Elements of Scientific Computing

J. Hartmut Bleher

KommunikationsSysteme,
Bismarckstr. 31, D-72622 Nürtingen, Germany.
kommsys@t-online.de

Abstract. In the past decades, computer performance has increased dramatically and is still doing so, putting tremendous power at the fingertips of every computer user. Many of these capabilities are used for new and complex functions and for a (hopefully) better user interface, but also sometimes wasted for questionable gadgets. Unfortunately, in the same time frame little has been done to increase the *confidence in the answers computers produce* in the scientific field, because, so far, computer designers are much more motivated by mass market needs than by scientific requirements. With the circuit densities achievable on computer chips today and with 'clean' architectures already in place, only very little additional effort would have to be spent to provide the means to *allow computers to deliver results with guarantees*, i.e. with verified accuracy.

1 Hardware Requirements

Scientific computing is done mostly with *Floating Point representation* of numbers. A 'clean' Floating Point architecture is a recognized requirement for the survival of today's single-chip or multi-chip computers. This means that for basic arithmetic functions ($+$, $-$, $\times$, $/$) and even for a set of standard functions (sqrt, sin, tan, exp, etc.) *the presented accuracy is one half of the Unit in the Last Place* (1/2 ULP).

It is in principle immaterial which Floating Point format is used, like Hexadecimal as in IBM /370, /390 architecture or Binary as in IEEE 754 architecture, although a Decimal format would sometimes alleviate conversion to 'human-readable' results.

It is in principle also relatively unimportant which precision of the numerical data presented is chosen, i.e. how many bits are used in the mantissa and in the exponent of the Floating Point number. The precision, being Single (Short), Double (Long) or Extended (Quadrupel), is mainly influenced by the data paths of the computer core and the associated memory (32, 64, 128 or 256 bits). It may be freely selected by the programmer as long as it allows computation without loss of accuracy in combined and iterative arithmetic functions, and representation of the results with the necessary number of accurate digits. However, the *danger of losing accuracy in solving (ill-conditioned) problems* which span a very large area of the computer representable number space is still immanent, even with the highest practical

precision. Therefore, at least one more encompassing basic arithmetic function has to be provided which completely avoids loss of accuracy, especially *through cancellation* in the 'Multiply-and-Add' function sequence applied to an arbitrarily large number set.

It has been shown by U. Kulisch et al. that an *architected inner product* (Dot Product •) can satisfy this requirement in an advantageous way, and will thus become a key element of scientific computing. In essence, it requires the entire architectured computer number space to be in Fixed Point representation for the 'Multiply-and-Add' function. This can be easily accomplished by a moderately enhanced addressing and storing mechanism for the L1-cache which is not difficult at today's circuit count of computer chips.

Computation with absolute accuracy requires *Directed Rounding* (up, down, to 0, to nearest) to avoid accumulation of rounding errors and to allow bounding of results using iterative algorithms. All rounding modes must be intrinsically executable with the basic arithmetic instructions, requiring individual OP-codes to not lose any cycles for setting a rounding mode. This *architected Interval Arithmetic capability* is another key element of today's scientific computing.

It would be quite helpful to have, besides the *REAL Floating Point number space*, an extension of the basic computer data types to the *COMPLEX Floating Point number space* in hardware, with all the intricacies of errorfree handling of boundaries and exceptions being removed from the hands of the not so experienced programmer.

An extended set of Floating Point registers may simultaneously contain upper and lower bounds of COMPLEX Floating Point numbers fed by multiple arithmetic and logic units *executing in parallel COMPLEX interval arithmetic instructions.*

Hardware Performance, usually measured in Floating Point Operations Per Second (FLOPS), is determined by the number of machine cycles required to execute a Floating Point instruction in a particular precision. Again, the high circuit count of today's processor chips makes the objective of single cycle Floating Point instructions a realistic goal, even for the highest precision implemented. Additional circuitry may be used for instruction and data pipelining and for providing numbers of identical execution units working in parallel on larger data sets. Thus, several Floating Point instructions may be executed per machine cycle in a multi-way fashion on a multi-million circuit computer chip in the very near future.

