Interval Computations, No 4, 1994, pp. 100–129.

Extended Interval Arithmetic in IEEE Floating-Point Environment

Evgenija D. Popova[1]

Abstract

This paper describes an implementation of a general interval arithmetic extension, which comprises the following extensions of the conventional interval arithmetic: (1) extension of the set of normal intervals by improper intervals; (2) extension of the set of arithmetic operations for normal intervals by nonstandard operations; (3) extension by infinite intervals. We give a possible realization scheme of such an universal interval arithmetic in any programming environment supporting IEEE floating-point arithmetic. A PASCAL-XSC module is reported which allows easy programming of numerical algorithms formulated in terms of conventional interval arithmetic or of any of the enlisted extended interval spaces, and provides a common base for comparison of such numerical algorithms.

1 Introduction

Conventional interval arithmetic [1], [30] has been extended in the following three main directions:

- Extension of the set of normal (proper) intervals by *improper* intervals, which involves an extension of the definitions of the interval-arithmetic operations for the set of proper and improper intervals. The corresponding extended interval arithmetic structure $\mathcal{K}$ has been studied by E. Kaucher [14]–[16], H. -J. Ortolf [31], E. Gardenes [11], [12] and others. The conditional distributivity in $\mathcal{K}$ has been recently formulated [8].

- Extension of the set of arithmetic operations for normal intervals by *nonstandard* operations. The corresponding extended interval arithmetic structure $\mathcal{M}$ has been investigated in [24], [26] and applied in a number of numerical algorithms [7], [25], [28] etc.

- Extensions by *infinite* intervals [13], [14], [23], [26].

It has been demonstrated [8], [26], [32] that the extended interval arithmetic structures $\mathcal{K}$ and $\mathcal{M}$ are strongly interrelated and can be equally well used for practical purposes. All the above three extensions can be naturally combined into a common universal structure based on the extended concept of interval.

Except for SIGLA-PL/1 [11] (using radial representation of the intervals) to our knowledge there is no implementation of the extended interval arithmetic for generalized intervals. Meanwhile, a number of numerical algorithms (see Section 5) based on the enlisted

[1]This work has been partially supported by the National Science Fund of the Ministry of Science and Education under grant No. MM 10/91.

above extensions of the conventional interval arithmetic appeared. But their usage and comparison are hampered by the lack of an appropriate base software.

Another reason for the development of this collection of routines includes possible applications of the extended interval arithmetic. According to the extended concept of interval a generalized interval $[a,b] = \{x \in R \mid a \leq x \leq b, \text{ if } a \leq b;\ b \leq x \leq a, \text{ if } a > b\}$ is a subset of real values supplemented by a direction. The reacher set of generalized (directed) intervals implies reacher algebraic properties of the arithmetic operations involved. The interval structures considered here have a potential for applications to computation of sharp bounds for the ranges of functions, inner and outer numerical approximations etc. which are reviewed in Section 5.

Most of the implementations of the conventional interval arithmetic can actually perform interval operations on directed (proper/improper) intervals as well. This is due to the fact that the structure components of the interval data type are user accessible and no checking procedure for the type of the arguments is provided by the interval operation routines. The arithmetic routines designed for proper intervals can also produce results if (some of) the arguments are improper intervals. To be more specific, a PASCAL-XSC [17] program for subtraction of intervals works if some of the arguments are improper, for example

$$[1.07, 2.82] - [3.59\text{E}2, 3.58\text{E}2] = [-3.569\text{E}2, -3.5718\text{E}2].$$

Addition and subtraction operations produce correct results regardless of the type of the operands (proper/improper) since the end-point expressions for these operations are simple and offer no choice. However this is not true for multiplication and division operations where different expressions can be used. Interval multiplication when implemented according to the expression

$$[a^-, a^+] \times [b^-, b^+] = [\min\{a^-b^-, a^-b^+, a^+b^-, a^+b^+\}, \max\{a^-b^-, a^-b^+, a^+b^-, a^+b^+\}]$$

always gives wrong result if some of the operands is improper interval. Procedures for interval multiplication which use an expression involving checking of the signs of the interval end-points give correct result when some of the operands are improper intervals not containing zero in its interior. The result is wrong if an operand is an improper interval containing zero. Moreover, different implementations give different results in this situation, e. g.

$$[2,3] \times [7,-5] = \begin{cases} [21,-10] & \text{in PASCAL-SC,} \\ [14,-15] & \text{in MODULA-SC,} \\ [14,-10] & \text{in PASCAL-XSC.} \end{cases}$$

The correct answer in $\mathcal{K}$ is $[21,-15]$. The results of the division for operands of different types are analogous to those for multiplication. There is an obvious need to implement the correct definitions for all interval arithmetic operations for directed (proper/improper) intervals. This can be done at no additional cost in comparison to standard interval arithmetic implementation.

Main purpose of this paper is to present a convenient, portable and universal programming tool supporting ordinary interval arithmetic as well as the enlisted extensions. Sections 2–3 give the necessary theoretical base which underlies the implementation and the usage of the interval arithmetic based on the extended concept of interval. Concise end-point representations are given which are particularly suitable for computer implementation. Section 3 summarizes the semimorphic definitions of the computer interval

operations and some of their properties. A PASCAL-XSC [17] module EXI_ARI is presented in Section 4 as a current implementation of extended interval arithmetic in programming environment supporting IEEE floating-point arithmetic [2], [3]. In order to preserve type compatibility within the existing structure of PASCAL-XSC language, the EXI_ARI module does not support outer (Kahan's) infinite intervals [13], which implementation requires new interval data type. Based on a special theoretic consideration, given in Section 2, a specific scheme is proposed for implementation of the interval division operation, extended to perform division by interval containing zero. Most recent interval arithmetic implementations are based on floating-point arithmetic conforming the IEEE standard 754 [2]. But no agreement exists in the interval community on how to deal with interval arithmetic exceptions and no standard concerning interval arithmetic on no numbers has been proposed. In this paper we pay attention to those interval operations, which may have doubtful implementation in an IEEE floating-point environment and give the corresponding definitions of the interval arithmetic exceptional situations and their default response used in this realization.

2 Extended interval arithmetic

In this section we shall briefly outline some basic formulae of the extended interval arithmetic structure $\mathcal{K}$ proposed by E. Kaucher [14]–[16]. This structure is of major importance for our software implementation since it involves a new concept of interval. We shall omit any discussion of the nonstandard extended arithmetic [24]–[26] since it only concerns new arithmetic operations which present no problem for the implementation.

The set of all finite normal (proper) intervals $IR = \{[a,b] \mid a,b \in R,\ a \le b\}$ is extended into the set $H = \{[a,b] \mid a,b \in R\} \cong R^2$ of all ordered couples of finite real numbers further called directed intervals. A directed interval $A = [a^-, a^+] \in H$ is either proper if $a^- \le a^+$, or improper if $a^- > a^+$, so that

$$H = \{[a,b] \mid a,b \in R\} = IR \cup \overline{IR}, \qquad \overline{IR} = \left\{[a^-, a^+] \mid a^- \ge a^+;\ a^-, a^+ \in R\right\}.$$

Denote $\mathcal{T} = \{A \in IR \mid a^- a^+ \le 0\} \cup \{A \in \overline{IR} \mid a^- a^+ \le 0\} = Z \cup \overline{Z}$. For $A \in H$ the symbol a^s with $s \in \{+,-\}$ denotes certain end-point of A and the "product" st for $s,t \in \{+,-\}$ is defined by $++ = -- = +$ and $+- = -+ = -$, so that a^{s+} is well defined.

