1

Fundamentals of Interval Computing

Ralph Baker Kearfott[1] and Chenyi Hu[2]

[1] Department of Mathematics, University of Louisiana at Lafayette, Box 4-1010, Lafayette, LA 70504-1010, USA. rbk@louisiana.edu
[2] Department of Computer Science, University of Central Arkansas, 201 Donaghey Avenue, Conway, AR 72035-0001, USA. chu@uca.edu

This volume deals, generally, with innovative techniques for automated knowledge representation and manipulation when such knowledge is subject to significant uncertainty, as well as with automated decision processes associated with such uncertain knowledge. Going beyond traditional probability theory and traditional statistical arguments, the techniques herein make use of interval techniques, of fuzzy knowledge representation and fuzzy logic, and of the combination of interval techniques with fuzzy logic and with probability theory.

In this chapter, we introduce interval computing, giving reasons for its development and references to historical work. We also preview the remainder of the book, contrasting the underlying philosophy and range of application with prevalent views among experts in interval computation.

1.1 Intervals and Their Representation

By the term *interval* we mean the set of all real numbers between specified lower and upper bounds, a and b (i. e. $\{x|a \leq x \leq b; a, b, x \in \mathbb{R}\}$). Intervals are denoted in various ways within the interval computations community. For examples, see [39, 18, 3, 40, 23]. In this book, we use lowercase boldface letters to denote intervals. For example, $\boldsymbol{x}$ is an interval. The lower and upper bounds of an interval $\boldsymbol{x}$ are specified as $\underline{x}$ and $\overline{x}$, respectively. Hence, $\boldsymbol{x} = [\underline{x}, \overline{x}]$. We call this the *endpoint representation* of an interval.[3] An empty interval - an interval that contains no real numbers - is simply the empty set $\emptyset$. Various computer representations are possible in implementations for $\emptyset$.

The *midpoint* of a nonempty interval is the algebraic average of its lower and upper bounds. The width of a nonempty interval is the difference between

[3] Alternate representations include midpoint-radius representation and tolerance representation. We discuss midpoint-radius (or midpoint-width) representation below.

C. Hu et al. (eds.), *Knowledge Processing with Interval and Soft Computing*,
DOI: 10.1007/978-1-84800-326-2_1,

its upper and lower bounds. We use the uppercase letters M and W in subscripts to specify the midpoint and width of a nonempty interval, respectively. The midpoint and width of the interval $\boldsymbol{x} = [\underline{x}, \overline{x}]$ are $x_M = (\underline{x} + \overline{x})/2$ and $x_W = \overline{x} - \underline{x}$, respectively. We define two unary interval operators, $m(\ )$ and $w(\ )$, that return the midpoint and width of an interval, respectively; that is, $m(\boldsymbol{x}) = x_M$ and $w(\boldsymbol{x}) = x_W$.

The lower bound of an interval is the same as the difference between its midpoint and one-half of its width. Similarly, the upper bound of an interval is the same as the sum of its midpoint and one-half of its width. We can represent an interval by its midpoint and width. This is called the *midpoint-width representation* of an interval. For example, the interval $\boldsymbol{x} = [\underline{x}, \overline{x}]$, in its midpoint-width representation, is $\boldsymbol{x} = [x_M - x_W/2, x_M + x_W/2]$. If the lower and upper bounds of an interval are the same, we say that is a *trivial* or *degenerate* or a *thin* interval. Obviously, the width of a degenerate interval is zero.

The above discussion can be extended to interval vectors and interval matrices. The entries of an interval vector or matrix are intervals. To unambiguously distinguish point vectors and interval vectors, and point matrices and interval matrices, in this book we use boldface letters to specify intervals. We use boldface lowercase letters with arrows on top to denote interval vectors. For example, $\vec{\boldsymbol{x}}$ is an interval vector. Its lower and upper bounds are $\underline{\vec{x}}$ and $\overline{\vec{x}}$, respectively. We omit the top arrow when it would not cause confusion. For example, $\vec{\boldsymbol{x}}$ (or $\boldsymbol{x}$) with

$$\vec{\boldsymbol{x}} = \begin{pmatrix} [1,2] \\ [3,4] \end{pmatrix}$$

can denote an interval vector.

