Institut für
Angewandte
Mathematik

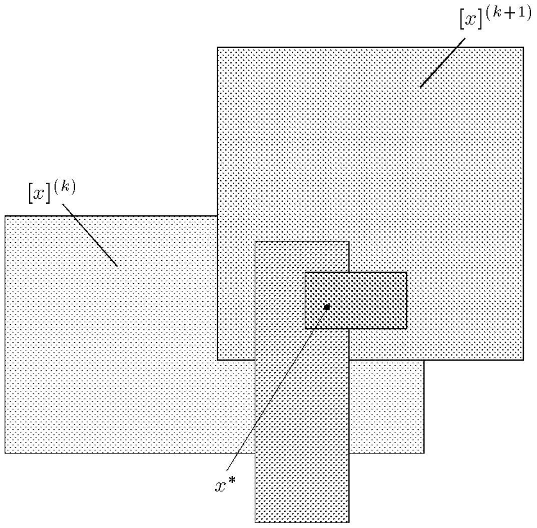
Universität Karlsruhe (TH)
D-76128 Karlsruhe

Inclusion Isotone Extended Interval Arithmetic

— A Toolbox Update —

Dietmar Ratz

F orschungsschwerpunkt
C omputerarithmetik,
I ntervallrechnung und
N umerische Algorithmen mit
E rgebnisverifikation



Bericht 5/1996

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

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

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

Inclusion Isotone Extended Interval Arithmetic

— A Toolbox Update —

Dietmar Ratz

Contents

Zusammenfassung

Inclusionsisotone Erweiterte Intervallarithmetik: In diesem Bericht behandeln wir die korrekte Formulierung einer speziellen erweiterten Intervallarithmetik für die Anwednung in Intervall-Newton-ähnlichen Verfahren. Wir demonstrieren zunächst einige Probleme mit ausgewählten älteren Definitionen. Danach beschreiben wir Ziel, Konzept und Eigenschften, die für die Definition einer korrekten erweiterten Intervalldivision wichtig sind. Schließlich, zeigen wir einen geeigneten Weg auf, die in unserem speziellen Kontext benötigten erweiterten Intervalloperationen zu realisieren, und wir beweisen deren Inklusionsisotonie. Weiterhin geben wir einige Bespielanwendungen an. Den Abschluß bilden Implementierungs-Updates unserer erweiterten Intervalloperationen in den Toolbox-Umgebungen [2] und [3].

Abstract

Inclusion Isotone Extended Interval Arithmetic: In this report we deal with the correct formulation of a special extended interval arithmetic in the context of interval Newton like methods. We first demonstrate some of the problems arising from selected older definitions. Then we investigate the basic aim, concept, and properties important for defining a correct extended interval division. Finally, we give a proper way for defining the extended interval operations needed in our special context, and we prove their inclusion isotonicity. Additionally, we give some sample applications. We conclude with two updated implementations of our extended interval operations in the toolbox environments [2] and [3].

1 Introduction

In the usual rules for interval division (see [1] for example), the case of division by an interval containing zero is excluded. Nevertheless, this case can be very useful in the context of interval Newton (Gauss-Seidel) methods applied in the field of *global methods*, which aim at finding all roots of a nonlinear system of equations or all solutions of a global optimization problem within a specified region. Here, this so-called *extended interval division* helps to eliminate parts of the search region which are guaranteed not to contain a solution (see the sample applications below). For this purpose, we must be able to compute the result of the expression

$$\mathcal{W} := r - \frac{[x]}{[y]}, \tag{1}$$

where $r \in I\!R$ and $[x], [y] \in I I\!R$ independently of $[y]$ containing zero or not. The result $\mathcal{W}$ is used as an intermediate value only, because the set $\mathcal{W}$ is intersected with an interval before it is used for further computation.

The origin of this expression is the problem of finding the set $\mathcal{W}$ of all solutions w with

$$y \cdot (r - w) = x, \quad \text{for any } x \in [x] \text{ and } y \in [y]. \tag{2}$$

Since 1968, many different authors have dealt with different kinds of definitions of extended interval arithmetic ([2], [4], [5], [6], [7], [8], [12], [13], [15], [16], and many others). Some of these definitions are very general, others very special for the case described above. Taking a closer look on some of the special definitions of extended interval division, we will see that they do not treat all important cases correctly or that they even violate the rules of inclusion isotonicity.

In this paper, we first demonstrate the difficulties arising from some of the old definitions. Then we formulate the basic aim, concept, and properties important for defining the extended interval division. Finally, we give a proper way for defining the extended interval operations needed in our special case of interval Newton like methods. In addition, we present sample applications and implementations of our extended interval arithmetic.

2 Notation and Basic Properties

Throughout the present paper, we denote real numbers by x, y, etc. and real bounded and closed intervals by $[x] = [\underline{x}, \overline{x}], [y] = [\underline{y}, \overline{y}]$, etc., where $\inf[x] = \min[x] = \underline{x}$, $\sup[x] = \max[x] = \overline{x}$, $\inf[y] = \min[y] = \underline{y}$, $\sup[y] = \max[y] = \overline{y}$, etc. The set of compact real intervals is denoted by $I\!\!R := \{[\underline{a}, \overline{a}] \mid \underline{a} \leq \overline{a},\ \underline{a}, \overline{a} \in I\!\!R\}$. The midpoint of the interval $[x] \in I\!\!R$ is defined by $m([x]) = (\underline{x} + \overline{x})/2$.

The elementary real operations $\circ \in \{+, -, \cdot, /\}$ are extended to interval arguments $[x], [y]$ by defining the result of an *elementary interval operation* to be the set of real numbers which results from combining any two numbers contained in $[x]$ and in $[y]$. That is,

$$[x] \circ [y] := \{x \circ y \mid x \in [x], y \in [y]\}. \tag{3}$$

Of course, the definition of $[x]/[y]$ is restricted to intervals $[y]$ with $0 \notin [y]$.

The elementary operations are *inclusion isotonic.* That means:

$$[x] \subseteq [x'],\ [y] \subseteq [y'] \implies [x] \circ [y] \subseteq [x'] \circ [y'], \quad \circ \in \{+, -, \cdot, /\}.$$

We call a function $F : I\!I\!R \to I\!I\!R$ an *inclusion function* of $f : I\!R \to I\!R$ in $[x] \in I\!I\!R$, if $x \in [x]$ implies $f(x) \in F([x])$. In other words, $f_{\mathrm{rg}}([x]) \subseteq F([x])$, where $f_{\mathrm{rg}}([x])$ is the range of the function f on $[x]$. The inclusion function of the derivative of f is denoted by F'. It is assumed in the following that the inclusion functions have the *isotonicity* property, i.e. $[x] \subseteq [y]$ implies $F([x]) \subseteq F([y])$.

Moreover, we use the following notation for $[x], [y] \in I\!I\!R$

$$[x] \overset{\circ}{\subset} [y] :\Longleftrightarrow \underline{y} < \underline{x} \wedge \overline{x} < \overline{y},$$

which we call the *inner inclusion* relation.

We extend the space of real intervals by permitting the bounds of intervals to be one of the ideal points $-\infty$ and $+\infty$. Thus, the set of *extended real intervals* is

$$I\!I\!R^\star := I\!I\!R \cup \{[-\infty, r] \mid r \in I\!R\} \cup \{[r, +\infty] \mid r \in I\!R\} \cup \{[-\infty, +\infty]\}. \tag{4}$$

That is, $I\!I\!R^\star$ is the set of real intervals completed by those unbounded intervals whose lower and upper bounds may be infinity. For instance, $[-\infty, r]$ is another notation for the set $\{x \in I\!R \mid x \leq r\}$, and $[-\infty, +\infty]$ denotes the entire real axis.

