23

The XSC tools for extended scientific computing

Ulrich Kulisch
Universität Karlsruhe
Institut für Angewandte Mathematik, Postfach 6980, 76128 Karlsruhe, Germany. Fax: +49-721-69 52 83.

Abstract
This paper summarizes work in developing tools for verified scientific computing in a variety of computing environments.

Keywords
Automatic result verification, ACRITH-XSC, C-XSC, FORTRAN-XSC, PASCAL-XSC

1 SUMMARY

Many applications require that rigorous mathematics can be done with a computer using floating-point numbers. As an example, this is essential in simulation runs (e.g., fusion reactors) or mathematical modeling where the user has to distinguish between computational artifacts and genuine reactions of the model. The model can only be developed systematically if errors entered by the computation can be excluded essentially. There are many other applications, which can be solved more adequately and simply, or can be solved at all, if an expanded arithmetic capability is available. This talk illustrated this by a number of examples. Automatic result verification reintegrates numerical computing into real mathematics.

Numerical mathematics has devised algorithms which deliver highly accurate and automatically verified results by applying mathematical fixed-point theorems. Their implementation requires suitable arithmetic support and powerful programming tools. The development of the XSC tools has aimed at providing these tools within the PASCAL, FORTRAN and C++ setting. The XSC tools are particularly suited for the development of numerical algorithms that deliver highly accurate and automatically verified results.

Three XSC programming systems, PASCAL-XSC, ACRITH-XSC and C-XSC, have been developed and are available for diverse platforms. FORTRAN-XSC, based on Fortran

90, is still under development. These systems have been designed for scientific computing and, in particular, to support programming of numerical algorithms with automatic result verification. They simplify programming in the area of scientific and technical computing significantly.

In a quite natural way, these systems provide a number of concepts which are vital in a contemporary programming environment: a module concept, an operator concept, functions and operators with general result type, overloading of functions, procedures and operators, dynamic arrays, subarray slices, a string concept, overloading of the assignment, of read and write, and of others. Using these concepts, data types and operators are predefined for real and complex numbers, real and complex intervals, as well as for vectors and matrices over these types. All these operators provide results with maximum accuracy. Twenty-four standard mathematical functions are provided with their generic names for real and interval arguments. The computed values are of highest accuracy. By operator overloading, a long real arithmetic (as an array of reals) and long real interval arithmetic, including corresponding elementary functions, are also available. Arithmetics for automatic differentiation and generation of Taylor coefficients are provided as part of the runtime system of the XSC tools, i.e., derivatives, Taylor coefficients, gradients, Jacobian and Hessian matrices, or enclosures of these can be computed directly out of expressions by a simple type change of operands. Using the XSC tools numerical algorithms come considerably closer to usual mathematical notation. Programs are easier to read and to write. They are easier to debug, and therefore more reliable. Many programs can be read like a technical report.

Problem solving routines with automatic result verification are available for many standard problems of numerical analysis, including linear and nonlinear systems of equations, eigenproblems, sharp evaluation of expressions, numerical quadrature, problems for differential and integral equations, etc.

A vector arithmetic coprocessor for the PC has been developed in CMOS VLSI technology. It can be used on an PC with an industry standard PCI bus. In comparison with traditional floating-point arithmetic, the chip speeds up vector and matrix operations. It computes them to full accuracy or with only one final rounding. In a dot product computation the products are accumulated into a long fixed-point register which is kept on the arithmetic unit. A special carry resolution technique is used. The chip's functionality is directly coupled to the operator symbols in vector and matrix expressions of the XSC languages. The chip is the world's first hardware implementation of the *GAMM/IMACS Proposal for Accurate Floating-Point Vector Arithmetic.*

Books on PASCAL-XSC and C-XSC have been published by Springer-Verlag. PASCAL-XSC is now available in German, English and Russian. A series of three volumes, *Toolbox for Verified Scientific Computing* is under preparation. Volume 1, containing programs and algorithms, is now available on the level of PASCAL-XSC and C-XSC. Both books have been published by Springer-Verlag. Volume 2 is scheduled to appear in 1996.

REFERENCES

Adams, E. and Kulisch U., eds. (1993) *Scientific Computing with Automatic Result Verification.* Academic Press, New York.

Alefeld, G. and Herzberger, J. (1983) *An Introduction to Interval Computations.* Academic Press, New York.

Hammer, R., Hocks, M., Kulisch, U. and Ratz, D. (1993) *Numerical Toolbox for Verified Computing I, Algorithms and Pascal-XSC Programs.* Springer, Berlin.

