

# IEEE Standard for Interval Arithmetic

IEEE Computer Society

Sponsored by the  
IEEE Microprocessor Standards Committee

Currently in preview, click buy full version

# IEEE Standard for Interval Arithmetic

Sponsor

**Microprocessor Standards Committee**  
of the  
**IEEE Computer Society**

Approved 11 June 2015

**IEEE-SA Standards Board**

Currently in preview, click buy full version

**Abstract:** This standard specifies basic interval arithmetic (IA) operations selecting and following one of the commonly used mathematical interval models. This standard supports the IEEE 754<sup>TM</sup> floating-point formats of practical use in interval computations. Exception conditions are defined, and standard handling of these conditions is specified. Consistency with the interval model is tempered with practical considerations based on input from representatives of vendors, developers and maintainers of existing systems.

The standard provides a layer between the hardware and the programming language levels. It does not mandate that any operations be implemented in hardware. It does not define any realization of the basic operations as functions in a programming language.

**Keywords:** arithmetic, computing, decoration, enclosure, hull, IEEE 1788<sup>TM</sup>, interval, operation, verified

---

The Institute of Electrical and Electronics Engineers, Inc.  
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2015 by The Institute of Electrical and Electronics Engineers, Inc.  
All rights reserved. Published 30 June 2015. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-0-7381-9720-3 STD20228  
Print: ISBN 978-0-7381-9721-0 STDPD20228

*IEEE prohibits discrimination, harassment, and bullying.*

*For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.*

*No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.*

## **Important Notices and Disclaimers Concerning IEEE Standards Documents**

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

### **Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents**

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (“IEEE-SA”) Standards Board. IEEE (“the Institute”) develops its standards through a consensus development process, approved by the American National Standards Institute (“ANSI”), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: result, and workmanlike effort. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person relying on any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

### **Translations**

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

### **Official statements**

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear

that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

### **Comments on standards**

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board  
445 Hoes Lane  
Piscataway, NJ 08854 USA

### **Laws and regulations**

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

### **Copyrights**

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

### **Photocopies**

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

### **Updating of IEEE Standards documents**

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at <http://ieeexplore.ieee.org/xpl/standards.jsp> or contact IEEE at the address listed previously. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE-SA Website at <http://standards.ieee.org>.

### **Errata**

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

### **Patents**

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

## Participants

At the time this IEEE standard was completed, the Interval Standard Working Group had the following membership:

**Nathalie Revol**, *Chair*  
**R. Baker Kearfott**, *Vice Chair and Acting Chair*  
**William Edmonson**, *Secretary*  
**J. Wolff von Gudenberg**, *Web Master*  
**Guillaume Melquiond**, *Archivist*  
**George Corliss**, *Voting Tabulator*  
**John Pryce**, *Senior Technical Editor*  
**Christian Keil**, *Deputy Technical Editor*  
**Michel Hack**, **Vincent Lefèvre**, **Ian McIntosh**, **Dmitry Nadezhin**,  
**Ned Nedialkov** and **J. Wolff von Gudenberg**, *Assistant Technical Editors*

