Karlsruher Institut für Technologie

Up-to-date Interval Arithmetic From Closed Intervals to Connected Sets of Real Numbers

Ulrich Kulisch

Preprint 15/02

INSTITUT FÜR WISSENSCHAFTLICHES RECHNEN
UND MATHEMATISCHE MODELLBILDUNG

KIT – Universität des Landes Baden-Württemberg und
nationales Forschungszentrum in der Helmholtz-Gemeinschaft

www.kit.edu

Anschriften der Verfasser:

Prof. Dr. Ulrich Kulisch
Institut für Angewandte und Numerische Mathematik
Karlsruher Institut für Technologie (KIT)
D-76128 Karlsruhe

Up-to-date Interval Arithmetic
From closed intervals to connected sets of real numbers

Ulrich Kulisch

Institut für Angewandte und Numerische Mathematik
Karlsruher Institut für Technologie
Kaiserstrasse 12
D-76128 Karlsruhe GERMANY
Ulrich.Kulisch@kit.edu

Abstract. This paper unifies the representations of different kinds of computer arithmetic. It is motivated by ideas developed in the book *The End of Error* by John Gustafson [5]. Here interval arithmetic just deals with connected sets of real numbers. These can be closed, open, half-open, bounded or unbounded.

The first chapter gives a brief informal review of computer arithmetic from early floating-point arithmetic to the IEEE 754 floating-point arithmetic standard, to conventional interval arithmetic for closed and bounded real intervals, to the proposed standard IEEE P1788 for interval arithmetic, to advanced computer arithmetic, and finally to the just recently defined and published unum and ubound arithmetic [5].

Then in chapter 2 the style switches from an informal to a pure and strict mathematical one. Different kinds of computer arithmetic follow an abstract mathematical pattern and are just special realizations of it. The basic mathematical concepts are condensed into an abstract axiomatic definition. A computer operation is defined via a monotone mapping of an arithmetic operation in a complete lattice onto a complete sublattice. Essential properties of floating-point arithmetic, of interval arithmetic for closed bounded and unbounded real intervals, and for advanced computer arithmetic can directly be derived from this abstract mathematical model.

Then we consider unum and ubound arithmetic. To a great deal this can be seen as an extension of arithmetic for closed real intervals to open and halfopen real intervals. Essential properties of unum and ubound arithmetic are also derived from the abstract mathematical setting given in chapter 2. Computer executable formulas for the arithmetic operations of ubound arithmetic are derived on the base of pure floating-point arithmetic. These are much simpler, easier to implement and faster to execute than alternatives that would be obtained on the base of the IEEE 754 floating-point arithmetic standard which extends pure floating-point arithmetic by a number of exceptions.

The axioms of computer arithmetic given in section 2 also can be used to define ubound arithmetic in higher dimensional spaces like complex numbers, vectors and matrices with real and interval components. As an example section 4 indicates how this can be done in case of matrices with ubound components. Execution of the resulting computer executable formulas once more requires an exact dot product.

In comparison with conventional interval arithmetic *The End of Error* may be a too big step to easily get accepted by manufacturers and computer users. So in the last section we mention a reduced but still great step that might easier find its way into computers in the near future.

1 Introduction

The first chapter briefly reviews the development of arithmetic for scientific computing from a mathematical point of view from the early days of floating-point arithmetic to conventional interval arithmetic until the latest step of unum and ubound arithmetic.

1.1 Early Floating-Point Arithmetic

Early computers designed and built by Konrad Zuse, the Z3 (1941) and the Z4 (1945), are among the first computers that used the binary number system and floating-point for number representation [4, 25]. Both machines carried out the four basic arithmetic operations of addition, subtraction, multiplication, division, and the square root by hardware. In the Z4 floating-point numbers were represented by 32 bits. They were used in a way very similar to what today is IEEE 754 single precision arithmetic. The technology of those days was poor (electromechanical relays, electron tubes). It was complex and expensive. To avoid frequent interrupts special representations and corresponding wirings were available to handle the three special values: 0, ∞, and *indefinite* (for $0/0$, $\infty \cdot 0$, $\infty - \infty$, or ∞/∞ and others).

These early computers were able to execute about 100 flops (**f**loating-point **o**perations **p**er **s**econd). For comparison: With a mechanic desk calculator or a modern pocket calculator a trained person can execute about 1000 arithmetic operations somewhat reliably on a day. The computer now could do this in 10 seconds. This was a gigantic increase in computing speed by a factor of about 10^7.

Over the years the computer technology was permanently improved. This permitted an increase of the word size and of speed. Already in 1965 computers were on the market (CDC 6600) that performed 10^5 flops. At these speeds a conventional error analysis of numerical algorithms, that estimates the error of each single arithmetic operation, becomes questionable. Examples can be given which illustrate that computers after very few operations deliver a completely absurd result [29]. It easily can be shown that for a certain system of two linear equations with two unknowns even today computers delivers a result of which possibly not a single digit is correct. Such results strongly suggest to use the computer more for computing close bounds for the solution instead of murkey approximations.

1.2 The Standard for Floating-Point Arithmetic IEEE 754

Continuous progress in computer technology allowed extra features such as additional word sizes and differences in the coding and numbers of special cases. To stabilize the situation a standard for floating-point arithmetic was developed and internationally adopted in 1985. It is known as the IEEE 754 floating-point arithmetic standard. Until today the most used floating-point format is double precision. It corresponds to about 16 decimal digits. A revision of the standard IEEE 754, published in 2008, added another word size of 128 bits.

During a floating-point computation exceptional events like underflow, overflow or division by zero may occur. For such events the IEEE 754 standard reserves some bit patterns to represent special quantities. It specifies special representations for $-\infty$, $+\infty$, -0, $+0$, and for `NaN` (not a number). Normally, an overflow or division by zero would cause a computation to be interrupted. There are, however, examples for which it makes sense for a computation to continue. In IEEE 754 arithmetic the general strategy upon an exceptional event is to deliver a result and continue the computation. This requires the result of operations on or resulting in special values to be defined. Examples are: $4/0 = \infty$, $-4/0 = -\infty$, $0/0 = \texttt{NaN}$, $\infty - \infty = \texttt{NaN}$, $0 \cdot \infty = \texttt{NaN}$, $\infty/\infty = \texttt{NaN}$, $1/(-\infty) = -0$, $-3/(+\infty) = -0$, $\log 0 = -\infty$, $\log x = \texttt{NaN}$ when $x < 0$, $4 - \infty = -\infty$. When a `NaN` participates in a floating-point operation, the result is always a `NaN`. The purpose of these special operations and results is to allow programmers to postpone some tests and decisions to a later time in the program when it is more convenient.

The standard for floating-point arithmetic IEEE 754 has been widely accepted and has been used in almost every processor developed since 1985. This has greatly improved the portability of floating-point programs. IEEE 754 floating-point arithmetic has been used successfully in the past. Many computer users are familiar with all details of IEEE 754 arithmetic including all its exceptions like *underflow*, *overflow*, $-\infty$, $+\infty$, `NaN`, -0, $+0$, and so on. Seventy years of extensive use of floating-point arithmetic with all its exceptions makes users believe that this is the only reasonable way of using the computer for scientific computing.

By the time the original standard IEEE 754 was developed, early microprocessors were on the market. They were made with a few thousand transistors, and ran at 1 or 2 MHz. Arithmetic was provided by an 8-bit adder. Dramatic advances in computer technology, in memory size, and in speed have been made since 1985. Arithmetic speed has gone from megaflops (10^6 flops), to gigaflops (10^9 flops), to teraflops (10^{12} flops), to petaflops (10^{15} flops), and it is already approaching the exaflops (10^{18} flops) range. This even is a greater increase of computing speed since 1985 than the one from a hand calculator to the first electronic computers! A qualitative difference goes with it. At the time of the megaflops computer a conventional error analysis was recommended in every numerical analysis textbook. Today the PC is a gigaflops computer. For the teraflops or petaflops computer conventional error analysis is no longer practical.

Computing indeed has already reached astronomical dimensions! With increasing speed, problems that are dealt with become larger and larger. Extending pure floating-point arithmetic by operations for elements that are not real numbers and perform trillions of operations with them appears questionable. What seemed to be reasonable for slow speed computers needs not to be so for computers that perform trillions of operations in a second. A compiler could detect exceptional events and ask the user to treat them as for any other error message.

The capability of a computer should not just be judged by the number of operations it can perform in a certain amount of time without asking whether the computed result is correct. It should also be asked how fast a computer can compute correctly to 3, 5, 10 or 15 decimal places for certain problems. If the question were asked that way, it would very soon lead to better computers. Mathematical methods that give an answer to this question are available for many problems. Computers, however, are at present not designed in a way that allows these methods to be used effectively. Computer arithmetic must move strongly towards more reliability in computing. Instead of the computer being merely a fast calculating tool it must be developed into a scientific instrument of mathematics.

1.3 Conventional Interval Arithmetic

For reasons just mentioned interval arithmetic was invented. Conventional interval arithmetic just deals with **bounded and closed real intervals.** Formulas for the basic arithmetic operations for these are easily derived. Interval arithmetic became popular after the book [23] by R. E. Moore was published in 1966. It was soon further exploited by other well known books by G. Alefeld and J. Herzberger [1, 2] or by E. Hansen [6, 7] for instance, and others. Interval mathematics using conventional interval arithmetic has been developed to a high standard over the last few decades. It provides methods which deliver results with guarantees.

Since the 1970-ies until lately [12, 13, 26, 32] attempts were undertaken to extend the arithmetic for closed and bounded real intervals to unbounded intervals. However, inconsistencies to deal with $-\infty$ and $+\infty$ have occured again and again. If the real numbers $\mathbb{R}$ are extended by $-\infty$ and $+\infty$ then unusual and unsatisfactory operations are to be dealt with like $\infty - \infty$, $0 \cdot \infty$, or ∞/∞.

1.4 The Proposed Standard for Interval Arithmetic IEEE P1788

In April 2008 the author of this article published a book [19] in which the problems with the infinities and other exceptions are definitely eliminated. Here interval arithmetic just deals with sets of real numbers. Since $-\infty$ and $+\infty$ are not real numbers, they cannot be elements of a real interval. They only can be bounds of a real interval. Formulas for the arithmetic operations for bounded and closed real intervals are well established in conventional interval arithmetic. It is shown in the book that these formulas can be extended to closed and unbounded real intervals by a continuity principle. For a bound $-\infty$ or $+\infty$ in an interval operand the bounds for the resulting interval can be obtained from the formulas for bounded real intervals by applying well established rules of real analysis for computing with $-\infty$ and $+\infty$. It is also shown in the book that obscure operations like $\infty - \infty$ or ∞/∞ do not occur in the formulas for the operations for unbounded real intervals. *This new approach to arithmetic for bounded and unbounded closed real intervals leads to an algebraically closed calculus which is free of exceptions. It remains free of exceptions if the operations are mapped on a floating-point screen by the monotone, upwardly directed rounding*, for definition see 3.2. Intervals bring the continuum on the computer. An interval between two floating-point bounds represents the continuous set of real numbers between these bounds.

A few months after publication of the book [19] the IEEE Computer Society founded a committee IEEE P1788 for developing a standard for interval arithmetic in August 2008. A motion, presented by the author, to include arithmetic for unbounded real intervals where $-\infty$ and $+\infty$ may be bounds but not elements of unbounded real intervals has been accepted by IEEE P1788.

With little hardware expenditure interval arithmetic can be made as fast as simple floating-point arithmetic. The lower and the upper bound of an arithmetic operation easily can be computed simultaneously. With more suitable processors, rigorous methods based on interval arithmetic could be comparable in speed to today's approximate methods. As computers speed up, interval arithmetic becomes a principal and necessary tool for controlling the precision of a computation as well as the accuracy of the computed result.

The standard for interval arithmetic IEEE P1788 takes the elements of the standard IEEE 754 for floating-point arithmetic as the basic set of numbers. This introduces all the IEEE 754 exceptions into interval arithmetic. It makes interval arithmetic clumsy and complicates its implementation and understanding. This can be avoided if interval arithmetic is developed over the set of real numbers as demonstrated in [19]. All kinds of speculation should be removed from computer arithmetic.

1.5 Advanced Computer Arithmetic

The book [19] deals with computer arithmetic in a more general sense than usual. It shows how the arithmetic and mathematical capability of the digital computer can be enhanced in a quite natural way. This is motivated by the desire and the need to improve the accuracy of numerical computing and to control the quality of computed results.

