Definition of the Arithmetic Operations and Comparison Relations for an Interval Arithmetic Standard*

Gerd Bohlender and Ulrich Kulisch
Institut für Angewandte und Numerische Mathematik,
Universität Karlsruhe, D-76128 Karlsruhe, Germany
bohlender@kit.edu,ulrich.kulisch@kit.edu

Abstract

Concepts of interval arithmetic like arithmetic operations, comparison relations, distances, convergence, etc. are clearly defined terms of pure mathematics. The development of a standard for interval arithmetic should be based on these concepts. This paper considers the definition of the arithmetic operations and of comparison relations for an interval arithmetic computation standard. For derivation see [6] and [5] in particular. Hardware support for the operations and comparison relations is given in [3].

Keywords: computer arithmetic, floating-point arithmetic, interval arithmetic, elementary functions, arithmetic standards
AMS subject classifications: 65G30, 65G99, 65G50, 65Y04, 68M07

1 Interval Sets and Mappings

Interval arithmetic over the real numbers deals with closed[1] and connected sets of the real numbers $\mathbb{R}$. An interval is denoted by a bold lower case letter. It represents an ordered pair, written as $\boldsymbol{a} = [\underline{a}, \overline{a}]$, with $\underline{a}, \overline{a} \in \mathbb{R}$. $\underline{a}$ denotes the lower bound and $\overline{a}$ the upper bound. The lower bound shall not be greater than the upper bound. The set of nonempty, closed and bounded real intervals is denoted by $\mathbb{IR}$.

The set of all bounded or unbounded intervals is denoted by $\overline{\mathbb{IR}}$, $\emptyset \in \overline{\mathbb{IR}}$. If a bound is $-\infty$ or $+\infty$ the bound is not an element of the interval. Such intervals may also be written as $(-\infty, a]$ or $[b, +\infty)$ with $a, b \in \mathbb{R}$ or $(-\infty, +\infty)$ where the parentheses indicate that the bounds $-\infty$ and $+\infty$ are not elements of the interval. With respect to set inclusion as an order relation $\{\overline{\mathbb{IR}}, \subseteq\}$ is a complete lattice. It is bounded from below by the empty set $\emptyset$ and from above by the set $(-\infty, +\infty)$.

On the computer real numbers are approximated by the subset of floating-point numbers as defined by the IEEE 754 floating-point arithmetic standard, for instance.

*Submitted: January 19, 2009; Revised: February 10, 2010; Accepted: February 20, 2010.
[1] A subset of $\mathbb{R}$ is called closed if its complement is open.

The set of floating-point numbers is denoted by $\mathbb{F}$. The subset of $\mathbb{IR}$ with bounds of $\mathbb{F}$ is denoted by $\mathbb{IF}$. The subset of all bounded or unbounded intervals of $\overline{\mathbb{IR}}$ with finite bounds of $\mathbb{F}$ is denoted by $\overline{\mathbb{IF}}$, $\emptyset \in \overline{\mathbb{IF}}$. Intervals over the real numbers, arithmetic operations, and comparison relations for these are approximated by intervals, arithmetic operations, and comparison relations for intervals of the set $\overline{\mathbb{IF}}$.

A real number or an interval over the real numbers is mapped onto the smallest floating-point interval that contains the number or interval respectively. This mapping $\diamondsuit : \overline{\mathbb{IR}} \to \overline{\mathbb{IF}}$ is characterized (uniquely defined) by the following properties:

(R1) $\diamondsuit \boldsymbol{a} = \boldsymbol{a}$, for all $\boldsymbol{a} \in \overline{\mathbb{IF}}$,

(R2) $\boldsymbol{a} \subseteq \boldsymbol{b} \Rightarrow \diamondsuit \boldsymbol{a} \subseteq \diamondsuit \boldsymbol{b}$, for $\boldsymbol{a}, \boldsymbol{b} \in \overline{\mathbb{IR}}$,

(R3) $\boldsymbol{a} \subseteq \diamondsuit \boldsymbol{a}$, for all $\boldsymbol{a} \in \overline{\mathbb{IR}}$,

(R4) $\diamondsuit(-\boldsymbol{a}) = -\diamondsuit \boldsymbol{a}$, for all $\boldsymbol{a} \in \overline{\mathbb{IR}}$.