For a directed interval A define "sign" $\sigma : H \setminus \{[a^-, a^+] \mid a^- a^+ < 0\} \to \{+,-\}$ by

$$\sigma(A) \quad = \quad \begin{cases} +, & \text{if } (0 \le a^-)\ \&\ (0 \le a^+); \\ -, & \text{if } (a^- \le 0)\ \&\ (a^+ \le 0) \quad (\text{but } A \ne [0,0]), \end{cases}$$

and a binary variable "direction" by

$$\tau(A) = \begin{cases} +, & \text{if } a^- \le a^+, \\ -, & \text{otherwise}. \end{cases} \tag{1}$$

The operations of the extended interval arithmetic structure $\mathcal{K} = \{H, +, \times, \subseteq\}$ are extensions of the interval arithmetic relation and operations from the conventional interval

space $\{IR, +, \times, /, \subseteq\}$ into H

$$A \subseteq B \iff (b^- \le a^-) \ \& \ (a^+ \le b^+), \quad \text{for } A, B \in H; \tag{2}$$

$$A + B = [a^- + b^-, a^+ + b^+], \quad \text{for } A, B \in H\ ; \tag{3}$$

$$A \times B = \begin{cases} [a^{-\sigma(B)} b^{-\sigma(A)}, a^{\sigma(B)} b^{\sigma(A)}], & \text{for } A, B \in H \setminus \mathcal{T}, \\ [a^{\delta\tau(B)} b^{-\delta}, a^{\delta\tau(B)} b^{\delta}], & \delta = \sigma(A), \quad \text{for } A \in H \setminus \mathcal{T},\ B \in \mathcal{T}, \\ [a^{-\delta} b^{\delta\tau(A)}, a^{\delta} b^{\delta\tau(A)}], & \delta = \sigma(B), \quad \text{for } A \in \mathcal{T},\ B \in H \setminus \mathcal{T}, \\ [\min\{a^- b^+, a^+ b^-\},\ \max\{a^- b^-, a^+ b^+\}], & \text{for } A, B \in Z, \\ [\max\{a^- b^-, a^+ b^+\},\ \min\{a^- b^+, a^+ b^-\}], & \text{for } A, B \in \overline{Z}, \\ 0\ , \quad \text{for } A \in Z,\ B \in \overline{Z} \quad \text{or} \quad A \in \overline{Z},\ B \in Z\ . \end{cases} \tag{4}$$

Note that according to definition (2) any improper interval $A = [a^-, a^+]$ such that $a^+ \le b \le a^-$ is contained in the point interval $B = [b, b]$, $A \subseteq B$.

From (4) we have $(-1) \times B = [-b^+, -b^-] = -B$ for $B \in H$. Thus the extension of the conventional interval subtraction into H can be obtained as a composite operation

$$A - B = A + (-B) = [a^- - b^+, a^+ - b^-], \quad A, B \in H. \tag{5}$$

The substructures $(H, +, \subseteq)$ and $(H \setminus \mathcal{T}, \times, \subseteq)$ of $\mathcal{K}$ are isotone groups [14]. The inverse elements with respect to the operations $+$ and $\times$ are:

$$\begin{aligned} -_h A &= [-a^-, -a^+], \quad \text{for } A \in H; \\ 1/_h A &= [1/a^-, 1/a^+], \quad \text{for } A \in H \setminus \mathcal{T}. \end{aligned}$$

The monadic operator conjugation (called dual in [11], [12]) defined by

$$A_- = [a^+, a^-] = -_h(-A) = -(-_h A) \tag{6}$$

expresses an element-to-element symmetry in H and has the properties:

$$A \subseteq B \Longleftrightarrow A_- \supseteq B_-, \qquad (A \circ B)_- = A_- \circ B_-, \qquad \circ \in \{+, -, \times, /\}.$$

For $A \in H \setminus \mathcal{T}$ there exists also an unique operator "set inversion" $1/A = 1/_h A_- = [1/a^+, 1/a^-]$ such that $1/_h(1/A) = 1/(1/_h A) = A_-$. The extension of the conventional interval operation A/B for $A \in H$, $B \in H \setminus \mathcal{T}$ is thus obtained as a composite operation, too

$$A/B = A \times (1/B) = \begin{cases} [a^{-\sigma(B)}/b^{\sigma(A)}, a^{\sigma(B)}/b^{-\sigma(A)}], & \text{for } A, B \in H \setminus \mathcal{T}, \\ [a^{-\delta}/b^{-\delta\tau(A)}, a^{\delta}/b^{-\delta\tau(A)}], & \delta = \sigma(B), \quad \text{for } A \in \mathcal{T},\ B \in H \setminus \mathcal{T}. \end{cases} \tag{7}$$

H is a lattice with respect to $\subseteq$ with the following lattice operations:

$$\inf_{\subseteq}(A, B) = A \ \wedge \ B = [\max\{a^-, b^-\}, \min\{a^+, b^+\}], \tag{8}$$

$$\sup_{\subseteq}(A, B) = A \ \vee \ B = [\min\{a^-, b^-\}, \max\{a^+, b^+\}]. \tag{9}$$

The lattice operations satisfy the following properties

$$\begin{aligned} (A \circ B) + C &= (A + C) \circ (B + C) \qquad \text{for } A, B, C \in H \ \text{ and } \ \circ \in \{\wedge, \vee\}, \\ (A \circ B) \times C &= (A \times C) \circ (B \times C) \quad \text{for } A, B, C \in H \setminus \mathcal{T} \ \text{ and } \ \circ \in \{\wedge, \vee\}, \\ (A \wedge B)_- &= A_- \vee B_-. \end{aligned}$$

Another order relation is also defined by

$$A \preceq B \iff (a^- \leq b^-) \,\&\, (a^+ \leq b^+) \tag{10}$$

with the properties

$$A \preceq B \Longleftrightarrow A_- \preceq B_-, \quad A \preceq B \Longleftrightarrow -A_- \preceq -B_-, \quad A \preceq B \Longrightarrow A + C \preceq B + C.$$

Kaucher [14] introduces the so-called hyperbolic product as

$$A \times_h B = [a^- b^-, a^+ b^+], \qquad A, B \in H. \tag{11}$$

The interval arithmetic addition (3) together with the hyperbolic multiplication (11) form the semifield $\mathcal{H} = \{H \setminus \mathcal{T}, +, \times_h\}$ [15]. The inverse elements $-_h A$, $1/_h A$ generate operations

$$A -_h B = A + (-_h B) = [a^- - b^-, a^+ - b^+], \quad A, B \in H, \tag{12}$$

$$A /_h B = A \times_h (1/_h B) = [a^-/b^-, a^+/b^+], \quad A \in H,\ B \in H \setminus \mathcal{T}, \tag{13}$$

called hyperbolic subtraction, resp. hyperbolic division.

Let $R^* = R \cup \{-\infty, \infty\}$. Denote by $H_{\mathcal{I}} = \{[a, b] \mid a, b \in R^*\}$ the set of all finite and infinite directed intervals. The intervals from $H_{\mathcal{I}}$ are called *inner directed intervals* in contrast to the *outer directed intervals* (proper or improper), obtained by division by intervals containing/contained in zero to be defined below.

Using that $-\infty \leq a \leq +\infty$ for all $a \in R^*$ and the conventional rules for manipulations with infinities (see for example [21]) the definitions of the relation $\subseteq$ and the arithmetic operations $+, -, \times, /$ are extended from $H \times H$ into $H_{\mathcal{I}} \times H_{\mathcal{I}}$ by replacing H with $H_{\mathcal{I}}$ in (2)–(5), (7). The special cases of end-points of the form $\pm(\infty - \infty)$ or $\pm(0 \cdot \infty)$ are considered as exceptions (see Section 4.3).

