C–XSC 2.0: A C++ Class Library for Extended Scientific Computing

Walter Krämer and Werner Hofschuster
University of Wuppertal, Germany

The original version of the C++ class library C–XSC [4] is about ten years old. But in the last decade the underlying programming language C++ has been developed significantly. Since November 1998 the C++ standard [3] is available and more and more compilers support (most of) the features of this standard. The new version C–XSC 2.0 [2] conforms to the C++ standard.

For those who are not familiar with C–XSC let us first motivate the library by quoting essential parts (with slight modifications) from the preface of the book [4]: The programming environment C–XSC (C for eXtended Scientific Computing) is a powerful and easy to use programming tool, especially for scientific and engineering applications. C–XSC makes the computer more powerful arithmetically and significantly simplifies programming in the field of scientific computing (especially in the field of interval mathematics occasionally called mathematical numerics[1]). C–XSC is implemented as a numerical class library in the programming language C++ .

The speed of digital computers is ever increasing. But at processor speeds of gigaFLOPS it is a significant question whether floating-point arithmetic, which may fail already in simple calculations, is still adequate to be used in computers for huge problems.

Mathematicians have contrived algorithms which deliver highly accurate and automatically verified results by dappling mathematical fixed-point theorems. This means that these computations carry their own accuracy control. However, their implementations require suitable arithmetic support and powerful programming tools which were not previously available. The development of C–XSC has aimed at providing these tools within C++. C–XSC is particularly suited for the development of numerical algorithms that deliver highly accurate and automatically verified results, which are essential, for example, in simulation runs where the user has to distinguish between computational artifacts and genuine reactions of the model. Problem-solving functions with automatic result verification have been developed in C–XSC for several standard problems of numerical analysis.

[1] In contrast to Numerical Mathematics, where results are sometimes merely speculative, Mathematical Numerics delivers solutions of numerical problems that are mathematically proven to be correct.

C – XSC consists of a run time system written in ANSI C and C++ including an optimal dot product and many predefined data types for elements of the most commonly used vector spaces such as real and complex numbers, vectors, and matrices. Operators for elements of these types are predefined and can be called by their usual operator symbols. Thus, arithmetic expressions and numerical algorithms are expressed in a notation that is very close to the usual mathematical notation. C – XSC allows to write verification algorithms in a way which is very near to pseudo-code used in scientific publications. All predefined numerical operators are of highest accuracy. That is, the computed result differs from the correct result by at most one rounding.

While the emphasis in computing is traditionally on speed, in C – XSC, the emphasis is more on accuracy and reliability of results. The total time for solving a problem is the sum of the programming effort, the processing time, and the time for the interpretation of results. We contend that C – XSC reduces this sum considerably.

C++ programmers should be able to use and write programs in C – XSC immediately. C – XSC simplifies programming by providing many predefined data types and arithmetic operators. Programs are much easier to read, to write, and to debug.

What is new in C – XSC 2.0?

- All routines are now in the namespace `cxsc`
- Explicit typecast constructors
- Constant values passed by reference are now passed by const reference
- The error handling is done according to the C++ error handling using exception classes
- Modification in the field for subvectors and submatrices
- The library uses templates extensively
- The source code of C-XSC 2.0 is freely available from `http://www.math.uni-wuppertal.de/~xsc/xsc/download.html`
- Older C – XSC programs have to be modified slightly to run with C – XSC 2.0. We will discuss this point in our talk
- The source code of a new version of the C++ Toolbox for Verified Computing [1] which works with C-XSC 2.0 is freely available from `http://www.math.uni-wuppertal.de/~xsc/xsc/download.html`

The following areas are covered by the **C++ Toolbox for Verified Computing** [1] (Its current version is freely available and runs with C – XSC 2.0):

- One-dimensional problems: Extended interval division, Evaluation of polynomials, Automatic differentiation, Nonlinear equations in one variable, Global optimization, Accurate evaluation of arithmetic expressions, Zeros of complex polynomials.
- Multi-dimensional problems: Linear systems of equations, Linear optimization, Automatic differentiation for gradients, Jacobians, and Hessian, Nonlinear systems of Equations, Global optimization, Initial value problems in ordinary differential equations[2].

Additional self-verifying problem solving routines based on C–XSC 2.0 are available (e.g., a slope arithmetic in forward and in reverse mode, numerical quadrature and cubature [9], validated bounds for Taylor coefficients [5]; for downloads see
`http://www.math.uni-wuppertal.de/~xsc/xsc/cxsc_software.html`) or will be made available. Further developments will be discussed in our talk. We will also comment on other interval tools, especially on INTLAB [6] and Sun compilers [7] supporting intervals. We will see that so called containment computations [8] can be used, e.g., to compute (with rather small programming effort) enclosures of all roots of functions over finite and infinite domains.

Acknowledgments: Many colleagues and scientists (see [2] Paragraph 1) have directly and indirectly contributed to the realization of C–XSC and C–XSC 2.0. The authors would like to thank each of them for his or her cooperation.

Keywords: C–XSC, C++ Class Library, Interval Mathematics, Validated Numerics, Enclosure Methods, INTLAB, Containment Computations

References

[1] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz, *C++ Toolbox for Verified Computing: Basic Numerical Problems*, Springer-Verlag, Berlin, 1995.

[2] W. Hofschuster, W. Krämer, S. Wedner, and A. Wiethoff, *C–XSC 2.0: A C++ Class Library for Extended Scientific Computing*, University of Wuppertal, 2001, pp. 1–24.

[3] ISO/IEC 14882: *Standard for the C++ Programming Language*, 1998.

[4] R. Klatte, U. Kulisch, C. Lawo, M. Rauch, and A. Wiethoff, *C–XSC – A C++ Class Library for Scientific Computing*. Springer-Verlag, Berlin, 1993.

[5] M. Neher, "Validated Bounds for Taylor Coefficients of Analytic Functions", *Reliable Computing*, 2001, Vol. 7, No. 4.

[6] S. M. Rump, "INTLAB –INTerval LABoratory", in T. Csendes (ed.), *Developments in Reliable Computing*, Kluwer, 1999.

[2] Available from R. Lohner (see http://www.uni-karlsruhe.de/~Rudolf.Lohner/)

[7] Sun Forte[TM] Developer 6 update 2 compilers (see `http://www.sun.com/forte/index.html`)

[8] G. W. Walster, et al., *The "Simple" Closed Interval System*, Sun Microsystems, Feb 2000.

[9] S. Wedner, *Verifizierte Bestimmung singulärer Integrale – Quadratur und Kubatur*. Dissertation, Universität Karlsruhe, 2000.