Uppercase letters are used to denote matrices. Boldface uppercase letters denote interval matrices. For example, $\boldsymbol{A}$ is an interval matrix, and its lower and upper bounds are $\underline{A}$ and $\overline{A}$, respectively. The midpoint and width of a nonempty interval vector (or matrix) are real vectors (or matrices). Therefore, $\boldsymbol{A}$ with

$$\boldsymbol{A} = \begin{pmatrix} [1,2] & [3,4] \\ [5,6] & [7,8] \end{pmatrix}$$

can denote an interval matrix, and its lower and upper bounds are point matrices

$$\underline{A} = \begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix},$$

$$\overline{A} = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}.$$

The notation we adopt in this volume has been recommended (but not required) for the journal *Reliable Computing* and has been proposed as a voluntary standard (although not universally adopted).

1.2 Origins and Reason for Development

Interval computing, specifically interval arithmetic, began primarily as a means of automating error analysis during the process of computing floating point approximations to solutions of scientific problems. The basic idea is that the true value x of some quantity appearing in some computation is not known exactly, but it is bounded in some interval $\boldsymbol{x}$, $x \in \boldsymbol{x}$. In exact arithmetic, each operation[4] $\odot$, $\odot \in \{+, -, \times, \div, \text{etc.}\}$ is formally defined on these intervals in such a way that the result $\boldsymbol{x} \odot \boldsymbol{y}$ is equal to the set $x \odot y$ as x ranges over all values in $\boldsymbol{x}$ and y ranges over all values in $\boldsymbol{y}$; that is,

$$\boldsymbol{x} \odot \boldsymbol{y} = \{x \odot y \mid x \in \boldsymbol{x} \text{ and } y \in \boldsymbol{y}\}. \tag{1.1}$$

For example, let $\boldsymbol{x} = [1, 2]$ and $\boldsymbol{y} = [-3, -1]$. We then have $\boldsymbol{x} + \boldsymbol{y} = [-2, 1]$, $\boldsymbol{x} - \boldsymbol{y} = [2, 5]$, $\boldsymbol{x} * \boldsymbol{y} = [-6, -1]$, and $\boldsymbol{x} \div \boldsymbol{y} = [-2, -1/3]$.

Using the basic interval arithmetic operations, we can perform linear algebra operations such as interval vector dot products, interval matrix-vector multiplications, interval matrix-matrix operations, etc. We will discuss and apply these in later chapters.

In addition to binary arithmetic operations, logic and set operations can also be performed on intervals. For example, $[0, 2] \subset [1, 2]$ returns false, $[0, 2] \cup [1, 3] = [0, 3]$, and $[0, 2] \cap [1, 3] = [1, 2]$. With these additional features of interval computing, we develop new algorithms to solve problems that are hard to solve in classical point arithmetic. By extending interval operations further with fuzzy logic and probability theory, this book presents additional algorithms for knowledge processing, especially in handling uncertainty.

Most modern literature on interval arithmetic is traceable to [36], although there are earlier independent works, such as [53], that are often overlooked but contain many, if not most, of the developments in [36]; some of these early works are available at `http://www.cs.utep.edu/interval-cmp/early.html`.

There is extensive literature on interval computations. Classic introductions are the books [37], [38], and [2] and the latter's translation to English [3]. A numerical analysis textbook that introduces interval arithmetic in appropriate places is [41]. An introduction to interval analysis with numerous computational examples within the MATLAB environment, as well as discussion of the most successful applications, appears in [39]. An early, carefully written reference on interval arithmetic and its implementation on actual machines is [29].

[4] In actual implementations, an operation $\odot$ can include not only the four elementary arithmetic operations but also all of the usual preprogrammed functions, such as sin, exp, etc., that are available in compiler libraries for scientific programming languages.