3 Problems with Some of the "Older" Definitions of "/"

Ratschek, Rokne (1988)

In [15], Ratschek and Rokne slightly modify a definition of Hansen [4]. For finite intervals $[x], [y] \in I\!\!R$ they define the *extended interval division* by

$$[x]/[y] := \begin{cases} [-\infty, +\infty] & \text{if } \underline{x} < 0 < \overline{x} \text{ or } [y] = 0, \\ [\overline{x}/\underline{y}, +\infty] & \text{if } \overline{x} \le 0 \text{ and } \underline{y} < \overline{y} = 0, \\ [-\infty, \overline{x}/\overline{y}] \cup [\overline{x}/\underline{y}, +\infty] & \text{if } \overline{x} \le 0 \text{ and } \underline{y} < 0 < \overline{y}, \\ [-\infty, \overline{x}/\overline{y}] & \text{if } \overline{x} \le 0 \text{ and } 0 = \underline{y} < \overline{y}, \\ [-\infty, \underline{x}/\underline{y}] & \text{if } 0 \le \underline{x} \text{ and } \underline{y} < \overline{y} = 0, \\ [-\infty, \underline{x}/\underline{y}] \cup [\underline{x}/\overline{y}, +\infty] & \text{if } 0 \le \underline{x} \text{ and } \underline{y} < 0 < \overline{y}, \\ [\underline{x}/\overline{y}, +\infty] & \text{if } 0 \le \underline{x} \text{ and } 0 = \underline{y} < \overline{y}. \end{cases} \tag{5}$$

First of all, the different cases of Definition (5) are overlapping, and therefore it is possible to produce two different results for $[x] = [0, 0]$.

Second, the case $0 \in [x] \wedge 0 \in [y]$ is not treated correctly. For example, we want to apply one interval Newton step to the perturbed problem $f(x) = 0$ for $f(x) = [0, 4] \cdot x^3$ in $[x] = [1, 2]$. It is easy to see, that all points in $[x]$ must be in the solution set of that problem. Now, if we compute

$$N([x]) = (m([x]) - f(m([x]))/f'([x])) \cap [x],$$

we get (applying the last case in (5))

$$\begin{aligned} N([1,2]) &= (1.5 - [0, 13.5]/[0, 48]) \cap [1, 2] \\ &= (1.5 - [0, +\infty]) \cap [1, 2] \\ &= [-\infty, 1.5] \cap [1, 2] \\ &= [1, 1.5], \end{aligned}$$

and half of the solution set is lost!

Moreover, Definition (5) is not inclusion isotonic in all cases. With $a, b \in I\!\!R$ and $a, b > 0$, we have $[0] \subseteq [0, b]$, but (5) delivers

$$[0, a]/[0] = [-\infty, +\infty] \not\subseteq [0, +\infty] = [0, a]/[0, b].$$

Hansen (1992)

For finite intervals $[x], [y] \in I\!\!R$ with $\underline{y} < \overline{y}$, Hansen defines the *extended interval division* in his book from 1992 [5] by

$$[x]/[y] := \begin{cases} [\overline{x}/\underline{y}, +\infty] & \text{if } \overline{x} \le 0 \text{ and } \underline{y} < \overline{y} = 0, \\ [-\infty, \overline{x}/\overline{y}] \cup [\overline{x}/\underline{y}, +\infty] & \text{if } \overline{x} \le 0 \text{ and } \underline{y} < 0 < \overline{y}, \\ [-\infty, \overline{x}/\overline{y}] & \text{if } \overline{x} \le 0 \text{ and } 0 = \underline{y} < \overline{y}, \\ [-\infty, +\infty] & \text{if } \underline{x} < 0 < \overline{x}, \\ [-\infty, \underline{x}/\underline{y}] & \text{if } 0 \le \underline{x} \text{ and } \underline{y} < \overline{y} = 0, \\ [-\infty, \underline{x}/\underline{y}] \cup [\underline{x}/\overline{y}, +\infty] & \text{if } 0 \le \underline{x} \text{ and } \underline{y} < 0 < \overline{y}, \\ [\underline{x}/\overline{y}, +\infty] & \text{if } 0 \le \underline{x} \text{ and } 0 = \underline{y} < \overline{y}. \end{cases} \tag{6}$$

As with the Ratschek/Rokne definition, the different cases of Definition (6) are overlapping, and we can obtain two different results for $[x] = [0, 0]$. Therefore, the definition can also not be inclusion isotonic. For example, with $a, b \in I\!\!R$ and $a, b > 0$, we have that $[0] \subseteq [0, a]$ but cases 3 and 7 of Definition (6) deliver

$$[0]/[0, b] = [-\infty, 0] \not\subseteq [0, +\infty] = [0, a]/[0, b].$$

Second, it is assumed that $\underline{y} < \overline{y}$. So, the case $[x]/[0, 0]$ is not treated at all (but it is a possible case, see the sample application).

Moreover, as for Definition (5), the case $0 \in [x] \ \wedge \ 0 \in [y]$ is not treated correctly. These rules produce the same results for our example from above.

Hammer, Hocks, Kulisch, Ratz (1993)

In the toolbox book [2], the authors define the *extended interval division* for finite intervals $[x], [y] \in I\!\!R$ by

$$[x]/[y] := \begin{cases} [-\infty, +\infty] & \text{if } \underline{x} < 0 < \overline{x} \text{ or } [x] = 0 \text{ or } [y] = 0, \\ [\overline{x}/\underline{y}, +\infty] & \text{if } \overline{x} \le 0 \text{ and } \underline{y} < \overline{y} = 0, \\ [-\infty, \overline{x}/\overline{y}] \cup [\overline{x}/\underline{y}, +\infty] & \text{if } \overline{x} \le 0 \text{ and } \underline{y} < 0 < \overline{y}, \\ [-\infty, \overline{x}/\overline{y}] & \text{if } \overline{x} \le 0 \text{ and } 0 = \underline{y} < \overline{y}, \\ [-\infty, \underline{x}/\underline{y}] & \text{if } 0 \le \underline{x} \text{ and } \underline{y} < \overline{y} = 0, \\ [-\infty, \underline{x}/\underline{y}] \cup [\underline{x}/\overline{y}, +\infty] & \text{if } 0 \le \underline{x} \text{ and } \underline{y} < 0 < \overline{y}, \\ [\underline{x}/\overline{y}, +\infty] & \text{if } 0 \le \underline{x} \text{ and } 0 = \underline{y} < \overline{y}. \end{cases} \tag{7}$$

Here, the different cases of Definition (7) are non-overlapping if we assume $0 \neq [x]$ in cases 2 to 7 (as it is assumed by the authors in their implementation of (7), see [2, page 95]). Therefore the results are unique. Moreover, the case $[y] = 0$ is at least incorporated.

As for Hansen's definition, the case $0 \in [x] \ \wedge \ 0 \in [y]$ is not treated correctly. These rules produce the same results for our example from above.

Similar to Definition (5), Definition (7) is is not inclusion isotonic in all cases. With $a, b \in I\!R$ and $a, b > 0$, we have that $[0] \subseteq [0, a]$ and $[0] \subseteq [0, b]$ but (7) delivers

$$[0]/[0, b] = [-\infty, +\infty] \not\subseteq [0, +\infty] = [0, a]/[0, b]$$

and

$$[0, a]/[0] = [-\infty, +\infty] \not\subseteq [0, +\infty] = [0, a]/[0, b].$$

These three examples demonstrate the need of a properly defined extended interval division.

4 A New Approach

From the definition of the standard interval division (3) with $0 \notin [y]$, we have that the quotient $[x]/[y] = [x] \cdot [1/\overline{y}, 1/\underline{y}]$ is the set

$$S([x], [y]) := \{z \in I\!R \mid z = x/y, \quad x \in [x], y \in [y]\}. \tag{8}$$

But if we allow $[y]$ to contain the point zero, we cannot use this definition because $x/0$ is not defined.

Therefore, we go back to the original problem (2). So, dividing $[x]$ by $[y]$ can be interpreted as the aim of finding all points z for which $y \cdot z = x$ for any $x \in [x]$ and $y \in [y]$. Thus we fix

Definition 4.1 *Let $[x], [y] \in II\!R$. Then we call the set*

$$\mathcal{S}^\star([x], [y]) := \{z \in I\!R \mid y \cdot z = x, \quad x \in [x], \; y \in [y]\}. \tag{9}$$

the extended solution set for $[x]/[y]$.

