INFORMATICA, 2005, Vol. 16, No. 1, 145–154

Comparison of Packages for Interval Arithmetic

Julius ŽILINSKAS

Institute of Mathematics and Informatics
Akademijos 4, 08663, Vilnius, Lithuania
e-mail: julius.zilinskas@mii.lt

Received: January 2004

Abstract. In this paper public available C and C++ packages for interval arithmetic are investigated and experimentally compared. The results of comparison give suggestions which packages and when are preferable.

Key words: interval arithmetic, packages for interval arithmetic, interval global optimization.

1. Introduction

Many problems in engineering, physics, economic may be formulated as global optimization problems. Mathematically the problem is formulated as optimization of a nonlinear objective function of continuous variables over a multidimensional feasible region. Interval global optimization methods are based on interval arithmetic proposed in (Moore, 1966). The lower and upper bounds for the function values in the sub-region are estimated applying the interval operations with intervals instead of the real operations with real variables in the algorithm of calculation the function values. The bounds are useful to detect the sub-regions of the feasible region not containing a global minimizer.

Because of necessary outward rounding in interval arithmetic, the implementation of interval arithmetic is not straightforward. There are several packages implementing interval arithmetic, but there is no experimental comparison of them. Only some completely different implementations (one in Matlab, one in C++, and one in Fortran-90) are compared in (Kearfott *et al.*, 2004). However it is not clear which package for interval arithmetic is preferable when a researcher implements his interval method. For example, to make experiments with balanced random interval arithmetic (Žilinskas and Bogle, 2003; Žilinskas and Bogle, 2004) the author has tried to implement his own interval library, to use fi_lib (Hofschuster and Krämer, 1997) and filib++ (Lerch *et al.*, 2001), because there was no available experimental comparison of packages for interval arithmetic published in scientific literature. In this paper public available C and C++ packages for interval arithmetic are investigated and compared experimentally.

2. Interval Arithmetic

Interval arithmetic has been proposed in (Moore, 1966). Interval arithmetic operates with real intervals $\underline{x} = [\underline{x}, \overline{x}] = \{x \in \Re | \underline{x} \leqslant x \leqslant \overline{x}\}$, defined by two real numbers $\underline{x} \in \Re$ and $\overline{x} \in \Re$, $\underline{x} \leqslant \overline{x}$. For any real arithmetic operation $x \circ y$ the corresponding interval arithmetic operation $\underline{x} \circ \underline{y}$ is defined whose result is an interval containing every possible number produced by real operation with real numbers from each interval:

$$\underline{x} \circ \underline{y} = \left\{ x \circ y \,|\, x \in \underline{x},\ y \in \underline{y} \right\}.$$

Basic interval arithmetic operations are defined as:

$$\underline{x} + \underline{y} = \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right],$$
$$\underline{x} - \underline{y} = \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right],$$
$$\underline{x} \times \underline{y} = \begin{cases} \left[\underline{x}\underline{y}, \overline{x}\overline{y}\right], & \underline{x} > 0, \quad \underline{y} > 0, \\ \left[\overline{x}\underline{y}, \overline{x}\overline{y}\right], & \underline{x} > 0, \quad \underline{y} \ni 0, \\ \left[\overline{x}\underline{y}, \underline{x}\overline{y}\right], & \underline{x} > 0, \quad \underline{y} < 0, \\ \left[\underline{x}\overline{y}, \overline{x}\overline{y}\right], & \underline{x} \ni 0, \quad \underline{y} > 0, \\ \left[\min(\underline{x}\overline{y}, \overline{x}\underline{y}),\ \max(\underline{x}\underline{y}, \overline{x}\overline{y})\right], & \underline{x} \ni 0, \quad \underline{y} \ni 0, \\ \left[\overline{x}\underline{y}, \underline{x}\underline{y}\right], & \underline{x} \ni 0, \quad \underline{y} < 0, \\ \left[\underline{x}\overline{y}, \overline{x}\underline{y}\right], & \underline{x} < 0, \quad \underline{y} > 0, \\ \left[\underline{x}\overline{y}, \underline{x}\underline{y}\right], & \underline{x} < 0, \quad \underline{y} \ni 0, \\ \left[\overline{x}\overline{y}, \underline{x}\underline{y}\right], & \underline{x} < 0, \quad \underline{y} < 0, \end{cases}$$
$$\underline{x} / \underline{y} = \begin{cases} \left[\underline{x}/\overline{y}, \overline{x}/\underline{y}\right], & \underline{x} > 0, \quad \underline{y} > 0, \\ \left[\overline{x}/\overline{y}, \underline{x}/\underline{y}\right], & \underline{x} > 0, \quad \underline{y} < 0, \\ \left[\underline{x}/\underline{y}, \overline{x}/\underline{y}\right], & \underline{x} \ni 0, \quad \underline{y} > 0, \\ \left[\overline{x}/\overline{y}, \underline{x}/\overline{y}\right], & \underline{x} \ni 0, \quad \underline{y} < 0, \\ \left[\underline{x}/\underline{y}, \overline{x}/\overline{y}\right], & \underline{x} < 0, \quad \underline{y} > 0, \\ \left[\overline{x}/\underline{y}, \underline{x}/\overline{y}\right], & \underline{x} < 0, \quad \underline{y} < 0. \end{cases}$$