1.3 Computer Implementation and Software

In practice, (1.1) is not achievable exactly in floating point arithmetic on computers. However, on modern computers (and, in particular, on those for which the IEEE 754 binary floating point standard is implemented), *directed rounding* can be used, so that instead of the exact value $\boldsymbol{x} \odot \boldsymbol{y}$ defined by (1.1), an interval $\boldsymbol{z}$ is computed such that

$$\{x \odot y \mid x \in \boldsymbol{x} \text{ and } y \in \boldsymbol{y}\} \subseteq \boldsymbol{z}$$

and such that the complement of $\{x \odot y \mid x \in \boldsymbol{x} \text{ and } y \in \boldsymbol{y}\}$ in $\boldsymbol{z}$ is very small (on the order of the roundoff unit). In this way, if intervals $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ are substituted for the variables $\{x_1, \ldots, \boldsymbol{x}_n\}$ in an expression $E(x_1, \ldots, x_n)$, and E is evaluated using interval arithmetic, then the interval result $\boldsymbol{E}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ contains the range

$$\{E(x_1, \ldots, x_n) \mid x_i \in \boldsymbol{x}_i, \quad 1 \le i \le n\}, \tag{1.2}$$

that is, completion of a computation using interval arithmetic with directed rounding (also called *outwardly rounded interval arithmetic*), yielding a result $\boldsymbol{z}$, provides a mathematically rigorous proof[5] that the actual result is contained in $\boldsymbol{z}$.

There are numerous software tools and applications for interval computing written in mainstream languages, such as C, C++, Fortran, Java, Lisp, as well as in computational algebra systems, such as Maple, MATLAB, and Mathematica. For example, INTLAB [49] is an interval computing toolbox in MATLAB. An interval software development environment [22] is available in FORTRAN-90. Software in C++ supporting interval arithmetic includes the Boost C++ source libraries (`http://www.boost.org/libs/numeric/interval/doc/interval.htm`), filib++ [31], PROFIL/BIAS [26, 27], Sun's Studio C++, and Fortran [52], an object-oriented interval matrix computing environment [43], and others. Interval arithmetic is slated to be embodied in a technical report for the next C++ standards document [11]. We present an object-oriented interval toolbox in C++, named "IntBox," in Chapter 10 of this book.

Application software packages using interval computing include COSY-Infinity [9, 14] (and also `http://bt.pa.msu.edu/index_cosy.htm`) that is based on Taylor models and interval methods for validated solution of ordinary differential equations, quadrature, and range bounding, CGAL (`http://www.cgal.org/`) that makes geometric computations robust and efficient, GlobSol [23, 25] that finds reliable solutions for global nonlinear optimization problems with interval analysis, iCOs for interval constraint satisfaction and global optimization `http://ylebbah.googlepages.com/icos`, and others, both old and new. For additional applications, see [39].

[5] Assuming the computer is programmed correctly and is not malfunctioning.

1.4 Present Uses of Interval Arithmetic

People soon discovered that interval computations can be applied in more situations than merely error analysis in existing floating point algorithms. The fact that the result of a computation carried out with outwardly rounded interval arithmetic rigorously bounds the range of the computation is useful in various contexts. On a mathematical level, such uses include the following:

- rigorously bounding the ranges of functions over wide domains;
- including bounded uncertainties in the inputs in models;
- rigorously proving existence and uniqueness of solutions to systems of equations using computational versions of classical fixed point theorems.

As evidenced by the ubiquitous appearance of Lipschitz constants and moduli of continuity in classical hard analysis, bounding ranges of a quantity over sizable domains is an important computation in various contexts. Furthermore, using preexisting interval software and modern programming languages, bounding such ranges reduces to programming the computation of the quantity. Within this framework, computations that are equivalent to or sharper than using moduli of continuity or Lipschitz constants can be carried out automatically.

Classical fixed point theorems, such as the Brouwer fixed point theorem, state that if the image of a region $\boldsymbol{x}$ under an operator G is contained in $\boldsymbol{x}$, then there is an $x \in \boldsymbol{x}$ such that $G(x) = x$. With interval arithmetic, the range of G can be bounded, and the hypotheses of the theorem are satisfied if the interval evaluation of G over $\boldsymbol{x}$ is contained in $\boldsymbol{x}$. Furthermore, although such classical theorems can be applied directly, *interval Newton methods*, with a theoretical basis in classical fixed point theory, have been extensively developed to both compute narrow bounds on the solutions to systems of equations and to prove existence and uniqueness of those solutions within those bounds. For information on such methods, see, in addition to the general references on interval computations we have cited earlier, [40], [23], and [18]. The book [40] contains a thorough treatment of interval Newton methods, and [23] and [18] treat such methods in the context of algorithms for global optimization.