Alexandru Amaricai	Malgorzata Jankowska	Evgenija Popova
Ayman Bakr	Michel Kieffer	Tarek Raissi
Ahmed Belhani	Walter Krämer	Nacim Ramani
Gerd Bohlender	Vladik Kreinovich	Andreas Rahn
Gilles Chabert	Ulrich Kulisch	Gaby De Raes
Rudnei Dias da Cunha	Dorina Lanza	Michael Schulte
Hend Dawood	David Lester	Kyambal Shahriari
Bo Einarsson	Dominique Lohez	Stefan Siegel
Alan Eliassen	Wolfram Luther	Ilwona Skalna
Hossam A. H. Fahmy	Amin Maher	Mark Stadtherr
Richard Fateman	Svetoslav Markov	James Stine
Scott Ferson	Günter Mayer	Pipop Thienprapasith
Haitham Gad	Jean-Pierre Merlet	Warwick Tucker
Kathy Gerber	Jean-Michel Muller	Alfredo Vaccaro
Alexandre Goldsztejn	Humberto Munoz	Maarten van Emden
Frederic Goualard	Jose Antonio Munoz	Erik-Jan van Kampen
Michael Groszkiewicz	Kaori Nagata	Van Snyder
Mohamed Guerfel	Mitsuhiko Nakao	Josep Vehi
Robert Hanek	Marcel Neher	Julio Villalba-Moreno
Behnam Hashemi	Manfred Neumeier	G. William Walster
Nathan Hayes	Liep Nguyen	Yan Wang
Oliver Heimlich	Michael Nooner	Lee Winter
Timothy Hickey	Shinichi Oishi	Pierre-Alain Yvars
Werner Hofschuster	Sylvain Pion	Sergei Zhilin
Chenyi Hu	Antony Popov	Mohamed Zidan
Trevor Jackson, III		Dan Zuras

The working group wishes to record with regret the loss of two members. Antony Popov of Sofia University, Bulgaria, died suddenly in 2012 at the young age of 49. Walter Krämer, of the Bergische Universität Wuppertal, Germany, died in October 2014 at the age of 62. Despite illness, Walter had remained an active participant until June 2014.

The following members of the individual balloting committee voted on this guide. Balloters may have voted for approval, disapproval, or abstention.

Bakul Banerjee	Oliver Heimlich	JeanMichel Muller
Juan Carreon	Werner Hoelzl	Dmitry Nadezhin
Keith Chow	Chenyi Hu	Marco Nehmeier
George Corliss	Piotr Karocki	John Pryce
Hossam Fahmy	Ralph Kearfott	Nathalie Revol
Andrew Fieldsend	Vladik Kreinovich	Eugene Stoudenmire
Alexander Gelman	Vincent Lefevre	Gerald Stueve
Frederic Goualard	Vincent Lipsio	J. Wolff Von Gutenberg
Randall Groves	William Lumpkins	Forrest Wright
Michel Hack	Guillaume Melquiond	Oren Yuen
Peter Harrod	James Moore	

When the IEEE-SA Standards Board approved this standard on 11 June 2015, it had the following membership:

**John Kulick**, *Chair*  
**Jon Walter Rosdahl**, *Vice Chair*  
**Richard H. Hulett**, *Past Chair*  
**Konstantinos Karachalios**, *Secretary*

Masayuki Ariyoshi	Joseph L. Koepfinger*	Stephen J. Shellhammer
Ted Burse	David J. Law	Adrian P. Stephens
Stephen Dukes	Hung Ling	Yatin Trivedi
Jean-Philippe Faure	Andrew Myles	Phillip Winston
J. Travis Griffith	T. W. Olsen	Don Wright
Gary Hoffman	Glenn Parsons	Yu Yuan
Michael Janezic	Ronald C. Petersen	Daidi Zhong
	Annette D. Reilly	

\*Member Emeritus

Don Messina  
*IEEE-SA Content Production and Management*

Jonathan Goldberg  
*IEEE-SA Operational Program Management*

## Introduction

This introduction is not part of IEEE Std 1788-2015, IEEE Standard for Interval Arithmetic.

This introduction explains some of the alternative interpretations, and sometimes competing objectives, that influenced the design of this standard. Implementers should study it for a fuller understanding of the design choices made in this standard among these interpretations and objectives. For more information on interval computations, including history, applications and software, see<sup>a</sup> e.g. [B1, B14] and the references therein, and also the interval computations web site [B5].