The guaranteed lower and upper bounds for the function values in the region defined by intervals of variables can be estimated applying interval operations with intervals instead of real operations in the algorithm to calculate the function values. The evaluated bounds always enclose the full range of the function values in the defined region:

$$\left\{ f(X) \,|\, X \in \overline{X},\ \ \underline{X} \in \Re^n,\ \ \overline{X} \in \Re^n \right\} \subseteq \underline{f}(\overline{X}),$$

where f: $\Re^n \to \Re$, $\underline{f}$: $[\Re, \Re]^n \to [\Re, \Re]$.

The bounds may be used in global optimization to detect the sub-regions of the feasible region not containing a global minimizer. Such sub-regions may be discarded from the further search. If the objective function is differentiable it is possible to compute the intervals of the derivatives and discard the sub-regions where the objective function is monotone. If the objective function is twice continuously differentiable it is possible to

compute the intervals of the second derivatives and discard the sub-regions where the objective function is concave. If the objective function is twice differentiable the special interval Newton method can be applied to reduce the sub-regions, and discard the sub-regions where there are no stationary points (Hansen, 1992).

Computers with limited precision can not represent all real numbers exactly. To guarantee the interval enclosure outward rounding should be used: during interval computation the number representing the lower end of the resulting interval ($\underline{x}$) should be rounded toward $-\infty$ and the number representing the upper end of the resulting interval ($\overline{x}$) should be rounded toward $+\infty$. Interval arithmetic with outward rounding may be used to find guaranteed interval enclosure of result of real operations, while real arithmetic may produce not correct results because of rounding errors in real computations. Although computers with limited precision can not represent all real numbers exactly, they can represent guaranteed enclosures using intervals, for example $\pi = [3.1415, 3.1416]$.

Division by interval containing 0 is not defined in standard interval arithmetic and is forbidden. The same holds for logarithm of interval containing 0 and other mathematical interval functions which are not defined over all space of real numbers. This is serious drawback, because computer programs are terminated when an arithmetic error exception is generated. Extended interval arithmetic has been proposed to overcome problems with partially defined interval operations and mathematical functions. In extended interval arithmetic no arithmetic error exception is generated, but the resulting interval may be extended towards negative or positive infinity. Special values to represent infinities and NotaNumber defined in the IEEE floating-point standard 754 helps implementation of the extended interval arithmetic.

3. Packages for Interval Arithmetic

3.1. *Developments at Hamburg–Harburg Technical University*

First public available C package for interval arithmetic was BIAS – Basic Interval Arithmetic Subroutines (Knüppel, 1993a). BIAS development was guided by the idea of BLAS (Basic Linear Algebra Subroutines) – to provide an interface for basic vector and matrix operations with fast implementations on various computers. The idea of BIAS was to provide such an interface for interval operations with the objectives: portability, independence of a specific interval representation and very efficient use of hardware. To reach portability subroutines have been implemented using ANSI C. The routines cover the basic scalar, vector and matrix interval operations. Hardware specific subroutines for the switching of rounding modes have been implemented. When the arithmetic exception is generated (e.g., division by interval containing zero), if possible the result is silently set (e.g., to $[-\infty, +\infty]$) without program termination.

The set of BIAS routines can be included in higher-level user-friendly programming environment, what was done with a C++ class library PROFIL – Programmer's Runtime Optimized Fast Interval Library (Knüppel, 1993b; Knüppel, 1994), which provides a C++

interface to BIAS with the advantage that it is independent from the internal representation and the implementation of interval types. PROFIL introduces data types: vectors, matrices, intervals, interval vectors and matrices, integer vectors and matrices, and operations between them, as well as several commonly used routines as linear interval system solver.