2 Arithmetic Operations for Intervals

With the mapping $\diamondsuit : \overline{\mathbb{IR}} \to \overline{\mathbb{IF}}$ and its properties listed at the end of Section 1 arithmetic operations for intervals of $\overline{\mathbb{IF}}$ are uniquely defined by:

(RG) $\boldsymbol{a} \diamondsuit \boldsymbol{b} := \diamondsuit(\boldsymbol{a} \circ \boldsymbol{b})$, for all $\boldsymbol{a}, \boldsymbol{b} \in \overline{\mathbb{IF}}$ and all $\circ \in \{+, -, *, /\}$.

The IEEE floating-point arithemtic standard 754 specifies arithmetic with four roundings: to the nearest floating-point number, downwards, upwards, and towards zero. For these operations the following notations will be used:
$+, -, *, /$ for the operations with rounding to the nearest floating-point number,
$\triangledown$, $\bigtriangledown$, $\triangledown$, $\bigtriangledown$ for the operations with rounding downwards,
$\triangle$, $\bigtriangleup$, $\triangle$, $\bigtriangleup$ for the operations with rounding upwards,[2] and
$*|, -|, +|, /|$ for the operations with rounding towards zero (chopping).[3]

With these notations for bounded intervals $\boldsymbol{a} = [\underline{a}, \overline{a}]$, $\boldsymbol{b} = [\underline{b}, \overline{b}] \in \mathbb{IF}$ the following arithmetic operations $+, -, *$, and $/$ can be derived from (RG)[4]:

Addition $[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\underline{a} \triangledown \underline{b}, \overline{a} \triangle \overline{b}]$.

Subtraction $[\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\underline{a} \triangledown \overline{b}, \overline{a} \triangle \underline{b}]$.

[2] In our PASCAL extension (available since 1980) and the Fortran extension we developed for and with IBM (available 1990) pairs of keyboard symbols $+<$, $-<$, $*<$, $/<$ and $+>$, $->$, $*>$, $/>$ have been used for the operations with rounding downwads and upwards, respectively.

[3] Frequently used programming languages do not allow 4 plus, minus, multiply, and divide operators for floating-point numbers. A future interval arithmetic standard could or should specify names for low level operations with the directed roundings. They could be: $addp, subp, mulp, divp, addn, subn, muln$, and $divn$. Here p stands for rounding toward positive and n for rounding toward negative. With these routines interval operations would be fully transferable from one processor to another.

[4] For details see [5].

Multiplication $[\underline{a},\overline{a}]*[\underline{b},\overline{b}]$	$[\underline{b},\overline{b}]$ $\overline{b}\le 0$	$[\underline{b},\overline{b}]$ $\underline{b}<0<\overline{b}$	$[\underline{b},\overline{b}]$ $\underline{b}\ge 0$
$[\underline{a},\overline{a}],\overline{a}\le 0$	$[\overline{a}\mathbin{\bar{\triangledown}}\overline{b},\underline{a}\mathbin{\bar{\triangle}}\underline{b}]$	$[\underline{a}\mathbin{\bar{\triangledown}}\overline{b},\underline{a}\mathbin{\bar{\triangle}}\underline{b}]$	$[\underline{a}\mathbin{\bar{\triangledown}}\overline{b},\overline{a}\mathbin{\bar{\triangle}}\underline{b}]$
$\underline{a}<0<\overline{a}$	$[\overline{a}\mathbin{\bar{\triangledown}}\underline{b},\underline{a}\mathbin{\bar{\triangle}}\underline{b}]$	$[min(\underline{a}\mathbin{\bar{\triangledown}}\overline{b},\overline{a}\mathbin{\bar{\triangledown}}\underline{b}),$ $max(\underline{a}\mathbin{\bar{\triangle}}\underline{b},\overline{a}\mathbin{\bar{\triangle}}\overline{b})]$	$[\underline{a}\mathbin{\bar{\triangledown}}\overline{b},\overline{a}\mathbin{\bar{\triangle}}\overline{b}]$
$[\underline{a},\overline{a}],\underline{a}\ge 0$	$[\overline{a}\mathbin{\bar{\triangledown}}\underline{b},\underline{a}\mathbin{\bar{\triangle}}\overline{b}]$	$[\overline{a}\mathbin{\bar{\triangledown}}\underline{b},\overline{a}\mathbin{\bar{\triangle}}\overline{b}]$	$[\underline{a}\mathbin{\bar{\triangledown}}\underline{b},\overline{a}\mathbin{\bar{\triangle}}\overline{b}]$

