08021 Abstracts Collection
Numerical Validation in Current Hardware Architectures
— Dagstuhl Seminar —

Wolfram Luther[1], Annie Cuyt[2], Walter Krämer[3] and Peter Markstein[4]

[1] Univ. Duisburg-Essen, DE
luther@inf.uni-due.de
[2] U Antwerp, BE
annie.cuyt@ua.ac.be
[3] BU Wuppertal, DE
[4] HP Labs, Palo Alto, US
peter@markstein.org

Abstract. From 06.01. to 11.01.2008, the Dagstuhl Seminar 08021 "Numerical Validation in Current Hardware Architectures" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Computer arithmetic, arbitrary precision, floating-point arithmetic standardization, language support, reliable libraries,high-precision special functions, reliablealgorithms, reliable floating-point and interval computing on different platforms

08021 Summary – Numerical Validation in Current Hardware Architectures

Numerical validation in current hardware architectures - From embedded system to high-end computational grids.

Keywords: Computer arithmetic, arbitrary precision, floating-point arithmetic standardization, language support, reliable libraries, high-precision special functions, reliablealgorithms, reliable floating-point and interval computing on different platforms

Joint work of: Cuyt, Annie; Krämer, Walter; Luther, Wolfram; Markstein, Peter

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2008/1433

SmartMOBILE and Its Applications to Biomechanics

Ekaterina Auer (Universität Duisburg-Essen, D)

To automatize modeling and simulation of mechanical systems for industry and research, a number of tools were developed, among which the program MOBILE plays a considerable role. However, such tools cannot guarantee the correctness of their results. Verified methods, for example, intervals and Taylor models, can be used there to avoid numerical errors. In this talk, we present a modeling and simulation tool SmartMOBILE based on MOBILE, which provides results guaranteed to be correct within the constraints of the considered model of a mechanical system. The use of verified methods there allows us additionally to take into account the uncertainty in measurements and study its influence on simulation. Moreover, the facility of verified sensitivity computation has been recently added to SmartMOBILE, which provides a means for analyzing a model with respect to its parameters. We focus on applications of SmartMOBILE to biomechanics and the problem of identification of muscle activation profiles. In particular, we analyze the given rough MOBILE model and the influence of small changes in its parameters on simulation. The model's sensitivity to several of its parameters is obtained based on the adjustment of the extended version of the verified initial value problem solver ValEncIA-IVP. Further steps toward verification of the whole muscle activation process are outlined. In general, this talk gives an insight into how various guaranteed methods can be applied to mechanical modeling and simulation and point out the advantages and shortcomings of these techniques with respect to this application area.

Numerical Verification Assessment in Computational Biomechanics

Ekaterina Auer (Universität Duisburg-Essen, D)

In this paper, we present several aspects of the recent project PROREOP, in which a new prognosis system is developed for optimizing patient-specific preoperative surgical planning for the human skeletal system. We address verification and validation assessment in PROREOP with special emphasis on numerical accuracy and performance. To assess numerical accuracy, we propose to employ graded instruments, including accuracy tests and error analysis. The use of such instruments is exemplified for the process of accurate femur reconstruction. Moreover, we show how to verify the simulation results and take into account measurement uncertainties for a part of this process using tools and techniques developed in the project TellHIM&S.

Keywords: Numerical verification assessment, validation, uncertainty, result verification

Joint work of: Auer, Ekaterina; Luther, Wolfram

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1437

Improving the Performance of a Verified Linear System Solver Using Optimized Libraries and Parallel Computation

Gerd Bohlender (Universität Karlsruhe, D)

A parallel version of the self-verified method for solving linear systems was presented on PARA and VECPAR conferences in 2006. In this research we propose improvements aiming at a better performance. The idea is to implement an algorithm that uses technologies as MPI communication primitives associated to libraries as LAPACK, BLAS and C-XSC, aiming to provide both self-verification and speed-up at the same time. The algorithms should find an enclosure even for very ill-conditioned problems. In this scenario, a parallel version of a self-verified solver for dense linear systems appears to be essential in order to solve bigger problems. Moreover, the major goal of this research is to provide a free, fast, reliable and accurate solver for dense linear systems.

Keywords: Linear systems, result verification, parallel computing

Joint work of: Kolberg, Mariana; Bohlender, Gerd; Claudio, Dalcidio

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1438

A Note on Some Applications of Interval Arithmetic in Hierarchical Solid Modeling

Eva Dyllong (Universität Duisburg-Essen, D)

Techniques of reliable computing like interval arithmetic can be used to guarantee a reliable solution even in the presence of numerical round-off errors. The need to trace bounds for the error function separately can be eliminated using these techniques. In this talk, we focus on some demonstrations how the techniques and algorithms of reliable computing can be applied to the construction and further processing of hierarchical solid representations using the octree model as an example.

An octree is a common hierarchical data structure to represent 3D geometrical objects in solid modeling systems or to reconstruct a real scene. The solid representation is based on recursive cell decompositions of the space. Unfortunately, the data structure may require a large amount of memory when it uses a set of very small cubic nodes to approximate a solid.

In this talk, we present a novel generalization of the octree model created from a CSG object that uses interval arithmetic and allows us to extend the tests for classifying points in space as inside, on the boundary or outside the object to handle whole sections of the space at once. Tree nodes with additional information about relevant parts of the CSG object are introduced in order to reduce the depth of the required subdivision.

Furthermore, this talk is concerned with interval-based algorithms for reliable proximity queries between the extended octrees and with further processing of the structure. We conclude the talk with some examples of implementations.

Keywords: Reliable solid modeling, hierarchical data structure

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1440

Error bounds for rational matrix functions applied to a vector

Andreas Frommer (Universität Wuppertal, D)

Let A be a square matrix and r a rational function. We are interested in computing approximations to the action of the matrix function $r(A)$ on some vector b, i.e. we want to compute $r(A)b$. Since A is usually sparse and large, we consider a Krylov subspace framework. The function r is expressed as a partial fraction expansion, and for each partial term we compute the FOM iterate of a given degree. These iterates are then combined to yield the searched for approximation for $r(A)b$. An important question is whether it is possible to assess the error of the approximation by giving upper and lower bounds. We do so in the case where A is hermitian by using the fact that all residuals of the FOM iterates are then colinear. Bounds for the error can then be obtained by the minimization and maximization of an a posteriorily computable new rational function over an interval determined by the spectrum of A. Simple interval arithmetic optimzation methods are perfectly suitable to solve these optimization problems and we show that the error bounds obtained can very good. This is particularly so if the rational function is itself an approximation to the exponential, a computation which arises in important applications like exponential integration methods, We also demonstrate the efficiency of our approach for the matrix sign function used in lattice quantum chromodynamics.

Keywords: Matrix functions, error bounds, iterative methods, Krylov subspaces, matrix exponential, sign function

Joint work of: Frommer, Andreas; Simoncini, Valeria

Extending the Range of C-XSC: Some Tools and Applications for the use in Parallel and other Environments

Markus Grimmer (Universität Wuppertal, D)

There is a broad range of packages and libraries for verified numerical computation. C-XSC is a library combining one of the most extensive sets of functions and operations on the one hand with a wide range of applications and special features on the other hand.

As such it is an important task both to make use of its existing capabilities in applications and to develop further extensions giving access to additional areas and environments.

In this talk, we present some examples of extensions for C-XSC that have been developed lately. Among these are extensions that give access to further hardware and software environments as well as applications making use of these possibilities.

Software libraries for interval computation always imply great computation effort: One way to reduce computation times is the development of parallel methods to make use of parallel hardware. For this, it is important that the features and data types of the used library can be easily used in parallel programs. An MPI package for C-XSC data types allows to easily use C-XSC in parallel programs without bothering about the internal structure of data types. Another extension of C-XSC, the C-XSC Taylor arithmetic, is also covered by the MPI package. Parallel verified linear system solvers based on the package are available as well, and further development has been and is being done to integrate more efficient methods for interval linear system solution.

One application making use of the mentioned extensions is a parallel verified Fredholm integral equation solver. Some results are given to demonstrate the reduction of computation time and, at the same time, the accuracy gain that can be obtained using the increased computation power. Naturally, hardware interval support would offer still more possibilities towards optimal performance of verified numerical software.