Several extensions to PROFIL including a set of test matrices, subroutines for local optimization, general linear singly linked lists, automatic differentiation and sample programs have been added as a separate package – PROFIL/BIAS extensions (Knüppel and Simenec, 1993).

A public available implementation of the global unconstrained minimization method involving a combination of local search, branch-and-bound technique and interval arithmetic (Jansson and Knüppel, 1995) has been written using PROFIL/BIAS together with PROFIL extensions (Knüppel, 1995).

PROFIL, extensions and global optimization package have been implemented in C++. As well as BIAS, all packages intended to be portable. However they have been implemented before the standard for the C++ programming language (ISO/IEC, 1998) and are not compilable with recent compilers as GCC later than 2.95. To meet a C++ standard specification a new completely revised version of PROFIL/BIAS including extensions and global optimization – PROFIL V 2.0 (Knüppel, 1999) package has been implemented and is public available.

3.2. *Developments at Wuppertal University*

A fast interval library fi_lib (Hofschuster and Krämer, 1997) was originally developed at Karlsruhe University and latter extended at Wuppertal University. The most important aim of the library was the fast computation of guaranteed bounds for interval versions of a comprehensive set of elementary function. Fast table look-up algorithms are used for the basic functions like arctan, exp or log. All elementary function routines are supplied with reliable relative error bounds of high quality. The error estimates cover rounding errors, errors introduced by not exactly representable constants as well as approximation errors (best approximations with reliable error bounds). All error estimates are reliable worst-case estimates, which have been derived using interval methods. The routines do not manipulate the rounding mode of basic operations, because setting the rounding mode may be rather expensive. All computations are done using the IEEE-double format. To get good portability all programs are written in ANSI-C. Source code and some applications are public available.

A C++ class library for extended scientific computing: C-XSC (Hofschuster *et al.*, 2001) is a tool for the development of numerical algorithms delivering highly accurate and automatically verified results. The library integrates fast interval library fi_lib and provides a large number of predefined numerical data types implemented as C++ classes. C-XSC allows high-level programming of numerical applications in C++. The source codes of the C-XSC 2.0 are freely public available. C-XSC 2.0 conforms ISO/IEC C++ standard (ISO/IEC, 1998).

C++ toolbox for verified computing – CToolbox (Hammer *et al.*, 1995) is an addition to C-XSC library of problem-solving routines. It covers one-dimensional and multidimensional problems: accurate evaluation of polynomials, automatic differentiation, linear and nonlinear systems of equations, linear optimization, global optimization, zeros of complex polynomials. As well as C-XSC, the source codes of the toolbox are freely public available and it conforms ISO/IEC C++ standard.

A C++ extension of the interval library fi_lib – filib++ (Lerch *et al.*, 2001) extends it in two aspects. First, it adds an extended interval mode, that extends the exception-free computation mode using special values to represent infinities and NotaNumber known from the IEEE floating-point standard 754 to intervals. Second, the new state of the art design uses templates and traits classes in order to get an efficient, easily extendable and potable library, fully according to the C++ standard (ISO/IEC, 1998).

3.3. *SUN Forte Compilers with Interval Support*

Differently from the public freely available packages discussed before, the SUN Forte Compilers are commercial programs. SUN Forte C++ compiler integrates interval arithmetic library. The goal of SUN interval support in C++ was to stimulate development of commercial interval solver libraries and applications by providing program developers with quality interval code, narrow-width interval results, rapidly executing interval code and an easy-to-use software development environment (SUN Microsystems, 2001).

Interval template specializations for intervals using float, double, and long double scalar types are provided in SUN Forte C++ compiler and functions for interval<double> are tuned for speed. Extended interval arithmetic is supported – valid results are produced for any possible operator-operand combination, including division by zero and other indeterminate forms involving zero and infinities. Interval arithmetic operations and mathematical functions form a closed mathematical system.

4. Comparison of Packages for Interval Arithmetic

The criteria of packages for interval arithmetic are:

- guarantee of enclosure – the evaluated interval should enclose the full range of the function values in the defined multidimensional interval of variables, interval global optimization algorithm may miss a global minimum if it uses package of interval arithmetic without guaranteed enclosure,
- realization of extended interval arithmetic,
- portability,
- available extensions,
- speed of interval arithmetic operations and mathematical functions, global optimization algorithm takes less time if interval operations and mathematical functions take less time,
- width of resulting intervals – the evaluated interval should be as narrow as possible with guaranteed enclosure, global optimization algorithm is potentially faster if it uses tighter ranges of objective function.