Lemma 4.2 *For $[x], [y] \in II\!R$ with $0 \notin [y]$, the result of the standard interval division satisfies $[x]/[y] = S([x], [y]) = \mathcal{S}^\star([x], [y])$.*

Proof: From [1] we know that $[x]/[y] = S([x], [y])$. Since $y \neq 0$ for all $y \in [y]$, we have

$$\begin{aligned} S([x], [y]) &= \{z \mid z = x/y, \quad x \in [x], y \in [y]\} \\ &= \{z \mid y \cdot z = x, \quad x \in [x], y \in [y]\} \\ &= \mathcal{S}^\star([x], [y]). \end{aligned}$$

□

To see how we can provide the extended rules for division, we first look at

Example 4.3 Let $[x]$, $[y] \in II\!R$ with $[x] = [4, 5]$ and $[y] = [-1, 2]$. Since $0 \notin [4, 5]$, there is no $x \in [4, 5]$ for which a $z \in I\!R$ exists with $0 \cdot z = 0 = x$. Therefore we can exclude 0 from $[y]$, and we have

$$\begin{aligned} \mathcal{S}^\star([x], [y]) = & \quad \{z \in I\!R \mid y \cdot z = x, \; x \in [4, 5], \; y \in [-1, 0)\} \\ \cup & \quad \{z \in I\!R \mid y \cdot z = x, \; x \in [4, 5], \; y \in (0, 2]\}. \end{aligned}$$

Let $\varepsilon > 0$, then set

$$\begin{aligned}\mathcal{S}^\star_\varepsilon([x],[y]) := &\quad \{z \in I\!R \mid y \cdot z = x,\ x \in [4,5],\ y \in [-1,-\varepsilon]\} \\ &\cup\ \{z \in I\!R \mid y \cdot z = x,\ x \in [4,5],\ y \in [\varepsilon,2]\},\end{aligned}$$

and we have $\mathcal{S}^\star_\varepsilon([x],[y]) \longrightarrow \mathcal{S}^\star([x],[y])$ for $\varepsilon \longrightarrow 0$. Thus,

$$\mathcal{S}^\star_\varepsilon([x],[y]) = [-5/\varepsilon, 4] \cup [2, 5/\varepsilon] \longrightarrow [-\infty, 4] \cup [2, +\infty] = \mathcal{S}^\star([x],[y]),$$

and the result is given as the union of two extended real intervals, $\mathcal{S}^\star([4,5],[-1,2]) = [-\infty,-4] \cup [2,+\infty]$.

Example 4.3 is typical for the common case of an extended interval division, which may be illustrated by punching out a little gap around the origin of the real axis. In general, the result may be represented by zero (empty set), one, or two extended intervals.

5 A Proper Extended Definition of "/" and "−"

Definition 5.1 *We call* $[\not\,]$ *with* $[\not\,] \subseteq [x]$ *for all* $[x] \in I\!I\!R^\star \cup \{[\not\,]\}$ *the* empty interval.

Definition 5.2 *For finite intervals* $[x],[y] \in I\!I\!R$*, we define the* extended interval division *by*

$$[x]/[y] := \begin{cases} [x]\cdot[1/\overline{y}, 1/\underline{y}] & \text{if } 0 \notin [y], \\ [-\infty,+\infty] & \text{if } 0 \in [x] \ \wedge\ 0 \in [y], \\ [\overline{x}/\underline{y}, +\infty] & \text{if } \overline{x} < 0 \ \wedge\ \underline{y} < \overline{y} = 0, \\ [-\infty, \overline{x}/\overline{y}] \cup [\overline{x}/\underline{y}, +\infty] & \text{if } \overline{x} < 0 \ \wedge\ \underline{y} < 0 < \overline{y}, \\ [-\infty, \overline{x}/\overline{y}] & \text{if } \overline{x} < 0 \ \wedge\ 0 = \underline{y} < \overline{y}, \\ [-\infty, \underline{x}/\underline{y}] & \text{if } 0 < \underline{x} \ \wedge\ \underline{y} < \overline{y} = 0, \\ [-\infty, \underline{x}/\underline{y}] \cup [\underline{x}/\overline{y}, +\infty] & \text{if } 0 < \underline{x} \ \wedge\ \underline{y} < 0 < \overline{y}, \\ [\underline{x}/\overline{y}, +\infty] & \text{if } 0 < \underline{x} \ \wedge\ 0 = \underline{y} < \overline{y}, \\ [\not\,] & \text{if } 0 \notin [x] \ \wedge\ 0 = [y]. \end{cases} \tag{10}$$

This definition coresponds to a definition used by Kearfott and Novoa in [14] (implemented in [9]), which, in contrast to our definition, makes use of the ideal points $-\infty$ and $+\infty$. In their definition, the empty interval corresponds to the set $\{-\infty\} \cup \{+\infty\}$ and all cases contain those two ideal points in the result set to preserve inclusion isotonicity.

A definition of extended interval division in the spirit of the present paper will also appear in the forthcoming book of Kearfott [10].

Theorem 5.3 *For $[x], [y] \in I\mathbb{R}$, the result of the extended interval division according to Definition 5.2 satisfies $[x]/[y] = \mathcal{S}^\star([x],[y])$.*

Proof: Let $\varepsilon > 0$, then treat the different cases of Definition 5.2 in the following manner:

a) $0 \notin [y]$: According to Lemma 4.2, we have

$$[x]/[y] = S([x],[y]) = \mathcal{S}^\star([x],[y]).$$

b) $0 \in [x] \ \wedge \ 0 \in [y]$: For $x = 0 \ \wedge \ y = 0$, all $z \in \mathbb{R}$ fulfill $y \cdot z = x$. Therefore

$$\mathcal{S}^\star([x],[y]) = \mathbb{R} = [-\infty, +\infty] = [x]/[y].$$

c) $\overline{x} < 0 \ \wedge \ \underline{y} < \overline{y} = 0$: Since $0 \notin [x]$, there is no $x \in [x]$, for which a $z \in \mathbb{R}$ exists with $0 \cdot z = x$. Therefore we have

$$\begin{aligned}
\mathcal{S}^\star([x],[y]) &= \{z \in \mathbb{R} \mid y \cdot z = x,\ x \in [\underline{x}, \overline{x}],\ y \in [\underline{y}, 0]\} \\
&= \{z \in \mathbb{R} \mid y \cdot z = x,\ x \in [\underline{x}, \overline{x}],\ y \in [\underline{y}, 0)\} \\
&= \lim_{\varepsilon \to +0} \{z \in \mathbb{R} \mid y \cdot z = x,\ x \in [\underline{x}, \overline{x}],\ y \in [\underline{y}, -\varepsilon]\} \\
&= \lim_{\varepsilon \to +0} S([\underline{x}, \overline{x}], [\underline{y}, -\varepsilon]) \\
&= \lim_{\varepsilon \to +0} [\underline{x}, \overline{x}]/[\underline{y}, -\varepsilon] \\
&= \lim_{\varepsilon \to +0} [\overline{x}/\underline{y}, -\underline{x}/\varepsilon] \\
&= [\overline{x}/\underline{y}, +\infty] \\
&= [x]/[y].
\end{aligned}$$

d) $\overline{x} < 0 \ \wedge \ \underline{y} < 0 < \overline{y}$: Since $0 \notin [x]$, there is no $x \in [x]$, for which a $z \in \mathbb{R}$ exists with $0 \cdot z = x$. Therefore we have

$$\begin{aligned}
\mathcal{S}^\star([x],[y]) &= \{z \in \mathbb{R} \mid y \cdot z = x,\ x \in [\underline{x}, \overline{x}],\ y \in [\underline{y}, \overline{y}]\} \\
&= \{z \in \mathbb{R} \mid y \cdot z = x,\ x \in [\underline{x}, \overline{x}],\ y \in [\underline{y}, 0)\} \ \cup \\
&\qquad \{z \in \mathbb{R} \mid y \cdot z = x,\ x \in [\underline{x}, \overline{x}],\ y \in (0, \overline{y}]\} \\
&= \lim_{\varepsilon \to +0} \Big(\{z \in \mathbb{R} \mid y \cdot z = x,\ x \in [\underline{x}, \overline{x}],\ y \in [\underline{y}, -\varepsilon]\} \ \cup \\
&\qquad\qquad \{z \in \mathbb{R} \mid y \cdot z = x,\ x \in [\underline{x}, \overline{x}],\ y \in [+\varepsilon, \overline{y}]\} \Big) \\
&= \lim_{\varepsilon \to +0} \Big(S([\underline{x}, \overline{x}], [\underline{y}, -\varepsilon]) \cup S([\underline{x}, \overline{x}], [+\varepsilon, \overline{y}]) \Big) \\
&= \lim_{\varepsilon \to +0} \Big([\underline{x}, \overline{x}]/[\underline{y}, -\varepsilon] \cup [\underline{x}, \overline{x}]/[+\varepsilon, \overline{y}] \Big) \\
&= \lim_{\varepsilon \to +0} \Big([\overline{x}/\underline{y}, -\underline{x}/\varepsilon] \cup [\underline{x}/\varepsilon, \overline{x}/\overline{y}] \Big) \\
&= [\overline{x}/\underline{y}, +\infty] \cup [-\infty, \overline{x}/\overline{y}] \\
&= [x]/[y].
\end{aligned}$$