Advanced computer arithmetic extends the accuracy requirements for the elementary floating-point operations as defined by the arithmetic standard IEEE 754 to the customary product spaces of computation: the complex numbers, the real and complex intervals, the real and complex vectors and matrices, and the real and complex interval vectors and interval matrices. All computer approximations of arithmetic operations in these spaces should deliver a result that differs from the correct result by at most one rounding. For all these product spaces this accuracy requirement leads to operations which are distinctly different from those traditionally available on computers. This expanded set of arithmetic operations is taken as a definition of what is called **advanced computer arithmetic** in [19]. Programming environments that provide advanced computer arithmetic have been available since 1980 [9, 10, 12, 21, 32, 33].

Advanced computer arithmetic is then used to develop algorithms for computing highly accurate and guaranteed bounds for a number of standard problems of numerical analysis like systems of linear equations, evaluation of polynomials or other arithmetic expressions, numerical integration, optimization problems, and many others [12, 13]. These can be taken as higher order arithmetic operations. Essential for achieving these results is an exact dot product.

In vector and matrix spaces[1] the dot product of two vectors is a fundamental arithmetic operation. It is fascinating that this basic operation is also a mean to increase the speed of computing besides of the accuracy of the computed result. Actually the simplest and fastest way for computing a dot product of two floating-point vectors is to compute it exactly. Here the products are just shifted and added into a wide fixed-point register on the arithmetic unit. By pipelining, the exact dot product can be computed in the time the processor needs to read the data, i.e., it comes with utmost speed. This high speed is obtained by totally avoiding slow intermediate access to the main memory of the computer.

Any method that computes a dot product correctly rounded to the nearest floating-point number also has to consider the values of the summands. This results in a more complicated method with the outcome that it is necessarily slower than a conventional computation of the dot product in floating-point arithmetic. Experience with a prototype development in 1994 [3, 17] shows that a hardware implementation of the exact dot product can be expected to be three to four times faster than the latter. The main difference, however, is accuracy. There are many applications where a correctly rounded or otherwise precise dot product does not suffice to solve the problem. For details see [27, 28, 30] and [19].

The hardware needed for the exact dot product is comparable to that for a fast multiplier by an adder tree, accepted years ago and now standard technology in every modern processor. The exact dot product brings the same speedup for accumulations at comparable costs.

In 2009 the author prepared a motion that requires inclusion of the exact dot product as essential ingredient for obtaining high accuracy in interval computations into the standard IEEE P1788. The motion was accepted. But in 2013, however, the motion was weakened by the committee to now just recommending an exact dot product. In practice a recommendation guarantees nonstandard behavior for different computing systems.

Advanced computer arithmetic certainly is a much more useful extension to pure floating-point arithmetic than all the exceptions provided by IEEE 754. All forms of speculation need to be removed from computing.

[1] for real, complex, interval, and complex interval data

1.6 Unum and Ubound Arithmetic

While about 70 scientists from all over the world have been working on a standard for interval arithmetic for more than 6 years since August 2008, all of a sudden like out of nothing John Gustafson publishes a book: *The End of Error* [5]. Reading this book became a big surprise. It is a sound piece of work and it is hard to believe that a single person could develop so many nice ideas and put them together into a sketch of what might become the future of computing. Reading the book is fascinating. The situation very much reminds me to a text by Friedrich Schiller in his work *Demetrius.* It says:

Was ist die Mehrheit? Die Mehrheit ist der Unsinn,
Verstand ist stets bei wen'gen nur gewesen.
....
Man soll die Stimmen waegen und nicht zaehlen;
der Staat muss untergehn, frueh oder spaet,
wo Mehrheit siegt und Unverstand entscheidet.

For almost 60 years interval arithmetic was defined for the set $\mathbb{IR}$ of closed and bounded real intervals. *The End of Error* expands this to the set $\mathbb{JR}$ of just connected sets real numbers. These can be closed, open, half-open, bounded, or unbounded. The book shows that arithmetic for this expanded set is closed under addition, subtraction, multiplication, division, also square root, powers, logarithm, exponential, and many other elementary functions needed for technical computing, i.e., arithmetic operations for intervals of $\mathbb{JR}$ always lead to intervals of $\mathbb{JR}$ again. The calculus is free of exceptions. It remains free of exceptions if the bounds are restricted to a floating-point screen, for proof see section 3.4. John Gustafson shows in his book that this new extension of conventional interval arithmetic opens new areas of applications and allows getting better results.

This paper is an attempt to describe different kinds of computer arithmetic by a unique representation. It might help to integrate unum and ubound arithmetic into the arithmetic standard IEEE P1788 after this has been released from the IEEE 754 exceptions.

2 Axiomatic Definition of Computer Arithmetic

Frequently mathematics is seen as the science of structures. Analysis carries three kinds of structures: an algebraic structure, an order structure, and a topological or metric structure. These are coupled by certain compatibility properties, as for instance: $a \leq b \Rightarrow a + c \leq b + c$.

It is well known that floating-point numbers and floating-point arithmetic do not obey the rules of the real numbers $\mathbb{R}$. However, the rounding is a monotone function. So the changes to the order structure are minimal. This is the reason why the order structure plays a key role for an axiomatic definition of computer arithmetic.

This study considers the two elementary models that are covered by the two IEEE arithmetic standards IEEE 754 and IEEE 1788, computer arithmetic on the reals and on real intervals, but also more recent refined developments called unum and ubound arithmetic. Abstract settings of computer arithmetic for higher dimensional spaces like complex numbers, vectors and matrices for real, complex, and interval data can be developed following similar schemes. We briefly sketch this in section 4. For more details see [19] and the literature cited there.

We begin by listing a few well-known concepts and properties of ordered sets.

Definition 1. *A relation* $\leq$ *in a set* M *is called an* order relation*, and* $\{M, \leq\}$ *is called an* ordered set[2] *if for all* $a, b, c \in M$ *the following properties hold:*

(O1) $a \leq a,$ (reflexivity)
(O2) $a \leq b \ \wedge \ b \leq c \ \Rightarrow \ a \leq c,$ (transitivity)
(O3) $a \leq b \ \wedge \ b \leq a \ \Rightarrow \ a = b,$ (antisymmetry)

An ordered set M *is called* linearly *or* totally ordered *if in addition*

(O4) $a \leq b \vee b \leq a$ *for all* $a, b \in M.$ (linearly ordered)

[2] Occasionally called a partially ordered set.

An ordered set M is called

(O5) *a* lattice *if for any two elements $a, b \in M$, the* $\inf\{a, b\}$ *and the* $\sup\{a, b\}$ *exist.* (lattice)

(O6) *It is called* conditional completely ordered *if for every bounded subset $S \subseteq M$, the* $\inf S$ *and the* $\sup S$ *exist.*

(O7) *An ordered set M is called* completely ordered *or a* complete lattice *if for every subset $S \subseteq M$, the* $\inf S$ *and the* $\sup S$ *exist.* (complete lattice)

With these concepts the real numbers $\{\mathbb{R}, \leq\}$ are defined as a conditional complete linearly ordered field.

In the definition of a complete lattice, the case $S = M$ is included. Therefore, $\inf M$ and $\sup M$ exist. Since they are elements of M, every complete lattice has a least and a greatest element.

If a subset $S \subseteq M$ of a complete lattice $\{M, \leq\}$ is also a complete lattice, $\{S, \leq\}$ is called a *complete sublattice* of $\{M, \leq\}$ if the two lattice operations inf and sup in both sets lead to the same result, i.e., if

$$\text{for all} \quad A \subseteq S, \quad \inf_M A = \inf_S A \quad \text{and} \quad \sup_M A = \sup_S A.$$

Definition 2. *A subset S of a complete lattice $\{M, \leq\}$ is called a* screen *of M, if every element $a \in M$ has upper and lower bounds in S and the set of all upper and lower bounds of $a \in M$ has a least and a greatest element in S respectively. If a minus operator exists in M, a screen is called* symmetric*, if for all $a \in S$ also $-a \in S$.*

As a consequence of this definition a complete lattice and a screen have the same least and greatest element. It can be shown that a screen is a complete sublattice of $\{M, \leq\}$ with the same least and greatest element, [19].

Definition 3. *A mapping $\square : M \to S$ of a complete lattice $\{M, \leq\}$ onto a screen S is called a* rounding *if (R1) and (R2) hold:*

(R1) *for all* $\quad a \in S, \quad \square\, a := a.$ (projection)

(R2) $a \leq b \Rightarrow \square\, a \leq \square\, b.$ (monotone)

A rounding is called downwardly directed *resp.* upwardly directed *if for all $a \in M$*

(R3) $\square\, a \leq a \quad$ *resp.* $\quad a \leq \square\, a.$ (directed)

If a minus operator is defined in M, a rounding is called antisymmetric *if*

(R4) $\square\,(-a) = -\,\square\, a, \quad$ *for all $a \in M$.* (antisymmetric)

The monotone downwardly resp. upwardly directed roundings of a complete lattice onto a screen are unique. For the proof see [19].

Definition 4. *Let $\{M, \leq\}$ be a complete lattice and $\circ : M \times M \to M$ a binary arithmetic operation in M. If S is a screen of M, then a rounding $\square : M \to S$ can be used to approximate the operation $\circ$ in S by*

(RG) $a \boxdot b := \square\,(a \circ b)$*, for $a, b \in S$.*

If a minus operator is defined in M and S is a symmetric screen of M, then a mapping $\square : M \to S$ with the properties (R1,2,4) and (RG) is called a semimorphism[3].

Semimorphisms with antisymmetric roundings are particularly suited for transferring properties of the structure in M to the subset S. It can be shown [19] that semimorphisms leave a number of reasonable properties of ordered algebraic structures (ordered field, ordered vector space) invariant.

If an element $x \in M$ is bounded by $a \leq x \leq b$ with $a, b \in S$, then by (R1) and (R2) the rounded image $\square\, x$ is bounded by the same elements: $a \leq \square\, x \leq b$, i.e., $\square\, x$ is either the least upper (supremum) or the greatest lower (infimum) bound of x in S. Similarly, if for $x, y \in S$ the result of an operation $x \circ y$ is bounded by $a \leq x \circ y \leq b$ with $a, b \in S$, then by (R1), (R2), and (RG) also $a \leq x \boxdot y \leq b$, i.e., $x \boxdot y$ is either the least upper or the greatest lower bound of $x \circ y$ in S. If the rounding is upwardly or downwardly directed the result is the least upper or the greatest lower bound respectively.

[3] The properties (R1,2,4) and (RG) of a semimorphism can be shown to be necessary conditions for a homomorphism between ordered algebraic structures. For more details see [19]

3 Particular Models

3.1 Floating-Point Arithmetic

The set $\{\overline{\mathbb{R}}, \leq\}$ with $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$ is a complete lattice. Let $\mathbb{F}$ denote the set of finite floating-point numbers and $\overline{\mathbb{F}} := \mathbb{F} \cup \{-\infty, +\infty\}$. Then $\overline{\mathbb{F}}$ is a screen of $\{\overline{\mathbb{R}}, \leq\}$. The least element of the set $\overline{\mathbb{R}}$ and of the subset $\overline{\mathbb{F}}$ is $-\infty$ and the greatest element is $+\infty$.

Definition 5. *With a rounding* $\square : \overline{\mathbb{R}} \to \overline{\mathbb{F}}$ *arithmetic operations* $\boxdot$ *in* $\overline{\mathbb{F}}$ *are defined by*

(RG) $a \boxdot b := \square\,(a \circ b)$*, for* $a, b \in \overline{\mathbb{F}}$ *and* $\circ \in \{+, -, *, /\}$*,* [4]

with $b \neq 0$ *in case of division.*[5]

If a and b are adjacent floating-point numbers and $x \in \mathbb{R}$ with $a \leq x \leq b$, then because of (R1) and (R2) also $a \leq \square\, x \leq b$, i.e., there is never an element of $\mathbb{F}$ between an element $x \in \mathbb{R}$ and its rounded image $\square\, x$. The same property holds for the operations defined by (RG): If for $x, y \in \mathbb{F}$, $a \leq x \circ y \leq b$ then by (R1), (R2), and (RG) also $a \leq x \boxdot y \leq b$, for all $\circ \in \{+, -, *, /\}$, i.e., $x \boxdot y$ is either the greatest lower or the least upper bound of $x \circ y$ in $\mathbb{F}$.

Frequently used roundings $\square : \overline{\mathbb{R}} \to \overline{\mathbb{F}}$ are antisymmetric. Examples are the rounding to the nearest floating-point number, the rounding toward zero, or the rounding away from zero. A semimorphism transfers a number of useful properties of the real numbers to the floating-point numbers. The mathematical structure of $\mathbb{F}$ can even be defined as properties of $\mathbb{R}$ which are invariant with respect to semimorphism, [19].