1.5 Pitfalls

Speed and sharpness were issues early in the study of interval methods, and there has been considerable discussion of these issues in the literature. However, present software implementations, using optimizing (in-lining) compilers and operator overloading, achieve, averaged over many operations, speed within a factor of 5 of hardware floating point operations, and some implementations within the compiler itself achieve interval evaluation that is, on average, less than a factor of 2 slower than floating point evaluations for some

of the standard functions. Such speed is usually not the determining factor in whether to use interval computations.

Similarly, there has been significant discussion and development related to making the result intervals as tight as possible. In particular, many existing software systems today compute an interval $\boldsymbol{z}$ that is the narrowest machine-representable interval that contains the exact range as defined in (1.2) when $\odot \in \{+, -, \times, \div\}$; likewise, libraries for interval evaluation of the standard functions are also of high quality in this sense. Thus, tightness of the basic operations and interval evaluations of standard functions are not primary issues in deciding whether to apply interval computations.

The following pitfalls are more significant when designing applications with interval computations.

1.5.1 Interval Dependency

One major pitfall in interval computations is commonly termed *interval dependency*. Interval dependency is most easily illustrated by examining the subtraction operation: If $\boldsymbol{x} = [\underline{x}, \overline{x}]$ and $\boldsymbol{y} = [\underline{y}, \overline{y}]$, then the exact range $\boldsymbol{x} - \boldsymbol{y}$ happens to be

$$\boldsymbol{x} - \boldsymbol{y} = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]. \tag{1.3}$$

However, suppose, say, $\boldsymbol{x} = [-1, 1]$ is interpreted to represent some specific real number $x \in [-1, 1]$ but unknown other than that it lies in $[-1, 1]$. Then $x - x = 0$, but if the computer encounters the expression $x - x$ (without simplification) and substitutes $[-1, 1]$ for both instances of x, it obtains

$$[-1, 1] - [-1, 1] = [-2, 2]. \tag{1.4}$$

Observe that $[-2, 2]$ does contain the range of $f(x) = x - x$ as x ranges over $[-1, 1]$, but it does not sharply bound the range. The "dependency" is from the fact that the computation (1.4) assumes implicitly that the quantity in the first instance of $[-1, 1]$ varies independently from the quantity in the second instance of $[-1, 1]$ whereas, in fact, the two values are correlated or "dependent." Observe, however, that $[-2, 2]$ *is* the exact range of $g(x, y) = x - y$ as x ranges over $[-1, 1]$ and y ranges over $[-1, 1]$.

Due partly to the interval dependency phenomenon, traditional floating point algorithms can seldom be converted to successful interval algorithms (that rigorously bound roundoff error or provide useful bounds on ranges) by simply replacing floating point numbers by intervals; instead, intervals need to be introduced in appropriate ways, and new algorithms appropriate for interval computation are developed.

One property of interval ranges that ameliorates interval dependency is that the amount of overestimation (i. e., the sum of the differences between the endpoints of the actual range and the interval arithmetic evaluation) decreases proportionally to the widths of the input intervals. In fact, for continuously differentiable quantities, the bounds on the range can be computed in such

a way that the decrease is proportional to the squares of the widths of the input intervals; in special circumstances, bounds can even be computed with overestimation proportional to even higher powers of the widths of the input intervals. Thus, interval computations give locally tight bounds on the ranges of computed quantities.