All criteria will be used in our comparison.

The authors of all discussed packages and libraries for interval arithmetic state, that the enclosure is guaranteed. PROFIL/BIAS, filib++ and SUN Forte C++ interval library support extended interval arithmetic.

BIAS and fi_lib are implemented in ANSI C and seem to be most portable. They are rarely used without their extensions, but recent version of the extensions PROFIL V 2.0 for BIAS and C-XSC 2.0 for fi_lib both conforms with ISO/IEC C++ standard and should be portable.

SUN Forte C++ is commercial program and all other discussed packages are freely public available.

C++ interval library filib++ and SUN Forte C++ interval library both support extended interval arithmetic and both use templates. The definitions of these two C++ interval libraries are very similar and they should be easily exchangeable. SUN Forte C++ interval library may be used when the commercial SUN Forte compiler is available and easily exchanged by filib++ otherwise.

However filib++ and SUN Forte C++ interval library have no extensions yet as BIAS and fi_lib have. PROFIL V 2.0 (an extension of BIAS) provides subroutines for local optimization, general linear singly linked lists, automatic differentiation and global optimization. C-XSC and CToolbox (extensions of fi_lib) provide subroutines for accurate evaluation of polynomials, automatic differentiation, linear and nonlinear systems of equations, linear optimization, global optimization, zeros of complex polynomials.

Speed and widths of resulting intervals of packages for interval arithmetic should be measured experimentally. Speed is measured using time required to evaluate interval arithmetic operation or mathematical function. The package is faster when it requires less time. Time required to evaluate interval operations and mathematical functions may depend on interval data therefore we use 1000 random intervals in our experiments. The same set of 1000 random intervals is used to estimate speed of all packages. To measure time the interval arithmetic operation or mathematical function is evaluated 10000 times with each random interval. The measured time shows how long does it take to evaluate the interval arithmetic operation or mathematical function ten million times.

The width of resulting interval shows overestimation. The package for interval arithmetic evaluate intervals more exactly when resulting intervals are narrower. We make relative measurements in our experiments – the mean ratios of widths of resulting intervals of pairs of packages are estimated:

$$mr = \frac{1}{N}\sum_{i=0}^{N-1}\frac{\overline{x_i} - \underline{x_i}}{\overline{y_i} - \underline{y_i}},$$

where $\overline{x_i}$ and $\overline{y_i}$ are resulting intervals of two different packages. Widths of resulting intervals are very similar, therefore the real numbers with values near 1.0 are summed and differences may be lost. It is more accurate to sum numbers with values near 0.0.

This is why we use relative criterion of widths defined bellow:

$$e = mr - 1.0 = \frac{1}{N}\sum_{i=0}^{N-1}\left(\frac{\overline{x_i} - \underline{x_i}}{\overline{y_i} - \underline{y_i}} - 1.0\right) = \frac{1}{N}\sum_{i=0}^{N-1}\frac{\left(\overline{x_i} - \underline{x_i}\right) - \left(\overline{y_i} - \underline{y_i}\right)}{\overline{y_i} - \underline{y_i}}, \quad (1)$$

which shows only non integer part of the mean ratio.

Experiments on two different computer systems have been performed. One is SUN UltraSPARC Station with Solaris 2.9 operating system, SUN Forte C++ compiler to compile program with SUN Forte library for interval arithmetic and GCC 2.95 to compile programs with other packages for interval arithmetic. Another system is ix86 Personal Computer with Linux operating system and GCC 2.95 compiler.

The results of experiments are shown in Tables 1 and 2. In the tables t is time in seconds required to evaluate the interval arithmetic operation or mathematical function ten million times. The package is faster when it requires less time. In the tables e is non integer part of the mean ratios of widths of resulting intervals of each package ($\overline{x}$ in equation (1)) and SUN Forte C++ interval library ($\underline{y}$ in equation (1)). The package is more accurate when the mean ratio is lower.