For some Newton-like algorithms using conventional interval arithmetic it is essential to divide by a zero containing interval, so we need to implement an extended division operation. Some interval arithmetic specifications [4] require a separate procedure for an extended division producing two semi-infinite intervals, other specifications [19] allow division by intervals having zero only as an end-point. In what follows we consider the extension of the interval division operation and point out the possibility of a compressed representation of the result of the division by interval having zero in its interior as a finite directed interval.

The operation $1/B$ for $B \in \mathcal{L} = \{[a^-, a^+] \mid a^- a^+ < 0\}$ is defined [14] as a set of two intervals as follows:

$$1/B = 1/[b^-, b^+] \;=\; \{[-\tau(B)\infty, 1/b^-],\ [1/b^+, \tau(B)\infty]\}.$$

Such a set of two equally directed inner intervals (one involving ∞, the other $-\infty$) is called *outer directed interval*. These intervals are generalization of the so called Kahan's intervals [13], [23]. The following new propositions present some simple expressions which have been used in our implementation.

Proposition 2.1 *For $A \in H \setminus \mathcal{T}$ and $B \in \mathcal{L}$*

$$A/B \;=\; \{[c^-, -\tau(C)\infty],\ [\tau(C)\infty, c^+]\},$$

where $C = [c^-, c^+] = [a^{\tau(B)}/b^{\sigma(A)}, a^{\tau(B)}/b^{-\sigma(A)}]$.

For $A \in \mathcal{T}$ and $B \in \mathcal{L}$ we obtain

$$A/B \;=\; \begin{cases} [-\infty,\infty]_{\tau(A)}, & \tau(B)=+; \\ \{[a^+/b^+, a^-/b^+],\ [a^-/b^-, a^+/b^-]\}, & \tau(B)=-. \end{cases}$$

The above proposition suggests a "compressed" form of presentation for the result of the division by interval containing/contained in zero through only one finite directed interval C and the corresponding rule for a subsequent backward splitting of C into two inner directed intervals according to the direction of C. This presentation is exploited in the computer implementation of the division operation avoiding thus the necessity of a separate procedure for extended division.

Extension of the hyperbolic inversion $1/_h B$ for $B \in \mathcal{L}$ gives
$1/_h B = \{[1/b^-, -\tau(B)\infty],\ [\tau(B)\infty, 1/b^+]\}$.

Proposition 2.2 *For $A \in H$ and $B \in \mathcal{L}$*

$$A/_h B \;=\; \{[a^-/b^+, sign(a^-/b^+)\infty],\ [sign(a^+/b^-)\infty, a^+/b^-]\}, \tag{14}$$

and

$$sign\left(\frac{a^-}{b^+}\right) = \sigma(A)\tau(B) \;=\; \begin{cases} sign(\frac{a^+}{b^-}), & if\ \ A \in \mathcal{T} \\ -sign(\frac{a^+}{b^-}), & if\ \ A \in H \setminus \mathcal{T}. \end{cases}$$

It can be seen from (14) that the hyperbolic division by interval containing/contained in zero is also suitable for compressed representation $C = [a^-/b^+, a^+/b^-]$ of the result. The two semi-infinite intervals can be composed by the end-points (and their signs) of the compressed result.

3 Computer arithmetic

Analogous to the conventional interval computer arithmetic [21] a computer arithmetic for directed intervals is defined by semimorphism [9].

Let SR^* be a symmetric screen over R^* and $SH_{\mathcal{I}} = \{[a^-, a^+] \in H_{\mathcal{I}} \mid a^-, a^+ \in SR^*\}$, then $\{SH_{\mathcal{I}}, \subseteq\}$ is a screen of $\{H_{\mathcal{I}}, \subseteq\}$.

Define rounding $\square : H_{\mathcal{I}} \longrightarrow SH_{\mathcal{I}}$ as a monotonic function with the properties:

1. $\square(A) = A, \quad A \in SH_{\mathcal{I}}$;
2. $A \subseteq B \Longrightarrow \square(A) \subseteq \square(B), \qquad$ for $A, B \in H_{\mathcal{I}}$;
3. For $A, B \in H_{\mathcal{I}}$

$$A \subseteq \square A, \quad \square = \Diamond \quad \text{(outward rounding)},$$
$$A \supseteq \square A, \quad \square = \bigcirc \quad \text{(inward rounding)},$$
$$\Diamond A = [\bigtriangledown a^-, \bigtriangleup a^+], \qquad \bigcirc A = [\bigtriangleup a^-, \bigtriangledown a^+],$$

where $\bigtriangledown$, $\bigtriangleup$ are the corresponding directed roundings $\bigtriangledown, \bigtriangleup : R^* \longrightarrow SR^*$ [21].

If $\circ \in \{+,-,\times,/\}$ is an arithmetic operation in $H_{\mathcal{I}}$, the corresponding computer operation $\boxdot$ in $SH_{\mathcal{I}}$ is defined by

$$A \boxdot B = \square\,(A \circ B), \quad \text{for} \quad A,B \in SH_{\mathcal{I}},\ \square \in \{\Diamond, \bigcirc\}.$$

The explicit formulae for the computation of the result of the extended interval operations in $SH_{\mathcal{I}}$ are summarized as follows:
For $A,B \in SH_{\mathcal{I}}$ and $\circ \in \{+,-,\times,/\}$

$$\begin{aligned} A \diamondsuit B := \Diamond\,(A \circ B) &= [\ \bigtriangledown\,(A \circ B)^-,\ \bigtriangleup\,(A \circ B)^+\ ]; \\ A \circledcirc B := \bigcirc\,(A \circ B) &= [\ \bigtriangleup\,(A \circ B)^-,\ \bigtriangledown\,(A \circ B)^+\ ]. \end{aligned}$$

The extended interval computer operations are inclusion isotone

$$A \subseteq B \Longrightarrow A \boxdot C \subseteq B \boxdot C, \quad \text{for} \quad A,B,C \in SH_{\mathcal{I}},\ \circ \in \{+,-,\times,/\},\ \square \in \{\Diamond, \bigcirc\}.$$

The following inclusion assertions [11] are of major importance in obtaining inner and outer numerical approximations:

- For $A \in SH_{\mathcal{I}}$

$$\begin{aligned} (\Diamond(A_-))_- = \bigcirc A \subseteq \quad A \quad \subseteq \Diamond A = \bigcirc(A_-) \\ \bigcirc(A_-) = (\Diamond A)_- \subseteq \quad A_- \quad \subseteq (\bigcirc A)_- = \Diamond(A_-). \end{aligned}$$

- For $A \in SH_{\mathcal{I}}$ and $\circ \in \{+,-,\times,/\}$

$$\begin{aligned} (A_- \diamondsuit B_-)_- \subseteq A \circ B \subseteq A \diamondsuit B \\ A \circledcirc B \subseteq A \circ B \subseteq (A_- \circledcirc B_-)_-\,. \end{aligned}$$

- Let $F[\{\circ_1,\ldots,\circ_m\},\{A_1,\ldots,A_n\}]$ be a rational function where $\circ_i \in \{+,-,\times,/\}$, $i = 1,\ldots,m$ and $A_j \in H_{\mathcal{I}}$, $j = 1,\ldots,n$, then

$$F[\{\circledcirc_i\}_{i=1}^m, \{\bigcirc A_j\}_{i=1}^n] \subseteq F[\{\circ_i\}_{i=1}^m, \{A_j\}_{j=1}^n] \subseteq F[\{\diamondsuit_i\}_{i=1}^m, \{\Diamond A_j\}_{j=1}^n]$$

and

$$\begin{aligned} F[\{\diamondsuit_i\}_{i=1}^m, \{\Diamond((A_j)_-)\}_{j=1}^n]_- &= F[\{\circledcirc_i\}_{i=1}^m, \{\bigcirc A_j\}_{i=1}^n] \\ F[\{\diamondsuit_i\}_{i=1}^m, \{\Diamond A_j\}_{j=1}^n] &= F[\{\circledcirc_i\}_{i=1}^m, \{\bigcirc((A_j)_-)\}_{j=1}^n]_- \end{aligned}$$

Note, that according to these inclusion relations both inner and outer computer approximations could be obtained through only one of the rounding modes.