1.5.2 Computational Complexity

Computational complexity and similar theoretical issues arise in interval algorithms. For instance, if $\boldsymbol{A}$ is a matrix with interval entries, $\boldsymbol{b}$ is a vector with interval entries, and the individual entries in $\boldsymbol{A}$ and $\boldsymbol{b}$ are assumed to vary independently, then it is known that, in general, finding an interval vector sharply bounding the solution set

$$\{x \mid Ax = b \text{ for some } A \in \boldsymbol{A} \text{ and } b \in \boldsymbol{b}\} \tag{1.5}$$

is an NP-hard problem. (Such results are collected in [28].) However, there are many algorithms that successfully compute usefully narrow bounds to the solution sets to even large linear systems; as one of many examples of success, see [42]. Furthermore, there is general software that is at least moderately successful at handling large, sparse systems of linear equations within a friendly user environment: the MATLAB toolbox INTLAB (see [48, 19]).

1.5.3 Problems with Coordinate Systems

Actual ranges of quantities as input values range over intervals are seldom sets of the form

$$\{(x_1, \ldots, x_n) \mid x_i \in [\underline{x}_i, \overline{x}_i] \text{ for } 1 \leq i \leq n\}\,. \tag{1.6}$$

(Such sets are commonly termed *boxes* or *interval vectors*.) For instance, the image of the vector-valued function $F(x, y) = (x+y, x-y)^T$ is the skewed parallelogram depicted in Figure 1.1, whereas the smallest interval vector containing this range is the substantially larger vertically oriented box in Figure 1.1. This problem can be severe in interval methods for bounding the solution sets to initial value problems for systems of ordinary differential equations, where it is known as the *wrapping effect* (see, for instance, [38]). It even is a significant problem in branch-and-bound methods for global optimization, where excessive subdivision of a domain may occur if the proper coordinate system is not used.

The wrapping effect has been, to a large extent, successfully ameliorated for initial value problems, such as in the COSY system [10]. In general, a fix is to somehow use an appropriate change of coordinates, as is proposed in [24].

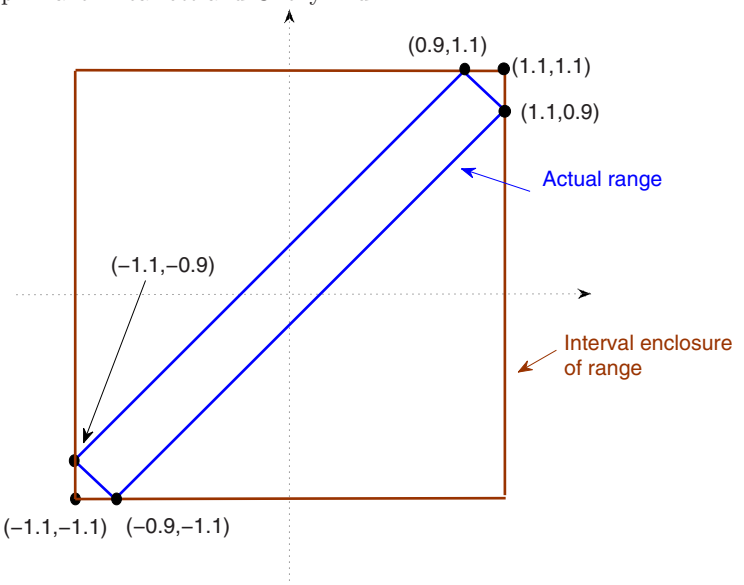

Fig. 1.1. Illustration of overestimation due to the choice of coordinates.

1.5.4 Successes Nonetheless

The concept of mathematical rigor in computer arithmetic is appealing, something that has caused enthusiasts to make excessive claims about its applicability and usefulness in the past. These claims, in turn, have stimulated naive experimentation, leading to disillusionment with the technology within the scientific computing community as a whole. However, with software crafted to utilize the strengths and avoid the weaknesses of interval computations, important problems are increasingly solved with interval techniques but not other methods. For instance, certain chemical kinetics equilibrium problems [50, 51, 15, 16] have been solved correctly with interval techniques, whereas earlier floating techniques had given approximate solutions that led to erroneous conclusions about the underlying physical problem. There have been similar successes in solving systems of equations arising in robotics [20, 30]. Inroads have even been made in the computation of parameters for boundary value problems [33, 32, 34, 35]. For additional details on these and other applications, see [39].