Division, $0\notin\boldsymbol{b}$ $[\underline{a},\overline{a}]/[\underline{b},\overline{b}]$	$[\underline{b},\overline{b}]$ $\overline{b}<0$	$[\underline{b},\overline{b}]$ $\underline{b}>0$
$[\underline{a},\overline{a}],\overline{a}\le 0$	$[\overline{a}\mathbin{\bar{\triangledown}}\underline{b},\underline{a}\mathbin{\bar{\triangle}}\overline{b}]$	$[\underline{a}\mathbin{\bar{\triangledown}}\underline{b},\overline{a}\mathbin{\bar{\triangle}}\overline{b}]$
$[\underline{a},\overline{a}],\underline{a}<0<\overline{a}$	$[\overline{a}\mathbin{\bar{\triangledown}}\overline{b},\underline{a}\mathbin{\bar{\triangle}}\overline{b}]$	$[\underline{a}\mathbin{\bar{\triangledown}}\underline{b},\overline{a}\mathbin{\bar{\triangle}}\underline{b}]$
$[\underline{a},\overline{a}],\underline{a}\ge 0$	$[\overline{a}\mathbin{\bar{\triangledown}}\overline{b},\underline{a}\mathbin{\bar{\triangle}}\underline{b}]$	$[\underline{a}\mathbin{\bar{\triangledown}}\overline{b},\overline{a}\mathbin{\bar{\triangle}}\underline{b}]$

Division, $0\in\boldsymbol{b}$ $[\underline{a},\overline{a}]/[\underline{b},\overline{b}]$	$\boldsymbol{b}=$ $[0,0]$	$[\underline{b},\overline{b}]$ $\underline{b}<\overline{b}=0$	$[\underline{b},\overline{b}]$ $0=\underline{b}<\overline{b}$
$[\underline{a},\overline{a}],\overline{a}<0$	$\emptyset$	$[\overline{a}\mathbin{\bar{\triangledown}}\underline{b},+\infty)$	$(-\infty,\overline{a}\mathbin{\bar{\triangle}}\overline{b}]$
$[\underline{a},\overline{a}],\underline{a}\le 0\le\overline{a}$	$(-\infty,+\infty)$	$(-\infty,+\infty)$	$(-\infty,+\infty)$
$[\underline{a},\overline{a}],\underline{a}>0$	$\emptyset$	$(-\infty,\underline{a}\mathbin{\bar{\triangle}}\underline{b}]$	$[\underline{a}\mathbin{\bar{\triangledown}}\overline{b},+\infty)$

Division by an interval that includes zero in the last table leads to unbounded intervals. To be complete, arithmetic operations for unbounded intervals now have to be defined also.

The first rule is that any operation with the empty set $\emptyset$ has the empty set as its result. Arithmetic operations for unbounded intervals of $\overline{\mathbb{IF}}$ can be performed on the computer by using the above formulas for bounded intervals if in addition a few formal rules for operations with $-\infty$ and $+\infty$ are applied. These rules are shown in the corresponding tables.

Addition	$-\infty$	b	$+\infty$
$-\infty$	$-\infty$	$-\infty$	
a	$-\infty$		$+\infty$
$+\infty$		$+\infty$	$+\infty$

Subtraction	$-\infty$	b	$+\infty$
$-\infty$		$-\infty$	$-\infty$
a	$+\infty$		$-\infty$
$+\infty$	$+\infty$	$+\infty$	

Multiplication	$-\infty$	$b<0$	0	$b>0$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	0	$-\infty$	$-\infty$
$a<0$	$+\infty$				$-\infty$
0	0				0
$a>0$	$-\infty$				$+\infty$
$+\infty$	$-\infty$	$-\infty$	0	$+\infty$	$+\infty$

Division	$-\infty$	$+\infty$
a	0	0

These rules are not new in principle. They are well established in real analysis and IEEE 754 provides them anyhow. The only rule that goes beyond IEEE 754 is

$$0 * (-\infty) = (-\infty) * 0 = 0 * (+\infty) = (+\infty) * 0 = 0. \tag{1}$$

This rule follows quite naturally from the definition of unbounded intervals. However, it should not be interpreted as a new mathematical law. It is just a short cut to easily compute the bounds of the result of an operation with unbounded intervals.