### Mathematical context

Interval computation is a collaboration between human programmer and machine infrastructure which, correctly done, produces mathematically proven numerical results about continuous problems—for instance, rigorous bounds on the global minimum of a function or the solution of a differential equation. It is part of the discipline of “constructive real analysis.” In the long term, the results of such computations might become sufficiently trusted to be accepted as contributing to legal decisions. The machine infrastructure acts as a body of theorems on which the correctness of an interval algorithm relies, so it must be made as reliable as is practical. In its logical chain are many links—hardware, underlying floating-point system, etc.—over which this standard has no control. The standard aims to strengthen one specific link, by defining interval objects and operations that are theoretically well-founded and practical to implement.

This document uses the standard notation  $[a, b]$  for “the interval between numbers  $a$  and  $b$ ,” with various detailed meanings depending on the underlying theory. The “classical” interval arithmetic (IA) of R.A. Moore [B8] uses only bounded, closed, nonempty intervals in the real numbers  $\mathbb{R}$ —that is,  $[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$  where  $a, b \in \mathbb{R}$  with  $a \leq b$ . So, for instance, division by an interval containing 0 is not defined in it. It was agreed early on that this standard should strictly extend classical IA in virtue of allowing an interval to be unbounded or empty.

Beyond this, various extensions of classical IA were considered. One choice that distinguishes between theories is: Are arithmetic operations purely algebraic, or do they involve topology? An example of the latter is containment set (cset) theory [B13], which extends functions over the reals to functions over the extended reals, e.g.,  $\sin(+\infty)$  is the set of all possible limits of  $\sin x$  as  $x \rightarrow +\infty$ , which is  $[-1, 1]$ . The complications of this were deemed to outweigh the advantages, and it was agreed that operations should be purely algebraic.

Another choice is: Is an interval a set—a subset of the number line—or is it something different? The most widely used forms of IA are *set-based* and define an interval to be a set of real numbers [B10]. They have established software to find validated solutions of linear and nonlinear algebraic equations, optimization problems, differential equations, etc.

However, *Kaucher* IA and the nearly equivalent *modal* IA have significant applications. In the former, an interval is formally a pair  $(a, b)$  of real numbers, which for  $a \leq b$  is “proper” and identified with the normal interval  $\{x \in \mathbb{R} \mid a \leq x \leq b\}$ , and for  $a > b$  is “improper.” In the latter, an interval is a pair  $(X, Q)$ , where  $X$  is a normal interval and  $Q$  is a quantifier, either  $\exists$  or  $\forall$ . At the time of writing, it finds commercial use in the graphics rendering industry. Both forms are referred to as Kaucher IA henceforth.

In view of their significance, it was decided to support both set-based and Kaucher IA. Because of their different mathematical bases, this led to the concept of *flavors* (see Clause 7). A flavor is a version of IA that extends classical IA in a precisely defined sense, such that when only classical intervals and restricted operations are used (avoiding, e.g., division by an interval containing zero), all flavors produce the same result at the mathematical level and also—up to roundoff—in finite precision.

Currently, the standard includes only the set-based flavor. Among other possible flavors are Kaucher/modal intervals; containment-sets; and the interval system of Siegfried Rump [B15], which handles the relation between floating-point numbers and intervals, including overflow, in an elegant way, as well as being able to support open and half-open intervals. All of these extend classical IA in the defined sense.

<sup>a</sup>The numbers in brackets correspond to those of the bibliography in Annex A.

Clause 1 through Clause 9 contain a common set of definitions and requirements that apply to all flavors. Clause 10 through Clause 14 contain the set-based flavor, in which:

- Intervals are subsets of the set  $\mathbb{R}$  of real numbers. At the mathematical level (Level 1 in the structure defined in Clause 5) they are precisely all topologically closed and connected subsets of  $\mathbb{R}$ . The finite-precision level (Level 2) uses the notion of an interval type, which is a finite set of Level 1 intervals.
- The interval version of an elementary function such as  $\sin x$  is essentially the natural extension to sets of the corresponding pointwise function on real numbers.

Fuzzy sets, like intervals, are a way to handle uncertain knowledge, and the two topics are related. However, to consider this relation was beyond the scope of this project.