e) $\overline{x} < 0 \;\wedge\; 0 = \underline{y} < \overline{y}$: Since $0 \notin [x]$, there is no $x \in [x]$, for which a $z \in I\!R$ exists with $0 \cdot z = x$. Therefore we have

$$\begin{aligned}
\mathcal{S}^\star([x],[y]) &= \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [0,\overline{y}]\} \\
&= \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in (0,\overline{y}]\} \\
&= \lim_{\varepsilon\to+0} \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [\varepsilon,\overline{y}]\} \\
&= \lim_{\varepsilon\to+0} S([\underline{x},\overline{x}],[\varepsilon,\overline{y}]) \\
&= \lim_{\varepsilon\to+0} [\underline{x},\overline{x}]/[\varepsilon,\overline{y}] \\
&= \lim_{\varepsilon\to+0} [\underline{x}/\varepsilon,\overline{x}/\overline{y}] \\
&= [-\infty,\overline{x}/\overline{y}] \\
&= [x]/[y].
\end{aligned}$$

f) $0 < \underline{x} \;\wedge\; \underline{y} < \overline{y} = 0$: Since $0 \notin [x]$, there is no $x \in [x]$, for which a $z \in I\!R$ exists with $0 \cdot z = x$. Therefore we have

$$\begin{aligned}
\mathcal{S}^\star([x],[y]) &= \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [\underline{y},0]\} \\
&= \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [\underline{y},0)\} \\
&= \lim_{\varepsilon\to+0} \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [\underline{y},-\varepsilon]\} \\
&= \lim_{\varepsilon\to+0} S([\underline{x},\overline{x}],[\underline{y},-\varepsilon]) \\
&= \lim_{\varepsilon\to+0} [\underline{x},\overline{x}]/[\underline{y},-\varepsilon] \\
&= \lim_{\varepsilon\to+0} [-\overline{x}/\varepsilon,\underline{x}/\underline{y}] \\
&= [-\infty,\underline{x}/\underline{y}] \\
&= [x]/[y].
\end{aligned}$$

g) $0 < \underline{x} \;\wedge\; \underline{y} < 0 < \overline{y}$: Since $0 \notin [x]$, there is no $x \in [x]$, for which a $z \in I\!R$ exists with $0 \cdot z = x$. Therefore we have

$$\begin{aligned}
\mathcal{S}^\star([x],[y]) &= \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [\underline{y},\overline{y}]\} \\
&= \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [\underline{y},0)\} \;\cup \\
&\quad\; \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in (0,\overline{y}]\} \\
&= \lim_{\varepsilon\to+0} \Big(\{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [\underline{y},-\varepsilon]\} \;\cup \\
&\qquad\qquad \{z \in I\!R \mid y \cdot z = x,\; x \in [\underline{x},\overline{x}],\; y \in [+\varepsilon,\overline{y}]\} \Big) \\
&= \lim_{\varepsilon\to+0} \Big(S([\underline{x},\overline{x}],[\underline{y},-\varepsilon]) \cup S([\underline{x},\overline{x}],[+\varepsilon,\overline{y}]) \Big) \\
&= \lim_{\varepsilon\to+0} \Big([\underline{x},\overline{x}]/[\underline{y},-\varepsilon] \cup [\underline{x},\overline{x}]/[+\varepsilon,\overline{y}] \Big) \\
&= \lim_{\varepsilon\to+0} \Big([-\overline{x}/\varepsilon,\underline{x}/\underline{y}] \cup [\underline{x}/\overline{y},\overline{x}/\varepsilon] \Big) \\
&= [-\infty,\underline{x}/\underline{y}] \cup [\underline{x}/\overline{y},+\infty] \\
&= [x]/[y].
\end{aligned}$$

h) $0 < \underline{x} \ \wedge \ 0 = \underline{y} < \overline{y}$: Since $0 \notin [x]$, there is no $x \in [x]$, for which a $z \in I\!R$ exists with $0 \cdot z = x$. Therefore we have

$$\begin{aligned}
\mathcal{S}^\star([x],[y]) &= \{z \in I\!R \mid y \cdot z = x,\ x \in [\underline{x},\overline{x}],\ y \in [0,\overline{y}]\} \\
&= \{z \in I\!R \mid y \cdot z = x,\ x \in [\underline{x},\overline{x}],\ y \in (0,\overline{y}]\} \\
&= \lim_{\varepsilon \to +0} \{z \in I\!R \mid y \cdot z = x,\ x \in [\underline{x},\overline{x}],\ y \in [\varepsilon,\overline{y}]\} \\
&= \lim_{\varepsilon \to +0} S([\underline{x},\overline{x}],[\varepsilon,\overline{y}]) \\
&= \lim_{\varepsilon \to +0} [\underline{x},\overline{x}]/[\varepsilon,\overline{y}] \\
&= \lim_{\varepsilon \to +0} [\underline{x}/\overline{y},\overline{x}/\varepsilon] \\
&= [\underline{x}/\overline{y},+\infty] \\
&= [x]/[y].
\end{aligned}$$

i) $0 \notin [x] \ \wedge \ 0 = [y]$: Since $0 \notin [x]$, there is no $x \in [x]$, for which a $z \in I\!R$ exists with $0 \cdot z = x$. Therefore, since 0 is the only element of $[y]$ we have

$$\mathcal{S}^\star([x],[y]) = \{z \in I\!R \mid y \cdot z = x,\ x \in [\underline{x},\overline{x}],\ y \in [\,/\!\!\!\,]\} = [\,/\!\!\!\,] = [x]/[y].$$

$\square$

Theorem 5.4 *The extended interval division according to Definition 5.2 is inclusion isotonic.*

Proof: From Theorem 5.3 we know that $[x]/[y] = \mathcal{S}^\star([x],[y])$ for all $[x],[y] \in I\!I\!R$. Now, let $[x],[x'],[y],[y'] \in I\!I\!R$, $[x] \subseteq [x']$, and $[y] \subseteq [y']$. Then

$$\begin{aligned}
[x]/[y] &= \mathcal{S}^\star([x],[y]) \\
&= \{z \in I\!R \mid y \cdot z = x,\ x \in [x],\ y \in [y]\} \\
&\subseteq \{z \in I\!R \mid y \cdot z = x,\ x \in [x'],\ y \in [y']\} \\
&= \mathcal{S}^\star([x'],[y']) \\
&= [x']/[y'].
\end{aligned}$$

$\square$

In most algorithms where extended interval arithmetic is applied, it is followed by an intersection with some finite interval. The result after intersection is an empty set (interval) or one or two finite intervals. In this sense, our extended interval arithmetic is a tool to generate and deal temporarily with infinite intervals.

Example 5.5 Consider Example 4.3. We look for $([x]/[y]) \cap [-5,4]$. The division yields $[-\infty,-4] \cup [2,+\infty]$. Intersection with $[-5,4]$ yields two finite intervals $[-5,-4] \cup [2,4]$. Similarly, $([x]/[y]) \cap [-2,4] = [2,4]$, and $([x]/[y]) \cap [-3,1] = [\,/\!\!\!\,]$.

In addition to the operation of an extended division, we only need one more extended operation, an extended subtraction.

Definition 5.6 *Let $r \in I\!R$ and $[z] \in I\!I\!R^{\star} \cup \{[\not\,]\}$. Then we call the set*

$$\mathcal{W}^{\star}(r,[z]) := \{w \in I\!R \mid w = r - z, \quad z \in [z]\}. \tag{11}$$

the extended solution set for $r - [z]$.