The hyperbolic computer operations $\circ_h \in \{-_h, \times_h, /_h\}$ are defined analogously.
For $A,B \in SH_{\mathcal{I}}$ and $\circ \in \{+,-,\times,/\}$

$$\begin{aligned} A \diamondsuit_h B := \Diamond\,(A \circ_h B) &= [\ \bigtriangledown\,(a^- \circ b^-),\ \bigtriangleup\,(a^+ \circ b^+)\ ]; \\ A \circledcirc_h B := \bigcirc\,(A \circ_h B) &= [\ \bigtriangleup\,(a^- \circ b^-),\ \bigtriangledown\,(a^+ \circ b^+)\ ]. \end{aligned}$$

Some of the inclusion properties of the computer hyperbolic operations are as follows

$$\begin{aligned} \Diamond\,(-_h A) = -_h(\bigcirc A), &\quad \text{for} \quad A \in H_{\mathcal{I}}; \\ A \subseteq B \Longrightarrow A \ominus_h C \subseteq B \diamondsuit_h C, &\quad \text{for} \quad A,B,C \in SH_{\mathcal{I}}; \\ A \subseteq B \Longrightarrow A \circledcirc_h C \subseteq B \diamondsuit_h C, &\quad \text{for} \quad A,B \in SH_{\mathcal{I}},\ C \in SH_{\mathcal{I}} \setminus S\mathcal{T},\ \circ \in \{\times,/\}. \end{aligned}$$

4 Implementation

4.1 Principles and requirements

Designing the extended interval arithmetic implementation our objective is to provide a comprehensive, portable and well-documented collection of routines, which allows easy programming of numerical computations in the extended interval spaces.

The only prerequisite concerning these routines is the presence of a reliable floating-point arithmetic with directed roundings conforming the IEEE floating-point standard [2] or [3]. Main reason for this choice were

- The IEEE standard provides mathematically well-defined computer arithmetic with maximum accuracy and directed roundings. This arithmetic ensures all obtained interval results to be accurate to 1 ULP (Unit in the Last Place);
- The IEEE standard gives accurate definitions of the floating-point exceptions and their handling, which is an appropriate base for defining the corresponding interval arithmetic exceptions and their handling;
- The IEEE floating-point arithmetic supports a set of non-numeric symbols — NaNs (Not-a-Number) and the two infinities, which is essential for the extended interval arithmetic;
- The IEEE standard ensures portability of the numerical software;
- The IEEE standard has been widely adopted to most hardware platforms and software implementations.

The major difficulty is that so far no common programming language allows access to the IEEE floating-point operations with directed roundings provided by some processors. Therefore, some machine-dependent routines should be written to provide first floating-point operations with maximum accuracy and directed roundings, and second suitable interface for testing and handling the exceptions. The alternative is to choose a SC-language ([10], [17], [18], [29]) which provides software emulation of the IEEE arithmetic. For current implementation we chose PASCAL-XSC as a wide-spread programming language for scientific computation.

Carefully designed language independent specifications containing clear mathematical definitions in computer arithmetic and definitions for the exceptions and their handling are developed for all routines.

The collection of extended interval arithmetic routines ensures full compatibility with the generally accepted interval operations and functions. Furthermore, the power of the conventional interval arithmetic is enhanced by

- extension of the definition domain of the conventional interval arithmetic providing thus tools for computations in extended interval spaces; in particular interval division operation is extended to perform division by an interval containing zero;
- providing additional set of arithmetic operations with inward rounding;
- providing routines supporting an additional order relation;
- providing hyperbolic arithmetic operations with inward and outward roundings;

- diversity of utility functions;
- certain definitions of the interval arithmetic exceptions and their handling providing thus consistency of the numerical results obtained through these operations.

In what follows we describe the PASCAL-XSC module EXI_ARI.

4.2 The PASCAL-XSC module EXI_ARI

This new PASCAL-XSC module for extended interval arithmetic is intended to be able to replace the existing module I_ARI for interval arithmetic in PASCAL-XSC language [17]. The new module supplies all operations, functions and procedures provided by I_ARI module and many other ones necessary for computations in extended interval arithmetic spaces.

The new module uses the definition of the type `INTERVAL`

type interval = **record** inf, sup : real **end;**

which is part of the language core of PASCAL-XSC. The `inf` component of the interval data type corresponds to the first component of a directed interval and the `sup` component corresponds to the second component of the directed interval comprising thus the definition for a real generalized (directed) interval as an ordered couple of real numbers.

All predefined arithmetic and lattice operators (Table 1) deliver an interval result. The two monadic operators `+, -` and the four basic operations `+, -, *, /` performing the corresponding operation in the module I_ARI with the rounding to the smallest enclosing interval (outward rounding) are predefined to perform the same operation for directed (proper/improper) intervals. An enclosing interval is a directed interval which contains according to the extended inclusion relation (2) the true interval solution.

Table 1. The operators of module EXI_ARI

left Operand / right Operand	integer real	interval
unary		$+$, $-$, _, `opp`
integer real	`AHO`, `AHI`, $\circ$ $+*$, $**$	$\diamond$, $\circ$, $\bullet$ $=$, $<>$, in, $><$ $\sqsubseteq$, $\trianglerighteq$ $+*$, $**$
interval	$\diamond$, $\circ$, $\bullet$ $=$, $<>$, $\sqsubseteq$, $\trianglerighteq$ $+*$, $**$	$\diamond$, $\circ$, $\bullet$ $=$, $<>$, in, $><$ $\sqsubseteq$, $\trianglerighteq$ $+*$, $**$

$\diamond \in \{+, -, *, /, +<, -<, *<, /<\}$, $\circ \in \{$`SHO`, `MHO`, `DHO`, `SHI`, `MHI`, `DHI`$\}$

$\bullet \in \{$`AI`, `SI`, `MI`, `DI`, `OA`, `OS`, `OM`, `OD`$\}$

$\sqsubseteq \in \{<, <=, >, >=\}$, $\trianglerighteq \in \{$`LT`, `LE`, `GT`, `GE`$\}$

Besides the usual interval operations with outward rounding, interval operations computing an inner inclusion of the true interval solution (inwardly directed rounding) can

also be very useful. An inner inclusion interval is a directed interval which is contained (according to the extended inclusion relation) in the true solution interval. The four operations `+<, -<, *<, /<` are predefined to perform the corresponding operation for directed intervals with inward rounding.

Two new monadic operations are defined: conjugation "_" of a directed interval, which is essential for the conversion between proper and improper intervals, and the hyperbolic unary minus `opp`.