2 Software Requirements

Operating Systems supporting scientific computing must be able to *effectively move large amounts of data* in a deep storage hierarchy. They also have to be able to *handle exception cases* on the system level without necessarily impeding the flow of data, i.e. *in a benign predictive manner.* For large

problems and large data sets appropriate *partitioning mechanisms* have to be put into place to effectively use the available storage and execution hardware resources.

Basic Algorithms and Procedures have to be provided to handle REAL and COMPLEX Floating Point interval number manipulation for basic and also for more complicated arithmetic functions in a *basic logic and arithmetic system* (blas). Special care is required to *minimize the path length* of individual procedures and to pay attention to the *efficiency of algorithms* used to achieve a high thruput, which is a measure for overall system performance.

The blas will become the base for Scientific Computing Languages, supporting in userfriendly dialects comprehensive *errorfree Operators on multiple* REAL and COMPLEX Floating Point (interval) *data types.*

Problem Solvers are needed to solve a large variety of mathematical and real life numerical problems in the fields of biology, chemistry, physics, and mechanical and electrical engineering with strong bounds of the results presented. They may be built using interval arithmetic and the software facilities described above. *Algorithmic proof of solutions, and verification and guarantee of result accuracy* can be provided using the comprehensive work of U. Kulisch and his associates.

3 Modelling Requirements

Having available the Hardware and Software Elements of Scientific Computing, one must not forget the fact that the *results delivered by the computer can only be as good as the theory, the model and the resulting input data* used to describe the real life circumstances and processes. Therefore it is imperative that the enhanced arithmetic capabilities of the computer described above are already considered when the models are described by equations, as well as during the process of representing the equations numerically. The scientist solving a particular problem has to decide whether to use straight forward traditional arithmetic or to embark on interval methods in order to verify the results of his/her computations even in ill-conditioned cases.

4 Conclusion

In today's world, with the rather mature hardware and software computer technologies in place and with the Very Large Scale Integration capabilities of the semiconductor foundries, there is no excuse whatsoever for not providing the key elements of scientific computing with each and every general purpose or scientific computer core delivered for main frames, servers and personal computers. *Interval Arithmetic for the REAL and COMPLEX Floating Point number space with single cycle execution* of basic instructions, including the Multiply-and-Add function for the architected Inner Product, shall be supported to allow guaranteed accuracy of computed results.