Because SUN Forte C++ interval library was used as the basis for evaluation of the ratios of widths, its values of e are always equal to 0, for the same interval data the resulting intervals are always the same. C++ interval library filib++ produces same resulting intervals as SUN Forte C++ interval library for interval arithmetic operations and square function. For other interval mathematical functions filib++ produced resulting intervals are a little bit wider. Interval library fi_lib produces wider resulting intervals than filib++ for interval arithmetic operations and square function and very similar resulting intervals as filib++ for other interval mathematical functions. PROFIL and PROFIL V 2.0 produce resulting intervals with the same width. They produce narrower resulting intervals than

Table 1

Experimental comparison of packages for interval arithmetic on SUN SPARC Station

	SUN Forte		filib++		fi_lib		PROFIL V 2.0	
	t	e	t	e	t	e	t	e
add	2.42	0	2.46	0	5.02	6.18e−16	2.05	−7.64e−17
sub	2.35	0	2.45	0	4.93	3.10e−16	2.03	−7.91e−17
mul	2.80	0	2.82	0	5.39	2.91e−16	2.09	−7.99e−17
div	3.18	0	3.19	0	5.64	2.96e−16	2.55	−7.83e−17
sin	4.11	0	9.98	3.37e−14	15.67	3.38e−14	84.62	9.04e−15
cos	3.84	0	8.70	2.24e−13	15.84	2.24e−13	627.65	7.51e−14
tan	5.49	0	11.62	4.25e−14	19.58	4.26e−14	87.46	6.13e−15
sqr	2.78	0	2.70	0	5.32	6.03e−16	2.49	−7.78e−17
sqrt	3.87	0	4.99	2.40e−15	8.07	2.55e−15	76.49	1.54e−14
exp	3.79	0	8.15	2.47e−14	11.05	2.48e−14	79.62	1.47e−14
log	3.60	0	9.24	6.13e−15	12.35	6.26e−15	79.56	2.84e−15

Table 2
Experimental comparison of packages for interval arithmetic on Personal Computer with Linux

	filib++		fi_lib		PROFIL		PROFIL V 2.0	
	t	e	t	e	t	e	t	e
add	1.93	0	1.89	6.18e−16	0.27	−1.94e−17	0.37	−1.94e−17
sub	1.89	0	1.96	3.10e−16	0.28	−2.75e−17	0.38	−2.75e−17
mul	2.12	0	2.05	2.91e−16	0.30	−4.85e−17	0.39	−4.85e−17
div	2.33	0	2.23	2.96e−16	0.56	−5.07e−17	0.58	−5.07e−17
sin	3.32	3.37e−14	4.79	3.38e−14	10.34	7.42e−15	10.35	7.42e−15
cos	3.13	2.24e−13	4.79	2.24e−13	95.93	7.42e−14	101.99	7.42e−14
tan	3.84	4.25e−14	6.60	4.26e−14	10.91	4.96e−15	10.96	4.96e−15
sqr	1.87	0	1.85	6.02e−16	0.51	−3.81e−17	0.63	−3.81e−17
sqrt	1.58	2.40e−15	2.77	2.55e−15	8.34	1.37e−14	8.32	1.37e−14
exp	3.24	2.47e−14	4.57	2.48e−14	10.72	1.19e−14	10.72	1.19e−14
log	4.61	6.13e−15	4.72	6.26e−15	10.33	2.43e−15	10.28	2.43e−15

SUN Forte C++ interval library for interval arithmetic operations and square function and wider resulting intervals than SUN Forte C++ interval library for other interval mathematical functions. The packages could be aligned by decreasing order of accuracy as:

- SUN Forte C++ interval library,
- PROFIL (both versions),
- C++ interval library filib++,
- interval library fi_lib.

However differences of widths of resulting intervals are very small and will not impact to the speed of global optimization algorithms.

The speed of SUN Forte C++ interval library and the speed of C++ interval library filib++ are very similar for interval arithmetic operations and square function. SUN Forte C++ interval library is approximately two times faster than filib++ for other interval mathematical functions. C++ interval library filib++ is approximately two times faster than interval library fi_lib on SUN SPARC Station, but their speed on Personal Computer with Linux is similar, although filib++ is a little bit faster. Speed of PROFIL and PROFIL V 2.0 is very similar as well as their resulting intervals. On SUN SPARC Station, PROFIL is a little bit faster than other packages for interval arithmetic operations and square function, but it is much slower for other interval mathematical functions. On Personal Computer with Linux, PROFIL is three to seven times faster than filib++ for interval arithmetic operations and square function, but three to thirty times slower than filib++ for other interval mathematical functions. The reason of fast interval arithmetic operations in PROFIL is that it uses hardware efficiently. The mathematical functions are slow in PROFIL because it does not use fast table look-up algorithms as other packages do. The packages could be aligned by decreasing speed as:

- PROFIL, if no interval mathematical function other than square function is used,
- SUN Forte C++ interval library,

- C++ interval library filib++,
- interval library fi_lib,
- PROFIL, if interval mathematical functions are used what is usually the case.