Another possibility to extend the range of C-XSC is to make results available for further computations in other software environments as, for example, computer algebra packages. An example of this is presented for the Maple interval package intpakX. This kind of interfaces also allows the user to get access to further platforms like operating systems, compilers or even hardware.

References:

[1] ALiCEnext: http://www.alicenext.uni-wuppertal.de.

[2] Blomquist, F.; Hofschuster, W.; Kraemer, W.: Real and Complex Taylor Arithmetic in C-XSC. Preprint BUW-WRSWT 2005/4, University of Wuppertal, 2005.

[3] Grimmer, M.; Kraemer, W.: An MPI Extension for Verified Numerical Computations in Parallel Environments. In: Int. Conf. on Scientific Computing (CSC'07, Worldcomp'07) Las Vegas, June 25-28, 2007, Proceedings pp. 111-117, Arabnia et al. (eds.), 2007.

[4] Grimmer, M.: An MPI Extension for the Use of C-XSC in Parallel Environments. Preprint BUW-WRSWT 2005/3, University of Wuppertal, 2005.

[5] Grimmer, M.: Selbstverifizierende mathematische Softwarewerkzeuge im High Performance Computing. Dissertation, Logos Verlag, Berlin, 2007.

[6] Grimmer, M.: Interval Arithmetic in Maple with intpakX. In: PAMM - Proceedings in Applied Mathematics and Mechanics, Vol. 2, Nr. 1, p. 442-443, Wiley-InterScience, 2003.

[7] Hofschuster, W.; Kraemer, W.: C-XSC 2.0: A C++ Library for Extended Scientific Computing. Numerical Software with Result Verification, Lecture Notes in Computer Science, Volume 2991/2004, Springer-Verlag, Heidelberg, pp. 15 - 35, 2004.

[8] Klein, W.: Enclosure Methods for Linear and Nonlinear Systems of Fredholm Integral Equations of the Second Kind. In: Adams, Kulisch: Scientific Computing with Result Verification, Academic Press, 1993.

Keywords: C-XSC, Integral Equations, Interval Arithmetic, Maple, MPI, Parallel Environment, Taylor Arithmetic, Verified Linear System Solver

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1441

C-XSC: Highlights and new developments

Werner Hofschuster (Universität Wuppertal, D)

C-XSC is one of the most sophisticated software libraries to facilitate the development of reliable numerical methods. The source code of this C++ class library (open source) is available free of charge. A lot of predefined numeric data types (real, interval, complex, cinterval, dotprecision, idotprecision, cdotprecision, cidotprecision, and correspondig matrix/vector types) as well as corresponding arithmetic operators of maximum accuracy are provided. Additionally some kinds of multiple precision data types are available: `l_real`, `l_interval`, `L_real`, `L_interval`, `DotK`, `IDotK`, ...

An extensive set of elementary mathematical functions for real and complex interval arguments of high accuracy are available. This is true not only for the basic interval types but also for the so called staggered correction (multiple precision) formats. These formats rely heavily on the possibility to compute dot products of vectors with floating point entries without loss of information (the computed result is guaranteed to be error free i.e. no rounding errors occur). This C-XSC operation is also used to realize the arithmetic operations in all the matrix/vector spaces as semimorphisms, i.e. with only one final rounding. Up to now this basic operation is realized by software. There is no hardware suppport. Thus, the performance is still far away from being optimal. There are several papers showing that hardware support would be possible by increasing the complexity of current processors only a little bit. Thus, the community should insist on getting this operation in hardware. The benefits would be great: best possible matrix/vector operations outperforming traditional, not accurate dot product computations via loops.

C-XSC also offers a lot of higher numerical routines and additional software always providing validated results: global optimization, linear and nonlinear systems with interval entries, parametric linear systems, numerical integration, Cauchy principal values of improper integrals, multiple-precision computations, computations with very wide exponent range, automatic differentiation, Taylor

coefficients of analytic functions, Fredholm integral equations, MPI extension for the use of C-XSC in parallel environments, ...

A first version of a new online api documentation generated by the documentation system Doxygen is available. New high performance linear system solvers based on BLAS and a (modified) DotK algorithm. The prototype of an efficient parallel version for large matrices is also available. It has been tested successfully on the supercomputer ALiCEnext as well as on a cluster of workstations at the University of Wuppertal. Further development/discussions: Containment sets? Parallel solvers for sparse matrices. Simplified output (like IntLab)?, complete set of test cases, ... In the talk we will emphasize the most important features of C-XSC by small programs or code snippets. Some time measurements will also be given.

We feel it is appropriate to thank all friends/persons (see http://www.math.uni-wuppertal.de/~xsc/xsc/history.html) which have contributed and/or which are still contributing to the development of C-XSC. Getting verified numerical results is a challenging and exciting task.

Refernces:

[1]Download: http://www.xsc.de/ http://www.math.uni-wuppertal.de/ xsc/

[2]Blomquist, F.; Hofschuster, W.; Krämer, W.; Neher, M.: Complex Interval Functions in C-XSC Preprint 2005/2, Universitaet Wuppertal, 2005

[3] Braune, K.: Hochgenaue Standardfunktionen fuer reelle und komplexe Punkte und Intervalle in beliebigen Gleitpunktrastern. Dissertation, Universitaet Karlsruhe, 1987.

[4] Grimmer, M.: An MPI Extension for the Use of C-XSC in Parallel Environments Preprint 2005/3, Universitaet Wuppertal, 2005.

[5] Grimmer, M.: Selbstverifizierende mathematische Softwarewerkzeuge im High Performance Computing Dissertation, Universitaet Wuppertal, 2007.

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

[7] Hofschuster, W.: Zur Berechnung von Funktionswerteinschliessungen bei speziellen Funktionen der mathematischen Physik Dissertation, Universitaet Karlsruhe, 2000

[8] Hofschuster, W.; Krämer, W.: C-XSC 2.0: A C++ Library for Extended Scientific Computing. Numerical Software with Result Verification, Lecture Notes in Computer Science, Volume 2991/2004, Springer-Verlag, Heidelberg, pp. 15 - 35, 2004.

[9] Klatte, Kulisch, Wiethoff, Lawo, Rauch: "C-XSC - A C++ Class Library for Extended Scientific Computing", Springer-Verlag, Heidelberg, 1993. Due to the C++ standardization (1998) and dramatic changes in C++ compilers over the last years this documentation describes no longer the actual C-XSC environment. Please refer to more accurate documentation (e.g.[1]) available from the web site of our research group: http://www.math.uni-wuppertal.de/ xsc/

[10] Kirchner, R., Kulisch, U.: Hardware Support for Interval Arithmetic. Reliable Computing, Volume 12, Number 3, June 2006 , pp. 225-237(13).

[11] Kraemer, W.: Inverse Standardfunktionen fuer reelle und komplexe Intervallargumente mit a priori Fehlerabschaetzungen fuer beliebige Datenformate. Dissertation, Universitaet Karlsruhe, 1987.

[12] Kulisch, U.: Computer Arithmetic and Validity - Theory, Implementation. To appear.

[13] Lerch, M.; Tischler, G.; Wolff von Gudenberg, J.; Hofschuster, W; Krämer, W.: filib++, a Fast Interval Library Supporting Containment Computations. ACM TOMS, volume 32, number 2, pp. 299-324, 2006.

[14] Neher, M.: Complex standard functions and their implementation in the CoStLy library. ACM TOMS 33 (2007), 20-46.

[15] Ogita, T.; Rump, S. M. and Oishi, S.: Accurate sum and dot product. SIAM Journal on Scientific Computing, 26:6, 2005.

[16] Wedner, S.: Verifizierte Bestimmung singulärer Integrale - Quadratur und Kubatur Dissertation, Universitaet Karlsruhe, 2000.

[17] Wiethoff, A.: Verifizierte globale Optimierung auf Parallelrechnern Dissertation, Universitaet Karlsruhe, 1997.

[18] Zimmer, M.: Laufzeiteffiziente, parallele Loeser fuer lineare Intervallgleichungssysteme in C-XSC, Master thesis, University of Wuppertal, 2007.