For the monotone downwardly resp. upwardly directed roundings of $\overline{\mathbb{R}}$ onto $\overline{\mathbb{F}}$ often the special symbols $\bigtriangledown$ resp. $\bigtriangleup$ are used. These roundings are not antisymmetric. They are related by the property:

$$\bigtriangledown\,(-a) = -\bigtriangleup\, a \quad \text{and} \quad \bigtriangleup\,(-a) = -\bigtriangledown\, a. \tag{1}$$

Arithmetic operations defined by (RG) and these roundings are denoted by $\bigtriangledown\!\!\!\!\!\!\circ\;$ and $\bigtriangleup\!\!\!\!\!\!\circ\;$, respectively, for $\circ \in \{+, -, \cdot, /\}$. These are heavily used in interval arithmetic.

3.2 Interval Arithmetic

Conventional interval arithmetic extends the arithmetic for real numbers to closed and connected sets of real numbers.[6] Originally it was developed for closed and bounded real intervals [2, 23]. The set of all such intervals is denoted by $\mathbb{IR}$. An interval of $\mathbb{IR}$ is written as an ordered pair $[a_1, a_2]$, with $a_1, a_2 \in \mathbb{R}$ and $a_1 \leq a_2$. The first element is the lower bound and the second is the upper bound.

Later it was discovered [19] that the formulas for the arithmetic operations $+, -, \cdot, /$ for bounded intervals of $\mathbb{IR}$ can be extended to unbounded intervals in an exception-free manner. For the notation of unbounded real intervals $-\infty$ and $+\infty$ are used as bounds. Since $-\infty$ and $+\infty$ are not real numbers a bound $-\infty$ or $+\infty$ of an unbounded interval is not an element of the interval. Thus unbounded real intervals are written as $(-\infty, a]$ or $[b, +\infty)$ or $(-\infty, +\infty)$[7] with $a, b \in \mathbb{R}$. No exceptional operations like $\infty - \infty$, ∞/∞ occur in the explicit formulas for the arithmetic operations of unbounded real intervals. For proof see [19]. An international standard IEEE 1788 now defines the notation of the set of closed and bounded real intervals to be $\mathbb{IR}$ and $\overline{\mathbb{IR}}$ for closed, bounded and unbounded real intervals.

With respect to the subset relation as an order relation the set of real intervals $\{\overline{\mathbb{IR}}, \subseteq\}$ is a complete lattice. The subset of $\overline{\mathbb{IR}}$ where all finite bounds are floating-point numbers of $\mathbb{F}$ is denoted by $\overline{\mathbb{IF}}$. $\{\overline{\mathbb{IF}}, \subseteq\}$ is a screen of $\{\overline{\mathbb{IR}}, \subseteq\}$. In both sets $\overline{\mathbb{IR}}$ and $\overline{\mathbb{IF}}$ the infimum of a subset of $\overline{\mathbb{IF}}$ is the intersection and the supremum is the interval hull. The least element of both sets $\overline{\mathbb{IR}}$ and $\overline{\mathbb{IF}}$ is the empty set $\varnothing$ and the greatest element is the set $\mathbb{R} = (-\infty, +\infty)$.

[4] Here operations like $\infty - \infty$ or ∞/∞ can occur which in IEEE 754 are set to NaN. In the interval operations, however, such constellations do not occur. For proof see [19].

[5] In real analysis division by zero is not defined. It does not lead to a real number.

[6] A set of real numbers is called *closed*, if its complement in $\mathbb{R}$ is open. A set $M \subset \mathbb{R}$ is called *open* if for all $a \in M$ also an ϵ-neighborhood $(a - \epsilon, a + \epsilon)$ is entirely in M.

[7] These intervals are closed sets of real numbers. Nevertheless we use round brackets at the bounds $-\infty$ and $+\infty$ to indicate that the bounds are not elements of the interval.

Definition 6. *For intervals $\boldsymbol{a}, \boldsymbol{b} \in \overline{\mathbb{IR}}$ arithmetic operations $\circ \in \{+, -, \cdot, /\}$ are defined as set operations*

$$\boldsymbol{a} \circ \boldsymbol{b} := \{a \circ b \mid a \in \boldsymbol{a} \wedge b \in \boldsymbol{b}\}. \tag{2}$$

Here for division we assume that $0 \notin \boldsymbol{b}$.

It is a well established result of interval arithmetic that under this definition $\overline{\mathbb{IR}}$ is a closed calculus, i.e., the result $\boldsymbol{a} \circ \boldsymbol{b}$ again is an element of $\overline{\mathbb{IR}}$. For details see [19]. For $\boldsymbol{a} = [a_1, a_2] \in \overline{\mathbb{IR}}$ we obtain by (2) immediately

$$-\boldsymbol{a} := (-1) \cdot \boldsymbol{a} = [-a_2, -a_1] \in \overline{\mathbb{IR}}.^8 \tag{3}$$

With (3) the subtraction can be reduced to the addition by $\boldsymbol{a} - \boldsymbol{b} = \boldsymbol{a} + (-\boldsymbol{b})$.

If in (3) $\boldsymbol{a} \in \overline{\mathbb{IF}}$, then also $-\boldsymbol{a} \in \overline{\mathbb{IF}}$, i.e., $\overline{\mathbb{IF}}$ is a symmetric screen of $\overline{\mathbb{IR}}$.

Between the complete lattice $\{\overline{\mathbb{IR}}, \subseteq\}$ and its screen $\{\overline{\mathbb{IF}}, \subseteq\}$ the monotone upwardly directed rounding $\Diamond : \overline{\mathbb{IR}} \to \overline{\mathbb{IF}}$ is uniquely defined. It is characterized by the following properties:

(R1) $\Diamond\, \boldsymbol{a} = \boldsymbol{a}$, for all $\boldsymbol{a} \in \overline{\mathbb{IF}}$. (projection)
(R2) $\boldsymbol{a} \subseteq \boldsymbol{b} \Rightarrow \Diamond\, \boldsymbol{a} \subseteq \Diamond\, \boldsymbol{b}$, for $\boldsymbol{a}, \boldsymbol{b} \in \overline{\mathbb{IR}}$. (monotone)
(R3) $\boldsymbol{a} \subseteq \Diamond\, \boldsymbol{a}$, for all $\boldsymbol{a} \in \overline{\mathbb{IR}}$. (upwardly directed)

For $\boldsymbol{a} = [a_1, a_2] \in \overline{\mathbb{IR}}$ the result of the monotone upwardly directed rounding $\Diamond$ is

$$\Diamond\, \boldsymbol{a} = [\,\triangledown\, a_1,\ \triangle\, a_2]. \tag{4}$$

Using (1) and (2) it is easy to see that the monotone upwardly directed rounding $\Diamond : \overline{\mathbb{IR}} \to \overline{\mathbb{IF}}$ is antisymmetric, i.e.,

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

An interval $\boldsymbol{a} = [a_1, a_2]$ is frequently interpreted as a point in $\mathbb{R}^2$. This very naturally induces the order relation $\leq$ of $\mathbb{R}^2$ to the set of intervals $\overline{\mathbb{IR}}$. For two intervals $\boldsymbol{a} = [a_1, a_2]$ and $\boldsymbol{b} = [b_1, b_2]$ the relation $\leq$ is defined by $\boldsymbol{a} \leq \boldsymbol{b} :\Leftrightarrow a_1 \leq b_1 \wedge a_2 \leq b_2$.

For the $\leq$ relation for intervals compatibility properties hold between the algebraic structure and the order structure in great similarity to the real numbers. For instance:

(OD1) $\boldsymbol{a} \leq \boldsymbol{b} \Rightarrow \boldsymbol{a} + \boldsymbol{c} \leq \boldsymbol{b} + \boldsymbol{c}$, for all $\boldsymbol{c}$.
(OD2) $\boldsymbol{a} \leq \boldsymbol{b} \Rightarrow -\boldsymbol{b} \leq -\boldsymbol{a}$.
(OD3) $[0,0] \leq \boldsymbol{a} \leq \boldsymbol{b} \wedge \boldsymbol{c} \geq [0,0] \Rightarrow \boldsymbol{a} * \boldsymbol{c} \leq \boldsymbol{b} * \boldsymbol{c}$.
(OD4) $[0,0] < \boldsymbol{a} \leq \boldsymbol{b} \wedge \boldsymbol{c} > [0,0] \Rightarrow [0,0] < \boldsymbol{a}/\boldsymbol{c} \leq \boldsymbol{b}/\boldsymbol{c} \wedge \boldsymbol{c}/\boldsymbol{a} \geq \boldsymbol{c}/\boldsymbol{b} > [0,0]$.

With respect to set inclusion as an order relation arithmetic operations in $\{\overline{\mathbb{IR}}, \subseteq\}$ are inclusion isotone by (2), i.e., $\boldsymbol{a} \subseteq \boldsymbol{b} \Rightarrow \boldsymbol{a} \circ \boldsymbol{c} \subseteq \boldsymbol{b} \circ \boldsymbol{c}$ or equivalently

(OD5) $\boldsymbol{a} \subseteq \boldsymbol{b} \wedge \boldsymbol{c} \subseteq \boldsymbol{d} \Rightarrow \boldsymbol{a} \circ \boldsymbol{c} \subseteq \boldsymbol{b} \circ \boldsymbol{d}$, for all $\circ \in \{+, -, *, /\}, 0 \notin \boldsymbol{b}, \boldsymbol{d}$ for $\circ = /$. (inclusion isotone)

Setting $\boldsymbol{c}, \boldsymbol{d} = -1$ in (OD5) delivers immediately $\boldsymbol{a} \subseteq \boldsymbol{b} \Rightarrow -\boldsymbol{a} \subseteq -\boldsymbol{b}$ which differs significantly from (OD2).

Definition 7. *With the upwardly directed rounding $\Diamond : \overline{\mathbb{IR}} \to \overline{\mathbb{IF}}$ binary arithmetic operations in $\overline{\mathbb{IF}}$ are defined by semimorphism:*

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

Here for division we assume that $\boldsymbol{a}/\boldsymbol{b}$ is defined.

If an interval $\boldsymbol{a} \in \overline{\mathbb{IF}}$ is an upper bound of an interval $\boldsymbol{x} \in \overline{\mathbb{IR}}$, i.e., $\boldsymbol{x} \subseteq \boldsymbol{a}$, then by (R1), (R2), and (R3) also $\boldsymbol{x} \subseteq \Diamond\, \boldsymbol{x} \subseteq \boldsymbol{a}$. This means $\Diamond\, \boldsymbol{x}$ is the least upper bound, the supremum of $\boldsymbol{x}$ in $\overline{\mathbb{IF}}$. Similarly if for $\boldsymbol{x}, \boldsymbol{y} \in \overline{\mathbb{IF}}$, $\boldsymbol{x} \circ \boldsymbol{y} \subseteq \boldsymbol{a}$ with $\boldsymbol{a} \in \overline{\mathbb{IF}}$, then by (R1), (R2), (R3), and (RG) also $\boldsymbol{x} \circ \boldsymbol{y} \subseteq \boldsymbol{x} \mathbin{\Diamond\!\!\!\!\circ}\, \boldsymbol{y} \subseteq \boldsymbol{a}$, i.e., $\boldsymbol{x} \mathbin{\Diamond\!\!\!\!\circ}\, \boldsymbol{y}$ is the least upper bound, the supremum of $\boldsymbol{x} \circ \boldsymbol{y}$ in $\overline{\mathbb{IF}}$. Occasionally the supremum $\boldsymbol{x} \mathbin{\Diamond\!\!\!\!\circ}\, \boldsymbol{y}$ of the result $\boldsymbol{x} \circ \boldsymbol{y} \in \overline{\mathbb{IR}}$ is called the tightest enclosure of $\boldsymbol{x} \circ \boldsymbol{y}$.

Arithmetic operations in $\overline{\mathbb{IF}}$ are inclusion isotone, i.e.,

[8] An integral number a in an interval expression is interpreted as interval $[a, a]$.

(OD5) $\boldsymbol{a} \subseteq \boldsymbol{b} \wedge \boldsymbol{c} \subseteq \boldsymbol{d} \Rightarrow \boldsymbol{a} \mathbin{\Diamond\!\!\!\!\circ}\, \boldsymbol{c} \subseteq \boldsymbol{b} \mathbin{\Diamond\!\!\!\!\circ}\, \boldsymbol{d}$, for $\circ \in \{+,-,\cdot,/\}, 0 \notin \boldsymbol{b}, \boldsymbol{d}$ for $\circ = /$. (inclusion isotone)

This is a consequence of the inclusion isotony of the arithmetic operations in $\overline{\mathbb{IR}}$, of (R2) and of (RG).