1.6 Context of This Work

In this volume, we present novel techniques for new and improved algorithms in knowledge engineering. In contrast to much work utilizing interval arithmetic, the focus here is not on rigor and mathematical theorem proving but

on efficient ways of encompassing uncertain inputs to compute bounds on outputs. Along these lines, intervals are combined heuristically with methods from probability theory and other methods that, in their raw form, cannot be made rigorous by naive application of interval arithmetic. A guiding principle in the applications treated in this volume is that the real-world data often provide interval inputs to the problem and that the data should be so represented.

In some of the problems tackled in this volume, traditional statistical models have previously been used, but interval techniques are used creatively for improved modeling and predictive power. An example of this, appearing in Section 4.5 of Chapter 4, is the use of interval data to reduce the raw noisy data in stock market indices. This data reduction is combined with an innovative view[6] of interval values of the singular value decomposition and principal component analysis to both simplify and enhance the predictive power of the model. In this work and Chapter 5, intervals and interval arithmetic are used to describe variability and uncertainty in the model inputs and to divine relationships between the model inputs and model outputs; the model outputs are intervals that approximate the range of behavior over the input intervals but are not claimed to rigorously enclose that range.[7] This is reasonable in view of the fact that real-world models often have uncertainties that cannot be quantified, so proof that an exact result is rigorously enclosed may not make sense.

In contrast, the traditional literature on interval enclosures for eigenvalue- and singular-value-decompositions assumes that the problem is specified exactly (with a point matrix), and the goal is to compute mathematically rigorous bounds on the exact solutions to this point problem. Examples of this approach are in [21, 4, 5, 6, 7, 1], and [44, 45, 8, 54, 13] in applications to partial differential equations. Although there has been some work, such as [46, 47, 12, 13], on bounding the range of eigenvalues of an interval matrix, rigorous enclosure of the range (often called *outer enclosures*) is problematical for general matrices. For this reason, when we adopt a statistician's philosophy and employ intervals to reduce noisy data and obtain approximate bounds on ranges, we do not obtain mathematically rigorous results, but we may get reasonable "guesses" in problems that otherwise would be intractable or for which rigorous bounds are not meaningful. Furthermore, using interval technology, we obtain new approaches that compare favorably with traditional statistical methods.

Continued research on the methods in this volume should lead to additional mathematical rigor and explanations for the reasons the models appear to work so well.

[6] But similar to that in [17].

[7] Nonetheless, measures of quality of the interval result, in terms of how close it is to the exact range, are discussed.

References

1. Aberth, O., Schaefer, M.J.: Precise matrix eigenvalues using range arithmetic. SIAM Journal on Matrix Analysis and Applications 14(1), 235–241 (1993)
2. Alefeld, G., Herzberger, J.: Einführung in die Intervallrechnung. Springer-Verlag, Berlin, (1974)
3. Alefeld, G., Herzberger, J.: Introduction to Interval Computations. Academic Press Inc., New York (1983)
4. Alefeld, G.: Componentwise inclusion and exclusion sets for solutions of quadratic equations in finite-dimensional spaces. Numerische Mathematik 48(4), 391–416 (1986)
5. Alefeld, G., Spreuer, H.: Iterative improvement of componentwise errorbounds for invariant subspaces belonging to a double or nearly double eigenvalue. Computing 36, 321–334 (1986)
6. Behnke, H.: Inclusion of eigenvalues of general eigenvalue problems for matrices. In: U. Kulisch, H.J. Stetter (eds.) Scientific Computation with Automatic Result Verification, Computing. Supplementum, Vol. 6, pp. 69–78. Springer, New York (1988)
7. Behnke, H.: Bounds for eigenvalues of parameter-dependent matrices. Computing 49(2), 159–167 (1992)
8. Behnke, H., Mertins, U.: Bounds for eigenvalues with the use of finite elements. In: U. Kulisch, R. Lohner, A. Facius (eds.) Perspectives on Enclosure Methods: GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, September 2000, Karlsruhe, Germany, 119–132. Kluwer Academic Publishers, Amsterdam (2001)
9. Berz, M., Makino, K., Shamseddine, K., Hoffstätter, G.H., Wan, W.: COSY INFINITY and its applications to nonlinear dynamics. In: M. Berz, C. Bischof, G. Corliss, A. Griewank (eds.) Computational Differentiation: Techniques, Applications, and Tools, 363–365. SIAM, Philadelphia (1996)
10. Berz, M.: COSY INFINITY web page `http://cosy.pa.msu.edu/cosy.pa.msu.edu` (2000)
11. Brönnimann, H., Melquiond, G., Pion, S.: A proposal to add interval arithmetic to the C++ Standard Library. Technical proposal N1843-05-0103, CIS, Brooklyn Polytechnic University, S Brooklyn (2005)
12. Chen, S., Qiu, Z., Liu, Z.: A method for computing eigenvalue bounds in structural vibration systems with interval parameters. Computers and Structures 51(3), 309 (1994)
13. Chen, S., Qiu, Z., Liu, Z.: Perturbation method for computing eigenvalue bounds in structural vibration systems with interval parameters. Communications in Applied Numerical Methods 10(2), 121–134 (1994)
14. Corliss, G.F., Yu, J.: Testing COSY's interval and Taylor model arithmetic. In: R. Alt, A. Frommer, R.B. Kearfott, W. Luther (eds.) Numerical Software with Result Verification: Platforms, Algorithms, Applications in Engineering, Physics, and Economics, Lectures Notes in Computer Science, No. 2992, pp. 91–105. Springer, Heidelberg (2004)
15. Gau, C.Y., Stadtherr, M.A.: New interval methodologies for reliable chemical process modeling. Computers and Chemical Engineering 26, 827–840 (2002)
16. Gau, C.Y., Stadtherr, M.A.: Dynamic load balancing for parallel interval-Newton using message passing. Computers and Chemical Engineering 26, 811–815 (2002)