3 Remarks on the Arithmetic Operations

I. In the table for division by an interval that includes zero the case $\underline{b} < 0 < \overline{b}$ is missing. This needs some explanation.

A basic concept of mathematics is that of a function or mapping. A function consists of a pair (f, D_f). It maps each element x of its domain of definition D_f on a unique element y of the range R_f of f, $f : D_f \to R_f$.

In real analysis division by zero is not defined. Thus a rational function $y = f(x)$ where the denominator is zero for $x = c$ is not defined for $x = c$, i.e., c is not an element of the domain of definition D_f. Since the function $f(x)$ is not defined at $x = c$ it does not have any value or property there. In this strict mathematical sense, division by an interval $[\underline{b}, \overline{b}]$ with $\underline{b} < 0 < \overline{b}$ is not well posed. For division the set $\underline{b} < 0 < \overline{b}$ devolves into the two distinct sets $[\underline{b}, 0]$[5] and $[0, \overline{b}]$ and division by an interval $[\underline{b}, \overline{b}]$ with $\underline{b} < 0 < \overline{b}$ actually consists of two divisions the result of which again consists of two distinct sets. In each case the result is a single unbounded interval. The two divisions should be performed separately. Division by the two sets $[\underline{b}, 0]$ and $[0, \overline{b}]$ is shown in the corresponding table.

The situation can easily be identified by the signs of the bounds of the divisor before the division is executed. For interval multiplication or division a case selection has to be done (by hardware or software) anyhow before the operations are performed. In the case $\underline{b} < 0 < \overline{b}$ the sign of $\underline{b}$ is negative and the sign of $\overline{b}$ is positive.

In the user's program, however, the two divisions appear as a single operation, as division by an interval $[\underline{b}, \overline{b}]$ with $\underline{b} < 0 < \overline{b}$. So an arithmetic operation in the user's program delivers two distinct results. This is an unusual situation in conventional computing.[6]

A solution to the problem would be for the computer to provide a flag for *distinct intervals*. The situation occurs if the divisor is an interval that contains zero as an interior point. In this case the flag would be raised and signaled to the user. The user may then apply a routine of his choice to deal with the situation as is appropriate for his application.

This routine could be: Modify the operands and recompute, or continue the computation with one of the sets and ignore the other one, or put one of the sets on a list and continue the computation with the other one, or return the entire set of real numbers $(-\infty, +\infty)$ as result and continue the computation, or stop computing, or any other action.

[5] Since division by zero does not contribute to the solution set it does not matter whether a paranthesis or bracket is used here.

[6] It would be very convenient for computing if other operations would also deliver two answers: floating-point addition and subtraction the rounded result and the error, multiplication the product to the double length and division the quotient and the remainder.

A somewhat natural solution would be to continue the computation on different processors, one for each interval. But the situation can occur repeatedly. How many processors would we need? Future multicore units will provide a large number of processors. They will suffice for quite a while. A similar situation occurs in global optimization using subdivision. After a certain test several candidates may be left for further investigation.

Newton's method reaches its ultimate elegance and strength in the extended interval Newton method. It computes all (single) zeros in a given domain. If a function has several zeros in a given interval its derivative becomes zero in that interval also. Thus Newton's method applied to that interval delivers two distinct sets. This is how the extended interval Newton method separates different zeros. If the method is continued along two separate paths, one for each of the distinct intervals it finally computes all zeros in the given domain. If the method continues with only one of the two distinct sets and ignores the other one it computes an enclosure of only one zero of the given function. If the interval Newton method delivers the empty set, the method has proved that there is no zero in the initial interval.

II. If interval arithmetic is hardware supported then execution of the operations listed above is about as fast as execution of the corresponding floating-point operations. It is thus not reasonable to define and study operations between floating-point numbers and intervals in order to save computing time. Floating-point arithmetic and interval arithmetic are not the same calculus for approximate arithmetic for real numbers. They should be kept strictly separate.[7]

Of course, computing with result verification often makes use of floating-point computations. If executed in IEEE 754 arithmetic this may result in an exception. So there remains the question of how exceptions like $-\infty$, $+\infty$, NaN, -0, $+0$ can reasonably be mapped on floating-point intervals.