### Specification levels

The floating-point standard IEEE Std 754<sup>TM</sup>-2008 describes itself as layered into four Specification Levels. To manage complexity, the present standard uses a corresponding structure. It deals mainly with Level 1, of mathematical *interval theory*, and Level 2, the finite set of *interval datums* in terms of which finite-precision interval computation is defined. It has some concern with Level 3, of *representations* of intervals as data structures; and with Level 4, of interchange *encoding* in *bit strings*.

There is another important player: the programming language. It was a recognized omission of the first (1985) version of IEEE Std 754-2008 that it specified individual operations but not how they should be used in expressions. Optimizing compilers have, since well before that standard, used clever transformations so that it is impossible to know the precisions used and the roundings performed while evaluating an expression, or whether the compiler has even “optimized away”  $(1.0 + x) - 1.0$  to become simply  $x$ . The 2008 revision specifies this by placing requirements on how operations should be used in expressions, though as of this writing, few programming languages have adopted that.

The lack of any restrictions is also a problem for intervals. Thus the standard makes requirements and recommendations on language implementations, thereby defining the notion of a standard-conforming implementation of intervals within a language.

The language does not constitute a fifth level in some linear sequence; from the user’s viewpoint, most current languages sit above datum level 2, alongside theory level 1, as a practical means to implement interval algorithms by manipulating Level 2 entities (though most languages have influence on Levels 3 and 4 also). This standard extends them to provide an instantiation of Level 2 entities.

### The Fundamental Theorem

Moore’s [B8] Fundamental Theorem of Interval Arithmetic (FTIA) is central to interval computation. Roughly, it says as follows. Let  $f$  be an *explicit arithmetic expression*—that is, it is built from finitely many elementary functions (arithmetic operations) such as  $+$ ,  $-$ ,  $\times$ ,  $\div$ ,  $\sin$ ,  $\exp$ ,  $\dots$ , with no non-arithmetic operations such as intersection, so that it defines a real function  $f(x_1, \dots, x_n)$ . Then evaluating  $f$  “in interval mode” over any interval input box  $(\mathbf{x}_1, \dots, \mathbf{x}_n)$  is guaranteed to enclose (i.e., give a set that contains) the range of  $f$  over those inputs. Typically there are sub-cases, where extra conditions lead to stronger conclusions such as  $f$  being continuous on the input box.

A version of the FTIA holds in all variants of interval theory, but with varying hypotheses and conclusions. In the context of this standard, an expression should be evaluated entirely in one flavor, and inferences made strictly from that flavor’s FTIA; otherwise, a user might believe an FTIA holds in a case where it does not, with possibly serious effects in applications. As stated, the FTIA is about the mathematical level. Moore’s achievements were to see that “outward rounding” makes the FTIA hold also in finite precision and to follow through the consequences. An advantage of the level structure used by the standard is that the mapping between Levels 1 and 2 defines a framework where it is easily proved that

Each flavor’s finite-precision FTIA holds in any conforming implementation.

Generally, during program execution it can only be decided *after* evaluating an expression whether the conditions for any sub-case of the FTIA hold. The purpose of each flavor’s *decoration system* is to make such decisions computable, see 6.4 and Clause 7.

For the set-based flavor, see [B12] for background on its decoration system, 6.4 for a statement of its FTIA, and Annex B for a statement and proof of its FTDIA.

## Operations

There are several interpretations of *evaluation outside an operation's domain* and *operations as relations rather than functions*. This includes classical alternative meanings of division by an interval containing zero, or square root of an interval containing negative values. To illustrate the different interpretations, consider  $\mathbf{y} = \sqrt{\mathbf{x}}$  where  $\mathbf{x} = [-1, 4]$ .