17. Gioia, F., Lauro, C.N.: Principal component analysis on interval data. Computational Statistics 21(2), 343–363 (2006)
18. Hansen, E.R., Walster, W.: Global Optimization Using Interval Analysis, 2nd ed. Marcel Dekker, New York (2003)
19. Hargreaves, G.I.: Interval analysis in MATLAB. Master's thesis, Department of Mathematics, University of Manchester (2002)
20. Jaulin, L., Keiffer, M., Didrit, O.,Walter, E.: Applied Interval Analysis. Springer-Verlag, Berlin (2001)
21. Kalmykov, S.A.: To the problem of determination of the symmetric matrix eigenvalues by means of the interval method. In: Numerical Analysis, Collect. Sci. Works, pp. 55–59. Sov. Acad. Sci., Sib. Branch, Inst. Theor. Appl. Mech., Novosibirsk, USSR (1978) (in Russian)
22. Kearfott, R.B.: A Fortran 90 environment for research and prototyping of enclosure algorithms for nonlinear equations and global optimization. ACM Transactions on Mathematical Software 21(1), 63–78 (1995)
23. Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Nonconvex Optimization and Its Applications. No. 13. Kluwer Academic, Norwell, MA (1996)
24. Kearfott, R.B.: Verified branch and bound for singular linear and nonlinear programs: An epsilon-inflation process (April 2007), Submitted
25. Kearfott, R.B.: GlobSol User Guide. Optimization Methods and Software (2008). Submitted
26. Knüppel, O.: PROFIL/BIAS - A fast interval library. Computing 53(3–4), 277–287 (1994)
27. Knüppel, O.: PROFIL/BIAS v 2.0. Bericht 99.1, Technische Universität Hamburg-Harburg, Harburg, Germany (1999). Available from `http://www.ti3.tu-harburg.de/profil_e`
28. Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational Complexity and Feasibility of Data Processing and Interval Computations, Applied Optimization, Vol. 10. Kluwer Academic, Norwell, MA (1998)
29. Kulisch, U.W., Miranker, W.L.: Computer Arithmetic in Theory and Practice. Computer Science and Applied Mathematics. Academic Press Inc., New York (1981)
30. Lee, D., Mavroidis, C., Merlet, J.P.: Five precision point synthesis of spatial RRR manipulators using interval analysis. Journal of Mechanical Design 126, 842–849 (2004)
31. Lerch, M., Tischler, G., Gudenberg, J.W.V., Hofschuster, W., Krämer, W.: FILIB++, a fast interval library supporting containment computations. ACM Transactions on Mathematical Software 32(2), 299–324 (2006)
32. Lin, Y., Stadtherr, M.A.: Advances in interval methods for deterministic global optimization in chemical engineering. Journal of Global Optimization 29, 281–296 (2004)
33. Lin, Y., Stadtherr, M.A.: Lp strategy for interval-Newton method in deterministic global optimization. Industrial & Engineering Chemistry Research, 43, 3741–3749 (2004)
34. Lin, Y., Stadtherr, M.A.: Locating stationary points of sorbate-zeolite potential energy surfaces using interval analysis. J. Chemical Physics, 121, 10159-10166 (2004)
35. Lin, Y., Stadtherr, M.A.: Deterministic global optimization of molecular structures using interval analysis. J. Computational Chemistry 26, 1413–1420 (2005)