The following would be reasonable: -0 and $+0$ can only mean 0. Since NaN is not a real number it should be mapped on the empty set and since $-\infty$ and $+\infty$ are also not real numbers their image could or should also be the empty set. If the image of the result of a floating-point computation is the empty set a flag should be set.

III. The empty set $\emptyset$ may occur as result of an interval operation as listed in the tables of Section 2. The result of any operation with the empty set $\emptyset$ was defined to be the empty set. This suggests an encoding of the empty set by $\emptyset = [+NaN, -NaN]$. Then the rules for interval arithmetic listed in Section 2 can also be applied to the empty set. By the well established rules of IEEE 754 for NaN an operation with the empty set would then automatically produce the empty set as the result.

The encoding $\emptyset = [+NaN, -NaN]$ for the empty set also turns out to be useful for the definition of comparison relations for intervals. These will be studied in the next section.

IV. Success of interval arithmetic is based on two arithmetical features: One is double precision interval arithmetic. The other is variable precision interval arithmetic [8, 9, 11, 1]. An exact scalar product for the double precision format is the basic tool to achieve high speed variable (dynamic) precision arithmetic for real and interval data. Pipelining gives it high speed, and exactitude brings very high accuracy into computation. There is no way to compute a dot product faster than the exact result. By pipelining, it can be computed in the time the processor needs to read the data,

[7] The XSC-languages allow real and interval data and operations between these in an expression. However, all real data are immediately interpreted as intervals and all operations are performed as interval operations.

i.e., it comes with utmost speed [4, 5]. Variable length interval arithmetic fully benefits from such speed [5]. No software simulation can go as fast. With operator overloading variable length interval arithmetic is very easy to use.

4 Comparison Relations and Lattice Operations

Three comparison relations are important for intervals of $\overline{\mathbb{IF}}$:

$$\textit{equality}, \quad \textit{less than or equal}, \quad \text{and} \quad \textit{set inclusion}.^{8} \tag{2}$$

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be intervals of $\overline{\mathbb{IF}}$ with bounds $\underline{a} \le \overline{a}$ and $\underline{b} \le \overline{b}$ respectively. Then the relations *equality* and *less than or equal* in $\overline{\mathbb{IF}}$ are defined by:

$$\begin{aligned} \boldsymbol{a} = \boldsymbol{b} \quad &:\Leftrightarrow \underline{a} = \underline{b} \wedge \overline{a} = \overline{b}, \\ \boldsymbol{a} \le \boldsymbol{b} \quad &:\Leftrightarrow \underline{a} \le \underline{b} \wedge \overline{a} \le \overline{b}. \end{aligned}$$

Since bounds for intervals of $\overline{\mathbb{IF}}$ may be $-\infty$ or $+\infty$ these comparison relations are executed as if performed in the lattice $\{F^*, \le\}$ with $F^* := F \cup \{-\infty\} \cup \{+\infty\}$.

With the order relation $\le$, $\{\overline{\mathbb{IF}}, \le\}$ is a lattice. The *greatest lower bound* (glb) and the *least upper bound* (lub) of $\boldsymbol{a}, \boldsymbol{b} \in \overline{\mathbb{IF}}$ are the intervals

$$\begin{aligned} \mathrm{glb}(\boldsymbol{a}, \boldsymbol{b}) \quad &:= [\min(\underline{a}, \underline{b}), \min(\overline{a}, \overline{b})], \\ \mathrm{lub}(\boldsymbol{a}, \boldsymbol{b}) \quad &:= [\max(\underline{a}, \underline{b}), \max(\overline{a}, \overline{b})]. \end{aligned}$$

The greatest lower bound and the least upper bound of an interval with the empty set are both the empty set.