The hyperbolic operations can be derived from the extended interval arithmetic operations and functions [32]. A direct and thus faster implementation of the hyperbolic operations according to formulae (12), (11), (13) is provided by the new operators `SHO, MHO, DHO` for hyperbolic subtraction, multiplication and division with outward rounding, and by the operators `SHI, MHI, DHI` for the same three operations with inward rounding. The four hyperbolic operators with outward rounding for real operands (Table 1) deliver the smallest proper interval enclosing the corresponding true result; the corresponding operators with inward rounding deliver the corresponding improper interval so that

$$a \circ \texttt{HO}\, b \;=\; \_(a \circ \texttt{HI}\, b), \qquad\qquad \circ \in \{\texttt{A}, \texttt{S}, \texttt{M}, \texttt{D}\}.$$

The module also comprises eight new operators (Table 1): `AI, SI, MI, DI` — for non-standard addition, subtraction, multiplication and division operations between directed intervals [26] with inward rounding and `OA, OS, OM, OD` — for the corresponding non-standard arithmetic operations between directed intervals with outward rounding. The detailed description of the necessary expressions involving computer arithmetic will be the subject of another publication.

The relational operators `=` (equal) and `<>` (not equal) are to be interpreted as the corresponding set-theoretic operators. The operator `=` is implemented in such a manner that it delivers true if and only if all components of the interval data type fulfill the equality. $A <> B := \texttt{not}(A = B)$. The relational operations `<, <=, >, >=` are predefined according to the extended order relation (2) for directed (proper/improper) intervals and

$$\begin{aligned} A < B &:= (A <= B)\ \texttt{and}\ (A <> B), \\ A > B &:= B < A, \\ A >= B &:= B <= A. \end{aligned} \tag{15}$$

Four new relational operations `LT, LE, GT, GE` are defined to supply testing the second order relation (10) between directed intervals. The operator `LE` satisfies

$$A\ \texttt{LE}\ B \Longleftrightarrow (A.\text{inf} <= B.\text{inf})\ \ \textbf{and}\ \ (A.\text{sup} <= B.\text{sup}).$$

The implementation of `LT, GT, GE` is according to ordering rules analogous to (15).

The operators **in** and `><` are predefined to test the set-theoretic relations "proper subset", resp. "disjointedness" between two intervals or between a real- and an interval operand. An interval A is a proper subset of an interval B if the proper part of A is contained in the proper part of B. Notationaly, operator **in** satisfies

$$A\ \textbf{in}\ B \Longleftrightarrow (b^{-\tau(B)} < a^{-\tau(A)})\ \ \textbf{and}\ \ (a^{+\tau(A)} < b^{+\tau(B)}).$$

Two directed intervals are disjoint if the intersection of their proper parts is an empty set.

The two lattice operations `+*` and `**` are predefined according to the corresponding extended definitions (9) and (8). Note, that both operations make sense even for two

real/integer operands and therefore can be used as conversion operations: `+*` for transfer from reals to a proper interval, and `**` for transfer from reals to an improper interval.

A set of utility functions is provided:

Function	Result Type	Meaning
intval (r1, r2)	*interval*	Interval with $inf = r1$ and $sup = r2$
intval (r)	*interval*	Interval with $inf = sup = r$
prop (i)	*interval*	The corresponding to i proper interval
inf (i)	*real*	The smaller end-point of i
sup (i)	*real*	The greater end-point of i
first (i)	*real*	The left end-point of i
second (i)	*real*	The right end-point of i
split (i,a)	*interval*	The a-th of the two semi-infinite intervals or the interval itself
abs (i)	*interval*	Absolute value $\mid i \mid = \{\mid r \mid : r \in i\}$
mid (i)	*real*	Midpoint of i
diam (i)	*real*	Diameter of i

a = integer expression, r, r1, r2 = real expression, i = interval expression

Two integer functions `sign` and `drc` are defined for a directed interval A to give its sign, resp. direction as:

$$\texttt{sign}(A) = \begin{cases} 1, & \text{if } a^- \le 0 \ \& \ a^+ \le 0,\ A \neq [0,0]; \\ 0, & \text{if } a^- a^+ < 0, \text{ or } A \text{ involves NaN}; \\ -1, & \text{if } a^- \ge 0 \ \& \ a^+ \ge 0; \end{cases} \qquad \texttt{drc}(A) = \begin{cases} 1, & \text{if } a^- \le a^+; \\ 0, & \text{if } A \text{ involves NaN}; \\ -1, & \text{if } a^- > a^+. \end{cases}$$

Current implementation of EXI_ARI module provides mathematical standard functions F for a directed interval argument X by using the predefined interval standard functions of the PASCAL-XSC module I_ARI

$$F_{\text{EXI\_ARI}}(X) = F_{\text{I\_ARI}}(\texttt{prop}(X))$$

and thus always deliver a proper interval as a result. An improved version of the EXI_ARI standard functions is designed to deliver a directed interval result; the direction of the result will depend on the monotonicity of the function and the direction of the input interval.

4.3 Exceptions

A number of exceptional situations such as *Invalid Operation*, *Overflow*, *Underflow*, *Division by Zero* and *Inexact Result* may arise during numerical computations in a floating-point environment conforming IEEE standards [2], [3]. Every exception, when it occurs must raise a flag that a program may subsequently sense and/or take a trap engineered to pass control to some code to handle the detected exceptional condition. The IEEE standards require that the default response to the exceptional situations is not to trap on them, but to compute and deliver to the destination a default result, specified in a reasonable way if not universally acceptable, for each possible exception. The proper handling of the exceptional conditions is an important part of any reliable numerical algorithm. A well-designed exception handling could result in a faster numerical algorithm [6]. IEEE floating-point arithmetic supports a set of special values called NaNs (Not-a-Number) for

communicating results of *Invalid Operation* exceptions, attempt to extract the square root of a negative number etc. There are two types of NaNs: *quiet* NaN which propagate through the arithmetic operations without precipitating exceptions and *signaling* NaN which precipitate an *Invalid Operation* exception whenever an attempt is made to use one as arithmetic operand. This provision of NaNs complicates IEEE definition of comparison operations and as a consequence the correct implementation of some interval arithmetic operations. An example will illustrate the necessity of some additional programmer's effort for the correct implementation of some interval arithmetic operations in standard conforming environments.

A typical implementation of the operation for convex hull of two intervals is as in the following MODULA-SC operator.

```
OPERATOR ChulII (A,B:INTERVAL) +*:  INTERVAL;
VAR Temp:INTERVAL;
BEGIN
 IF A.INF <= B.INF THEN
    Temp.INF := A.INF
 ELSE
    Temp.INF := B.INF
 END;
 IF A.SUP >= B.SUP THEN
    Temp.SUP := A.SUP
 ELSE
    Temp.SUP := B.SUP
 END;
 RETURN Temp
END ChulII;
```

Let us suppose that the interval values $A = [\mathtt{qNaN}, -5]$ and $B = [12, 16]$ are obtained as a result of some computational process in MODULA-SC [10], which floating-point arithmetic is in full conformance with IEEE Std 754. Although line 4 of the above code will signal *Invalid Operation* exception on floating-point comparison, a continued computation in non-trapping mode without checking for this exception in line 7 at the body of the operator will produce a misleading result $[12, 16]$ for those intervals instead of the correct indeterminate result $[\mathtt{qNaN}, 16]$. Therefore a test whether some of the operands is NaN or a test for *Invalid Operation* exception should be provided in the operator above in order a correct result to be supplied.

Same is the reason for producing

$$[-3, 6] = [-3, \mathtt{qNaN}] \times [-2, 1]$$

by interval multiplication in MODULA-SC. Actually none of the interval arithmetic specifications known to us pays special attention to the IEEE floating-point exceptions arising during the execution of the interval arithmetic operations and the definition of an undoubted interval default result. Below we give a short description of the interval arithmetic exceptions and their handling as they were adopted for this implementation.