Since the arithmetic operations $\boldsymbol{x} \circ \boldsymbol{y}$ in $\overline{\mathbb{IR}}$ are defined as set operations by (2) the operations $\boldsymbol{x} \mathbin{\Diamond\!\!\!\!\circ}\, \boldsymbol{y}$ for intervals of $\overline{\mathbb{IF}}$ defined by (RG) are not directly executable. The step from the definition of interval arithmetic by set operations to computer executable operations still requires some effort. But it is straight forward. For details see [19].

Comments: A frequent argument says that $-\infty$ and $+\infty$ must be considered as real numbers since they occur as values of real functions like $\ln x$ for $x = 0$ or $\tan x$ for $x = \pi/2$. A closer look at details, however, shows that this is not the case. The logarithm is defined as $\ln x := \int_1^x (1/t)dt$, for $x > 0$. It is not defined for $x = 0$. $-\infty$ is a lower bound of the function's values but not a value of the function $\ln x$.

The function $\tan x$ is defined as the quotient $\tan x := \sin x / \cos x$. Since $\cos \pi/2 = 0$, $\tan x$ is not defined for $x = 0$. $+\infty$ is an upper bound of the function's values but not a value of the function $\tan x$.

Remark 1, Irregularity: In real analysis division by zero is not defined. In interval arithmetic, however, the interval in the denominator of a quotient may contain zero. If division by an interval that contains zero is permitted the result can consist of two unbounded intervals. The general rule for computing the set $\boldsymbol{a}/\boldsymbol{b}$ with $0 \in \boldsymbol{b}$ is to remove its zero from the interval $\boldsymbol{b}$ and perform the division with the remaining set.[9] When zero is an interior point of the denominator, the set $[b_1, b_2]$ splits into the two distinct sets $[b_1, 0)$ and $(0, b_2]$. Division by these sets delivers two distinct unbounded intervals. This irregularity has successfully been used to develop the extended interval Newton method. It allows to compute all zeros of a function in a given domain. For details see [19].

Remark 2, Mincing: Evaluating an arithmetic expression or function for an interval X leads to a superset of the range of the function's values over the interval X. This overestimation decreases with the width of the interval X. It decreases quadratically if the centered form is used to represent the function. Subdivision or mincing is a common method for decreasing the overestimation. Its power is rather limited on a sequential computer. On a large parallel machine, however, it can be a very powerful and useful tool.

3.3 Unum and Ubound Arithmetic

In his recently published book *The End of Error* [5] John Gustafson develops a computing environment for real numbers and for sets of real numbers which is superior to conventional floating-point and interval arithmetic. A new number format, the *unum*[10], can more efficiently be used on computers with respect to many desirable properties like power consumption, storage requirements, bandwidth, parallelism concerns, and even speed. It gets mathematical rigor that even conventional interval arithmetic is not able to attain.

By obvious reasons John Gustafson's book strives for being upward compatable with IEEE 754 floating-point arithmetic and with traditional interval arithmetic. From the mathematical point of view, however, there is no need for doing this. Here we show that the new computing environment perfectly fits into an abstract mathematical approach to computer arithmetic as sketched in section 2. Like conventional closed real intervals also unums and ubounds just deal with sets of real numbers. $-\infty$ and $+\infty$ are not real numbers. They are just used as bounds to describe sets of real numbers. They are, however, themselves not elements of these sets. There is absolutely no need for introducing non mathematical entities like $-0, +0, NaN$ (not a number) or NaI (not an interval) in this new computing environment. Focusing on the mathematical core of the new computing scheme leads to several additional simplifications.

A *unum* is a bit string of variable length that has six subfields: the *sign bit s, exponent, fraction, uncertainty bit u (ubit), exponent size,* and *fraction size.* The first three subfields describe a floating-point number. If the ubit is 0, the number is exact. If it is 1, it is inexact. An inexact unum represents the set of all real numbers in the open interval between the floating-point part of the unum and the floating-point number one bit further from zero. The last two subfields, the exponent size and the fraction size are used to automatically shrink or enlarge the number of bits used for the representation

[9] This is in full accordance with function evaluation: When evaluating a function over a set, points outside its domain are simply ignored.

[10] stands for **u**niversal **num**ber.

s	*exponent*	*fraction*	*u*	*exp.size*	*fract.size*

Fig. 1. The universal number format *unum*.

of the exponent and the fraction part of the unum depending on results of operations. This automatic scaling adapts the word size to the needs of the computation. The set of all unums is denoted by $\mathbb{U}$. By the definition of unums $-\infty$ and $+\infty$ are elements of $\mathbb{U}$.

A *ubound* is a single unum or a pair of unums that represent a mathematical interval of the real line. Closed endpoints are represented by exact unums (ubit $= 0$), and open endpoints are represented by inexact unums (ubit $= 1$). So the ubit in a unbound's bound describes the kind of bracket that is used in the representation of the ubound. It is closed, if the ubit is 0 and it is open, if the ubit is 1. We denote the set of all ubounds by $\mathbb{JU}$. Later we shall occasionally denote an element $\boldsymbol{a} \in \mathbb{JU}$ by $\boldsymbol{a}$ $= \langle a_1, a_2\rangle$ where a_1, a_2 are floating-point numbers of $\overline{\mathbb{F}}$ and each one of the angle brackets $\langle$ and $\rangle$ can be open or closed.

The ubit after the floating-point part of a unum can be 0 or 1. So the set of unums $\mathbb{U}$ is a superset of the set of floating-point numbers, $\mathbb{U} \supset \mathbb{F}$. Nevertheless the unums are a linearly ordered set $\{\mathbb{U}, \leq\}$. For positive floating-point numbers the unum with ubit 0 is less than the unum with ubit 1 and for negative floating-point numbers the unum with ubit 0 is greater than the unum with ubit 1. In section 3.1 it was shown that with the following notations $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$ and $\overline{\mathbb{F}} := \mathbb{F} \cup \{-\infty, +\infty\}$ the ordered set $\{\overline{\mathbb{F}}, \leq\}$ is a screen of $\{\overline{\mathbb{R}}, \leq\}$. It is now easy to see that the ordered set of unums $\{\mathbb{U}, \leq\}$ is also a screen of $\{\overline{\mathbb{R}}, \leq\}$. It is a larger, i.e., a finer screen than $\{\overline{\mathbb{F}}, \leq\}$.[11]

The directed roundings $\bigtriangledown$ resp. $\bigtriangleup$ can now be extended as mappings from the extended set of real numbers $\overline{\mathbb{R}}$ onto the set of unums $\mathbb{U}$, $\bigtriangledown : \overline{\mathbb{R}} \to \mathbb{U}$ and $\bigtriangleup : \overline{\mathbb{R}} \to \mathbb{U}$. It is easy to see that (1) also holds for these newly defined mappings.

These roundings $\bigtriangledown$ and $\bigtriangleup$ can most frequently be used to map intervals or sets of real numbers onto ubounds. Here $\bigtriangledown$ delivers the lower bound and $\bigtriangleup$ the upper bound. This allows to express the ubit of the unum by the bracket of the ubound. Exact unums are expressed by closed endpoints, by square brackets. A closed endpoint is an element of the ubound. Inexact unums are expressed by open endpoints, by round brackets. An open endpoint is just a bound but not an element of the ubound.

We illustrate these roundings by simple examples. We use the decimal number system, a fraction part of three digits, and a space before the ubit. The following results are possible:

$$\begin{array}{ll}
\bigtriangledown(0.543216) = 0.543\ 1 = (0.543, & \bigtriangleup(0.543216) = 0.543\ 1 = 0.544), \\
\bigtriangledown(0.543) = 0.543\ 0 = [0.543, & \bigtriangleup(0.543) = 0.543\ 0 = 0.543], \\
\bigtriangledown(-0.543216) = -0.543\ 1 = (-0.544, & \bigtriangleup(-0.543216) = -0.543\ 1 = -0.543), \\
\bigtriangledown(-0.543) = -0.543\ 0 = [-0.543, & \bigtriangleup(-0.543) = -0.543\ 0 = -0.543].
\end{array}$$

Let now $\mathbb{JR}$ denote the set of bounded or unbounded real intervals where each bound can be open or closed. So $\mathbb{JR}$[12] denotes the set of open or closed or half-open intervals of real numbers. Besides of the empty set every interval of $\mathbb{JR}$ can be expressed by round and/or square brackets. If the bracket adjacent to a bound is round, the bound is not an element of the interval; if it is square the bound is an element of the interval.

With set inclusion as an order relation the ordered set $\{\mathbb{JR}, \subseteq\}$ is a complete lattice. The infimum of two or more elements of $\{\mathbb{JR}, \subseteq\}$ is the intersection and the supremum is the convex hull. The subset of $\mathbb{JR}$ where all bounds are unums of $\mathbb{U}$ is denoted by $\mathbb{JU}$. Then $\{\mathbb{JU}, \subseteq\}$ is a screen of $\{\mathbb{JR}, \subseteq\}$. In both sets $\mathbb{JR}$ and $\mathbb{JU}$ the infimum of two or more elements of $\mathbb{JR}$ and $\mathbb{JU}$ is the intersection and the supremum is the interval (convex) hull. The least element of both sets $\mathbb{JR}$ and $\mathbb{JU}$ is the empty set $\varnothing$ and the greatest element is the set $\mathbb{R} = (-\infty, +\infty)$. Elements of $\mathbb{JR}$ and $\mathbb{JU}$ are denoted by bold letters.

[11] This makes it plausible that unum arithmetic can lead to better results than floating-point arithmetic.

[12] We do not introduce a separate symbol for the subset of bounded such intervals here as in the case of real intervals.

Definition 8. *For elements $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{JR}$ we define arithmetic operations $\circ \in \{+, -, \cdot, /\}$ as set operations*

$$\boldsymbol{a} \circ \boldsymbol{b} := \{a \circ b \mid a \in \boldsymbol{a} \wedge b \in \boldsymbol{b}\}. \tag{5}$$

Here for division we assume that $0 \notin \boldsymbol{b}$.

Explicit formulas for the operations $\boldsymbol{a} \circ \boldsymbol{b}, \circ \in \{+, -, \cdot, /\}$ can be obtained in great similarity to the operations in $\overline{\mathbb{IR}}$. For derivation see [19]. However, each bound of the resulting interval in $\mathbb{JR}$ can now be open or closed.

It is a well established result that under Definition (5) $\mathbb{JR}$ is a closed calculus, i.e., the result $\boldsymbol{a} \circ \boldsymbol{b}$ is again an element of $\mathbb{JR}$. For details see [5].

Remark 3: A bound of the result $\boldsymbol{a} \circ \boldsymbol{b}$ in (5) is closed if and only if the ubit of the adjacent number is zero, i.e., the number is an exact unum. This can only happen if both operands for computing the bound come from closed interval bounds or if one of the operands is zero. In case of an inexact unum in any of the operands the bound is open.

Let us now denote an interval $\boldsymbol{a} \in \mathbb{JR}$ by $\boldsymbol{a} = \langle a_1, a_2 \rangle$, where each one of the angle brackets $\langle$ and $\rangle$ can be open or closed. Then we obtain by (5) immediately

$$-\boldsymbol{a} := \text{[13]}(-1)\cdot\boldsymbol{a} = (-1) \cdot \{x \mid a_1 \leq x \leq a_2\} = \{x \mid -a_2 \leq x \leq -a_1\} = \langle -a_2, -a_1 \rangle \in \mathbb{JR}. \tag{6}$$

$$-\langle a_1, a_2 \rangle = \langle -a_2, -a_1 \rangle. \tag{7}$$

More precisely: If the lower bound of the interval $\boldsymbol{a}$ is open (resp. closed) then the upper bound of $-\boldsymbol{a}$ is open (resp. closed), and if the upper bound of $\boldsymbol{a}$ is open (resp. closed) then the lower bound in $-\boldsymbol{a}$ is open (resp. closed).

With (7) subtraction can be reduced to addition by $\boldsymbol{a} - \boldsymbol{b} = \boldsymbol{a} + (-\boldsymbol{b})$.

If in (7) $\boldsymbol{a} \in \mathbb{JU}$, then also $-\boldsymbol{a} \in \mathbb{JU}$, i.e., $\mathbb{JU}$ is a symmetric screen of $\mathbb{JR}$.