MSC Subject Classifications: 68N30, 68N19, 65F99, 65G20, 65G30

Keywords: C-XSC, mathematical software, reliable libraries, language support

Joint work of: Hofschuster, Werner; Krämer, Walter

C-XSC and Closely Related Software Packages

Werner Hofschuster (Universität Wuppertal, D)

C-XSC and Closely Related Software Packages

Keywords: Mathematical software, reliable computing, C-XSC, CoStLy, ACETAF

Joint work of: Hofschuster, Werner; Krämer, Walter; Neher, Markus

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1442

Robustness of Boolean operations on subdivision-surface models

Di Jiang (Université de Montréal, CA)

This work was presented in two parts at Dagstuhl seminar 08021. The two presentations described work in progress, including a "backward bound" for a combined backward/forward error analysis for the problem mentioned in the title.

We seek rigorous proofs that representations of computed sets, produced by algorithms to compute Boolean operations, are well formed, and that the algorithms are correct. Such proofs should eventually take account of the use of finite-precision arithmetic, although the proofs presented here do not. The representations studied are based on subdivision surfaces. Such representations are being used more and more frequently in place of trimmed NURBS representations, and the robustness analysis for these new representations is simpler than for trimmed NURBS. The particular subdivision-surface representation used is based on the Loop subdivision scheme. The analysis is broken into three parts. First, it is established that the input operands are well-formed two-dimensional manifolds without boundary. This can be done with existing methods.

Secondly, we introduce the so-called "limit mesh", and view the limit meshes corresponding to the input sets as defining an approximate problem in the sense of a backward error analysis. The presentations mentioned above described a proof of the corresponding error bound. The third part of the analysis corresponds to the "forward bound": this remains to be done.

Keywords: Robustness, finite-precision arithmetic, Boolean operations, subdivision surfaces

Joint work of: Jiang, Di; Stewart, Neil

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1443

Issues for General Users of Automatically Verified Optimization Software: What Does the Answer Mean?

R. Baker Kearfott (Univ. of Louisiana - Lafayette, USA)

With additional experience supporting users of GlobSol, several issues have become apparent. In particular, we state that verified global optimization packages "cannot lie" in the sense that the software always outputs all answers. However, such software has tolerances dealing with the fineness of the domain resolution.

Because of these tolerances, the answers presented to the user can be misleading. Documentation for the software must therefore carefully define how users should interpret the answers. Several examples will be given. These examples are part of an updated User Guide for GlobSol.

Keywords: Global optimization, automatic result verification

GlobSol User Guide

R. Baker Kearfott (Univ. of Louisiana - Lafayette, USA)

We explain installation and use of the GlobSol package for mathematically rigorous bounds on all solutions to constrained and unconstrained global optimization problems, as well as nonlinear systems of equations.

This document should be of use both to people with optimization problems to solve and to people incorporating GlobSol's components into other systems or providing interfaces to GlobSol.

Keywords: Global optimization, automatic result verification

Efficient 16-bit Floating-Point Interval Processor for Embedded Systems

Michel Kieffer (CWRS - Supélec - Université Paris-Sud, F)

In the last ten years, interval techniques have allowed original solutions for many problems in engineering to be proposed. One of the main features of interval techniques is their ability to provide *guaranteed* results, *i.e.*, with a verified accuracy or which are numerically *proved*. Consider for example, a bounded-error parameter estimation problem: the value of some parameter vector has to be estimated from measured data using a given model structure and bounded measurement errors. In such a context, one may obtain a set which can be proved to contain all values of the parameter vector that are consistent with the model structure, the measured data, and the hypotheses on the noise. Nevertheless, the application of interval techniques in embedded real-time applications is far less developed. The lack of efficient interval hardware support may be a reason for this slower development.

Hardware implementations of interval arithmetic have been mentioned twenty years ago. Extension of existing hardware platforms have been proposed. Nevertheless, chip builders were not yet convinced of the usefulness of performing specific adaptation of chips to implement interval analysis. This is why interval analysis is mainly performed by software implementations on general-purpose processors. Interval computations are however quite inefficiently performed on such processors, since the recurrent rounding mode switchings required by interval computations results in recurrent flushes of the processor pipeline. This specific problem led people to study and design dedicated floating-point units (FPU) well suited to double rounding modes (towards $-\infty$ and towards $+\infty$). Moreover, in many applications, 32-bit FPU are oversized. Measurements, corrupted by errors, do not require to be processed with such an accuracy and in many cases, smaller FPU with reduced precision may fit the application constraints and provide a satisfying accuracy. Thus, for example, 16-bit floating-point computations is an efficient way to tackle both accuracy and dynamic problems encountered in signal and image processing, for filtering and convolution-based algorithms.

This talk introduces 16-bit floating-point arithmetic adapted to interval computations. The main idea is inspired by [Kolla99], which proposed to implement two 32-bit FPU on the 64-bit FPU of a general-purpose processor. Here, similarly, noticing that a 16-bit FPU is smaller than a 32-bit FPU, two 16-bit FPU (managing the two rounding modes required for interval computations) are shown not being much bigger than a single 32-bit FPU. The main advantage

is that no rounding mode switching is required, preventing them from flushing the processor pipeline. The implementation of such a 16-bit FPU is performed on the FPGA based NIOS-II soft processor, which allows instructions to be added to its instruction set. Customizable processors represent an opportunity to propose efficient and low-cost on-chip interval applications which may be used in embedded applications.

To compare the performance of 16 and 32-bit FPUs, an example of source localization using a network of acoustic or electromagnetic sensors is considered. In such network of sensors, power consumption and computational complexity are strong constraints when one is concerned with the increase of operability and autonomy. Distributed interval constraint propagation has been proposed as an efficient and low-complexity solution for source localization using a network of wireless sensors.

The talk recalls first the distributed source localization problems and sketches the solutions based on interval analysis. Then, the architecture of the 16-bit FPU is presented. Attention is paid to accuracy and dynamic range. Results provided by a 32-bit FPU are compared to those obtained with two 16-bit FPU on realistic simulated data. The hardware implementation on the three targeted architectures (Pentium4, Pentium 4-M, and NIOS-II) and provides benchmarks for execution time and energy consumption.

Keywords: Source localization, Interval analysis, Embedded applications, 16 bits floating-point Processor, FPGA

Joint work of: Piskorski, Stéphane; Kieffer, Michel; Lacassagne, Lionel; Etiemble, Daniel

See also: S. Pikorski, L. Lacassagne, M. Kieffer and D. Etiemble, Efficient 16-bit floating point interval processor for embedded systems and applications, 2006, Duisburg, SCAN 2006

Distributed parameter and state estimation in a network of sensors

Michel Kieffer (CWRS - Supélec - Université Paris-Sud, F)

In this paper, we have considered distributed bounded-error state estimation applied to the problem of source tracking with a network of wireless sensors. Estimation is performed in a distributed context, *i.e.*, each sensor has only a limited amount of measurements available. A guaranteed set estimator is put at work. At each time instant, any sensor of the node has its own set estimate of the location of the source.

Keywords: Parameter estimation, state estimation, bounded errors, nonlinear estimation

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1444

A Modified Staggered Correction Arithmetic with Enhanced Accuracy and Very Wide Exponent Range

Walter Krämer (Universität Wuppertal, D)

AMS classification: 65G20, 65G30, 65Y99, 37M99, 30-04

The staggered correction arithmetic is based on accurate scalar product computations of floating point vectors.

A staggered correction number is given as the (exact) sum of the components of a list of floating-point numbers (components of a floating-point vector).

Thus the maximum precision of arithmetical operations depends strongly on the exponent range of the floating-point screen. Staggered correction numbers close to the underflow range are typically not as accurate as numbers with larger exponents. For the IEEE double precision format the range is about 4.9 10E-324 < |double| < 1.7 10E+308. Thus, up to 630 (decimal) mantissa digits may be handled using staggerd precision variables.

To get high accuracy in numerical calculations (intermediate) underflow and/or overflow situations must be avoided, i.e. an appropriate scaling of the operands of arithmetic operations is necessary. To this end we represent our modified staggered number as a pair (e, x). We refer to such pairs as extended staggered numbers.