- a) In *optimization*, when computing lower bounds on the objective function, it is generally appropriate to return the result  $\mathbf{y} = [0, 2]$ , and ignore the fact that  $\sqrt{\cdot}$  has been applied to negative elements of  $\mathbf{x}$ .
- b) In applications (such as solving differential equations) where one must check whether the hypotheses of a *fixed point theorem* are satisfied:
  - 1) one might need to be sure that the function is defined and continuous on the input and, hence, report an illegal argument when, as in the above case, this fails; or
  - 2) one might need the result  $\mathbf{y} = [0, 2]$ , but must flag the fact that  $\sqrt{\cdot}$  has been evaluated at points where it is undefined or not continuous.
- c) In *constraint propagation*, the equation is often to be interpreted as: find an interval enclosing all  $y$  such that  $y^2 = x$  for some  $x \in [-1, 4]$ . In this case the answer is  $[-2, 2]$ .

The standard provides means to meet these diverse needs, while aiming to preserve clarity and efficiency. A language might achieve this by binding one of the above three interpretations—usually some variant of b)—to its built-in operations, and providing the others as library procedures.

In the context of flavors, a key idea is that of *common operation instances*: those elementary interval calculations that at the mathematical level are required to give the same result in all flavors. For example  $[1, 2]/[3, 4] = [1/4, 2/3]$  is common, while division by an interval containing zero is not common.

## Decorations

Many interval algorithms are only valid if certain mathematical conditions are satisfied: for instance, one might need to know that a function  $f$ , defined by an expression, is everywhere continuous on a box in  $\mathbb{R}^n$  defined by  $n$  input intervals  $\mathbf{x}_1, \dots, \mathbf{x}_n$ . The IEEE 754 use of global flags to record events such as division by zero was considered inadequate in an era of massively parallel processing. In this standard, such events are recorded locally by *decorations*.

A *decorated interval* is an ordinary interval tagged with a few bits that encode the decoration. A small number of decorations is provided, designed for efficient propagation of such property information. For instance, if evaluation outputs an interval  $\mathbf{y}$  with the `dac` decoration, then  $f$  is *defined and continuous* on its input box, with range contained in  $\mathbf{y}$ . This allows a rigorous check, for instance, that the conditions for applying a fixed-point theorem hold. Depending on need, a programmer may use bare (undecorated) intervals, or decorated intervals, or (if provided) the optional *compressed* decorated interval arithmetic that offers less decoration capability in exchange for faster execution.

## The Basic standard

To make the standard more accessible and speed up production of implementations, a subset of the set-based standard called the Basic Standard for Interval Arithmetic (BSIA) has been written. It includes just one finite-precision interval type—intervals whose endpoints are IEEE 754 `binary64` numbers—and those operations that in the editors' view are most commonly used. A minimal implementation of the BSIA is not a conforming implementation of the full standard since some required operations of the latter are omitted or provided in a restricted form. However a program that runs using such an implementation should run, and give identical output within roundoff, using an implementation of the full standard. At the time of writing a project is underway to issue the BSIA as a separate IEEE standard.

## Contents

Part 1. General Requirements	2
1. Overview	2
1.1. Scope	2
1.2. Purpose	2
1.3. Inclusions	2
1.4. Exclusions	2
1.5. Word usage	3
1.6. The meaning of conformance	3
1.7. Programming environment considerations	3
1.8. Language considerations	4
2. Normative references	4
3. Notation, abbreviations, and special terms	5
3.1. Frequently used notation and abbreviations	5
3.2. Special terms	5
4. Conformance	10
4.1. Conformance overview	10
4.2. Set-based interval arithmetic	11
4.2.1. IEEE 754 conformance	11
4.2.2. Compressed decorated interval arithmetic	11
4.3. Conformance claim	11
4.4. Implementation conformance questionnaire	12
5. Structure of the standard in levels	13
6. Functions and expressions	14
6.1. Function definitions	14
6.2. Expression definitions	15
6.3. Function libraries	17
6.4. The FTIA	18
6.5. Related issues	19
7. Flavors	19
7.1. Flavors overview	19
7.2. Flavor basic properties	20
7.3. Common evaluations	21
7.4. Primary versions and Level 1 interval versions	21
7.4.1. Arithmetic operations	21
7.4.2. Nonarithmetic operations required in all flavors	22
7.4.3. Flavor-defined nonarithmetic operations	22
7.5. The relation of Level 1 to Level 2	22
7.5.1. Types	23
7.5.2. Hull	23
7.5.3. Level 2 operations	23
7.5.4. Measures of accuracy	24
8. Decoration system	25
8.1. Decorations overview	25
8.2. Decoration definition and propagation	26
8.3. Recognizing common evaluation	26
9. Operations and related items required in all flavors	27
9.1. Arithmetic operations	27
9.2. Cancellative addition and subtraction	29
9.3. Set operations	29
9.4. Numeric functions of intervals	29
9.5. Boolean functions of intervals	29
9.6. Operations on/with decorations	29
9.7. All-flavor interval and number literals	30