Between the complete lattice $\{\mathbb{JR}, \subseteq\}$ and its screen $\{\mathbb{JU}, \subseteq\}$ the monotone upwardly directed rounding $\Diamond : \mathbb{JR} \to \mathbb{JU}$ is uniquely defined by the following properties:

(R1) $\Diamond\, \boldsymbol{a} = \boldsymbol{a}$, for all $\boldsymbol{a} \in \mathbb{JU}$. (projection)
(R2) $\boldsymbol{a} \subseteq \boldsymbol{b} \Rightarrow \Diamond\, \boldsymbol{a} \subseteq \Diamond\, \boldsymbol{b}$, for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{JR}$. (monotone)
(R3) $\boldsymbol{a} \subseteq \Diamond\, \boldsymbol{a}$, for all $\boldsymbol{a} \in \mathbb{JR}$. (upwardly directed)

For $\boldsymbol{a} = \langle a_1, a_2 \rangle \in \mathbb{JR}$ the result of the monotone upwardly directed rounding $\Diamond$ can be expressed by

$$\Diamond\, \boldsymbol{a} = \langle \triangledown a_1, \triangle a_2 \rangle. \tag{8}$$

where again each one of the angle brackets $\langle$ and $\rangle$ can be open or closed.

Similarly to the case of closed real intervals of $\overline{\mathbb{IR}}$ we now define an order relation $\leq$ for intervals of $\mathbb{JR}$. For intervals $\boldsymbol{a} = \langle a_1, a_2 \rangle$, $\boldsymbol{b} = \langle b_1, b_2 \rangle \in \mathbb{JR}$, the relation $\leq$ is defined by $\boldsymbol{a} \leq \boldsymbol{b} :\Leftrightarrow \langle a_1 \leq \langle b_1 \wedge a_2 \rangle \leq b_2 \rangle$. So we have for instance: $[1, 2) \leq (1, 2]$, or $[-2, -1) \leq (-2, -1]$.

For the $\leq$ relation for intervals compatibility properties hold between the algebraic structure and the order structure in great similarity to the real numbers. For instance:

(OD1) $\boldsymbol{a} \leq \boldsymbol{b} \Rightarrow \boldsymbol{a} + \boldsymbol{c} \leq \boldsymbol{b} + \boldsymbol{c}$, for all $\boldsymbol{c}$.
(OD2) $\boldsymbol{a} \leq \boldsymbol{b} \Rightarrow -\boldsymbol{b} \leq -\boldsymbol{a}$.
(OD3) $0 \leq \boldsymbol{a} \leq \boldsymbol{b} \wedge \boldsymbol{c} \geq 0 \Rightarrow \boldsymbol{a} \cdot \boldsymbol{c} \leq \boldsymbol{b} \cdot \boldsymbol{c}$.
(OD4) $0 < \boldsymbol{a} \leq \boldsymbol{b} \wedge \boldsymbol{c} > 0 \Rightarrow 0 < \boldsymbol{a}/\boldsymbol{c} \leq \boldsymbol{b}/\boldsymbol{c} \wedge \boldsymbol{c}/\boldsymbol{a} \geq \boldsymbol{c}/\boldsymbol{b} > 0$.

With respect to set inclusion as an order relation arithmetic operations in $\{\mathbb{JR}, \subseteq\}$ are inclusion isotone by (5), i.e., $\boldsymbol{a} \subseteq \boldsymbol{b} \Rightarrow \boldsymbol{a} \circ \boldsymbol{c} \subseteq \boldsymbol{b} \circ \boldsymbol{c}$ or equivalently

(OD5) $\boldsymbol{a} \subseteq \boldsymbol{b} \wedge \boldsymbol{c} \subseteq \boldsymbol{d} \Rightarrow \boldsymbol{a} \circ \boldsymbol{c} \subseteq \boldsymbol{b} \circ \boldsymbol{d}$, for all $\circ \in \{+, -, \cdot, /\}, 0 \notin \boldsymbol{b}, \boldsymbol{d}$ for $\circ = /$. (inclusion isotone)

Setting $\boldsymbol{c}, \boldsymbol{d} = -1$ in (OD5) delivers immediately $\boldsymbol{a} \subseteq \boldsymbol{b} \Rightarrow -\boldsymbol{a} \subseteq -\boldsymbol{b}$ which differs significantly from (OD2).

Using (1) and (5) it is easy to see that the monotone upwardly directed rounding $\Diamond : \mathbb{JR} \to \mathbb{JU}$ is antisymmetric, i.e.,

[13] An integral number a in a ubound expression is interpreted as ubound $[a, a]$.

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

Definition 9. *With the upwardly directed rounding* $\Diamond : \mathbb{IR} \to \mathbb{IU}$ *binary arithmetic operations in* $\mathbb{IU}$ *are defined by semimorphism:*

(RG) $\boldsymbol{a} \mathbin{\Diamond\!\!\!\!\!\circ\,} \boldsymbol{b} := \Diamond\, (\boldsymbol{a} \circ \boldsymbol{b})$, *for all* $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{IU}$ *and all* $\circ \in \{+, -, \cdot, /\}$.

Here for division we assume that $\boldsymbol{a}/\boldsymbol{b}$ *is defined.*

If a ubound $\boldsymbol{a} \in \mathbb{IU}$ is an upper bound of a ubound $\boldsymbol{x} \in \mathbb{IR}$, i.e., $\boldsymbol{x} \subseteq \boldsymbol{a}$, then by (R1), (R2), and (R3) also $\boldsymbol{x} \subseteq\ \Diamond\, \boldsymbol{x} \subseteq \boldsymbol{a}$. This means $\Diamond\, \boldsymbol{x}$ is the least upper bound, the supremum of $\boldsymbol{x}$ in $\mathbb{IU}$. Similarly if for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{IU}$, $\boldsymbol{x} \circ \boldsymbol{y} \subseteq \boldsymbol{a}$ with $\boldsymbol{a} \in \mathbb{IU}$, then by (R1), (R2), (R3), and (RG) also $\boldsymbol{x} \circ \boldsymbol{y} \subseteq \boldsymbol{x} \mathbin{\Diamond\!\!\!\!\!\circ\,} \boldsymbol{y} \subseteq \boldsymbol{a}$, i.e., $\boldsymbol{x} \mathbin{\Diamond\!\!\!\!\!\circ\,} \boldsymbol{y}$ is the least upper bound, the supremum of $\boldsymbol{x} \circ \boldsymbol{y}$ in $\mathbb{IU}$. Occasionally the supremum $\boldsymbol{x} \mathbin{\Diamond\!\!\!\!\!\circ\,} \boldsymbol{y} \in \mathbb{IU}$ of the result $\boldsymbol{x} \circ \boldsymbol{y} \in \mathbb{IR}$ is called the tightest enclosure of $\boldsymbol{x} \circ \boldsymbol{y}$.

Arithmetic operations in $\mathbb{IU}$ are inclusion isotone, i.e.,

(OD5) $\boldsymbol{a} \subseteq \boldsymbol{b} \wedge \boldsymbol{c} \subseteq \boldsymbol{d} \Rightarrow \boldsymbol{a} \mathbin{\Diamond\!\!\!\!\!\circ\,} \boldsymbol{c} \subseteq \boldsymbol{b} \mathbin{\Diamond\!\!\!\!\!\circ\,} \boldsymbol{d}$, for $\circ \in \{+, -, \cdot, /\}, 0 \notin \boldsymbol{b}, \boldsymbol{d}$ for $\circ = /$. (inclusion isotone)

This is a consequence of the inclusion isotony of the arithmetic operations in $\mathbb{IR}$, of (R2) and of (RG).

Ubound arithmetic as conventional interval arithmetic deals with sets of real numbers. Three kinds of unbounded intervals can occur in $\mathbb{IR}$ and $\mathbb{IU}$: $(-\infty, a\rangle$, $\langle b, +\infty)$ and $(-\infty, +\infty)$ with $a, b \in \mathbb{R}$, where the angle brackets can be open or closed. Since $-\infty$ and $+\infty$ are not real numbers the brackets adjacent to $-\infty$ and $+\infty$ can only be open, i.e., $-\infty$ and $+\infty$ are not elements of the unbounded intervals. This very naturally leads to the following rules:

$$(-\infty, a\rangle \cdot 0 = \langle b, +\infty) \cdot 0 = (-\infty, +\infty) \cdot 0 = 0.$$

Since the arithmetic operations $\boldsymbol{x} \circ \boldsymbol{y}$ in $\mathbb{IR}$ are defined as set operations by (5) the operations $\boldsymbol{x} \mathbin{\Diamond\!\!\!\!\!\circ\,} \boldsymbol{y}$ for ubounds of $\mathbb{IU}$ defined by (RG) are not directly executable. The step from the definition of arithmetic by set operations to computer executable operations still requires some effort. We discuss this question in the next section. For details see also [19] and [5].

3.4 Executable Ubound Arithmetic

We now consider the question how executable formulas for ubound arithmetic can be obtained. Let $\boldsymbol{a} = \langle a_1, a_2\rangle, \boldsymbol{b} = \langle b_1, b_2\rangle \in \mathbb{IR}$. Arithmetic in $\mathbb{IR}$ is defined by

$$\boldsymbol{a} \circ \boldsymbol{b} := \{a \circ b \mid a \in \boldsymbol{a} \wedge b \in \boldsymbol{b}\}, \tag{9}$$

for all $\circ \in \{+, -, \cdot, /\}$, $0 \notin \boldsymbol{b}$ in case of division. The function $a \circ b$ is continuous with respect to both variables. The set $\boldsymbol{a} \circ \boldsymbol{b}$ is the range of the function $a \circ b$ over the product set $\boldsymbol{a} \times \boldsymbol{b}$ with or without the boundaries depending on the open-closedness of $\boldsymbol{a}$ and $\boldsymbol{b}$. Since $\boldsymbol{a}$ and $\boldsymbol{b}$ are intervals of $\mathbb{IR}$ the set $\boldsymbol{a} \times \boldsymbol{b}$ is a simply connected subset of $\overline{\mathbb{R}}^2$, ($\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$). In such a region the range $\boldsymbol{a} \circ \boldsymbol{b}$ of the function $a \circ b$ is also simply connected. Therefore

$$\boldsymbol{a} \circ \boldsymbol{b} = \langle \inf(\boldsymbol{a} \circ \boldsymbol{b}), \sup(\boldsymbol{a} \circ \boldsymbol{b})\rangle, \tag{10}$$

i.e., for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{IR}$, $0 \notin \boldsymbol{b}$ in case of division, $\boldsymbol{a} \circ \boldsymbol{b}$ is again an interval of $\mathbb{IR}$.

The angle brackets on the right hand side of (10) depend on the open-closed endpoints of the intervals $\boldsymbol{a}$ and $\boldsymbol{b}$. The elements $-\infty$ and $+\infty$ can occur as bounds of real intervals. But they are themselves not elements of these intervals.

Neither the set definition (9) of the arithmetic operations $\boldsymbol{a} \circ \boldsymbol{b}$, $\circ \in \{+, -, \cdot, /\}$, nor the form (10) can be executed on the computer. So we have to derive more explicit formulas.

We demonstrate this in case of addition. By (OD1) we obtain $a_1 \leq \boldsymbol{a}$ and $b_1 \leq \boldsymbol{b} \Rightarrow a_1 + b_1 \leq \inf(\boldsymbol{a} + \boldsymbol{b})$. On the other hand $\inf(\boldsymbol{a} + \boldsymbol{b}) \leq a_1 + b_1$. From both inequalities we obtain by (O3): $\inf(\boldsymbol{a} + \boldsymbol{b}) = a_1 + b_1$. Analogously one obtains $\sup(\boldsymbol{a} + \boldsymbol{b}) = a_2 + b_2$. Thus

$$\boldsymbol{a} + \boldsymbol{b} = \langle \inf(\boldsymbol{a} + \boldsymbol{b}),\ \sup(\boldsymbol{a} + \boldsymbol{b})\rangle = \langle a_1 + b_1, a_2 + b_2\rangle.$$

Similarly by making use of (OD1,2,3,4) for intervals of $\mathbb{JR}$ and the simple sign rules $-(\boldsymbol{a}\cdot\boldsymbol{b}) = (-\boldsymbol{a})\cdot\boldsymbol{b} = \boldsymbol{a}\cdot(-\boldsymbol{b}), -(\boldsymbol{a}/\boldsymbol{b}) = (-\boldsymbol{a})/\boldsymbol{b} = \boldsymbol{a}/(-\boldsymbol{b})$ explicit formulas for all interval operations can be derived, [19].

Actually the infimum and supremum in (10) is taken for operations with the bounds. For bounded intervals $\boldsymbol{a} = \langle a_1, a_2\rangle$ and $\boldsymbol{b} = \langle b_1, b_2\rangle$ the following formula holds for all operations with $0 \notin \boldsymbol{b}$ in case of division:

$$\boldsymbol{a}\circ\boldsymbol{b} = \langle \min_{i,j=1,2}(a_i\circ b_j), \max_{i,j=1,2}(a_i\circ b_j)\rangle \text{ for } \circ\in\{+,-,\cdot,/\}. \tag{11}$$

In summary we can state:

Arithmetic in $\mathbb{JR}$ is an algebraically closed subset of the arithmetic in the powerset of the real numbers.