The integer e denotes an exponent with repect to the base 2 and x denotes an ordinary staggerd number (i.e. a vector of floating-point numbers). The value v of the modified staggered variable (e,x) is v = (e,x) := 2**e * sum(x). Here, sum(x) means the exact sum of all floating-point components of the staggered variabe x. The pair-representation allows the handling of very small and very large numbers: 1.3E-487564, 4.1E9999999, or numbers with even larger exponents may be used in numerical calculations. We know several test cases where multiple-precision calculations using computer algebra packages like Maple and/or Mathematica fail whereas our extended staggered software returns the expected result.

The new package offers real interval and complex interval arithmetic operations and also a rather complete set of mathematical functions for the new data types L_interval (extended real staggered intervals) and L_cinterval (extended complex staggerd intervals). The trigonometric functions, the inverse trigonometric functions, the hyperbolic and the inverse hyperbolic functions as well as several other functions are provided for (rectangular) extended complex staggered intervals.

Some details of the arithmetic operations as well as some details on the implementation of elementary mathematical functions will be discussed in the talk.

Applications (logistic equation, limit calculations) will be presented to show the superior behavior of the new arithmetic over the more traditional one. We will also discuss hardware requirements to get the staggered arithmetic working extremely fast.

The source code of the new package will be distributed under the GNU general public license. Up to now it is an independent supplement to our C-XSC library. The source code will be made available within the next weeks.

See http://www.math.uni-wuppertal.de/~xsc/xsc/cxsc_software.html

References:

[1] Auzinger, W. and Stetter H.J.: Accurate Arithmetic Results for Decimal Data on Non-Decimal Computers. Computing 35, 141-151, 1985.

[2] Blomquist, F.; Hofschuster, W.; Kraemer, W.:
Realisierung der hyperbolischen Cotangens-Funktion in einer Staggered Correction Intervallarithmetik in C-XSC Preprint BUW-WRSWT 2004/3, Universitaet Wuppertal, 2004.

[3] Blomquist, F.; Hofschuster, W.; Kraemer, W.:
Vermeidung von Ueber- und Unterlauf und Verbesserung der Genauigkeit bei reeller und komplexer staggered Intervall-Arithmetik mit Hilfe der C-XSC Klassen L_interval, L_cinterval. Preprint BUW-WRSWT 2007/8, Universitaet Wuppertal, 2007.

[4] Hofschuster, W.; Kraemer, W.:
C-XSC 2.0: A C++ Library for Extended Scientific Computing.
Numerical Software with Result Verification, Lecture Notes in Computer Science, Volume 2991/2004, Springer-Verlag, Heidelberg, pp. 15 - 35, 2004.

[5] Klatte, Kulisch, Wiethoff, Lawo, Rauch:
C-XSC - A C++ Class Library for Extended Scientific Computing, Springer-Verlag, Heidelberg, 1993.

[6] Kraemer, W.: Mehrfachgenaue reelle und intervallmaessige Staggered-Correction Arithmetik mit zugehoerigen Standardfunktionen, Report of the Institut fuer Angewandte Mathematik, Universitaet Karlsruhe, Germany, 1988.

[7] Lohner, R.: Interval arithmetic in staggered correction format.
In: Adams, E., Kulisch, U.(Eds): Scientific Computing with Automatic Result Verification. Academic Press, San Dieago, pp 301-321, 1993.

[8] Stetter, H.J.: Staggered Correction Representation, a Feasible Approach to Dynamic Precision, Proceedings of the Symposium on Scientific Software, edited by Cai, Fosdick, Huang, China University of Science and Technology Press, Beijing, China, 1989.

Keywords: Staggered correction, multiple precision, C-XSC, accurate dot product, wide exponent range, logistic equation, complex functions

Joint work of: Krämer, Walter; Blomquist, Frithjof; Hofschuster, Werner

A Modified Staggered Correction Arithmetic with Enhanced Accuracy and Very Wide Exponent Range

Walter Krämer (Universität Wuppertal, D)

A so called staggered precision arithmetic is a special kind of a multiple precision arithmetic based on the underlying floating point data format (typically IEEE double format) and fast floating point operations as well as exact dot product computations.

Due to floating point limitations it is not an arbitrary precision arithmetic. However, it typically allows computations using several hundred mantissa digits. A set of new modified staggered arithmetics for real and complex data as well as for real interval and complex interval data with very wide exponent range is presented. Some applications show the increased accuracy of computed results compared to ordinary staggered interval computations. The very wide exponent range of the new arithmetic operations allows computations far beyond the IEEE data formats.

The new arithmetics would be extremly fast, if an exact dot product was available in hardware (the fused accumulate and add instruction is only one step in this direction).

Keywords: Staggered correction, multiple precision, C-XSC, interval computation, wide exponent range, reliable numerical computations, complex interval functions

Joint work of: Blomquist, Frithjof; Hofschuster, Werner; Krämer, Walter

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1445

A proposed standard for interval arithmetic and complete arithmetic

Ulrich Kulisch (Universität Karlsruhe, D)

Based on a new book entitled "Computer Arithmetic and Validity" (to be published this year) the talk will discuss some background material that led to a proposal of the IFIP Working Group 2.5 to add Interval Arithmetic and Complete Arithmetic to the proposed IEEE Arithmetic Standard P754.

Complete Interval Arithmetic and its Implementation on the Computer

Ulrich Kulisch (Universität Karlsruhe, D)

A Complete Interval Arithmetic and its Implementation is discussed.

Keywords: Interval Arithmetic, implementation

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1446

Extended precision on the CELL processor

Jean-Luc Lamotte (Université Pierre et Marie Curie, F)

The CELL processor, jointly developed by a Sony, Toshiba, and IBM, provides great potential for scientific computing with a peak performance in single precision over 200 Gflop/s. But, this performance is obtained with an SIMD processor which is not fully IEEE compliant. For example, the single precision floating operations are performed with the rounding mode toward zero. There is no denormalized number, etc. The first part of the talk is devoted to the CELL architecture.

In the second part, we will explain how to implement the double working precision library, named double-single on the SPEs (Synergistic Processing Element) which are the workhorse processors of the CELL. The approach is close to those used in [2] for quad-double precision arithmetic. Firstly, algorithms based on error free transformations for the operators (+,-,*,/) are proposed for the rounding mode toward zero. We also prove their exactitude and we provide error bounds on the precision of the double-single floating-point arithmetic.

The third part focuses on implementation on the SIMD processor, taking into account the advantages of the characteristics of the SPE processor, among which the fully pipelined set of instructions in single precision and the FMA (Fused Multiplier-Add) operator are the most important. We have managed to implement the error-free transformations very efficiently. The performances of our implementation are presented. The theoretical peak performance of the library is much less than the performance of the double precision of the machine, which is about 2.7 Gflop/s in comparison with the 14.4 Gflop/s of the double precision. The results of our test show that it is not so bad. When 8 SPE are used to compute operations on very large vectors, the performance of the double-single and the true double floating point numbers are nearly equal.

In the future, with the same approach, we will continue our work to quad-single precision. With the next CELL processor which will provide a double precision fully-pipelined SIMD processor on SPE, we will modify slightly our code to reach double-double precision and the quad double precision.

References:

[1] Dekker, T. J.: 1971, A floating-point technique for extending the available precision. Numer. Math. 18, 224-242.

[2] Gschwind, M., H. P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki: 2006, 'Synergistic Processing in Cell's Multicore Architecture'. IEEE Micro 26(2), 10-24.

[3] Hida, Y., X. S. Li, and D. H. Bailey: 2001, 'Algorithms for Quad-Double Precision Floating Point Arithmetic'. In: Proc. 15th IEEE Symposium on Com-

puter Arithmetic. pp. 155-162, IEEE Computer Society Press, Los Alamitos, CA, USA.

[4] Jacobi, C., H.-J. Oh, K. D. Tran, S. R. Cottier, B. W. Michael, H. Nishikawa, Y. Totsuka, T. Namatame, and N. Yano: 2005, The Vector Floating-Point Unit in a Synergistic Processor Element of a CELL Processor. In: ARITH '05: Proceedings of the 17th IEEE Symposium on Computer Arithmetic. Washington, DC, USA, pp. 59-67, IEEE Computer Society.

[5] Kahle, J. A., M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy: 2005, Introduction to the cell multiprocessor. IBM J. Res. Dev. 49(4/5), 589-604.