Definition 5.7 *For $r \in I\!R$ and $[z] \in I\!I\!R^{\star} \cup \{[\not\,]\}$, we define the* extended interval subtraction *by*

$$r - [z] := \begin{cases} [r - \overline{z}, r - \underline{z}] & \textit{if } [z] \in I\!I\!R \\ [-\infty, +\infty] & \textit{if } [z] = [-\infty, +\infty] \\ [-\infty, r - \underline{z}] & \textit{if } [z] = [\underline{z}, +\infty] \\ [r - \overline{z}, +\infty] & \textit{if } [z] = [-\infty, \overline{z}] \\ [\not\,] & \textit{if } [z] = [\not\,] \end{cases} \tag{12}$$

Theorem 5.8 *For $r \in I\!R$ and $[z] \in I\!I\!R^{\star} \cup \{[\not\,]\}$, the result of the extended interval subtraction according to Definition 5.7 satisfies $r - [z] = \mathcal{W}^{\star}(r,[z])$.*

Proof: Let $\varepsilon > 0$, then we treat the different cases of Definition 5.7 in the following manner:

a) $[z] \in I\!I\!R$: This case corresponds to the standard definition of interval subtraction, since $[z]$ is finite.

b) $[z] = [-\infty, +\infty]$:

$$\begin{aligned} \mathcal{W}^{\star}(r,[z]) &= \{w \in I\!R \mid w = r - z,\ z \in [-\infty, \infty]\} \\ &= \lim_{\varepsilon \to +0} \{w \in I\!R \mid w = r - z,\ z \in [-1/\varepsilon, 1/\varepsilon]\} \\ &= \lim_{\varepsilon \to +0} [r - 1/\varepsilon, r + 1/\varepsilon] \\ &= [-\infty, +\infty] \\ &= r - [z]. \end{aligned}$$

c) $[z] = [\underline{z}, +\infty]$:

$$\begin{aligned} \mathcal{W}^{\star}(r,[z]) &= \{w \in I\!R \mid w = r - z,\ z \in [\underline{z}, \infty]\} \\ &= \lim_{\varepsilon \to +0} \{w \in I\!R \mid w = r - z,\ z \in [\underline{z}, 1/\varepsilon]\} \\ &= \lim_{\varepsilon \to +0} [r - 1/\varepsilon, r - \underline{z}] \\ &= [-\infty, r - \underline{z}] \\ &= r - [z]. \end{aligned}$$

d) $[z] = [-\infty, \overline{z}]$:

$$\begin{aligned} \mathcal{W}^\star(r,[z]) &= \{w \in I\!R \mid w = r - z,\ z \in [-\infty, \overline{z}]\} \\ &= \lim_{\varepsilon \to +0} \{w \in I\!R \mid w = r - z,\ z \in [-1/\varepsilon, \overline{z}]\} \\ &= \lim_{\varepsilon \to +0} [r - \overline{z}, r + 1/\varepsilon] \\ &= [r - \overline{z}, +\infty] \\ &= r - [z]. \end{aligned}$$

e) $[z] = [/\!\!\!\;]$:

$$\begin{aligned} \mathcal{W}^\star(r,[z]) &= \{w \in I\!R \mid w = r - z,\ z \in [/\!\!\!\;]\} \\ &= [/\!\!\!\;] \\ &= r - [z]. \end{aligned}$$

□

Theorem 5.9 *The extended interval subtraction according to Definition 5.7 is inclusion isotonic.*

Proof: From Theorem 5.8 we know that $r - [z] = \mathcal{W}^\star(r,[z])$ for all $r \in I\!R$ and all $[z] \in I\!I\!R^\star$. Now, let $[z], [z'] \in I\!I\!R^\star \cup \{[/\!\!\!\;]\}$ and $[z] \subseteq [z']$. Then

$$\begin{aligned} r - [z] &= \mathcal{W}^\star(r,[z]) \\ &= \{w \in I\!R \mid w = r - z,\ z \in [z]\} \\ &\subseteq \{w \in I\!R \mid w = r - z,\ z \in [z']\} \\ &= \mathcal{W}^\star(r,[z']) \\ &= r - [z']. \end{aligned}$$

□

6 Sample Applications

Example 6.1 We compare *extended interval Newton iteration*

$$[x]^{(k+1)} = N^\star([x]^{(k)}) = N([x]^{(k)}) \cap [x]^{(k)}, \qquad k = 0, 1, 2, \ldots,$$

where

$$N([x]) = m([x]) - f(m([x]))/F'([x])$$

with *standard interval Newton bisection iteration*, i.e.

$$[x]^{(k+1)} = B([x]^{(k)}), \qquad k = 0, 1, 2, \ldots,$$

where

$$B([x]) = \begin{cases} N([x]) \cap [x] & \text{if } 0 \notin F'([x]), \\ [\underline{x}, m([x])] \cup [m([x]), \overline{x}] & \text{otherwise.} \end{cases}$$

For both methods, $[x]^{(k)} \in I\!P I\!R$ for $k > 0$, i.e.

$$[x]^{(k)} = \bigcup_{j=1}^{n_k} [x]_j^{(k)}$$

with $[x]_j^{(k)} \in I\!R$ for $j = 1, \ldots, n_k$ during the iteration, and we define

$$N^\star([x]^{(k)}) := \bigcup_{j=1}^{n_k} N^\star([x]_j^{(k)})$$

and

$$B([x]^{(k)}) := \bigcup_{j=1}^{n_k} B([x]_j^{(k)}).$$

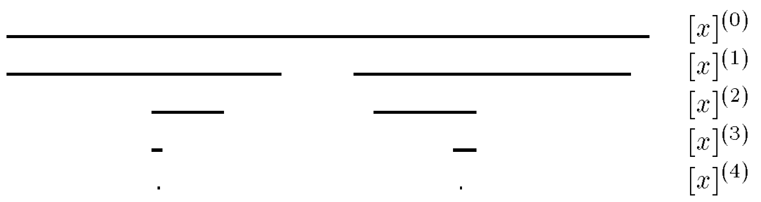
If we apply the extended interval Newton iteration to the function

$$f(x) = x^2 - 4x + 3$$

and starting interval $[x]^{(0)} = [0, 4.25]$, we obtain

$$\begin{aligned} [x]^{(1)} &= [0.0000, 1.814] \cup [2.296, 4.125], \\ [x]^{(2)} &= [0.9555, 1.431] \cup [2.425, 3.102], \\ &\vdots \\ [x]^{(5)} &= [0.9999, 1.001] \cup [2.999, 3.001]. \end{aligned}$$

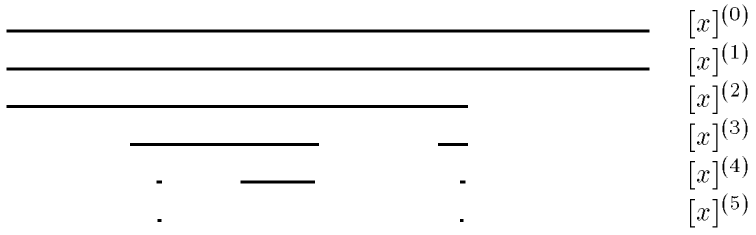
We need only 9 applications of the extended interval Newton operator $N^\star$ to guarantee a relative diameter of 10^{-4} for the final intervals. After two iterations, we are already able to verify the local uniqueness of a zero within the intervals $[x]_1^{(2)}$ and $[x]_2^{(2)}$, since $[x]_1^{(2)} \overset{\circ}{\subset} [x]_1^{(1)}$ and $[x]_2^{(2)} \overset{\circ}{\subset} [x]_2^{(1)}$. All printed results are rounded outwards to four significant digits. Presenting the interval iterates graphically, we have



If we use the standard interval Newton bisection iteration, we obtain

$$\begin{aligned} [x]^{(1)} &= [0.000, 2.063] \cup [2.062, 4.125], \\ [x]^{(2)} &= [0.000, 1.032] \cup [1.031, 2.063] \cup [2.062, 3.048], \\ &\vdots \\ [x]^{(6)} &= [0.9999, 1.001] \cup [2.999, 3.001]. \end{aligned}$$

We need a total number of 5 bisections and 14 applications of the standard interval Newton operator to guarantee a relative diameter of 10^{-4} for the final intervals. Uniqueness can not be verified until the final iteration. Presenting the interval iterates graphically, we have



Example 6.2 Let $f : I\!R^2 \to I\!R^2$ with $f_1(x) = x_1^2 \cdot (1 - x_2)$ and $f_2(x) = x_1 + x_2^2 - 1$. Then the points $(0,1)^\mathsf{T}$ and $(0,-1)^\mathsf{T}$ are the two zeros of f within in the box $([x_1],[x_2])^\mathsf{T} = ([-1,1],[-1,1])^\mathsf{T}$.

Now we want to perform special componentwise one-dimensional interval Newton steps with the perturbed functions $g : I\!R \to I\!R$ and $h : I\!R \to I\!R$ defined by

$$g(x_2) := [x_1]^2 \cdot (1 - x_2)$$

and

$$h(x_1) := x_1 + [x_2]^2 - 1.$$

Thus we use

$$\begin{aligned} N_g([x_2]) &:= m([x_2]) - \frac{g(m([x_2]))}{G'([x_2])} \\ &= m([x_2]) + \frac{[x_1]^2 \cdot (1 - m([x_2])}{[x_1]^2} \end{aligned}$$

and

$$\begin{aligned} N_h([x_1]) &:= m([x_1]) - \frac{h(m([x_1]))}{H'([x_1])} \\ &= m([x_1]) - \frac{[x_2]^2 + m([x_1]) - 1}{1}. \end{aligned}$$

We now compare the result using definitions (5), (6), or (7) with the result using Definition 5.2 when applying N_g, then N_h, and then again N_g, to narrow the enclosure $([x_1],[x_2])$ of the zeros of f.