Now we get by (RG) for intervals of $\mathbb{JU}$

$$\boldsymbol{a} \mathbin{\Diamond\mkern-13mu{\scriptscriptstyle\circ}\mkern5mu} \boldsymbol{b} := \ \Diamond\ (\boldsymbol{a}\circ\boldsymbol{b}) = \langle\ \triangledown\ \min_{i,j=1,2}(a_i\circ b_j),\ \vartriangle\ \max_{i,j=1,2}(a_i\circ b_j)\rangle$$

and by the monotonicity of the roundings $\triangledown$ and $\vartriangle$:

$$\boldsymbol{a} \mathbin{\Diamond\mkern-13mu{\scriptscriptstyle\circ}\mkern5mu} \boldsymbol{b} = \langle \min_{i,j=1,2}(a_i \mathbin{\triangledown\mkern-13mu{\scriptscriptstyle\circ}\mkern5mu} b_j), \max_{i,j=1,2}(a_i \mathbin{\vartriangle\mkern-13mu{\scriptscriptstyle\circ}\mkern5mu} b_j)\rangle.$$

For bounded and nonempty intervals $\boldsymbol{a} = \langle a_1, a_2\rangle$ and $\boldsymbol{b} = \langle b_1, b_2\rangle$ of $\mathbb{JU}$ the unary operation $-\boldsymbol{a}$ and the binary operations addition, subtraction, multiplication, and division are shown in the following tables. For details see [19]. Therein the operator symbols for intervals are simply denoted by $+, -, \cdot, /$.

Minus operator $\qquad -\boldsymbol{a} = \langle -a_2, -a_1\rangle$.
Addition $\qquad \langle a_1, a_2\rangle + \langle b_1, b_2\rangle = \langle a_1 \mathbin{\triangledown\mkern-13mu{\scriptscriptstyle +}\mkern5mu} b_1, a_2 \mathbin{\vartriangle\mkern-13mu{\scriptscriptstyle +}\mkern5mu} b_2\rangle$.
Subtraction $\qquad \langle a_1, a_2\rangle - \langle b_1, b_2\rangle = \langle a_1 \mathbin{\triangledown\mkern-13mu{\scriptscriptstyle -}\mkern5mu} b_2, a_2 \mathbin{\vartriangle\mkern-13mu{\scriptscriptstyle -}\mkern5mu} b_1\rangle$.

Multiplication $\langle a_1, a_2\rangle \cdot \langle b_1, b_2\rangle$	$\langle b_1, b_2\rangle$ $b_2 \le 0$	$\langle b_1, b_2\rangle$ $b_1 < 0 < b_2$	$\langle b_1, b_2\rangle$ $b_1 \ge 0$
$\langle a_1, a_2\rangle, a_2 \le 0$	$\langle a_2 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_2, a_1 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_1\rangle$	$\langle a_1 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_2, a_1 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_1\rangle$	$\langle a_1 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_2, a_2 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_1\rangle$
$a_1 < 0 < a_2$	$\langle a_2 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_1, a_1 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_1\rangle$	$\langle min(a_1 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_2, a_2 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_1),$ $max(a_1 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_1, a_2 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_2)\rangle$	$\langle a_1 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_2, a_2 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_2\rangle$
$\langle a_1, a_2\rangle, a_1 \ge 0$	$\langle a_2 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_1, a_1 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_2\rangle$	$\langle a_2 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_1, a_2 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_2\rangle$	$\langle a_1 \mathbin{\triangledown\mkern-12.5mu\cdot\mkern6mu} b_1, a_2 \mathbin{\vartriangle\mkern-12.5mu\cdot\mkern6mu} b_2\rangle$

Division, $0 \notin \boldsymbol{b}$ $\langle a_1, a_2\rangle / \langle b_1, b_2\rangle$	$\langle b_1, b_2\rangle$ $b_2 < 0$	$\langle b_1, b_2\rangle$ $b_1 > 0$
$\langle a_1, a_2\rangle, a_2 \le 0$	$\langle a_2 \mathbin{\triangledown\mkern-12mu/\mkern4mu} b_1, a_1 \mathbin{\vartriangle\mkern-12mu/\mkern4mu} b_2\rangle$	$\langle a_1 \mathbin{\triangledown\mkern-12mu/\mkern4mu} b_1, a_2 \mathbin{\vartriangle\mkern-12mu/\mkern4mu} b_2\rangle$
$\langle a_1, a_2\rangle, a_1 < 0 < a_2$	$\langle a_2 \mathbin{\triangledown\mkern-12mu/\mkern4mu} b_2, a_1 \mathbin{\vartriangle\mkern-12mu/\mkern4mu} b_2\rangle$	$\langle a_1 \mathbin{\triangledown\mkern-12mu/\mkern4mu} b_1, a_2 \mathbin{\vartriangle\mkern-12mu/\mkern4mu} b_1\rangle$
$\langle a_1, a_2\rangle, 0 \le a_1$	$\langle a_2 \mathbin{\triangledown\mkern-12mu/\mkern4mu} b_2, a_1 \mathbin{\vartriangle\mkern-12mu/\mkern4mu} b_1\rangle$	$\langle a_1 \mathbin{\triangledown\mkern-12mu/\mkern4mu} b_2, a_2 \mathbin{\vartriangle\mkern-12mu/\mkern4mu} b_1\rangle$

In real analysis division by zero is not defined. In interval arithmetic, however, the interval in the denominator of a quotient may contain zero. We consider this case also.

The general rule for computing the set $\boldsymbol{a}/\boldsymbol{b}$ with $0 \in \boldsymbol{b}$ is to remove its zero from the interval $\boldsymbol{b}$ and perform the division with the remaining set.[14] Whenever zero in $\boldsymbol{b}$ is an endpoint of $\boldsymbol{b}$, the result of the division can be obtained directly from the above table for division with $0 \notin \boldsymbol{b}$ by the limit process $b_1 \to 0$ or $b_2 \to 0$ respectively. The results are shown in the table for division with $0 \in \boldsymbol{b}$. Here, the round brackets stress that the bounds $-\infty$ and $+\infty$ are not elements of the interval. When zero is an interior point of the denominator, the set $\langle b_1, b_2\rangle$ splits into the distinct sets $\langle b_1, 0)$ and $(0, b_2\rangle$,

[14] This is in full accordance with function evaluation: When evaluating a function over a set, points outside its domain are simply ignored.

Division, $0 \in \boldsymbol{b}$	$\boldsymbol{b} =$	$\langle b_1, b_2\rangle$	$\langle b_1, b_2\rangle$
$\langle a_1, a_2\rangle / \langle b_1, b_2\rangle$	$\langle 0, 0\rangle$	$b_1 < b_2 = 0$	$0 = b_1 < b_2$
$\langle a_1, a_2\rangle = \langle 0, 0\rangle$	$\varnothing$	$\langle 0, 0\rangle$	$\langle 0, 0\rangle$
$\langle a_1, a_2\rangle, a_1 < 0, a_2 \leq 0$	$\varnothing$	$\langle a_2 \bigtriangledown b_1, +\infty)$	$(-\infty, a_2 \bigtriangleup b_2\rangle$
$\langle a_1, a_2\rangle, a_1 < 0 < a_2$	$\varnothing$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$\langle a_1, a_2\rangle, 0 \leq a_1, 0 < a_2$	$\varnothing$	$(-\infty, a_1 \bigtriangleup b_1\rangle$	$\langle a_1 \bigtriangledown b_2, +\infty)$

and the division by $\langle b_1, b_2\rangle$ actually means two divisions. The results of the two divisions are already shown in the table for division by $0 \in \boldsymbol{b}$.

However, in the user's program the two divisions appear as a single operation, as division by an interval $\langle b_1, b_2\rangle$ with $b_1 < 0 < b_2$, an operation that delivers two distinct results.

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: return the entire set of real numbers $(-\infty, +\infty)$ as result and continue the computation, or set a flag and 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 modify the operands and recompute, or stop computing, or some other action.

An alternative would be to provide a second division which in case of division by an interval that contains zero as an interior point generally delivers the result $(-\infty, +\infty)$. Then the user can decide when to use which division in his program.

Deriving explicit formulas for the result of interval operations $\boldsymbol{a} \circ \boldsymbol{b}, \circ \in \{+, -, \cdot, /\}$, it was assumed so far that the intervals $\boldsymbol{a}$ and $\boldsymbol{b}$ are nonempty and bounded. Four kinds of *extended intervals* come from division by an interval of $\mathbb{JU}$ that contains zero:

$$\varnothing, \quad (-\infty, a\rangle, \quad \langle b, +\infty), \quad \text{and} \quad (-\infty, +\infty).$$

To extend the operations to these more general intervals the first rule is that any operation with the empty set $\varnothing$ returns the empty set. Intervals of $\mathbb{JU}$ are connected sets of real numbers. $-\infty$ and $+\infty$ are not elements of these intervals. So multiplication of any such interval by 0 can only have 0 as the result. This very naturally leads to the following rules:

$$(-\infty, a\rangle \cdot 0 = \langle b, +\infty) \cdot 0 = (-\infty, +\infty) \cdot 0 = 0.$$

For intervals of $\mathbb{JU}$ we can now state:

Arithmetic for closed, open, and half-open, bounded or unbounded real intervals of $\mathbb{JU}$ is free of exceptions, i.e., arithmetic operations for intervals of $\mathbb{JU}$ always lead to intervals of $\mathbb{JU}$ again.

This is in sharp contrast to other models of interval arithmetic which consider $-\infty$ and $+\infty$ as elelements of unbounded real intervals. In such models obscure arithmetic operations like $\infty - \infty$, ∞/∞, $0 \cdot \infty$ occur which require introduction of unnatural superficial objects like NaI (Not an Interval).

High speed by support of hardware and programming languages is vital for all kinds of interval arithmetic to be more widely accepted by the scientific computing community. Right now no commercial processor provides interval arithmetic or unum and ubound arithmetic by hardware. In the author's book *Computer Arithmetic and Validity - Theory, Implementation, and Applications*, second edition 2013 [19] considerable emphasis is put on speeding up interval arithmetic. The book shows that interval arithmetic for diverse spaces can efficiently be provided on the computer if two features are made available by fast hardware:

I. Fast and direct hardware support for double precision interval arithmetic and

II. a fast and exact multiply and accumulate operation or, an exact dot product (EDP).

Realization of I. and II. is discussed at detail in the book [19]. It is shown that I. and II. can be obtained at very little hardware cost. With I. interval arithmetic would be as fast as simple floating-point arithmetic. The simplest and fastest way for computing a dot product is to compute it exactly. To make II. conveniently available a new data format *complete* is used together with a few very restricted arithmetic operations. By pipelining the EDP can be computed in the time the processor needs to read the data, i.e., it comes with utmost speed. I. and II. would boost both the speed of a computation and the accuracy of the result. Fast hardware support for I. and II. must be supported by future processors. Computing the dot product exactly even can be faster than computing it conventionally in double or extended precision floating-point arithmetic.

Modern processor architecture is coming considerably close to what is requested here. See [8], and in particular pp.1-1 to 1-3 and 2-5 to 2-6. These processors provide register space of 16 K bits. Only about 4 K bits suffice for a *complete register* which allows computing a dot product exactly at extreme speed for the double precision format. In the following section we discuss a frequent application of this.

4 A Sketch of Arithmetic for Matrices with Ubound Components

The axioms for computer arithmetic shown in section 2 also can be applied to define computer arithmetic in higher dimensional spaces like complex numbers, vectors and matrices for real, complex, interval and ubound data, for instance. Here we briefly sketch how arithmetic for matrices with interval and ubound components could be embedded into the axiomatic definition of computer arithmetic outlined in section 2.

Let $\{\overline{\mathbb{R}}, +, \cdot, \leq\}$ be the completely ordered set of real numbers and $\{U, \leq\}$ the symmetric screen of unums. In the ordered set of $n \times n$ matrices $\{M_n\mathbb{R}, +, \cdot, \leq\}$ we consider intervals $\mathbb{J}M_n\mathbb{R}$ and $\mathbb{J}M_nU$ where all bounds can be open or closed. Let $\mathbb{P}M_n\mathbb{R}$ denote the power set[15] of $M_n\mathbb{R}$. Then $\mathbb{P}M_n\mathbb{R} \supset \mathbb{J}M_n\mathbb{R} \supset \mathbb{J}M_n\mathbb{U}$. $\mathbb{J}M_n\mathbb{R}$ is an upper[16] screen of $\mathbb{P}M_n\mathbb{R}$ and $\mathbb{J}M_n\mathbb{U}$ is a screen of $\mathbb{J}M_n\mathbb{R}$. We consider the monotone upwardly directed roundings $\square : \mathbb{P}M_n\mathbb{R} \to \mathbb{J}M_n\mathbb{R}$ and $\Diamond : \mathbb{J}M_n\mathbb{R} \to \mathbb{J}M_nU$. They are uniquely defined.

For matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{J}M_n\mathbb{R}$ the set definition of arithmetic operations

$$\boldsymbol{A} \circ \boldsymbol{B} := \{A \circ B \mid A \in \boldsymbol{A} \wedge B \in \boldsymbol{B}\}, \circ \in \{+, \cdot\} \tag{12}$$

does not lead to an interval again. The result is a more general set. It is an element of the power set of matrices. To obtain an interval again the upwardly directed rounding from the power set onto the set of intervals of $\square : \mathbb{P}M_n\mathbb{R} \to \mathbb{J}M_n\mathbb{R}$ has to be applied. With it arithmetic operations for intervals $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{J}M_n\mathbb{R}$ are defined by

(RG) $\boldsymbol{A} \boxed{\circ} \boldsymbol{B} := \square\, (\boldsymbol{A} \circ \boldsymbol{B}), \circ \in \{+, -, \cdot\}$.

As in the case of conventional intervals subtraction can be expressed by negation and addition.

The set $\mathbb{J}M_nU$ of intervals of computer representable matrices is a screen of $\mathbb{J}M_n\mathbb{R}$. To obtain arithmetic for intervals $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{J}M_nU$ once more the monotone upwardly directed rounding, now denoted by $\Diamond : \mathbb{J}M_n\mathbb{R} \to \mathbb{J}M_nU$ is applied:

(RG) $\boldsymbol{A} \mathbin{\Diamond\kern-0.75em\circ\kern0.3em} \boldsymbol{B} := \Diamond\, (\boldsymbol{A} \boxed{\circ} \boldsymbol{B}), \circ \in \{+, \cdot\}$.

This leads to the best possible operations in the interval spaces $\mathbb{J}M_n\mathbb{R}$ and $\mathbb{J}M_nU$.

Because of the set definition of the arithmetic operations, however, these best possible operations are not directly executable on a computer. Therefore, we are now going to express them in terms of computer executable formulas. For details see [19].

To do this, we consider the set of $n \times n$ matrices $M_n\mathbb{J}\mathbb{R}$. The elements of this set have components that are intervals of $\mathbb{J}\mathbb{R}$. With the operations and the order relation $\leq$ of the latter, we define operations $\boxplus$, $\boxdot$, and an order relation $\leq$ in $M_n\mathbb{J}\mathbb{R}$ by employing the conventional definition of the operations for matrices. With $\boldsymbol{A} = (\boldsymbol{a}_{\mathbf{ij}})$, $\boldsymbol{B} = (\boldsymbol{b}_{\mathbf{ij}}) \in M_n\mathbb{J}\mathbb{R}$ let be

$$\boldsymbol{A} \boxplus \boldsymbol{B} := (\boldsymbol{a}_{\mathbf{ij}} + \boldsymbol{b}_{\mathbf{ij}}) \;\wedge\; \boldsymbol{A} \boxdot \boldsymbol{B} := \left(\sum_{\nu=\mathbf{1}}^{\mathbf{n}} \boldsymbol{a}_{\mathrm{i}\nu} \cdot \boldsymbol{b}_{\nu\mathrm{j}}\right) \;\wedge\; \boldsymbol{A} \leq \boldsymbol{B} :\Leftrightarrow \boldsymbol{a}_{\mathbf{ij}} \leq \boldsymbol{b}_{\mathbf{ij}}, \mathrm{i,j} = 1(1)\mathrm{n}.$$

[15] The power set of a set M is the set of all subsets of M.

[16] For definition see [19].

Here $+, \cdot$ are the operations in $\mathbb{J}\mathbb{R}$ as defined in (5) and $\sum$ denotes the repeated summation in $\mathbb{J}\mathbb{R}$.

Remark 4: The bounds of the components of the product matrix $\boldsymbol{A} \boxdot \boldsymbol{B}$ will be open in the majority of cases. This is a simple consequence of Remark 3. In a bit weaker form this also holds for the addition $\boldsymbol{A} \boxplus \boldsymbol{B}$.

We now define a mapping

$$\chi : M_n\mathbb{J}\mathbb{R} \to \mathbb{J}M_n\mathbb{R}$$

which for matrices $\boldsymbol{A} = (\boldsymbol{a}_{\mathbf{ij}}) \in M_n\mathbb{J}\mathbb{R}$ with $\boldsymbol{a}_{\mathbf{ij}} = \langle a_{ij}^{(1)}, a_{ij}^{(2)} \rangle \in \mathbb{J}\mathbb{R}$,[17] $i, j = 1(1)n$, has the property

$$\chi \boldsymbol{A} = \chi(\boldsymbol{a}_{\mathbf{ij}}) = \chi(\langle a_{ij}^{(1)}, a_{ij}^{(2)} \rangle) := \langle (a_{ij}^{(1)}), (a_{ij}^{(2)}) \rangle. \tag{13}$$

Obviously χ is a one-to-one mapping of $M_n\mathbb{J}\mathbb{R}$ onto $\mathbb{J}M_n\mathbb{R}$ and an order isomorphism with respect to $\leq$. It can be shown that χ is also an algebraic isomorphism for the operations addition and multiplication, i.e.,

$$\chi\boldsymbol{A} \mathbin{\square\mkern-13mu\circ\mkern4mu} \chi\boldsymbol{B} = \chi(\boldsymbol{A} \mathbin{\square\mkern-13mu\circ\mkern4mu} \boldsymbol{B}), \circ \in \{+, \cdot\}.$$

For the proof in case of closed intervals $\boldsymbol{a}_{\mathbf{ij}}, \boldsymbol{b}_{\mathbf{ij}} \in \mathbb{I}\mathbb{R}$ see [19].

Whenever two structures are isomorphic, corresponding elements can be identified with each other. This allows us to define an inclusion relation even for elements $\boldsymbol{A} = (\boldsymbol{a}_{\mathbf{ij}})$, $\boldsymbol{B} = (\boldsymbol{b}_{\mathbf{ij}}) \in M_n\mathbb{J}\mathbb{R}$ by

$$\boldsymbol{A} \subseteq \boldsymbol{B} :\Leftrightarrow \boldsymbol{a}_{\mathbf{ij}} \subseteq \boldsymbol{b}_{\mathbf{ij}}, \text{ for all i,j=1(1)n.}$$

and

$$(a_{ij}) \in \boldsymbol{A} = (A_{ij}) :\Leftrightarrow a_{ij} \in A_{ij}, \text{ for all } i, j = 1(1)n.$$

This convenient definition allows for the interpretation that a matrix $\boldsymbol{A} = (\boldsymbol{a}_{\mathbf{ij}}) \in M_n\mathbb{J}\mathbb{R}$ also represents a set of matrices as demonstrated by the following identity:

$$\boldsymbol{A} = (A_{ij}) \equiv \{(a_{ij}) \mid a_{ij} \in A_{ij}, \text{ i,j=1(1)n}\}.$$[18]

Both matrices contain the same elements.

With the monotone upwardly directed rounding $\Diamond : \mathbb{J}\mathbb{R} \to \mathbb{J}\mathbb{U}$ a rounding $\Diamond : M_n\mathbb{J}\mathbb{R} \to M_n\mathbb{J}\mathbb{U}$ and operations in $M_n\mathbb{J}\mathbb{U}$ can now be defined by

$$\begin{aligned} \Diamond\, \boldsymbol{A} &:= (\Diamond\, \boldsymbol{a}_{\mathbf{ij}}), \\ \boldsymbol{A} \mathbin{\Diamond\mkern-12.5mu{\scriptstyle\circ}\mkern5mu} \boldsymbol{B} &:= \Diamond\, (\boldsymbol{A} \mathbin{\square\mkern-13mu\circ\mkern4mu} \boldsymbol{B}), \; \circ \in \{+, \cdot\}. \end{aligned}$$

Now it can be shown (for the proof in case of closed intervals see [19]) that the mapping χ establishes an isomorphism

$$\chi\boldsymbol{A} \mathbin{\Diamond\mkern-12.5mu{\scriptstyle\circ}\mkern5mu} \chi\boldsymbol{B} = \chi(\boldsymbol{A} \mathbin{\Diamond\mkern-12.5mu{\scriptstyle\circ}\mkern5mu} \boldsymbol{B}), \circ \in \{+, \cdot\},$$

i.e., the structures $\{M_n\mathbb{J}\mathbb{U}, \mathbin{\Diamond\mkern-12.5mu{\scriptstyle\circ}\mkern5mu}, \Diamond, \leq, \subseteq\}$ and $\{\mathbb{J}M_n\mathbb{U}, \mathbin{\Diamond\mkern-12.5mu{\scriptstyle\circ}\mkern5mu}, \Diamond, \leq, \subseteq\}$ can be identified with each other.

This isomorphism reduces the optimal, best possible but not computer executable operations in $\mathbb{J}M_n\mathbb{U}$, to the operations in $M_n\mathbb{J}\mathbb{U}$. We analyze these operations more closely.

For matrices $\boldsymbol{A} = (\boldsymbol{a}_{\mathbf{ij}})$, $\boldsymbol{B} = (\boldsymbol{b}_{\mathbf{ij}}) \in M_n\mathbb{J}\mathbb{U}$, $\boldsymbol{a}_{\mathbf{ij}}, \boldsymbol{b}_{\mathbf{ij}} \in \mathbb{J}\mathbb{U}$ arithmetic operations are defined by

$$\boldsymbol{A} \mathbin{\Diamond\mkern-12.5mu{\scriptstyle+}\mkern5mu} \boldsymbol{B} := \Diamond\, (\boldsymbol{A} \boxplus \boldsymbol{B}) \quad \wedge \quad \boldsymbol{A} \mathbin{\Diamond\mkern-12mu\cdot\mkern6mu} \boldsymbol{B} := \Diamond\, (\boldsymbol{A} \boxdot \boldsymbol{B})$$

with the rounding $\Diamond\, \boldsymbol{A} := (\Diamond\, \boldsymbol{a}_{\mathbf{ij}})$. This leads to the following formulas for the operations in $M_n\mathbb{J}\mathbb{U}$:

$$\boldsymbol{A} \mathbin{\Diamond\mkern-12.5mu{\scriptstyle+}\mkern5mu} \boldsymbol{B} = (\Diamond\, (\boldsymbol{a}_{\mathbf{ij}} + \boldsymbol{b}_{\mathbf{ij}})) = (\boldsymbol{a}_{\mathbf{ij}} \mathbin{\Diamond\mkern-12.5mu{\scriptstyle+}\mkern5mu} \boldsymbol{b}_{\mathbf{ij}}), \tag{14}$$

$$\boldsymbol{A} \mathbin{\Diamond\mkern-12mu\cdot\mkern6mu} \boldsymbol{B} = \Diamond\, (\boldsymbol{A} \boxdot \boldsymbol{B}) = \left(\Diamond \sum_{\nu=1}^{n} (\boldsymbol{a}_{\mathbf{i}\nu} \cdot \boldsymbol{b}_{\nu\mathbf{j}}) \right). \tag{15}$$

These operations are executable on a computer. The componentwise addition in (14) can be performed by means of the addition in $\mathbb{J}\mathbb{U}$. The multiplications in (15) are to be executed using the

[17] The angle brackets $\langle$ and $\rangle$ here denote the interval bounds. Each one of them can be open or closed.

[18] The round brackets here denote the matrix braces.

multiplication in $\mathbb{JR}$. Then the lower bounds and the upper bounds are to be added in $\mathbb{R}$. Finally the rounding $\Diamond : \mathbb{JR} \to \mathbb{JU}$ has to be executed.

With $\boldsymbol{a}_{\mathbf{ij}}=\langle a_{ij}^1, a_{ij}^2\rangle$, $\boldsymbol{b}_{\mathbf{ij}}=\langle b_{ij}^1, b_{ij}^2\rangle \in \mathbb{JU}$, (15) can be written in a more explicit form:

$$\boldsymbol{A} \Diamond \boldsymbol{B} = \left(\langle \bigtriangledown \sum_{\nu=1}^{n} \min_{r,s=1,2} (a_{i\nu}^r b_{\nu j}^s),\ \bigtriangleup \sum_{\nu=1}^{n} \max_{r,s=1,2} (a_{i\nu}^r b_{\nu j}^s) \rangle \right). \tag{16}$$

Here the products $a_{i\nu}^r b_{\nu j}^s$ are elements of $\mathbb{R}$ (and in general not of $\mathbb{U}$). **The summands (products of double length) are to be correctly accumulated in $\mathbb{R}$ by the exact scalar product.** Finally the sum of products is rounded only once by $\bigtriangledown$ resp. $\bigtriangleup$ from $\mathbb{R}$ onto $\mathbb{U}$. The angle brackets in (16) denote the interval bounds. Each one of them can be open or closed. The large round brackets denote the matrix braces. In the vast majority of cases the angle brackets will be open. Only in the very rare case that a sum before rounding is an exact unum the angle bracket is closed.