[6] Knuth, D. E.: 1998, The Art of Computer Programming, Volume 2, Seminumerical Algorithms. Reading, MA, USA: Addison-Wesley, third edition.

[7] Priest, D. M.: 1992, On Properties of Floating Point Arithmetics: Numerical Stability and the Cost of Accurate Computations. Ph.D. thesis, Mathematics Department, University of California, Berkeley, CA, USA.
ftp: //ftp.icsi.berkeley.edu/pub/theory/priest-thesis.ps.Z.

[8] Nguyen, H. D.: 2007, Calcul precis et efficace sur le processeur CELL'. Master report, LIP6, UPMC (P. and M. Curie University), Paris, France
http://www-pequan.lip6.fr/ graillat/papers/rapport_Diep.pdf.

[9] Williams, S., J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick: 2006, 'The potential of the cell processor for scientic computing'. In: CF '06:

Proceedings of the 3rd conference on Computing frontiers. New York, NY, USA, pp. 9-20, ACM Press.

Keywords: Extended precision, CELL processor

Joint work of: Nguyen Hong, Diep; Graillat, Stef; Lamotte, Jean-Luc

Verified Computation of Spherical t-Designs - Challenges, Approaches, and Current Limitations

Bruno Lang (Universität Wuppertal, D)

We present result-verifying methods for computing spherical t-designs. A spherical t-design is a set of N points on the unit sphere, such that the integral of each polynomial of degree $\leq t$ over the sphere is obtained by just adding the polynomial values at these points. We treat two particular cases, which require completely different approaches and pose different computational challenges, also w.r.t. the underlying hardware. In the "small t" case we are considering cases (t, N), for which it is not known whether an N-point spherical t-design does exist. Here we were able to prove that a 7-point spherical 3-design does not exist. In the "large t" case we are constructing $(t+1)^2$-point spherical t-designs, where the number of points is one order of magnitude smaller than in a known approach, and only by a factor of 4 above a theoretical lower bound. We were able to verify the existence of such t-designs up to $t = 60$.

Keywords: Spherical t-designs, verification of existence

Joint work of: Lang, Bruno; Beelitz, Thomas; Chen, Xiaojun; Frommer, Andreas; Ueberholz, Peer; Willems, Paul

Searching for Some Worst Cases for the Correct Rounding of the Power Functions in Double Precision

Vincent Lefèvre (ENS - Lyon, F)

I'll present an efficient algorithm to search for the worst cases of a numerically regular unary function. Obtaining complete results for the power function x^y is out of reach. But interesting partial results could be obtained. I'll briefly talk about the case of the integer power function x^n (see Jean-Michel Muller's presentation) and give the current results. I'll also present the application to the detection of the exact cases of x^y (joint work with Christoph Lauter).

Keywords: Floating-point arithmetic, correct rounding, power function

Compensated Horner scheme in k times the working precision

Nicolas Louvet (ENS - Lyon, F)

The backward stability of the Horner scheme when evaluating a given polynomial p at the point x justifies its practical interest. Nevertheless, the computed result can be arbitrarily less accurate than the working precision u when the evaluation of $p(x)$ is ill-conditioned. This is the case for example in the neighborhood of multiple roots where all the digits or even the order of the computed value of $p(x)$ can be false. Several techniques and softwares intend to improve the accuracy of results computed in floating point arithmetic. When an IEEE-754 floating point arithmetic is available, "double-double" and "quad-double" software libraries are effective solutions to simulate respectively twice our four times the working precision [1].

In [2], we have already described a compensated Horner scheme: with this scheme, the result is of the same quality as if computed in doubled working precision. We present here another compensated algorithm that computes an approximate r of $p(x)$ of the same quality as if computed in k times the working precision ($k \geq 2$). More exactly, this means that r satisfies

$$\frac{|r - p(x)|}{|p(x)|} \leq u + O(u^k)(p, x), \tag{1}$$

where (p, x) is the classical condition number that describes the sensitivity of the polynomial evaluation. This algorithm only requires an IEEE-754 floating point arithmetic with rounding to the nearest.

Our main tool to improve the accuracy of the computed result is an "error-free transformation" [3] (EFT) for the polynomial evaluation with the Horner scheme. We prove that recursive application of this EFT allows us to compute an approximate that satisfies the previous inequality.

Experimental results show that the time penalty due to the improvement of the accuracy is very reasonable for $k \leq 4$. In particular, our routine runs about 40% faster than the corresponding routine based on the quad-double library. This justifies the practical interest of the method when only a small increase of the working precision is needed.

References:

[1] Y. Hida, X. S. Li, and D. H. Bailey, Algorithms for quad-double precision floating point arithmetic.

In Proc. 15th IEEE Symposium on Computer Arithmetic, pp. 155–162.

IEEE Computer Society, 2001.

[2] P. Langlois and N. Louvet, "How to ensure a faithful polynomial evaluation with the compensated Horner algorithm?".

In Proc. 15th IEEE Symposium on Computer Arithmetic, pp. 141–149.

IEEE Computer Society, Jun. 2007, pp. 141–149.

[3] T. Ogita, S. M. Rump, and S. Oishi, Accurate sum and dot product.

SIAM J. Sci. Comp., 26(6):1955–1988, 2005.

Joint work of: Langlois, Philippe; Louvet, Nicolas

Verification and Validation Assessment in Computational Biomechanics

Wolfram Luther (Universität Duisburg-Essen, D)

In a recent project, PROREOP (development of a new prognosis system to optimize patient-specific preoperative surgical planning for the human skeletal system), we address V&V assessment with special emphasis on numerical accuracy and performance.

Keywords: Verification, validation, assessment

A Note on Solving Problem 7 of the SIAM 100-Digit Challenge Using C-XSC

Mariana Lüderitz Kolberg (Faculdade de Informática PUCRS - Porto Alegre, BR)

C-XSC is a powerful C++ class library which simplifies the development of selfverifying numerical software.

But C-XSC is not only a development tool, it also provides a lot of predefined highly accurate routines to compute reliable bounds for the solution to standard numerical problems.

In this note we discuss the usage of a reliable linear system solver to compute the solution of problem 7 of the SIAM 100-digit challenge. To get the result we have to solve a 20 000 Œ 20 000 system of linear equations using interval computations. To perform this task we run our software on the advanced Linux cluster engine ALiCEnext located at the University of Wuppertal and on the high performance computer HP XC6000 at the computing center of the University of Karlsruhe.

The main purpose of this note is to demonstrate the power/weakness of our approach to solve linear interval systems with a large dense system matrix using C-XSC and to get feedback from other research groups all over the world concerned with the topic described. We are very much interested to see comparisons concerning different methods/algorithms, timings, memory consumptions, and different hardware/software environments. It should be easy to adapt our main routine (see Section 3 below) to other programming languages, and different computing environments. Changing just one variable allows the generation of arbitrary large system matrices making it easy to do sound (reproducible and comparable) timings and to check for the largest possible system size that can be handled successfully by a specific package/environment.

Keywords: C-XSC, reliable computing, 100-digit challenge, reliable linear system solver, high performance computing, large dense linear systems

Joint work of: Kolberg, Mariana; Krämer, Walter; Zimmer, Michael

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1447

The New IEEE-754 Standard for Floating Point Arithmetic

Peter Markstein (HP - Palo Alto, USA)

The current IEEE-754 floating point standard was adopted 23 years ago. IEEE chartered a committee to revise the standard to include new common practice in floating point arithmetic, to incorporate decimal floating point into the standard, and to address the issue of reproducible results. This talk will visit these issues, based on the current work of the IEEE-754 revisions committee, which expects that a new standard will be adopted sometime in 2008.

Keywords: Floating point arithmetic, standards

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2008/1448

Certifying numerical programs

Guillaume Melquiond (INRIA-MSR - Orsay, F)

This talk will present Gappa, a tool designed to help users to formally certify that clever floating-point and fixed-point applications are correct. The mechanisms of Gappa will be described, as well as its use on a few examples, including the proof of an elementary function of the CRlibm library.

Keywords: Floating-point arithmetic, program certification, formal proof

Some algorithmic improvements due to the availability of an FMA

Jean-Michel Muller (ENS - Lyon, F)

We give some examples (correctly-rounded multiplication by an "exact" constant such as π, computation of integer powers) of functions that can be computed easily and accurately when an FMA instruction becomes available.