9.7.1.	Overview	30
9.7.2.	All-flavor number literals	31
9.7.3.	Unit in last place	31
9.7.4.	All-flavor bare interval literals	31
9.7.5.	All-flavor decorated interval literals	31
9.7.6.	Grammar for all-flavor literals	32
9.8.	Constructors	32
Part 2. Set-Based Intervals		34
10.	Level 1 description	34
10.1.	Non-interval Level 1 entities	34
10.2.	Intervals	34
10.3.	Hull	35
10.4.	Functions and expressions	35
10.5.	Required operations	36
10.5.1.	Interval literals	36
10.5.2.	Interval constants	36
10.5.3.	Forward-mode elementary functions	36
10.5.4.	Reverse-mode elementary functions	36
10.5.5.	Two-output division	37
10.5.6.	Cancellative addition and subtraction	38
10.5.7.	Set operations	38
10.5.8.	Constructors	38
10.5.9.	Numeric functions of intervals	39
10.5.10.	Boolean functions of intervals	39
10.6.	Recommended operations	40
10.6.1.	Forward-mode elementary functions	40
10.6.2.	Slope functions	41
10.6.3.	Boolean functions of intervals	41
10.6.4.	Extended interval comparisons	41
11.	The decoration system at Level 1	44
11.1.	Decorations and decorated intervals overview	44
11.2.	Definitions and basic properties	44
11.3.	The ill-formed interval	45
11.4.	Permitted combinations	45
11.5.	Operations on/with decorations	45
11.5.1.	Initializing	45
11.5.2.	Disassembling and assembling	46
11.5.3.	Comparisons	46
11.6.	Decorations and arithmetic operations	46
11.7.	Decoration of non-arithmetic operations	47
11.7.1.	Interval-valued operations	47
11.7.2.	Non-interval-valued operations	47
11.8.	User-supplied functions	47
11.9.	Notes on the <code>com</code> decoration	48
11.10.	Compressed arithmetic with a threshold (optional)	49
11.10.1.	Motivation	49
11.10.2.	Compressed interval types	49
11.10.3.	Operations	50
12.	Level 2 description	51
12.1.	Level 2 introduction	51
12.1.1.	Types and formats	51
12.1.2.	Operations	51
12.1.3.	Exception behavior	52