Using (5), (6), or (7), we get

$$\begin{aligned} [x_2] &:= [x_2] \cap N_g([x_2]) \\ &= [-1,1] \cap (0 + \frac{[-1,1]^2}{[-1,1]^2}) \\ &= [-1,1] \cap \frac{[0,1]}{[0,1]} \end{aligned}$$

$$\begin{aligned}
&= [-1,1] \cap [0,+\infty] \\
&= [0,1], \\
[x_1] &:= [x_1] \cap N_h([x_1]) \\
&= [-1,1] \cap (0 - \frac{[0,1]^2 - 1}{1}) \\
&= [-1,1] \cap -\frac{[0,1] - 1}{1} \\
&= [-1,1] \cap [0,1] \\
&= [0,1], \\
[x_2] &:= [x_2] \cap N_g([x_2]) \\
&= [0,1] \cap (0.5 + \frac{[0,1]^2 \cdot 0.5}{[0,1]^2}) \\
&= [0,1] \cap (0.5 + [0,+\infty]) \\
&= [0,1] \cap [0.5,+\infty] \\
&= [0.5,1].
\end{aligned}$$

Therefore, $(0,-1)^\mathsf{T} \notin ([x_1],[x_2])^\mathsf{T} = ([0,1],[0.5,1])^\mathsf{T}$, so one zero is lost!

Using Definition 5.2, this error is corrected, and we get

$$\begin{aligned}
[x_2] &:= [x_2] \cap N_g([x_2]) \\
&= [-1,1] \cap (0 + \frac{[-1,1]^2}{[-1,1]^2}) \\
&= [-1,1] \cap \frac{[0,1]}{[0,1]} \\
&= [-1,1] \cap [-\infty,+\infty] \\
&= [-1,1] \\
[x_1] &:= [x_1] \cap N_h([x_1]) \\
&= [-1,1] \cap (0 - \frac{[-1,1]^2 - 1}{1}) \\
&= [-1,1] \cap -\frac{[0,1] - 1}{1} \\
&= [-1,1] \cap [0,1] \\
&= [0,1] \\
[x_2] &:= [x_2] \cap N_g([x_2]) \\
&= [-1,1] \cap (0 + \frac{[0,1]^2}{[0,1]^2}) \\
&= [-1,1] \cap [-\infty,+\infty]) \\
&= [-1,1],
\end{aligned}$$

which indicates that a bisection is necessary to achieve further progress in narrowing $[x_2]$.

Example 6.3 If we apply the routine AllZeros from the toolbox book [2] to the (constant) function $f(x) = x - x + 5$ for the starting search region $[-1000, 1000]$, we get

```
Computing all zeros of the function  f(x) = x - x + 5

Search interval      : [-1000,1000]
Tolerance (relative) : 1e-10

1023 calls of XINewton needed!

0 interval enclosure(s)

There is no zero within the search interval.
```

The final result is correct. However, since automatic differentiation delivers $f'([x]) = [0,0]$ for all $[x] \in I\!R$, we have a denominator $[0,0]$, and the extended interval division according to (7) results in $[-\infty, +\infty]$. Therefore, many bisections and 1023 recursive calls of XINewton are necessary.

Now, we equip the toolbox modules with the extended interval division according to Definition 5.2. If we then apply the routine AllZeros to the (constant) function $f(x) = x - x + 5$ and the starting search region $[-1000, 1000]$ we get

```
Computing all zeros of the function  f(x) = x - x + 5

Search interval      : [-1000,1000]
Tolerance (relative) : 1e-10

1 call of XINewton needed!

0 interval enclosure(s)

There is no zero within the search interval.
```

This improved behaviour is due to the fact, that for the denominator $[0,0]$ the extended interval division now results in $[\,/\!\!\!\!\;]$. Therefore, only one call of XINewton is necessary to verify that there is no zero in the search region.

7 Two Implementations

In the following, we list two implementations of (10) and (12) in PASCAL–XSC [11] and in C++. These modules are updated versions of the public domain toolbox modules described in [2] and in [3].