Keywords: Floating-point arithmetic, fused multiply-add instruction, fma

Complex Inclusion Functions in the CoStLy C++ Library

Markus Neher (Universität Karlsruhe, D)

In this talk, we report on the C++ class library CoStLy (Complex Standard Functions Library) for the rigorous computation of complex function values or ranges. Rectangular complex interval arithmetic is used for the computations. The set of all rectangular complex intervals is denoted by IC in the following. In the CoStLy procedures, all truncation and roundoff errors are calculated during the course of the floating-point computation and enclosed into the result.

The library contains procedures for root and power functions, the exponential, trigonometric and hyperbolic functions, their inverse functions, and some auxiliary functions, such as the absolute value or the argument function.

The design of the respective inclusion functions has been guided by the paradigm that range bounds must be valid in any circumstance. For a single-valued complex function f, its inclusion function $F : IC- > IC$, and a given rectangular complex interval Z, this means that $F(Z)$ must contain the set $f(z)|zinZ$. For a multi-valued complex function, the meaning of a valid enclosure is less obvious. For example, the definition of $sqrt(-1)$ depends very much on the context of the computation. Possible values include $+i$, $-i$, $+i, -i$, $i*[-1,1]$, or the empty set.

For each multi-valued function f in the library, CoStLy contains an inclusion function F_s for the single-valued principal branch of f. Usually, F_s is only

defined on a subset of IC. Some alternative types of inclusion functions for multi-valued functions have also been implemented. These include inclusion functions F_c that are defined on IC, but enclose function values of different branches of f. For this type of inclusion function, the user of the software much check that the respective calculation is justified by theory.

Where applicable, inclusion functions F_a containing all function values of f have been implemented. For example, such inclusion functions are available for roots. For some multi-valued complex functions, however, the set of all function values is generally unbounded. In this case, no such inclusion functions are available.

For all inclusion functions in the CoStLy library, optimal range bounds are computed in exact arithmetic. The implementation of the algorithms in floating point arithmetic is generally obstructed by overflow, underflow or cancellation (OUC) in intermediate expressions. Avoiding OUC exceptions often requires many case distinctions. In the CoStLy library, all expression subject to potential overflow have been removed to avoid program abortion or invalid results. For the sake of accuracy, most of the cancellation and underflow exceptions have also been treated, even though these often only produce overestimations for arguments that are unlikely to appear in applications, such as arguments with very large or very small absolute values. All OUC exceptions are documented in the CoStLy source code.

The CoStLy library has been extensively tested for arguments with absolute values ranging from 1.0E-300 to 1.0E+300. For most arguments, the computed bounds for function values are highly accurate. In many test cases, the observed precision of the result was about 50 correct bits (out of the 53 bits available in IEEE 754 floating point arithmetic) for point arguments.

Numerical examples are presented in the talk.

The CoStLy C++ Class Library

Markus Neher (Universität Karlsruhe, D)

CoStLy Complex Standard Functions Library) has been developed as a C++ class library for the validated computation of function values and of ranges of complex standard functions. If performed in exact arithmetic, the inclusion functions for principal branches compute optimal range bounds. For the sake of accuracy, a major effort has been made in the implementation of the algorithms in floating point arithmetic to eliminate all intermediate expressions subject to numerical overflow, underflow, or cancellation. The CoStLy library has been extensively tested for arguments with absolute values ranging from 1.0E-300 to 1.0E+300. For most arguments, the computed bounds for function values are highly accurate. In many test cases, the observed precision of the result was about 50 correct bits (out of the 53 bits available in IEEE 754 floating point arithmetic) for point arguments.

Keywords: Complex interval arithmetic, inclusion functions

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1449

Iterative Refinement for Ill-Conditioned Linear Systems

Shin'ichi Oishi (Waseda Univ. / JST - Tokyo, J)

In this talk, we will consider the convergence of iterative refinement for a linear equation:

$$Ax = b, \tag{2}$$

where A is a floating point $n \times n$ matrix and b is a floating point n dimensional vector.

Let u be the unit round-off of the working precision and $\kappa(A)$ be the condition number of the problem.

For well posed problems, *i.e.*, in case of $u\kappa(A) < 1$, it has been shown that the iterative refinement improve the forward and backward errors of computed solutions provided that the residuals are evaluated by extended precision, in which the unit round off $\overline{u}$ is the order of u^2, then the result is rounded to the working precision. In this talk, we will treat ill-conditioned problems with

$$1 < u\kappa(A) < \infty. \tag{3}$$

Keywords: Accurate Dot Product Algorithm, Convergence Theorem

On the Interoperability between Interval Software

Evgenija D. Popova (Bulgarian Academy of Sciences, BG)

The increased appreciation of interval analysis as a powerful tool for controlling round-off errors and modelling with uncertain data leads to a growing number of diverse interval software. Beside in some other aspects, the available interval software differs with respect to the environment in which it operates and the provided functionality. Some specific software tools are built on the top of other more general interval software but there is no single environment supporting all (or most) of the available interval methods. On another side, most recent interval applications require a combination of diverse methods. It is difficult for the end-users to combine and manage the diversity of interval software tools, packages, and research codes, even the latter being accessible. Two recent initiatives: [1], directed toward developing of a comprehensive full-featured library of validated routines, and [3] intending to provide a general service framework for validated

computing in heterogeneous environment, reflect the realized necessity for an integration of the available methods and software tools.

It is commonly understood that quality comprehensive libraries are not compiled by a single person or small group of people over a short time [1]. Therefore, in this work we present an alternative approach based on interval software interoperability.

While the simplest form of interoperability is the exchange of data files, we will focus on the ability to run a particular routine executable in one environment from within another software environment, and vice-versa, via communication protocols. We discuss the motivation, advantages and some problems that may appear in providing interoperability between the existing interval software.

Since the general-purpose environments for scientific/technical computing like Matlab, Mathematica, Maple, etc. have several features not attributable to the compiled languages from one side and on another side most problem solving tools are developed in some compiled language for efficiency reasons, it is interesting to study the possibilities for interoperability between these two kinds of interval supporting environments.

More specifically, we base our presentation on the interoperability between Mathematica [5] and external C-XSC programs [2] via MathLink communication protocol [4]. First, we discuss the portability and reliability of interval arithmetic in Mathematica. Then, we present MathLink technology for building external MathLink-compatible programs. On the example of a C-XSC function for solving parametric linear systems, called from within a Mathematica session, we demonstrate some advantages of interval software interoperability.

Namely, expanded functionality for both environments, exchanging data without using intermediate files and without any conversion but under dynamics and interactivity in the communication, symbolic manipulation interfaces for the compiled language software that often make access to the external functionality from within Mathematica more convenient even than from its own native environment. Once established, MathLink connection to external interval libraries or problem-solving software opens up an array on new possibilities for the latter.

References:

[1] G. Corliss, R. B. Kearfott, N. Nedialkov, S. Smith: Towards an Interval Subroutine Library, Workshop on Reliable Engineering Computing, Svannah, Georgia, USA, Feb. 22-24, 2006.

[2] W. Hofschuster: C-XSC: Highlights and new developments. In: Numerical Validation in Current Hardware Architectures. Number 08021 Dagstuhl Seminar, Internationales Begegnungs- und Forschungszentrum für Informatik, Schloss Dagstuhl, Germany, 2008.

[3] W. Luther, W. Kramer: Accurate Grid Computing, 12th GAMM-IMACS Int. Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006), Duisburg, Sept. 26-29, 2006.

[4] Ch. Miyaji, P. Abbot eds.: Mathlink: Network Programming with Mathematica, Cambridge Univ. Press, Cambridge, 2001.

[5] Wolfram Research Inc.: Mathematica, Version 5.2, Champaign, IL, 2005.

Keywords: Software interoperability, interfacing, interval software, C-XSC, MathLink, Mathematica

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1450

Second note on basic interval arithmetic for IEEE754R

John D. Pryce (Cranfield University, GB)

The IFIP Working Group 2.5 on Numerical Software (IFIPWG2.5) wrote on 5th Septem- ber 2007 to the IEEE Standards Committee concerned with revising the IEEE Floating- Point Arithmetic Standards 754 and 854 (IEEE754R), expressing the unanimous request of IFIPWG2.5 that the following requirement be included in the future computer arithmetic standard:

For the data format double precision, interval arithmetic should be made available at the speed of simple floating-point arithmetic.