Since the current implementation of the extended interval arithmetic operations is based on an underlying floating-point arithmetic conforming the IEEE standard [2], most of the interval arithmetic operations themselves will signal no exceptions. The exceptions

arising on an interval operation are all the exceptions caused by exceptional operands and exceptional results on the underlying floating-point operations. As they are, for example, on the interval addition and subtraction. For other interval operations such as the unary "+" and "−" the exceptions will depend on whether the underlying floating-point arithmetic considers copying and unary minus as arithmetic operations or not.

Invalid Operation exception is signaled on all interval relational operations if NaN is involved in some of the operands. `FALSE` is delivered as a default result when the exception occurs without a trap.

Unlike floating-point arithmetic where *quiet* NaNs propagate through arithmetic operations without raising exceptions, some of the interval arithmetic operations whose algorithms involve floating-point comparison will signal *Invalid Operation* exception if some of the operands involves quiet or signaling NaN (see the example above). Such operations are the operations for multiplication, division, convex hull, intersection and some auxiliary functions.

Lattice operations convex hull and intersection of interval, real and mixed (interval and real) arguments will signal an *Invalid Operation* exception if some of the operands involves NaN. The default result delivered when the exception occurs without a trap is an interval with a *quiet* NaN (qNaN) at that end-point at which NaN is involved in the arguments.

Beside the exceptions on the corresponding underlying floating-point operations, interval multiplication and division operations will signal *Invalid Operation* exception if some of the operands involves NaN. The default result delivered in non-trapping mode will involve at least one qNaN as end-point.

Division by Zero exception will arise on interval division when *Division by Zero* exception arise on the corresponding floating-point division operation involved in the interval division or when the interval divisor contains zero as an internal point. The result delivered to the destination when the trap is disable is either an infinite/semi-infinite interval corresponding to a divisor having zero end-point, or a finite directed interval. The latter is a concise representation of the two semi-infinite intervals resulting the division by interval containing zero in its interior (see section 3.2). An interval function `split` is provided for a backward splitting of the finite interval into two semi-infinite intervals and to return one of them.

Invalid Operation exception will also be signaled on the auxiliary mathematical functions `sign, drc, mid, diam` if their argument involves NaN. The delivered default result by `sign` and `drc` is zero, the others deliver qNaN.

For all mathematical standard functions *Invalid Operation* exception will be signaled if the argument does not belong to the definition domain of the corresponding function.

Due to the extended inclusion relation the result of the extended interval arithmetic operations with inward rounding will never be an empty set and no such exception will be signaled. The same is valid for the extended operation `**` . If computations in one of the subsets IR (proper intervals) or $\overline{IR}$ (improper intervals) of the extended interval space $\mathcal{H} = IR \cup \overline{IR}$ have to be performed, a possible test for an empty set result in the corresponding subset is the different direction of the delivered result (i. e. an improper interval delivered as a result of the operation `**` between two proper intervals will mean an empty set intersection).

5 Some application hints

In this section we shall present some simple examples illustrating the advantages of the implemented extensions of the conventional interval arithmetic.

An interpretation of the elements of H as "directed" ranges of monotone functions [26], [27] leads to an important application of the extended interval arithmetic to obtain sharp bounds for the range of a function over an interval. Let f be a continuous and monotone function on the interval $T \in IR$ and its range be $f(T) = \{f(t) \mid t \in T\}$. The type of monotonicity of f determines the "direction" into which the range $f(T)$ is traced when the argument t ranges its interval domain T in a fixed direction, say from left to right. Indeed, if f is isotone (nondecreasing) in T, then the interval $f(t)$ is traced from its lower to its upper bound whenever t traces T from left to right. Alternatively $f(T)$ is traced from its upper to its lower bound if f is antitone (nonincreasing) in T and t ranges T from left to right. That is why the interval $f[T] = [f(t^-), f(t^+)] \in H$ is called *directed range* of the function f over the interval T. The binary variable $\tau(A)$, defined by (1), called direction of the interval $A \in H$, represents the type of the monotonicity of the corresponding monotone function which directed range is A.

Consider the function $f(x) = f_1(x) \circ f_2(x)$, here $\circ \in \{+, -, \times, /\}$. We seek the range of the function using the already known ranges $f_1(X)$, $f_2(X)$. Since f_1 and f_2 are continuous on R then the ranges $f_1(X)$ and $f_2(X)$ are intervals and for the range of f we have $f(X) = \{f_1(x) \circ f_2(x) \mid x \in X\} \subseteq f_1(X) \circ f_2(X)$. It is highly desirable to obtain an equality in the above relation. However, such equality relation could be achieved by the conventional interval arithmetic only when both functions are equally monotone on the interval X. The familiar interval arithmetic can not provide an exact expression for $f(X)$ when f_1 and f_2 have different monotonicity on X. In [24] Markov defines an extension of the conventional interval arithmetic by introducing four special interval operations which provide equality relation for differently monotone functions. Nonstandard operations of $\mathcal{M}$ were used for a better bessel function evaluation [35] and in some numerical algorithms [7], [28] etc. Some authors use special techniques to achieve the result of the nonstandard interval operations. See for example [34] where an "interior difference" of two sets is defined with the meaning of the nonstandard subtraction operation, or [5] where a special algorithm based on the monotonicity of the functions is proposed to obtain sharp bounds for ranges of functions. Note, that any assertion in $\mathcal{M}$ has equivalent one in $\mathcal{K}$ and $\mathcal{H}$ and vise verse. Transition formulae [32] between the arithmetic operations of the extended interval spaces $\mathcal{K}$ and $\mathcal{M}$, and between them and the hyperbolic operations can be helpful in transferring numerical algorithms between $\mathcal{K}$, $\mathcal{M}$, $\mathcal{H}$. So, the exact representation of the range of monotone functions can also be achieved by means of directed intervals as follows:

Assume $CM(T)$ is the set of continuous and monotone functions defined on $T \in IR$.

Proposition 5.1 *For f, g, $f \circ g \in CM(D)$, $X \subseteq D$, $\circ \in \{+, -, \times, /\}$ and $g[X] \in D \setminus \mathcal{L}$ for $\circ = /$, we have*

$$(f \circ g)[X] = f[X] \circ_h g[X].$$

In the conventional interval computations when a given variable occurs more than once, it causes so called dependency problem. Considerable effort has been expended by interval analysts in attempting to produce systematic methods for representing an interval function to most sharply bound the range of a given real function over an interval. It follows from the above proposition that the dependency problem does not occur for rational monotone

functions in the hyperbolic semifield. Monotonicity of rational functions can be assured by subdividing an interval into parts where the given function is monotonic.

Corolary 5.1 *Let $f(x)$ be a rational function which computational graph consists of p functions $f_1, f_2, \ldots, f_p$, so that $f(x) \equiv F\left(\{\circ_i\}_{i=1}^m, \{f_j(x)\}_{j=1}^p\right)$. If $f, f_1, \ldots, f_p \in CM(X)$, then*

$$f[X] \quad = \quad F[\{\circ_h^i\}_{i=1}^m, \{f_j[X]\}_{j=1}^p]$$

is the exact directed range of f over $X \in H$.

Example 5.1 *Evaluate the function $f(x) = sin(x)/x$ over the interval $X = [\pi/2, \pi]$.*

We have

$$f[X] = sin[X]/_h X = [1, 0]/_h[\pi/2, \pi] = [2/\pi, 0]$$

and f is decreasing in X since $f[X]$ is improper interval.