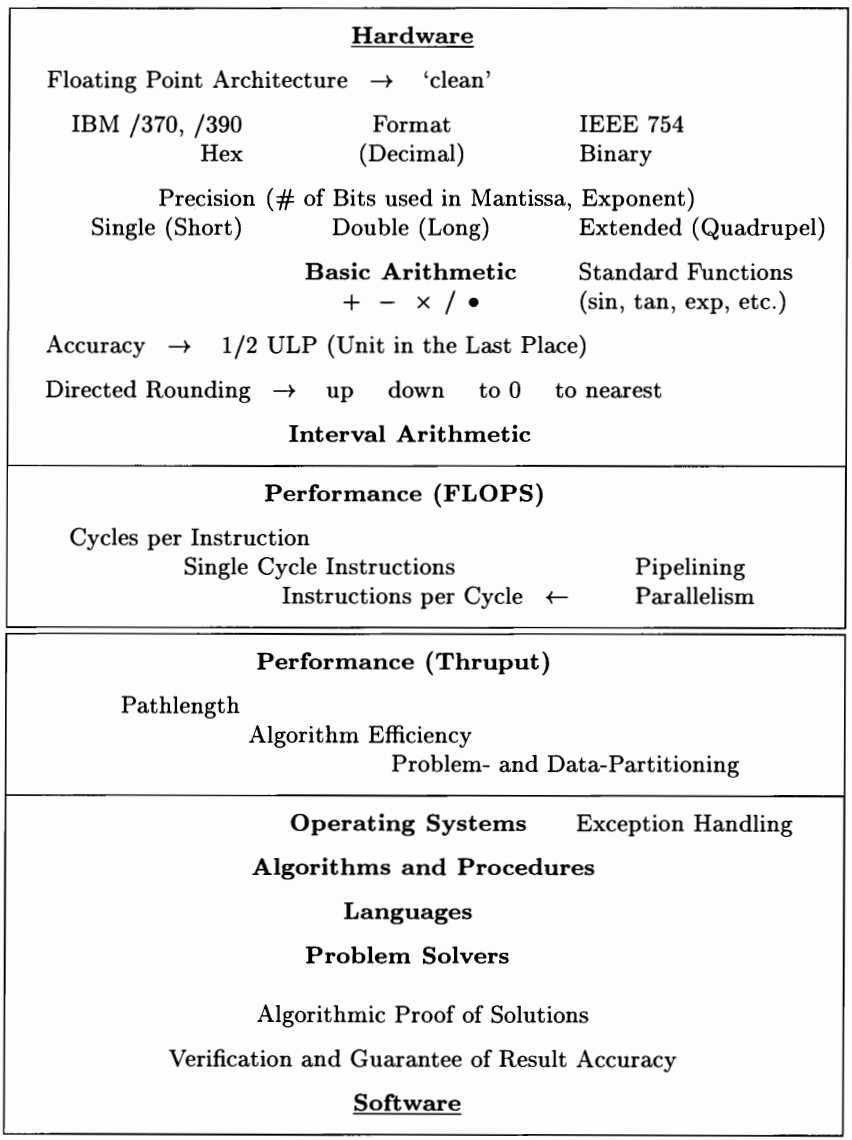

Fig. 1. Elements of Scientific Computing

References

1. Kulisch U. W., Miranker W. L. (1981) Computer Arithmetic in Theory and Practice, Academic Press, New York.
2. Kulisch U. W., Miranker W. L., eds. (1983) A New Approach to Scientific Computation, Academic Press, New York.
3. IBM (1983) High-Accuracy Arithmetic Subroutine Library, Program No. 5664-185, General Information Manual, GC-33-6163-0, and Program Description and User's Guide, SC-33-6164-1.
4. IBM (1984) IBM S/370 RPQ High-Accuracy Arithmetic, SA-22-7093-0.
5. ANSI/IEEE (1985) A Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std. 754-1985, New York, printed in SIGPLAN **22**, 2 (1987), pp. 9–25.
6. IBM (1986) High-Accuracy Arithmetic Subroutine Library, Program No. 5665-337/5666-320 General Information Manual, GC-33-6163-02, and Program Description and User's Guide, SC-33-6164-02.
7. Bleher J. H., Rump S. M., Kulisch U. W., Metzger M., Ullrich Ch., Walter W. (1987, 1988) FORTRAN-SC, A Study of a FORTRAN Extension for Engineering/Scientific Computation with Access to ACRITH, Computing **39** and Computing, Suppl. **6**.
8. Adams E., Kulisch U. W., eds. (1993) Scientific Computing with Automatic Result Verification, Academic Press, New York.
9. Kulisch U. W., Teufel T., Höfflinger B. (1994) Genauer und trotzdem schneller, ein neuer Coprozessor für hochgenaue Matrix- und Vektoroperationen, Electronic **26**, Franzis, Poing.
10. Gustafson J. (1998) Computational Verifiability and Feasibility of the ASCI Program, IEEE Computational Science and Engineering, January-March 1998.

Biography

J. Hartmut Bleher is a Consultant for Information and Communication Systems. He has been involved in Solid State and Low Temperature Physics and Integrated Circuit Technology at the Universities of Stuttgart and Aachen (Germany), and at the IBM Laboratory in East Fishkill, New York (U.S.A.). At the IBM Laboratory in Böblingen (Germany) he has managed Hardware and Software System Development for Computer Graphics and for Scientific Computing. He has been Product Manager for the IBM/9370 Processor Family and for IBM/370 and IBM/390 Integrated Parallel Systems. He has managed IBM's cooperations with the University of Karlsruhe (Germany) in the field of Scientific Computing and with the CERN High Energy Physics Laboratory in Geneva (Switzerland) in the field of Parallel Processing. J. Hartmut Bleher received his M.S. and his Ph.D. from the Technical University of Stuttgart (Germany).