86.42	82.79	78.50	82.21	82.37	71.96

For those readers who might be interested in more details we provide further statistics such as the number of Krawczyk operator calls, Hansen-Sengupta operator calls, etc., without comments, though.

References

[1] Alefeld, G., Herzberger, J.: Introduction to interval computations. New York: Academic Press 1983.
[2] Cleary, J. G.: Logical arithmetic. Future Comput. Syst. 125–149 (1987).
[3] Hansen, E., Sengupta, S.: Bounding solutions of systems of equations using interval analysis. BIT *21*, 203–211 (1981).
[4] Krawczyk, R.: Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. Computing *4*, 187–201 (1969).

[5] Mackworth, A. K.: Consistency in networks of relations. Art. Intell. *8*, 99–118 (1977).
[6] Moore, R. E.: Interval analysis. Englewood Cliffs: Prentice-Hall 1966.
[7] Moore, R. E.: A test for existence of solution to nonlinear systems. SIAM J. Numer. Anal. *14*, 611–615 (1977).
[8] Moore, R. E.: A computational test for convergence of iterative methods for nonlinear systems. SIAM J. Numer. Anal. *15*, 1194–1196 (1978).
[9] Moore, R. E., Jones, S. T.: Safe starting regions for iterative methods. SIAM J. Numer. Anal. *14*, 1051–1065 (1977).
[10] Moore, R. E., Qi, L.: A successive interval test for nonlinear systems. SIAM J. Numer. Anal. *19*, 845–850 (1982).
[11] Morgan, A.: Solving polynomial systems using continuation for engineering and scientific problems. Englewood Cliffs: Prentice-Hall 1987.
[12] Neumaier, A.: Interval methods for systems of equations. Cambridge: Cambridge University Press 1990.
[13] Older, W., Vellino, A.: Extending Prolog with constraint arithmetic on real intervals. In: Proceedings of the Eight Biennial Conference of the Canadian Society for Computational Studies of Intelligence, 1990.
[14] Qi, L.: A note on the Moore test for nonlinear system. SIAM J. Numer. Anal. *19*, 851–857 (1982).
[15] Rump, S. M.: Solving nonlinear systems with least significant bit accuracy. Computing *29*, 183–200 (1982).
[16] Rump, S. M.: Solution of linear and nonlinear algebraic problems with sharp, guaranteed bounds. Computing [Suppl.] *5*, 147–168 (1984).
[17] Shearer, J. M., Wolfe, M. A.: Some computable existence, uniquness, and convergence tests for nonlinear systems. SIAM J. Numer. Anal. *22*, 1200–1207 (1985).
[18] Wolfe, M. A.: Interval methods for algebraic equations. In: Reliability in computing, pp. 229–248. London: Academic Press 1988.

H. Hong
V. Stahl
Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz
Austria