Example 5.2 *Evaluate the function $f(x) = (x + 1/x) * 4^{-2x} - 2x$ over the interval $X = [-2, -1]$.*

We have

$$\begin{aligned} f[X] &= (X + 1/_h X) *_h 4^{-2X} -_h 2X \\ &= ([-2, -1] + [-0.5, -1]) *_h 4^{[4,2]} -_h [-4, -2] \\ &= [-2.5, -2] *_h [256, 16] -_h [-4, -2] \\ &= [-640, -32] -_h [-4, -2] = [-636, -30], \end{aligned}$$

while the result in the conventional interval arithmetic is $[-766, -20]$.

A major property of the extensions considered here is the algebraic completeness of the corresponding structures:

- The lattice operations are closed with respect to the extended inclusion relation (2).
- Due to the existence of inverse elements with respect to the addition and multiplication operations the equation $A+X = B$ has an unique solution $X = B-A_- = B-_h A$ in H and the equation $A * X = B$ has an unique solution $X = B/A_- = B/_h A_-$ for $A \in H \setminus \mathcal{T}$ and $B \in H$.
- Distributivity of the hyperbolic operations and conditional distributivity of the operations in $\mathcal{K}$ and $\mathcal{M}$.

Initiated by Gardenes et al. [12] the algebraic completeness of the extended interval arithmetic structure $\mathcal{K}$ has been used by several Russian authors [22], [33], [36], [37] to develop an *algebraic approach* in finding inner approximations of an interval system of linear equations.

Consider the linear interval system of equations

$$\mathbf{A}x = \mathbf{B}, \tag{16}$$

where $\mathbf{A} \in IR^{n\times n}$, $\mathbf{B} \in IR^n$. The solution set of (16) is defined as

$$\textstyle\sum_{\exists\exists} := \{x \in R^n \mid (\exists A \in \mathbf{A})(\exists b \in \mathbf{B})(Ax = b)\}.$$

A key role in the algebraic approach to the solutions of a linear interval system of equations plays the *interval algebraic solution*. Interval algebraic solution to (16) is an interval vector $\mathbf{x}_a$ such that substituting it into (16) and performing all interval arithmetic operations results in the valid equality $\mathbf{A}\mathbf{x}_a = \mathbf{B}$.

It has been proved [22], [33] that if $\mathbf{x}_a$ is an algebraic interval solution to the system $\mathbf{A}x = \mathbf{B}_-$ in $\mathcal{K}$ and all its components are improper intervals, then $(\mathbf{x}_a)_- \subseteq \sum_{\exists\exists}$. That is $(\mathbf{x}_a)_-$ is an inner approximation of the solution set of (16).

The following two solution sets of (16) are also important for numerous practical applications:

- the tolerable solution set $\sum_{\forall\exists} := \{x \in R^n \mid (\forall A \in \mathbf{A})(\exists b \in \mathbf{B})(Ax = b)\}$;
- the control solution set $\sum_{\exists\forall} := \{x \in R^n \mid (\exists A \in \mathbf{A})(\forall b \in \mathbf{B})(Ax = b)\}$.

Let $\mathbf{x}_a$ be the interval algebraic solution of (16) obtained in $\mathcal{K}$. If all components of $\mathbf{x}_a$ are proper intervals, then $\mathbf{x}_a \subseteq \sum_{\forall\exists}$; if all components of $\mathbf{x}_a$ are improper intervals, then $(\mathbf{x}_a)_- \subseteq \sum_{\exists\forall}$ [33].

The tolerable and control solution sets are closely related to some problems of identification and interpolation under bounded uncertainties with an impact on automatic control. Below we demonstrate the application of extended interval arithmetic on a simple practical example taken from [11].

Example 5.3 *Let $v = e * r/(\rho + r + s)$ be the equation describing an electrical circuit, where e, r and ρ are given constants varying in prescribed intervals: $e \in E$, $r \in R$, $\rho \in R_0$, $E, R, R_0 \in IR$. The goal is to determine an interval value S for the resistance, so that for any $s \in S$ the voltage v to be kept in prescribed bounds, that is $v \subseteq V \in IR$.*

Due to the inclusion monotonicity of the interval operations (conventional and extended) for any $s \in S$ we have $E * R/(R + R_0 + s) \subseteq V$ and S might be either proper or improper interval. Thus the algebraic solution of the interval equation

$$\frac{E * R}{R + R_0 + S} = V \tag{17}$$

over the wider set H will be the sought solution of the above problem. We shall find the algebraic solution S of (17) by some algebraic transformations:

Since $A/A_- = A_-/A = 1$ for $A \in H$, multiplying both sides of the equation (17) first by $(R + R_0 + S)_-$ and then by $1/V_-$, supposing $0 \notin V$ we obtain consecutively

$$\begin{aligned} E * R &= V * (R + R_0 + S)_-, \\ \frac{E * R}{V_-} &= R_- + (R_0)_- + S_-. \end{aligned}$$

Due to $A - A_- = 0$ subtraction of $(R + R_0)$ leads to $E * R/(V_-) - R - R_0 = S_-$, which after a conjugation becomes

$$S = (E * R)_-/V - (R + R_0)_-. \tag{18}$$

We have to compute an inner approximation $\bigcirc S \subseteq S$ in order to guarantee the inclusion $E * R/(R + R_0 + \bigcirc S) \subseteq V$ (extended interval operations are inclusion isotone).

Using the inclusion properties of the extended interval arithmetic computer operations (Section 3) we obtain the following expression for $\bigcirc S$ in computer arithmetic

$$\bigcirc S = (\Diamond E_-) \otimes (\Diamond R_-) \oslash (\bigcirc V) \ominus (\Diamond R)_- \ominus (\Diamond R_0)_-.$$

Assuming the data $E = [9.0, 11.0]$, $R = [2.0, 4.0]$ and $R_0 = [1.5, 2.5]$, if the voltage is to be kept inside the interval $V = [2.0, 4.0]$, from (18) we obtain $S = [7.5, 2.5]$. Since the resulting interval is improper, this is a control interval. In other words for any $s \in S$ there exist $e \in E$, $\rho \in R_0$ and $r \in R$ such that $e * r/(\rho + r + s) \in V$.

Allowing the voltage within the wider range $V = [2, 8]$ the resulting resistance would be $S = [2.0, 2.5]$ showing the tolerance for the electrical circuit, that is for any $s \in S$ and any $e \in E$, $\rho \in R_0$ and $r \in R$ $\ e * r/(\rho + r + s) \in V$.

Finally, we point out that the algebraic approach to solving some interval problems could be applied straightforward in a computer algebra system supporting extended interval arithmetic.

6 Conclusion

The implemented collection of extended interval arithmetic routines provides full compatibility with the generally accepted interval operations and functions. Furthermore, the power of the conventional interval arithmetic is enhanced by the operations with inward rounding and a natural extension of the definition domain of the conventional interval arithmetic providing thus tools for computations in extended interval spaces. Giving certain definitions of the interval arithmetic exceptions and their handling in an IEEE floating-point environment we allow a consistent transfer of this software to other IEEE software/hardware platforms and provide the corresponding consistency of the numerical results obtained by these interval arithmetic routines.

The extended interval arithmetic routines are a suitable base for transfer, testing and comparison of the numerical algorithms involving the ordinary interval arithmetic and the considered interval extensions as well as a necessary tool for implementing the forthcoming algorithms.

Acknowledgements

The author would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Alefeld, G.; Herzberger, J.: *Einführung in die Intervallrechnung.* Bibliographisches Institut Mannheim, 1974.

[2] American National Standards Institute/Institute of Electrical and Electronics Engineers: *IEEE Standard for Binary Floating-Point Arithmetic.* ANSI/IEEE Std 754–1985, New York, 1985.