7.1 PASCAL–XSC Implementation

```
{-----------------------------------------------------------------------}
{ Purpose: Definition of an extended interval arithmetic which allows    }
{    the division by an interval containing zero.                        }
{ Method: Overloading of operators for arithmetic and lattice operations }
{    of data type 'xinterval'.                                           }
{ Global types, operators, and functions:                                }
{    types      KindType      : component type of extended intervals     }
{               xinterval     : data type for extended intervals         }
{    operators  div, -, **    : operators of extended interval arithmetic}
{    function   EmptyIntval   : delivers empty set as irregular interval }
{-----------------------------------------------------------------------}
module xi_ari;

use i_ari;  { Standard interval arithmetic }

{-----------------------------------------------------------------------}
{ Global type definitions                                                }
{-----------------------------------------------------------------------}
{ An extended interval 'x', represented by the type 'xinterval', is      }
{ defined according to the following rules (a <= b):                     }
{                                                                        }
{ x = [a, b]              : x.kind = Finite,     x.inf = a, x.sup = b    }
{ x = [a, +oo]            : x.kind = PlusInfty,  x.inf = a, x.sup undef. }
{ x = [-oo, a]            : x.kind = MinusInfty, x.sup = a, x.inf undef. }
{ x = [-oo, a] v [b, +oo] : x.kind = Double,     x.inf = b, x.sup = a    }
{ x = [-oo, +oo]          : x.kind = Double,     x.inf = a, x.sup = a    }
{ x = [/]                 : x.kind = Empty,      x.inf  and x.sup undef. }
{                                                                        }
{ In this definition, 'v' stands for the set union and 'oo' for infinity.}
{-----------------------------------------------------------------------}
global type
  KindType     = (Finite, PlusInfty, MinusInfty, Double, Empty);
  xinterval    = global record                        { Extended intervals  }
                     kind     : KindType;             { according to the    }
                     inf, sup : real;                 { definition above    }
                   end;                               {---------------------}

{-----------------------------------------------------------------------}
{ Function 'EmptyIntval' delivers an empty interval (empty set)          }
{-----------------------------------------------------------------------}
global function EmptyIntval : interval;
begin
  EmptyIntval.inf :=  999999999;  { Definition of an irregular interval  }
  EmptyIntval.sup := -999999999;  { EmptyIntval = [999999999,-999999999] }
end;                              {--------------------------------------}

{---------------------------------------------------------------}
{ Extended interval division 'A / B', where 0 in 'B' is allowed. }
{---------------------------------------------------------------}
global operator div (A, B : interval) quotient : xinterval;
  var
    C : interval;
    Q : xinterval;
  begin
    if (0 in B) then { extended interval division }
      begin
        if (0 in A) then
          begin                   { Q = [-oo, +oo] = [-oo, 0] v [0, +oo] }
            Q.kind:= Double;      {--------------------------------------}
            Q.sup := 0;
            Q.inf := 0;
          end
        else if (0 = B) then
          begin                                                 { Q = [/] }
            Q.kind := Empty;                                    {---------}
          end
        else if ((sup(A) < 0) and (sup(B) = 0)) then
          begin                                        { Q = [Q.inf, +oo] }
            Q.kind:= PlusInfty;                        {------------------}
            Q.inf := sup(A) /< inf(B);
          end
        else if ((sup(A) < 0) and (inf(B) < 0) and (sup(B) > 0)) then
          begin                        { Q = [-oo, Q.sup] v [Q.inf, +oo] }
            Q.kind:= Double;           {---------------------------------}
            Q.sup := sup(A) /> sup(B);
            Q.inf := sup(A) /< inf(B);
          end
        else if ((sup(A) < 0) and (inf(B) = 0)) then
          begin                                        { Q = [-oo, Q.sup] }
            Q.kind:= MinusInfty;                       {------------------}
            Q.sup := sup(a) /> sup(b);
          end
        else if ((inf(A) > 0) and (sup(B) = 0)) then
          begin                                        { Q = [-oo, Q.sup] }
            Q.kind:= MinusInfty;                       {------------------}
            Q.sup := inf(A) /> inf(B);
          end
        else if ((inf(A) > 0) and (inf(B) < 0) and (sup(B) > 0)) then
          begin                        { Q = [-oo, Q.sup] v [Q.inf, +oo] }
            Q.kind:= Double;           {---------------------------------}
            Q.sup := inf(A) /> inf(B);
            Q.inf := inf(A) /< sup(B);
          end
        else if ((inf(A) > 0) and (inf(B) = 0)) then
          begin                                        { Q = [Q.inf, +oo] }
            Q.kind:= PlusInfty;                        {------------------}
            Q.inf := inf(A) /< sup(B);
          end
      end { 0 in B }
    else  { not (0 in B) ==> standard interval division }
      begin                                          { Q = [C.inf, C.sup] }
        C     := A / B;                              {--------------------}
        Q.kind:= Finite;
        Q.inf := C.inf;
        Q.sup := C.sup;
      end;
    quotient := Q;
  end;

{-------------------------------------------------------------------}
{ Subtraction of an extended interval 'B' from a real value 'a'. }
{-------------------------------------------------------------------}
global operator - (a : real; B : xinterval) difference : xinterval;
  var
    D : xinterval;
  begin
    case B.kind of
      Finite     : begin                             { D = [D.inf, D.sup] }
                      D.kind:= Finite;               {--------------------}
                      D.inf := a -< B.sup;
                      D.sup := a -> B.inf;
                   end;
      PlusInfty  : begin                                 { D = [inf, +oo] }
                      D.kind:= MinusInfty;               {----------------}
                   D.sup := a -> B.inf;
                 end;
      MinusInfty : begin                                   { D = [-oo, sup] }
                   D.kind:= PlusInfty;                     {----------------}
                   D.inf := a -< B.sup;
                 end;
      Double     : begin                 { D = [-oo, D.sup] v [D.inf, +oo] }
                   D.kind:= Double; {--------------------------------}
                   D.inf := a -< B.sup;
                   D.sup := a -> B.inf;
                   if (D.inf < D.sup) then
                     D.inf := D.sup;
                 end;
      Empty      : begin                                          { D = [/] }
                   D.kind:= Empty;                                {---------}
                 end;
    end;
    difference := D;
  end;

{------------------------------------------------------------------------}
{ Intersection of interval 'X' and extended interval 'Y'. The result     }
{ is given as a pair (vector) of intervals, where one or both of them    }
{ can be empty intervals ('EmptyIntval').                                }
{------------------------------------------------------------------------}
global operator ** (X: interval; Y: xinterval) Intersect : ivector[1..2];
  var
    H : interval;
  begin
    Intersect[1]:= EmptyIntval;
    Intersect[2]:= EmptyIntval;

    case Y.kind of
      Finite     : begin                               { X ** [Y.inf,Y.sup] }
                     H := intval(Y.inf,Y.sup);         {--------------------}
                     if not (X >< H) then
                       Intersect[1]:= X ** H;
                   end;
      PlusInfty  : if (X.sup >= Y.inf) then              { X ** [Y.inf,+oo] }
                   begin                                 {------------------}
                     if (X.inf > Y.inf) then
                       Intersect[1]:= X
                     else
                       Intersect[1]:= intval(Y.inf, X.sup);
                   end;
      MinusInfty: if (Y.sup >= X.inf) then               { X ** [-oo,Y.sup] }
                   begin                                 {------------------}
                     if (X.sup < Y.sup) then
                       Intersect[1]:= X
                     else
                       Intersect[1]:= intval(X.inf, Y.sup)
                   end;
      Double     : if ((X.inf <= Y.sup) and (Y.inf <= X.sup)) then
                     begin
                       Intersect[1]:= intval(X.inf,Y.sup);{ X**[-oo,Y.sup]}
                       Intersect[2]:= intval(Y.inf,X.sup);{ X**[Y.inf,+oo]}
                     end                                   {---------------}
                   else if (Y.inf <= X.sup) then
                     begin                               { X ** [Y.inf,+oo] }
                       if (X.inf >= Y.inf) then          {------------------}
                         Intersect[1]:= X
                       else
                         Intersect[1]:= intval(Y.inf, X.sup);
```

```
                    end
                  else if (X.inf <= Y.sup) then
                    begin                             { X ** [-oo,Y.sup] }
                      if (X.sup <= Y.sup) then        {------------------}
                        Intersect[1]:=  X
                      else
                        Intersect[1]:=  intval(X.inf, Y.sup);
                    end;
      Empty     : ;                                          { X ** [/] }
    end; { CASE kind OF ... }                                {----------}
  end; { OPERATOR ** ... }

{-----------------------------------------------------------------------}
{ Module initialization part                                            }
{-----------------------------------------------------------------------}
begin
  { Nothing to initialize }
end.
```