IEEE754R (we believe) welcomed this development. They had before them a document defining interval arithmetic operations but, to be the basis of a standards document, it needed more detail. Members of the Interval Subroutine Library (ISL) team were asked to comment, in an email from Ulrich Kulisch that enclosed one from Jim Demmel to Van Snyder raising the issue. This paper provides ISL's comments.

Keywords: Interval arithmetic, validated computation, floating point, standards, exceptions, not an interval

Joint work of: Pryce, John D.; Corliss, George C.; Kearfott, R. Baker; Nedialkov, Ned S.; Smith, Spencer

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1451

Towards the Development of an Interval Arithmetic Environment for Validated Computer-Aided Design and Verification of Systems in Control Engineering

Andreas Rauh (Universität Ulm, D)

Modern techniques for the design and analysis of control strategies for nonlinear dynamical systems are often based on the simulation of the open-loop as well as the closed-loop dynamical behavior of suitable mathematical models. In control engineering, continuous-time and discrete-time state-space representations are widely used which are given by sets of ordinary differential equations and difference equations, respectively. In addition to these representations, sets of differential algebraic equations are commonly used. Since we will focus on computational techniques which are applied for the design and mathematical verification of controllers for lumped parameter systems, i.e., systems which do

not contain elements with distributed parameters, partial differential equations will not be considered in this talk.

The prerequisite for the design and robustness analysis of each control system is the identification of mathematical models which describe the dynamics of the plant to be controlled as well as the available measurement devices with a sufficient accuracy. The model identification task comprises the derivation of physically motivated state equations, their parameterization based on measured data, as well as simplifications to apply specific approaches for controller design.

In the design stage, both open-loop and closed-loop control strategies can be considered. Since dynamical system models are subject to uncertain parameters and uncertain initial conditions in most practical applications, detailed mathematical formulations of the desired dynamics of the controlled system are necessary. These specifications involve the definition of robustness with respect to the above-mentioned uncertainties. For linear system representations, robustness is commonly specified in terms of regions in the complex domain containing all admissible poles of the closed-loop transfer functions (Γ-stability) or in terms of specifications of worst-case bounds for the frequency response ($\mathcal{B}$-stability) [1].

However, these specifications do not allow for inclusion of bounds for the state variables which are often available in the time domain if controllers are designed for safety critical applications. Especially for nonlinear dynamical systems, pole assignment based on the linearization of nonlinear mathematical models generally leads to the necessity for the analysis of asymptotic stability of the resulting closed-loop dynamics.

In this presentation, we will give an overview of the potential use of validated techniques for the analysis and design of controllers for nonlinear dynamical systems with uncertainties, where the systems under consideration will be subject to constraints for both state and control variables.

As an application scenario the design of robust control strategies for a biological wastewater treatment process will be discussed. In the design and the verification process, constraints for both state and control variables which are given by guaranteed interval bounds in the time domain are taken into account. Suitable computational techniques are, for example, based on an extension of the validated initial value problem solver VALENCIA-IVP [2,6]. For that purpose, differential sensitivities of the trajectories of all state variables with respect to variations of the parameters of the mathematical system model as well as the adaptation of controller parameters are computed. This information can then be used for online identification and adaptation of parameters during the operation of a closed-loop controller as well as in offline design, verification, and optimization. Here, the interval arithmetic routines for sensitivity analysis allow to compute guaranteed differential sensitivity measures for system models with both nominal parameters and interval uncertainties.

The presented interval arithmetic techniques are the basis for a general purpose tool for the analysis and the design of robust and optimal control strategies for uncertain dynamical systems. The presentation is concluded with an outlook on the formulation of control problems using sets of differential algebraic

equations. Possibilities for the extension of VALENCIA-IVP to this type of system representation will be summarized. Relations between the presented interval arithmetic approach and methods for stabilizing control of nonlinear dynamical systems which make use of structural system properties such as differential flatness [3] and exact feedback linearization are highlighted [4,5]. In the latter case, input-output linearization as well as (in special cases) input-to-state linearization are of practical importance.

References:

[1] J. Ackermann, P. Blue, T. Bünte, L. Güvenc, D. Kaesbauer, M. Kordt, M. Muhler, and D. Odenthal, *Robust Control: The Parameter Space Approach*, Springer–Verlag, London, 2nd edition, 2002.

[2] E. Auer, A. Rauh, E. P. Hofer, and W. Luther, *Validated Modeling of Mechanical Systems with* SMARTMOBILE*: Improvement of Performance by* VALENCIA-IVP, In Proceedings of Dagstuhl Seminar 06021: Reliable Implementation of Real Number Algorithms: Theory and Practice, Lecture Notes in Computer Science, Dagstuhl, Germany, 2006. In print.

[3] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, *Flatness and Defect of Nonlinear Systems: Introductory Theory and Examples*, International Journal of Control, vol. 61, pp. 1327–1361, 1995.

[4] H. K. Khalil, *Nonlinear Systems*, Prentice-Hall, Upper Saddle River, New Jersey, 3rd edition, 2002.

[5] H. J. Marquez, *Nonlinear Control Systems*, John Wiley & Sons, Inc., New Jersey, 2003.

[6] A. Rauh and E. Auer, www.valencia-ivp.com.

Keywords: Interval techniques, VALENCIA-IVP, controller design, robustness, validated integration of ODEs, parameter uncertainties, sensitivity analysis

Joint work of: Rauh, Andreas; Minisini, Johanna; Hofer, Eberhard P.

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1452

Automatic adaptation of the computing precision

Nathalie Revol (ENS - Lyon, F)

When a given computation does not yield the required accuracy, the computation can be done (either restarted or continued) using an increasing computing precision.

A natural question is how to increase the computing precision in order to minimize the time overhead, compared to the case where the optimal computing precision is known in advance and used for the computation.

Kreinovich and Rump have proven that when the computation must be restarted from scratch, then the minimal overhead is a factor 4.

In this presentation, we study the case where the computation can benefit from results obtained with a lower precision and thus is not restarted but rather

continued. Our first main contribution is to show that in such cases, the overhead is less than 4: for instance the minimal overhead is 2 for the Newton algorithm.

Then we present our second main contribution, an asymptotically optimal strategy for adapting the computing precision, which has an overhead tending to 1 when the optimal (unknown) precision tends to infinity.

[1]O. Beaumont, E.M. Daoudi, N. Maillard, P. Manneback and J.-L. Roch.
Tradeoff to minimize extra-computations and stopping criterion tests for parallel iterative schemes.
Parallel Matrix Algorithms and Applications, Marseille, 2004.
[2]J. van der Hoeven.
Computations with effective real numbers.
Theoretical Computer Science, vol 351, no 1, pp 52–60, 2006.
[3] V. Kreinovich and S. Rump.
Towards Optimal Use of Multi-Precision Arithmetic: A Remark.
Reliable Computing, vol 12, pp 365–369, 2006.
[4] N. Mueller.
The iRRAM: Exact Arithmetic in C++.
Workshop on Constructivity and Complexity in Analysis, Swansea, 2000.
[5] N. Revol.
Newton's algorithm using multiple precision interval arithmetic.
Numerical Algorithms, vol 34, no 2, pp 417–426, 2003.

Keywords: Computing precision, automatic adaptation, asymptotically optimal adaptation

Interval Arithmetic and Standardization

Jürgen Wolff von Gudenberg (Universität Würzburg, D)

Interval arithmetic is arithmetic for continuous sets. Floating-point intervals are intervals of real numbers with floating-point bounds.

Operations for intervals can be efficiently implemented. There is an unanimous agreement, how to define the basic operations, if we exclude division by an interval containing zero. Hence, it should be standardized. For division by zero, two options are possible, the clean exception free interval arithmetic or the containment arithmetic. They can be standardized as options.

Elementary functions for intervals can be defined. In some application areas loose evaluation of functions, i.e. evaluation over an interval which is not completely contained in the function domain, is recommended, In this case, however, a discontinuity flag has to be set to inform that Brouwer's fixed point theorem is no longer applicable in that case.