[3] American National Standards Institute/Institute of Electrical and Electronics Engineers: *IEEE Standard for Radix-Independent Floating-Point Arithmetic.* ANSI/IEEE Std 854–1987, New York, 1987.

[4] Corliss, G. F.: *Proposal for a Basic Interval Arithmetic Subroutines Library (BIAS).* preprint, 1990.

[5] Corliss, G. F.; Rall, L. B.: *Computing the Range of Derivatives.* In Kaucher, E.; Markov, S. M.; Mayer, G. (Eds.): *Computer Arithmetic, Scientific Computation and Mathematical Modelling.* IMACS Annals on Computing and Appl. Math., **12**, J. C. Balzer, Basel, 1992, pp. 195–212.

[6] Demmel, J.; Li, X.: *Faster Numerical Algorithms via Exception Handling*, In Swartzlander, E.; Irwin, M. J.; Jullien, G. (Eds.): Proceedings of the 11th Symposium on Computer Arithmetic, IEEE Computer Society Press, 1993, pp. 234–241.

[7] Dimitrova, N. S.: *On Some Properties of an Interval Newton Type Method and its Modification.* Computing, Suppl. **9**, 1993, pp. 21–32.

[8] Dimitrova, N.; Markov, S. M.; Popova, E.: *Extended Interval Arithmetics: New Results and Applications.* In Atanassova, L.; Herzberger, J. (Eds.): *Computer Arithmetic and Enclosure Methods.* Elsevier Sci. Publishers B. V., 1992, pp. 225–232.

[9] Durst, E.: *Realisierung einer erweiterten Intervallrechnung mit Überlaufarithmetik.* Diplomarbeit, Universität Karlsruhe, 1975.

[10] Falcó Korn, C.; Gutzwiller, S.; König, S.; Ullrich, Ch.: *Modula-SC. Motivation, Language Definition and Implementation.* In Kaucher, E.; Markov, S. M.; Mayer, G. (Eds.): *Computer Arithmetic, Scientific Computation and Mathematical Modelling.* IMACS Annals on Computing and Appl. Math., **12**, J. C. Balzer, Basel, 1992, pp. 161–181.

[11] Gardeñes, E.; Trepat, A.: *Fundamentals of SIGLA, an Interval Computing System over the Completed Set of Intervals.* Computing, **24**, 1980, pp. 161–179.

[12] Gardeñes, E.; Trepat, A.; Janer, J. M.: *Approaches to Simulation and to the Linear Problem in the SIGLA System.* Freiburger Interval-Berichte 81/8, 1981, pp. 1–28.

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

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

[15] Kaucher, E.: *Über Eigenschaften und Anwendungsmöglichkeiten der erweiterten Intervallrechnung und des hyperbolischen Fastkörpers über R.* Computing Suppl. **1**, 1977, pp. 81–94.

[16] Kaucher, E.: *Interval Analysis in the Extended Interval Space IR.* Computing Suppl. **2**, 1980, pp. 33–49.

[17] Klatte, R.; Kulisch, U.; Neaga, M.; Ratz, D.; Ullrich, Ch.: *PASCAL-XSC Language Reference with Examples.* Springer, Berlin, 1992.

[18] Klatte, R.; Kulisch, U.; Lawo, C.; Rauch, M.; Wiethoff, A.: *C-XSC A C++ Class Library for Extended Scientific Computation.* Springer-Verlag, Berlin, 1993.

[19] Knüppel, O.: *BIAS — Basic Interval Arithmetic Subroutines.* Bericht 93.3, Technische Universität Hamburg-Harburg, Hamburg, 1993.

[20] Kulisch, U. (Ed.): *PASCAL-SC: A PASCAL Extension for Scientific Computation; Information Manual and Floppy Disks;* Version IBM PC/AT, Operating System DOS. Wiley-Teubner Series in Comp. Sci., B. G. Teubner, J. Wiley & Sons, 1987.

[21] Kulisch, U., Miranker, W. L.: *Computer Arithmetic in Theory and Practice.* Academic Press, New York 1981.

[22] Kuprianova, L.: *A Method of Square Root for Solving Interval Linear Algebraic System.* Int. Conference on Interval and Computer-Algebraic Methods in Science and Engineering, Interval'94, St-Petersburg, 1994.

[23] Laveuve, S. E.: *Definition einer Kahan-Arithmetik und ihre Implementierung.* In Nickel, K. (Ed.): *Interval Mathematics.* Lecture Notes in Computer Science, **29**, Springer, Berlin, 1975, pp. 236–245.

[24] Markov, S. M.: *Extended Interval Arithmetic.* Compt. Rend. Acad. Bulg. Sci., **30**, 9, 1977, pp. 1239–1242.

[25] Markov, S. M.: *Some Applications of the Extended Interval Arithmetic to Interval Iterations.* Computing Suppl. **2**, 1980, pp. 69–84.

[26] Markov, S. M.: *Extended Interval Arithmetic Involving Infinite Intervals.* Mathematica Balkanica, New Series, **6**, 3, 1992, pp. 269–304.

[27] Markov, S. M.: *On the Presentation of Ranges of Monotone Functions using Interval Arithmetic.* Interval Computations, No 4(6), 1992, pp. 19–31.

[28] Markov, S. M.; Angelov, R.: *An Interval Method for System of ODE.* In Nickel, K.: *Interval Mathematics 1985.* Lecture Notes in Computer Science, **212**, Springer, 1986, pp. 103–108.

[29] Metzger, M.; Walter, W. V.: *FORTRAN-SC: A Programming Language for Engineering/Scientific Computation.* In Ullrich, Ch. (Ed.): *Contribution to Computer Arithmetic and Self-Validating Numerical Methods.* IMACS Annals on Computing and Appl. Math., **7**, J. C. Balzer, Basel, 1990, pp. 427–441.

[30] Moore, R. E.: *Interval Analysis.* Englewood Cliffs, N. Y., Prentice-Hall 1966.

[31] Ortolf, H. -J.: *Eine Verallgemeinerung der Intervallarithmetik.* Geselschaft für Mathematik und Datenverarbeitung, Bonn **11**, 1969, pp. 1-71.

[32] Popova, E.: *Transition Formulae between Interval Arithmetic Structures.* preprint, 1994.

[33] Shary, S. P.: *Algebraic Approach to the Interval Linear Static Identification, Tolerance and Control Problems or One More Application of Kaucher Arithmetic.* Int. Conference on Interval and Computer-Algebraic Methods in Science and Engineering, Interval'94, St-Petersburg, 1994.

[34] Stetter, H. J.: *Validated Solution of Initial Value Problems for ODE.* In Ullrich, Ch. (Ed.): *Computer Arithmetic and Self-Validating Numerical Methods.* Notes and Reports in Mathematics in Science and Engineering, **7**, Academic Press, 1990, pp. 171–187.

[35] Zakovic, S.: *Evaluation of Bessel-Ricatti Functions using Interval Arithmetic.* Working Paper, Numerical Optimisation Centre, University of Hertfordshire, 1994.

[36] Zakharov, A. V.: *Constructing an Interval Algebraic Solution in Extended Interval Arithmetic.* Proc. of All-Union Conference on Actual Problems of Applied Mathematics, Saratov, 1990, pp. 311–317. (in Russian)

[37] Zyuzin, V. S.: *On a Way of Finding Two-Sided Approximation to the Solution of System of Linear Interval Equations.* Differential Equations and the Theory of Functions, Saratov State University, Saratov, 1987, pp. 28–32. (in Russian)