7.2 C++ Implementation

```
//----------------------------------------------------------------------
// File: xi_ari (header)
// Purpose: Definition of an extended interval arithmetic which allows
//    the division by an interval containing zero.
// Global type:
//    KindType          : component type of extended intervals
// Global function:
//    EmptyIntval()     : empty set as irregular interval
// Class xinterval:
//    operators %, -, &: operators of extended interval arithmetic
//    operator =        : assignment operator
//----------------------------------------------------------------------
#ifndef __XI_ARI_HPP
#define __XI_ARI_HPP

#include <interval.hpp>     // Interval arithmetic
#include <ivector.hpp>      // Interval vector arithmetic

typedef enum { Finite, PlusInfty, MinusInfty, Double, Empty } KindType;

extern interval EmptyIntval ( );   // Irregular (empty) interval

class xinterval {                  // Extended intervals according
  private:                         // to the above definition
    KindType  kind;                //----------------------------
    real      inf, sup;

  public:
    xinterval ( );
    xinterval ( const KindType&, const real&, const real& );
    xinterval ( const xinterval& );

    xinterval& operator= ( xinterval& );

    friend xinterval operator- ( const real&, const xinterval& );
    friend xinterval operator% ( interval&, interval& );
    friend ivector   operator& ( interval&, const xinterval& );
};
#endif
```

```
//----------------------------------------------------------------------------
// File: xi_ari (implementation)
// Purpose: Definition of an extended interval arithmetic which allows the
//    division by an interval containing zero.
// Method: Overloading of operators for arithmetic and lattice operations
//    of data type 'xinterval'.
// Global type:
//    KindType         : component type of extended intervals
// Global function:
//    EmptyIntval()    : empty set as irregular interval
// Class xinterval:
//    operators %, -, &: operators of extended interval arithmetic
//    operator =       : assignment operator
//----------------------------------------------------------------------------
#include <i_util.hpp>     // Interval utility functions
#include <xi_ari.hpp>

//----------------------------------------------------------------------------
// An extended interval 'x', represented by the type 'xinterval', is
// defined according to the following rules (a <= b):
//
// x = [a, b]             :  x.kind = Finite,     x.inf = a, x.sup = b
// x = [a, +oo]           :  x.kind = PlusInfty,  x.inf = a, x.sup undef.
// x = [-oo, a]           :  x.kind = MinusInfty, x.sup = a, x.inf undef.
// x = [-oo, a] v [b, +oo] :  x.kind = Double,     x.inf = b, x.sup = a
// x = [-oo, +oo]         :  x.kind = Double,     x.inf = a, x.sup = a
// x = [/]                :  x.kind = Empty,      x.inf  and x.sup undef.
//
// In this definition, 'v' stands for the set union and 'oo' for infinity.
//----------------------------------------------------------------------------
interval EmptyIntval ( )                     // Irregular (empty) interval
{                                            //--------------------------
  interval x;
  x = _unchecked_interval(999999999.0,-999999999.0);
  return x;
}

//----------------------------------------------------------------------------
// Constructors and assignment operator
//----------------------------------------------------------------------------
xinterval::xinterval ( )
{
  kind = Finite;
  inf  = 0.0;
  sup  = 0.0;
}

xinterval::xinterval ( const KindType& k, const real& i, const real& s )
{
  kind = k;
  inf  = i;
  sup  = s;
}

xinterval::xinterval ( const xinterval& a )
{
  kind = a.kind;
  inf  = a.inf;
  sup  = a.sup;
}

xinterval& xinterval::operator= ( xinterval& a )
```

```
{
  kind = a.kind;
  inf  = a.inf;
  sup  = a.sup;
  return *this;
}

//----------------------------------------------------------------------------
// Extended interval division 'A / B' where 0 in 'B' is allowed.
//----------------------------------------------------------------------------
xinterval operator% ( interval& A, interval& B )
{
  interval  c;
  xinterval Q;

  if ( in(0.0, B) ) {
    if ( in(0.0, A) ) {
      Q.kind = Double;                 // Q = [-oo,+oo] = [-oo,0] v [0,+oo]
      Q.sup  = 0.0;                    //----------------------------------
      Q.inf  = 0.0;
    }
    else if ( B == 0.0 ) {                                      // Q = [/]
      Q.kind = PlusInfty;                                       //--------
      Q.inf  = divd(Sup(A),Inf(B));
    }
    else if ( (Sup(A) < 0.0) && (Sup(B) == 0.0) ) {     // Q = [Q.inf,+oo]
      Q.kind = PlusInfty;                               //----------------
      Q.inf  = divd(Sup(A),Inf(B));
    }
    else if ( (Sup(A) < 0.0) && (Inf(B) < 0.0) && (Sup(B) > 0.0) ) {
      Q.kind = Double;                     // Q = [-oo,Q.sup] v [Q.inf,+oo]
      Q.sup  = divu(Sup(A),Sup(B));        //------------------------------
      Q.inf  = divd(Sup(A),Inf(B));
    }
    else if ( (Sup(A) < 0.0) && (Inf(B) == 0.0) ) {     // Q = [-oo,Q.sup]
      Q.kind = MinusInfty;                              //----------------
      Q.sup  = divu(Sup(A),Sup(B));
    }
    else if ( (Inf(A) > 0.0) && (Sup(B) == 0.0) ) {     // Q = [-oo,Q.sup]
      Q.kind = MinusInfty;                              //----------------
      Q.sup  = divu(Inf(A),Inf(B));
    }
    else if ( (Inf(A) > 0.0) && (Inf(B) < 0.0) && (Sup(B) > 0.0) ) {
      Q.kind = Double;                     // Q = [-oo,Q.sup] v [Q.inf,+oo]
      Q.sup  = divu(Inf(A),Inf(B));        //------------------------------
      Q.inf  = divd(Inf(A),Sup(B));
    }
    else { // if ( (Inf(A) > 0.0) && (Inf(B) == 0.0) )
      Q.kind = PlusInfty;                               // Q = [Q.inf,+oo]
      Q.inf  = divd(Inf(A),Sup(B));                     //----------------
    }
  } // in(0.0,B)
  else {  // !in(0.0,B)
    c = A / B;                                        // Q = [C.inf,C.sup]
    Q.kind = Finite;                                  //------------------
    Q.inf  = Inf(c);
    Q.sup  = Sup(c);
  }

  return Q;
} // operator%

//----------------------------------------------------------------------------
```

```
// Subtraction of an extended interval 'B' from a real value 'a'.
//------------------------------------------------------------------------
xinterval operator- ( const real& a, const xinterval& B )
{
  xinterval D;

  switch (B.kind) {
    case Finite     : D.kind = Finite;                 // D = [D.inf,D.sup]
                      D.inf  = subd(a,B.sup);          //------------------
                      D.sup  = subu(a,B.inf);
                      break;
    case PlusInfty  : D.kind = MinusInfty;                 // D = [inf,+oo]
                      D.sup  = subu(a,B.inf);              //--------------
                      break;
    case MinusInfty : D.kind = PlusInfty;                  // D = [-oo,sup]
                      D.inf  = subd(a,B.sup);              //--------------
                      break;
    case Double     : D.kind = Double;    // D = [-oo,D.sup] v [D.inf,+oo]
                      D.inf  = subd(a,B.sup);//---------------------------
                      D.sup  = subu(a,B.inf);
                      if (D.inf < D.sup) D.inf = D.sup;
                      break;
    case Empty      : D.kind = Empty;                           // D = [/]
                      D.inf  = subd(a,B.sup);                   //--------
                      break;
  } // switch
  return D;
}

//------------------------------------------------------------------------
// Intersection of an interval 'X' and an extended interval 'Y'.
// The result is given as a pair (vector) of intervals, where one or both
// of them can be empty intervals.
//------------------------------------------------------------------------
ivector operator& ( interval& X, const xinterval& Y )
{
  interval H;
  ivector  IS(2);

  IS[1] = EmptyIntval();
  IS[2] = EmptyIntval();

  switch (Y.kind) {
    case Finite     : // [X.inf,X.sup] & [Y.inf,Y.sup]
                      //------------------------------
                      H = _interval(Y.inf,Y.sup);
                      if ( !Disjoint(X,H) ) IS[1] = X & H;
                      break;
    case PlusInfty  : // [X.inf,X.sup] & [Y.inf,+oo]
                      //----------------------------
                      if (Sup(X) >= Y.inf)
                        if (Inf(X) > Y.inf)
                          IS[1] = X;
                        else
                          IS[1] = _interval(Y.inf,Sup(X));
                      break;
    case MinusInfty : // [X.inf,X.sup] & [-oo,Y.sup]
                      //----------------------------
                      if (Y.sup >= Inf(X))
                        if (Sup(X)<Y.sup)
                          IS[1] = X;
                        else
                          IS[1] = _interval(Inf(X),Y.sup);
```

```
                           break;
    case Double          : if ( (Inf(X) <= Y.sup) && (Y.inf <= Sup(X)) ) {
                             IS[1] = _interval(Inf(X),Y.sup);// X & [-oo,Y.sup]
                             IS[2] = _interval(Y.inf,Sup(X));// X & [Y.inf,+oo]
                           }
                           else if (Y.inf <= Sup(X))         // X & [Y.inf,+oo]
                             if (Inf(X) >= Y.inf)            //----------------
                               IS[1] = X;
                             else
                               IS[1] = _interval(Y.inf,Sup(X));
                           else if (Inf(X) <= Y.sup)         // X & [-oo,Y.sup]
                             if (Sup(X) <= Y.sup)            //----------------
                               IS[1] = X;
                             else
                               IS[1] = _interval(Inf(X),Y.sup);
                           break;
    case Empty           : break;                                     // X ** [/]
  } // switch                                                         //---------

  return IS;
} // operator&
```

References

[1] G. ALEFELD, J. HERZBERGER, *Introduction to Interval Computations*. Academic Press, New York, 1983.

[2] R. HAMMER, M. HOCKS, U. KULISCH, D. RATZ, *Numerical Toolbox for Verified Computing I*, Springer-Verlag, Heidelberg, 1993.

[3] R. HAMMER, M. HOCKS, U. KULISCH, D. RATZ, *C++ Toolbox for Verified Computing I*, Springer-Verlag, Heidelberg, 1995.

[4] E. HANSEN. *Global Optimization Using Interval Analysis – The Multi-Dimensional Case.* Numerische Mathematik **34**, 247–270, 1980.

[5] E. HANSEN, *Global Optimization Using Interval Analysis.* Marcel Dekker, New York, 1992.

[6] KAHAN, W. M., *A More Complete Interval Arithmetic.* In: Lecture Notes for a Summer Course at the University of Michigan, 1968.

[7] E. KAUCHER, *Über metrische und algebraische Eigenschaften einiger beim numerischen Rechnen auftretender Räume.* Dissertation, Universität Karlsruhe, 1973.

[8] E. KAUCHER, *Interval Analysis in the Extended Interval Space.* Computing, Supplemnetum **2**, 33–49, 1990.

[9] R. B. KEARFOTT, *A Fortran 90 Environment for Research and Prototyping of Enclosure Algorithms for Nonlinear Equations and Global Optimization.* ACM Transactions of Mathematical Software, **21**, 63–78, 1995.

[10] R. B. Kearfott, *Rigorous Global Search: Continuos Problems.* Kluwer Academic Publishers, Boston, 1996.

[11] R. Klatte, U. Kulisch, M. Neaga, D. Ratz, Ch. Ullrich, *PASCAL–XSC – Language Reference with Examples*. Springer-Verlag, New York, 1992.

[12] S. E. Laveuve, *Definition einer Kahan-Arithmetik und ihre Implementierung.* In: K. Nickel (Ed.), *Interval Mathematics*. Lecture Notes in Computer Science, No. 29, 236–245, Springer-Verlag, Berlin, 1975.

[13] S. M. Markov, *Extended Interval Arithmetic.* Compt. rend. Acad. bulg. Sci., 30, 9, 1239–1242, 1977.

[14] M. Novoa III, *Theory of Preconditioners for the Interval Gauss-Seidel Method and Existence/Uniqueness Theory with Interval Newton Methods.* Unpublished material, 1993.

[15] H. Ratschek, J. Rokne, *New Computer Methods for Global Optimization*. Ellis Horwood Limited, Chichester, 1988.

[16] D. Ratz, *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen.* Dissertation, Universität Karlsruhe, 1992.