12.2.	Naming conventions for operations	52
12.3.	Tagging, and the meaning of equality at Level 2	52
12.4.	Number formats	53
12.5.	Bare and decorated interval types	54
12.5.1.	Definition	54
12.5.2.	Inf-sup and mid-rad types	55
12.6.	754-conformance	55
12.6.1.	Definition	55
12.6.2.	754-conforming mixed-type operations	55
12.7.	Multi-precision interval types	55
12.8.	Explicit and implicit types, and Level 2 hull operation	56
12.8.1.	Hull in one dimension	56
12.8.2.	Hull in several dimensions	56
12.9.	Level 2 interval extensions	56
12.10.	Accuracy of operations	57
12.10.1.	Measures of accuracy	57
12.10.2.	Accuracy requirements	58
12.10.3.	Documentation requirements	58
12.11.	Interval and number literals	58
12.11.1.	Overview	58
12.11.2.	Number literals	58
12.11.3.	Bare intervals	59
12.11.4.	Decorated intervals	59
12.11.5.	Grammar for portable literals	59
12.12.	Required operations on bare and decorated intervals	60
12.12.1.	Interval constants	60
12.12.2.	Forward-mode elementary functions	61
12.12.3.	Two-output division	61
12.12.4.	Reverse-mode elementary functions	61
12.12.5.	Cancellative addition and subtraction	61
12.12.6.	Set operations	62
12.12.7.	Constructors	62
12.12.8.	Numeric functions of intervals	63
12.12.9.	Boolean functions of intervals	64
12.12.10.	Interval type conversion	65
12.12.11.	Operations on/with decorations	65
12.12.12.	Reduction operations	65
12.13.	Recommended operations	66
12.13.1.	Forward-mode elementary functions	66
12.13.2.	Slope functions	66
12.13.3.	Boolean functions of intervals	66
12.13.4.	Extended interval comparisons	66
12.13.5.	Exact reduction operations	66
13.	Input and output (I/O) of intervals	67
13.1.	Overview	67
13.2.	Input	67
13.3.	Output	67
13.4.	Exact text representation	68
13.4.1.	Conversion of IEEE 754 numbers to strings	69
13.4.2.	Exact representations of comparable types	70
14.	Levels 3 and 4 description	70
14.1.	Overview	70
14.2.	Representation	70
14.3.	Operations and representation	71

14.4. Interchange representations and encodings	71
Annex A. Bibliography (informative)	74
Annex B. The fundamental theorem of decorated interval arithmetic for the set-based flavor (informative)	76
B1. Preliminaries	76
B2. The theorem	78

# IEEE Standard for Interval Arithmetic

*IMPORTANT NOTICE: IEEE Standards documents are not intended to ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.*

*This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading Important Notice or Important Notices and Disclaimers Concerning IEEE Documents. They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.*

PART 1

## General Requirements

### 1. Overview

#### 1.1 Scope

This standard specifies basic interval arithmetic (IA) operations selecting and following one of the commonly used mathematical interval models. This standard supports the IEEE 754<sup>TM</sup> floating-point formats of practical use in interval computations. Exception conditions are defined, and standard handling of these conditions is specified. Consistency with the interval model is tempered with practical considerations based on input from representatives of vendors, developers and maintainers of existing systems.

The standard provides a layer between the hardware and the programming language levels. It does not mandate that any operations be implemented in hardware. It does not define any realization of the basic operations as functions in a programming language.

#### 1.2 Purpose

The aim of the standard is to improve the availability of reliable computing in modern hardware and software environments by defining the basic building blocks needed for performing interval arithmetic. There are presently many systems for interval arithmetic in use; lack of a standard inhibits development, portability, and ability to verify correctness of codes.

#### 1.3 Inclusions

This standard specifies

- Types for interval data based on underlying numeric formats, with a special class of type derived from IEEE 754 floating-point formats.
- Constructors for intervals from numeric and character sequence data.
- Addition, subtraction, multiplication, division, fused multiply add, square root; other interval-valued operations for intervals.
- Midpoint, radius and other numeric functions of intervals.
- Interval comparison relations and other boolean functions of intervals.
- Elementary interval functions of intervals.
- Conversions between different interval types.
- Conversions between interval types and external representations as text strings.
- Interval-related exceptional conditions and their handling.

#### 1.4 Exclusions

This standard does not specify

- Which numeric formats supported by the underlying system shall have an associated interval type.
- How (for implementations supporting IEEE 754 arithmetic) operations act on the IEEE 754 status flags.
- How an implementation represents intervals at the level of programming language data types or bit patterns.