5 Short Term Progress

Compared with conventional interval arithmetic *The End of Error* [5] means a huge step ahead. For being more energy efficient and other reasons it controls the word size of the interval bounds in dependence of intermediate results and keeps it as small as possible. To avoid mathematical shortcomings it extends the basic set from closed real intervals to connected sets of real numbers. All this are laudable and most natural goals. The entire step, however, may be too big to get realized on computers that can be bought on the market in the near future.

So it may be reasonable to look for a smaller step which might have a more realistic chance. As such the introduction of the ubit into the floating-point bounds at the cost of shrinking the excessive exponent sizes of the IEEE 754 floating-point formats by one bit would already be a great step ahead. It would allow an extension of conventional interval arithmetic to closed, open, half-open, bounded, and unbounded sets of real numbers. By the way it would reduce the register memory for computing the dot product exactly in case of the double precision format, for instance, from excessive 4000 to only about 2000 bit. As side effect the exact dot product brings speed and associativity for addition.

Acknowledgement: The author owes thanks to Goetz Alefeld and Gerd Bohlender for useful discussions and comments on the paper. He gratefully acknowledges frequent e-mail exchange with John Gustafson on the contents of the paper.

References

1. G. Alefeld and J. Herzberger, *Einführung in die Intervallrechnung*, Informatik 12, Bibliographisches Institut, Mannheim Wien Zürich, 1974.
2. G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.
3. Ch. Baumhof, *A new VLSI vector arithmetic coprocessor for the PC*, in: Institute of Electrical and Electronics Engineers (IEEE), S. Knowles and W. H. McAllister (eds.), *Proceedings of 12th Symposium on Computer Arithmetic ARITH*, Bath, England, July 19–21, 1995, pp. 210–215, IEEE Computer Society Press, Piscataway, NJ, 1995.
4. W. De Beauclair, *Rechnen mit Maschinen*, Vieweg, Braunschweig, 1968.
5. J. L. Gustafson, *The End of Error*. To be published by CRC Press Taylor and Francis Group, A Chapman and Hall Book, 2014/15.
6. E. R. Hansen, *Topics in Interval Analysis*, Clarendon Press, Oxford, 1969.
7. E. R. Hansen, *Global Optimization Using Interval Analysis*, Marcel Dekker Inc., New York Basel Hong Kong, 1992.
8. INTEL, Intel Architecture Instruction Set Extensions Progamming Reference, 319433-017, December 2013, http://software.intel.com/en-us/file/319433-017pdf.
9. R. Klatte, U. Kulisch, M. Neaga, D. Ratz and Ch. Ullrich, *PASCAL-XSC – Sprachbeschreibung mit Beispielen*, Springer, Berlin Heidelberg New York, 1991.
   See also http://www2.math.uni-wuppertal.de/ xsc/ or http://www.xsc.de/.

10. R. Klatte, U. Kulisch, M. Neaga, D. Ratz and Ch. Ullrich, *PASCAL-XSC – Language Reference with Examples*, Springer, Berlin Heidelberg New York, 1992.
See also http://www2.math.uni-wuppertal.de/ xsc/ or http://www.xsc.de/.
Russian translation MIR, Moscow, 1995, third edition 2006.
See also http://www2.math.uni-wuppertal.de/ xsc/ or http://www.xsc.de/.
11. R. Hammer, M. Hocks, U. Kulisch and D. Ratz, *Numerical Toolbox for Verified Computing I: Basic Numerical Problems (PASCAL-XSC)*, Springer, Berlin Heidelberg New York, 1993.
Russian translation MIR, Moskau, 2005.
12. R. Klatte, U. Kulisch, C. Lawo, M. Rauch and A. Wiethoff, *C-XSC – A C++ Class Library for Extended Scientific Computing*, Springer, Berlin Heidelberg New York, 1993.
See also http://www2.math.uni-wuppertal.de/xsc/ or http://www.xsc.de/.
13. R. Hammer, M. Hocks, U. Kulisch and D. Ratz, *C++ Toolbox for Verified Computing: Basic Numerical Problems.* Springer, Berlin Heidelberg New York, 1995.
14. U. Kulisch, *An axiomatic approach to rounded computations*, TS Report No. 1020, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1969, and *Numerische Mathematik* 19 (1971), 1–17.
15. U. Kulisch, *Implementation and Formalization of Floating-Point Arithmetics,* IBM T. J. Watson-Research Center, Report Nr. RC 4608, 1 - 50, 1973. Invited talk at the Caratheodory Symposium, Sept. 1973 in Athens, published in: The Greek Mathematical Society, C. Caratheodory Symposium, 328 - 369, 1973, and in Computing 14, 323–348, 1975.
16. U. Kulisch, *Grundlagen des Numerischen Rechnens - Mathematische Begründung der Rechnerarithmetik*, Bibliographisches Institut, Mannheim Wien Zürich, 1976, ISBN 3-411-01517-9.
17. U. Kulisch, T. Teufel and B. Hoefflinger, Genauer und trotzdem schneller: Ein neuer Coprozessor für hochgenaue Matrix- und Vektoroperationen. Titelgeschichte, *Elektronik* **26** (1994), 52–56.
18. U. Kulisch, *Complete interval arithmetic and its implementation on the computer*, in: A. Cuyt, et al. (eds.), *Numerical Validation in Current Hardware Architectures*, LNCS, Vol. 5492, pp. 7–26, Springer-Verlag, Heidelberg, 2008.
19. U. Kulisch, *Computer Arithmetic and Validity – Theory, Implementation, and Applications*, de Gruyter, Berlin, 2008, ISBN 978-3-11-020318-9, second edition 2013, ISBN 978-3-11-030173-1.
20. U. Kulisch, V. Snyder, *The Exact Dot Product.* Prepared for and sent to IEEE P1788 in 2009. To be published.
21. U. Kulisch (Editor), *PASCAL-XSC, A PASCAL Extension for Scientific Computation, Information Manual and Floppy Disks*, B. G. Teubner, Stuttgart, 1987.
22. U. Kulisch, *Mathematics and Speed for Interval Arithmetic – A Complement to IEEE P1788.* Prepared for and sent to IEEE P1788 in January 2014. To be published.
23. R. E. Moore, *Interval Analysis*, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1966.
24. J. D. Pryce (Ed.), *P1788, IEEE Standard for Interval Arithmetic*,
http://grouper.ieee.org/groups/1788/email/pdfOWdtH2mOd9.pdf.
25. R. Rojas, Konrad Zuses Rechenmaschinen: sechzig Jahre Computergeschichte, in: *Spektrum der Wissenschaft*, pp. 54–62, Spektrum Verlag, Heidelberg, 1997.
26. Sun Microsystems, *Interval Arithmetic Programming Reference, Fortran 95*, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303, USA, 2000.
27. S. Oishi, K. Tanabe, T. Ogita and S. M. Rump, *Convergence of Rump's method for inverting arbitrarily ill-conditioned matrices,* Journal of Computational and Applied Mathematics 205 (2007), 533–544.
28. S. M. Rump, *Kleine Fehlerschranken bei Matrixproblemen*, Dissertation, Universität Karlsruhe, 1980.
29. S. M. Rump, *How Reliable are Results of Computers?* Jahrbuch berblicke Mathematik, 1983.
30. S. M. Rump, Solving algebraic problems with high accuracy, in: U. Kulisch and W. L. Miranker (eds.), *A New Approach to Scientific Computation*, Proceedings of Symposium held at IBM Research Center, Yorktown Heights, N.Y., 1982, pp. 51–120, Academic Press, New York, 1983.
31. IBM, *IBM System/370 RPQ. High Accuracy Arithmetic*, SA 22-7093-0, IBM Deutschland GmbH (Department 3282, Schönaicher Strasse 220, D-71032 Böblingen), 1984.
32. IBM, *IBM High-Accuracy Arithmetic Subroutine Library (ACRITH)*, IBM Deutschland GmbH (Department 3282, Schönaicher Strasse 220, D-71032 Böblingen), 1983, third edition, 1986.
    1. General Information Manual, GC 33-6163-02.
    2. Program Description and User's Guide, SC 33-6164-02.
    3. Reference Summary, GX 33-9009-02.
33. IBM, *ACRITH–XSC: IBM High Accuracy Arithmetic – Extended Scientific Computation. Version 1, Release 1*, IBM Deutschland GmbH (Department 3282, Schönaicher Strasse 220, D-71032 Böblingen), 1990.
    1. General Information, GC33-6461-01.
    2. Reference, SC33-6462-00.
    3. Sample Programs, SC33-6463-00.
    4. How To Use, SC33-6464-00.
    5. Syntax Diagrams, SC33-6466-00.

IWRMM-Preprints seit 2012

Nr. 12/01 Branimir Anic, Christopher A. Beattie, Serkan Gugercin, Athanasios C. Antoulas: Interpolatory Weighted-H2 Model Reduction

Nr. 12/02 Christian Wieners, Jiping Xin: Boundary Element Approximation for Maxwell's Eigenvalue Problem

Nr. 12/03 Thomas Schuster, Andreas Rieder, Frank Schöpfer: The Approximate Inverse in Action IV: Semi-Discrete Equations in a Banach Space Setting

Nr. 12/04 Markus Bürg: Convergence of an hp-Adaptive Finite Element Strategy for Maxwell's Equations

Nr. 12/05 David Cohen, Stig Larsson, Magdalena Sigg: A Trigonometric Method for the Linear Stochastic Wave Equation

Nr. 12/06 Tim Kreutzmann, Andreas Rieder: Geometric Reconstruction in Bioluminescence Tomography

Nr. 12/07 Tobias Jahnke, Michael Kreim: Error bound for piecewise deterministic processes modeling stochastic reaction systems

Nr. 12/08 Haojun Li, Kirankumar Hiremath, Andreas Rieder, Wolfgang Freude: Adaptive Wavelet Collocation Method for Simulation of Time Dependent Maxwell's Equations

Nr. 12/09 Andreas Arnold, Tobias Jahnke: On the approximation of high-dimensional differential equations in the hierarchical Tucker format

Nr. 12/10 Mike A. Botchev, Volker Grimm, Marlis Hochbruck: Residual, Restarting and Richardson Iteration for the Matrix Exponential

Nr. 13/01 Willy Dörfler, Stefan Findeisen: Numerical Optimization of a Waveguide Transition Using Finite Element Beam Propagation

Nr. 13/02 Fabio Margotti, Andreas Rieder, Antonio Leitao: A Kaczmarz Version of the Reginn-Landweber Iteration for Ill-Posed Problems in Banach Spaces

Nr. 13/03 Andreas Kirsch, Andreas Rieder: On the Linearization of Operators Related to the Full Waveform Inversion in Seismology

Nr. 14/01 Robert Winkler, Andreas Rieder: Resolution-Controlled Conductivity Discretization in Electrical Impedance Tomography

Nr. 14/02 Andreas Kirsch, Andreas Rieder: Seismic Tomography is Locally Ill-Posed

Nr. 14/03 Fabio Margotti, Andreas Rieder: An Inexact Newton Regularization in Banach Spaces based on the Nonstationary Iterated Tikhonov Method

Nr. 14/04 Robert Winkler, Andreas Rieder: Model-Aware Newton-Type Inversion Scheme for Electrical Impedance Tomography

Nr. 15/01 Tilo Arens, Thomas Rösch: A Collocation Method for Integral Equations with Super-Algebraic Convergence Rate

Nr. 15/02 Ulrich Kulisch: Up-to-date Interval Arithmetic from Closed Intervals to Connected Sets of Real Numbers

Eine aktuelle Liste aller IWRMM-Preprints finden Sie auf:

www.math.kit.edu/iwrmm/seite/preprints

Kontakt

Karlsruher Institut für Technologie (KIT)
Institut für Wissenschaftliches Rechnen
und Mathematische Modellbildung

Prof. Dr. Christian Wieners
Geschäftsführender Direktor

Campus Süd
Englerstraße 2
76131 Karlsruhe

E-Mail:Bettina.Haindl@kit.edu

www.math.kit.edu/iwrmm/

Herausgeber

Karlsruher Institut für Technologie (KIT)
Kaiserstraße 12 | 76131 Karlsruhe

April 2015

www.kit.edu