36. Moore, R.E.: Interval arithmetic and automatic error analysis in digital computing. Ph.D. dissertation, Department of Mathematics, Stanford University, Stanford, CA (1962)
37. Moore, R.E.: Interval Analysis. Prentice–Hall, Upper Saddle River, NJ (1966)
38. Moore, R.E.: Methods and Applications of Interval Analysis. Society for Industrial and Applied Mathematics, Philadelphia (1979)
39. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to interval algorithms and their applications with INTLAB: a MATLAB toolkit. Submitted
40. Neumaier, A.: Interval Methods for Systems of Equations. Encyclopedia of Mathematics and Its Applications, Vol. 37. Cambridge University Press, Cambridge (1990)
41. Neumaier, A.: Introduction to Numerical Analysis. Cambridge University Press, Cambridge (2001)
42. Neumaier, A., Pownuk, A.: Linear systems with large uncertainties, with applications to truss structures. Reliable Computing 13, 149–172 (2007)
43. Nooner, M., Hu, C.: A computational environment for interval matrices. In: R.L. Muhanna, R.L. Mullen (eds.) Proceedings of 2006 Workshop on Reliable Engineering Computing, pp. 65–74. Georgia Tech. University, Savanna (2006). http://www.gtsav.gatech.edu/workshop/rec06/proceedings.html
44. Oishi, S.: Fast enclosure of matrix eigenvalues and singular values via rounding mode controlled computation. Linear Algebra and its Applications 324(1–3), 133–146 (2001)
45. Plum, M.: Computer-assisted enclosure methods for elliptic differential equations. Linear Algebra and its Applications 324(1–3), 147–187 (2001)
46. Rohn, J., Deif, A.: On the range of eigenvalues of an interval matrix. Computing 47(3–4), 373–377 (1992)
47. Rohn, J.: Interval matrices: Singularity and real eigenvalues. SIAM Journal on Matrix Analysis and Applications 14(1), 82–91 (1993)
48. Rump, S.M.: INTLAB-INTerval LABoratory. In: T. Csendes (ed.) Developments in Reliable Computing: Papers presented at the International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics, Vol. 5(3), pp. 77–104. Kluwer Academic, Norwell, MA (1999)
49. Rump, S.M.: INTLAB - INTerval LABoratory (1999-2008) http://www.ti3.tu-harburg.de/rump/intlab/
50. Stadtherr, M.A.: Interval analysis: Application to phase equilibrium problems. In: A. Iserles (ed.) Encyclopedia of Optimization. Kluwer Academic, Norwell, MA (2001)
51. Stadtherr, M.A.: Interval analysis: Application to chemical engineering design problems. In: A. Iserles (ed.) Encyclopedia of Optimization. Kluwer Academic, Norwell, MA (2001)
52. Sun: Sun studio math libraries (1994-2007). Available from http://developers.sun.com/sunstudio/documentation/libraries/math_libraries.jsp
53. Sunaga, T.: Theory of interval algebra and its application to numerical analysis. RAAG Memoirs 2, 29–46 (1958)
54. Wieners, C.: A parallel Newton multigrid method for high order finite elements and its application on numerical existence proofs for elliptic boundary value equation. Zeitschrift für Angewandte Mathematik und Mechanik 76, 171–176 (1996)