The inclusion relation in $\overline{\mathbb{IF}}$ is defined by

$$\boldsymbol{a} \subseteq \boldsymbol{b} \quad :\Leftrightarrow \underline{b} \le \underline{a} \wedge \overline{a} \le \overline{b}. \tag{3}$$

With the relation $\subseteq$, $\{\overline{\mathbb{IF}}, \subseteq\}$ is also a lattice. The least element in $\{\overline{\mathbb{IF}}, \subseteq\}$ is the empty set $\emptyset$ and the greatest element is the interval $(-\infty, +\infty)$. The infimum of two elements $\boldsymbol{a}, \boldsymbol{b} \in \overline{\mathbb{IF}}$ is the intersection and the supremum is the interval hull (convex hull):

$$\begin{aligned} \inf(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a} \cap \boldsymbol{b} \quad &:= [\max(\underline{a}, \underline{b}), \min(\overline{a}, \overline{b})] \quad \text{or the empty set } \emptyset, \\ \sup(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a} \underline{\cup} \boldsymbol{b} \quad &:= [\min(\underline{a}, \underline{b}), \max(\overline{a}, \overline{b})]. \end{aligned}$$

The intersection of an interval with the empty set is the empty set. The interval hull with the empty set is the other operand.

If in the formulas for $glb(\boldsymbol{a}, \boldsymbol{b})$, $lub(\boldsymbol{a}, \boldsymbol{b})$, $\boldsymbol{a} \cap \boldsymbol{b}$, $\boldsymbol{a} \underline{\cup} \boldsymbol{b}$, a bound is $-\infty$ or $+\infty$ a parenthesis should be used at this interval bound to denote the resulting interval. This bound is not an element of the interval.

[8] Of course, also other order relations can be defined for intervals of $\overline{\mathbb{IR}}$ and $\overline{\mathbb{IF}}$. The theory of interval arithmetic is extensively studied in [5]. Only the three order realtions considered here are needed for it. So specification of these three realtions should suffice in order to keep an interval arithmetic standard simple. If a particular application needs another relation the user can easily define it ad hoc.

If in any of the comparison relations defined here both operands are the empty set, the result is true. If in 3 $\boldsymbol{a}$ is the empty set the result is true. Otherwise the result is false if in any of the three comparison relations only one operand is the empty set.[9]

A particular case of inclusion is the relation *element of*. It is defined by

$$a \in \boldsymbol{b} \quad :\Leftrightarrow \underline{b} \leq a \wedge a \leq \overline{b}.$$

Another useful check is for whether $[\underline{a}, \overline{a}]$ is an interval at all, that is, if $\underline{a} \leq \overline{a}$.

References

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[2] W. Kahan, *A More Complete Interval Arithmetic*, Lecture Notes prepared for a summer course at the University of Michigan, June 17–21, 1968.

[3] R. Kirchner and U. Kulisch, "Hardware support for interval arithmetic", *Reliable Computing*, vol. 12, no. 3, pp. 225–237, 2006.

[4] U. W. Kulisch, *Advanced Arithmetic for the Digital Computer – Design of Arithmetic Units*, Springer-Verlag, Wien, New York, 2002.

[5] U. W. Kulisch, *Computer Arithmetic and Validity – Theory, Implementation and Applications*, De Gruyter, Berlin, New York, 2008.

[6] U. W. Kulisch, "Complete Interval Arithmetic and its Implementation on the Computer", In: A. Cuyt et al. (eds.), *Numerical Validation in Current Hardware Architectures*, Lecture Notes in Computer Science LNCS, vol. 5492, Springer-Verlag Berlin Heidelberg, pp. 41–67, 2009.

[7] IFIPWG-IEEE754R, *Letter of the IFIP WG 2.5 to the IEEE Computer Arithmetic Revision Group*, 2007.[10]

[8] R. E. Moore, *Interval Analysis*, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1966.

[9] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, Pennsylvania, 1979.

[10] S. Ratz, *On Extended Interval Arithmetic and Inclusion Isotony*, Preprint, Institut für Angewandte Mathematik, Universität Karlsruhe, 1999.

[11] S. M. Rump, *Kleine Fehlerschranken bei Matrixproblemen*, Dissertation, Universität Karlsruhe, 1980.

[9] A convenient encoding of the empty set may be $\emptyset = [+NaN, -NaN]$. Then most comparison relations and lattice operations considered in this section would deliver the correct answer if conventional rules for NaN are applied. However, if $\boldsymbol{a} = \emptyset$ then set inclusion 3 and computing the interval hull do not follow this rule. So in these two cases whether $\boldsymbol{a} = \emptyset$ must be checked before the operations can be executed.

[10] See http://www.math.kit.edu/ianm2/∼kulisch.