The experimental comparison of the packages for interval arithmetic suggests that PROFIL should be used when no interval mathematical function other than square function is used. However it is rare case. Otherwise SUN Forte C++ interval library is most accurate and fast one. If this library is not available because it is commercial and non free, C++ interval library filib++ is most appropriate.

5. Conclusions

Public available C and C++ packages for interval arithmetic have been investigated and experimentally compared. Investigation and experimental comparison of C and C++ packages for interval arithmetic suggest to use SUN Forte C++ interval library when the commercial SUN Forte compiler is available and public freely available C++ interval library filib++ otherwise. These packages are most fast and accurate, they implement extended interval arithmetic. Interval algorithms should be implemented so that the used interval library could be exchanged easily and similar definitions of these two libraries are very helpful.

References

Hammer, R., M. Hocks, U. Kulish and D.Ratz (1995). *C++ Toolbox for Verified Computing*: *Basic Numerical Problems*. Springer, Berlin.

Hansen, E. (1992). *Global Optimization Using Interval Analysis*. Marcel Dekker, New York.

Hofschuster, W., and W. Krämer (1997). A fast public domain interval library in ANSI C. In: A. Sydow (Ed.), *Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*, Vol. 2. Berlin. pp. 395–400.

Hofschuster, W., W. Krämer, S. Wedner and A. Wiethoff (2001). *C-XSC 2.0 - A C++ Class Library for Extended Scientific Computing*. Preprint 2001/1, Universität Wuppertal.

ISO/IEC (1998). *14882*: *Standard for the C++ Programming Language*. ISO/IEC.

Jansson, C., and O. Knüppel (1995). A branch-and-bound algorithm for bound constrained optimization problems without derivatives. *Journal of Global Optimization*, **7**(3), 297–331.

Kearfott, R.B., M. Neher, S. Oishi and F. Rico (2004). Libraries, tools, and interactive systems for verified computations four case studies. *Lecture Notes in Computer Science*, **2991**, 36–63.

Knüppel, O. (1993a). *BIAS – Basic Interval Arithmetic Subroutines*. Report 93.3, Technische Universität Hamburg–Harburg.

Knüppel, O. (1993b). *PROFIL – Programmer's Runtime Optimized Fast Interval Library*. Report 93.4, Technische Universität Hamburg–Harburg.

Knüppel, O. (1994). PROFIL/BIAS – A fast interval library. *Computing*, **53**(3–4), 277–287.

Knüppel, O. (1995). *A PROFIL/BIAS Implementation of a Global Minimization Algorithm*. Report 95.4, Technische Universität Hamburg–Harburg.

Knüppel, O. (1999). *PROFIL/BIAS V 2.0*. Report 99.1, Technische Universität Hamburg–Harburg.

Knüppel, O., and T. Simenec (1993). *PROFIL/BIAS Extensions*. Report 93.5, Technische Universität Hamburg–Harburg.

Lerch, M., G. Tischler, J. W. von Gudenberg, W. Hofschuster and W. Krämer (2001). *The Interval Library filib++ 2.0 – Design, Features and Sample Programs*. Preprint 2001/4, Universität Wuppertal.

Moore, R. E. (1966). *Interval Analysis*. Prentice-Hall.
SUN Microsystems (2001). *C++ Interval Arithmetic Programming Reference*. Forte Developer 6 update 2 (Sun WorkShop 6 update 2). SUN Microsystems.
Žilinskas, J., and I.D.L. Bogle (2003). Evaluation ranges of functions using balanced random interval arithmetic. *Informatica*, **14**(3), 403–416.
Žilinskas, J., and I.D.L. Bogle (2004). Balanced random interval arithmetic. *Computers and Chemical Engineering*, **28**(5), 839–851.

J. Žilinskas is a researcher in Systems Analysis Department at the Institute of Mathematics and Informatics, Lithuania. He studied informatics at Kaunas University of Technology receiving his PhD in 2002. His research interests include global optimization and parallel computing.

Intervalų aritmetikos bibliotekų palyginimas

Julius ŽILINSKAS

Straipsnyje išnagrinėtos ir eksperimentiškai palygintos viešai prieinamos C ir C++ intervalų aritmetikos bibliotekos. Palyginimo rezultatai siūlo, kada ir kokias intervalų aritmetikos bibliotekas naudoti.