Keywords: Intervals, containment sets, IEEE754r

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1434

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2008/1434

Implementation of the reciprocal square root in MPFR

Paul Zimmermann (INRIA Lorraine, F)

We describe the implementation of the reciprocal square root — also called inverse square root — as a native function in the MPFR library. The difficulty is to implement Newton's iteration for the reciprocal square root on top's of GNU MP's MPN layer, while guaranteeing a rigorous 1/2 ulp bound on the roundoff error.

Keywords: Multiple precision, floating-point, inverse square root, correct rounding, MPFR library

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2008/1435

Full Paper:
http://www.mpfr.org/

Fast (parallel) Dense Linear Interval Systems Solving in C-XSC Using Error Free Transformations and BLAS

Michael Zimmer (Universität Wuppertal, D)

The traditional solver for linear interval systems available in C-XSC [6,1] is mathematically based on the Krawczyk[12] operator and modifications introduced by Rump[17]. The Krawczyk operator is composed of matrix/vector operations. These operations are realized in C-XSC with higest accuracy (only one final rounding) using a so called long accumulator (dotprecision variable). C-XSC dotprecision variables allow the error free computation of sums of floating point numbers as well as the error free computation of scalar products of floating point vectors. Thus, from a mathematical point of view these operations are perfect. Because actual hardware does not support these perfect scalar products all operations have to be realized by software. This fact leads to a tremendous time penalty (note: it has been shown that with modest additional hardware costs perfect scalar products can be made as fast as simple floating-point loops).

To speed up the C-XSC scalar product software-operations we adapt the so called DotK algorithm as published in [14]. Error free transformations[14,3,4,10] are used as basic building blocks to develop summation and scalar product algorithms simulating a K-fold precision. Compared to the perfect C-XSC operations these operations are fast. They are more accurate than simple floating-point loops (but of course no longer perfect in the mathematical sense). The fast operations are available in C-XSC via the new data types DotK, IDotK, CDotk and CIDotK. These new data types are composed in such a way that traditional C-XSC code using dotprecision variables can be adapted with minimal effort. It is possible to switch (at runtime!) from perfect computations to fast operations using K-fold precision (K equal 0 means traditional dotprecision computations)

and it is possible to hold intermediate results with corresponding error bounds for further summations or scalar product updates. The details are described in [19].

Additionaly, based on similar algorithms used in Intlab[16], BLAS and LAPACK libraries [2] are used in the $O(n^3)$ parts of the linear system solver. For matrix-matrix products, manipulation of the rounding mode of the processor is used to compute enclosures of the correct result.

Comparing the traditional solver with the new version shows that the class of problems which are solvable with the new version is smaller than the class of problems which can be solved using the solver based on perfect operations. But it seems that for real world problems also the new solver is appropriate. Using the new solver based on BLAS and simulating a quadrupel precision (i.e. k==2) the speedup comes close to 200(!). The new solver is nearly as fast as the corresponding IntLab[16] solver verifylss. Solving a real linaer system of dimension 1000 on a Pentium 4 with 3.2GHz takes about 2.8 seconds. In all cases tested the accuracy of our new solver was better and in some cases significantly better than the accuracy of the corresponding IntLab results. The new solver also allows solving larger (dense) problems than its IntLab counterpart. We also show some examples where IntLab falls down whereas our new solver still works.

A parallel version of this solver, based on ScaLAPACK, is also available. Unlike the previous parallel solver in C-XSC[5], this new solver does not depend on a root-node, which makes it possible to compute a verified solution even of very large linear systems.

In the talk we will discuss the new data types in more detail, we will emphasize our modifications to the DotK algorithm taken from the literature [14,15], we will show time measurements and we will present results concerning the accuracy of the computed enclosures. Our results will also be compared to corresponding results computed with the IntLab package. We also will comment on hardware features and compiler options which can/should be used to get reliable results on different platforms efficiently.

References:

[1] Downloads:

C-XSC library: http://www.math.uni-wuppertal.de/~xsc/xsc/cxsc.html Solvers: http://www.math.uni-wuppertal.de/~xsc/xsc/cxsc_software.html

[2] L.S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, An Updated Set of Basic Linear Algebra Subprograms (BLAS), ACM Trans. Math. Soft., 28-2 (2002), pp. 135–151.

[3] Bohlender, G.; Walter, W.; Kornerup, P.; Matula, D.W.; Kornerup, P.; Matula, D.W.:

Semantics for Exact Floating Point Operations.

Proceedings, 10th IEEE Symposium on Computer Arithmetic, 26-28 June 1991, IEEE, 1991.

[4] Dekker, T.J.: A floating-point technique for extending the available precision. Numer. Math., 18:224, 1971.

[5] Grimmer, M.: Selbstverifizierende Mathematische Softwarewerkzeuge im High-Performance Computing. Konzeption, Entwicklung und Analyse am Beispiel der parallelen verifizierten Loesung linearer Fredholmscher Integralgleichungen zweiter Art. Logos Verlag, 2007.

[6] Hofschuster, W.; Kraemer, W.:

C-XSC 2.0: A C++ Library for Extended Scientific Computing.

Numerical Software with Result Verification, Lecture Notes in Computer Science, Volume 2991/2004, Springer-Verlag, Heidelberg, pp. 15 - 35, 2004.

[7] Kersten, Tim: Verifizierende rechnerinvariante Numerikmodule, Dissertation, University of Karlsruhe, 1998

[8] Klatte, Kulisch, Wiethoff, Lawo, Rauch: "C-XSC - A C++ Class Library for Extended Scientific Computing", Springer-Verlag, Heidelberg, 1993.

Due to the C++ standardization (1998) and dramatic changes in C++ compilers over the last years this documentation describes no longer the actual C-XSC environment. Please refer to more accurate documentation (e.g.[1]) available from the web site of our research group: http...

[9] Kirchner, R., Kulisch, U.:

Hardware Support for Interval Arithmetic.

Reliable Computing, Volume 12, Number 3, June 2006 , pp. 225-237(13).

[10] Knuth, D.E.: The Art of Computer Programming: Seminumerical Algorithms.

Addison Wesley, 1969, vol. 2.

[11] Kulisch, U.: Computer Arithmetic and Validity - Theory, Implementation. To appear.

[12] Krawczyk, R.: Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken, Computing, 4:187-201, 1969.

[13] Lerch, M.; Tischler, G.; Wolff von Gudenberg, J.; Hofschuster, W; Kraemer, W.: filib++, a Fast Interval Library Supporting Containment Computations.

ACM TOMS, volume 32, number 2, pp. 299-324, 2006.

[14] Ogita, T., Rump, S.M., Oishi, S.: Accurate sum and dot product. SIAM Journal on Scientific Computing, 26:6, 2005.

[15] Oishi, S., Tanabe, K., Ogita, T., Rump, S.M., Yamanaka, N.:

A Parallel Algorithm of Accurate Dot Product.

Submitted for publication, 2007.

[16] Rump, S.M.: Intlab - Interval Laboratory. Developments in Reliable Computing, pp. 77-104, 1999.

[17] Rump, S.M.: Kleine Fehlerschranken bei Matrixproblemen, Dissertation, University of Karlsruhe, 1980

[18] Stroustrup, Bjarne: The C++-Programming Language, 3rd Edition, Addison-Wesley, 2000.

[19] Zimmer, Michael: Laufzeiteffiziente, parallele Loeser fuer lineare Intervallgleichungssysteme in C-XSC, Master thesis, University of Wuppertal, 2007.

AMS subject classification: 65H10, 15-04, 65G99, 65G10, 65-04

Keywords: Error-free transformations, K-fold accuracy, accurate dot product, C-XSC, high accuracy, dense linear systems, verified computation

Joint work of: Zimmer, Michael; Kraemer,Walter

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1436

A Software Library for Reliable Online-Arithmetic with Rational Numbers

Gregorio de Miguel Casado (University of Zaragoza, E)

An overview of a novel calculation framework for scientific computing in integrable spaces is introduced. This paper discusses some implementation issues adopted for a software library devoted to exact rational online-arithmetic operators for periodic rational operands codified in fractional positional notation.

Keywords: Computable analysis; online-arithmetic; rational numbers

Joint work of: de Miguel Casado, Gregorio; García Chamizo, Juan Manuel

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2008/1439