Hammer, R., Hocks, M., Kulisch, U. and Ratz, D. (1995) *C++ Toolbox for Verified Computing I.* Springer, Berlin.

Klatte, R., Kulisch, U., Neaga, M. Ratz, D. and Ullrich, Ch. (1991) *PASCAL-XSC-Sprachbeschreibung mit Beispielen.* Springer, Berlin. English translation, 1992. Russian translation, 1996.

Klatte, R., Kulisch, U., Neaga, M. Ratz, D. and Ullrich, Ch. (1992) *PASCAL-XSC Language Reference with Examples.* Springer, Berlin.

Klatte, R., Kulisch, U., Lawo, Ch., Rauch, M. and Wiethoff, A. (1993) *C-XSC, A C++ Class Library for Extended Scientific Computing.* Springer, Berlin.

Krämer, W., Kulisch, U. and Lohner, R. (1996) *Numerical Toolbox for Verified Computing II, Theory, Algorithms and Pascal-XSC Programs.* Springer, Berlin.

Kulisch, U. (1976) *Grundlagen des Numerischen Rechnens.* BI Wissenschaftsverlag, Mannheim.

Kulisch, U. and Miranker, W. (1981) *Computer Arithmetic in Theory and Practice.* Academic Press, New York.

Kulisch, U. and Miranker, W., eds. (1983) *A New Approach to Scientific Computation.* Academic Press, New York.

GAMM-IMACS (1993) Proposal for accurate floating-Point vector arithmetic. *Math. and Comp. in Simul.*, **35**(4). Also in *Rundbrief der GAMM*, Brief 2.

IBM (1986) *High Accuracy Arithmetic Subroutine Library (ACRITH), General Information Manual.* 3rd edition, GC33-6163-02.

IBM (1990) *High Accuracy Arithmetic—Extended Scientific Computation (ACRITH-XSC), General Information.* GC33-6461-01.

IBM (1984) *System/370 RPQ, High Accuracy Arithmetic.* SA22-7093-0..

DISCUSSION

Speaker : U. Kulisch

N. Higham : You have described the benefits of interval arithmetic. It's clear that interval arithmetic is not widely used in the scientific computing community as a whole. Why do you think this is?

U. Kulisch : Forty years of nearly exclusive use of floating-point arithmetic in scientific computing has formed and now dominates our thinking. Interval arithmetic requires a much higher level of abstraction than languages like Fortran 77, Pascal or C provide. If every single operation requires a procedure call, the user's attention and energy is forced down to the level of coding.

The development and implementation of powerful and convenient programming environments like PASCAL-XSC or ACRITH-XSC requires a large manpower of experienced and devoted scientists (about 20 man-years for each). There interval arithmetic, the elementary functions for the data types real and interval, a long real and and long real interval arithmetic including the elementary functions, vector and matrix arithmetic, differentiation and Taylor arithmetic both for real and interval data are provided by the runtime system of the compiler. All operations are called by the usual mathematical operator symbols. This releases the user's attention from coding difficulties. This means, for instance, that enclosures of a high derivative function over an interval can be computed by the same notation of the expression which is used to compute a real function value. The compiler interprets the operators according to the type specification of the data. The level of programming is really essential. It opens up a new era of numerical mathematical thinking.

G. Corliss : One reason why intervals are not widely used is the small number of interval researchers compared with workers in approximate methods. There is *plenty* of room for more workers on interval methods.

D. Lozier : I am developing a test system for mathematical functions, particularly the higher transcendental functions. I want to use software with error bounds to compute the reference values for my test system. What is the current status with respect to algorithms and software for interval special functions?

U. Kulisch : A number of higher transcendental functions have been implemented by Dr. F. Blomquist for real and interval data types for the decimal arithmetic of our PASCAL-XSC system. This system uses BCD arithmetic, so the functions can only be used to check for an accuracy of 13 decimal digits. More accurate elementary functions (14 hex digits) have been developed, implemented, and are available in the IBM program product ACRITH-XSC by Dr. K. Braune and Dr. W. Krämer at Karlsruhe, even for complex and complex interval data formats. Dr. Blomquist is now developing and implementing higher transcendental functions (the error function, the gamma function, Dawson's integral, etc.) for the double data format of IEEE arithmetic standard 754. All persons mentioned can be reached via my Institute.

W.V. Snyder : Do you or your colleagues have plans to extend work on interval computation libraries to Fortran?

U. Kulisch : Yes, we have already been working on an interval computation library for Fortran at my Institute. The group leader was Wolfgang V. Walter. Two years ago he became Professor at the Technische Universität Dresden. This has delayed the work a little. I hope that we will jointly finish the project and that sometime something like Fortran-XSC will be available.