Introduction to the Special Issue "Real Numbers and Computers"

This special issue contains a selection of papers presented during the international conference "*Real Numbers and Computers*", Saint-Étienne, France, April 1995.
Efficient handling of real numbers in a computer is not yet solved in a satisfying way, yet. Although the "floating-point" formats most often used in scientific computing usually give sufficient results, some reliability problems may occur. Program portability could imply high rewriting costs: some programs which work well with a machine, may become unreliable with another one. Users (from computer algebra, computational geometry, ...) may need results far more accurate than the ones obtained with usual number systems, if not "exact" results.
Many members of the scientific community are concerned by this problem, they could share their knowledge and come up with new solutions. But they do not have the opportunity to meet, they do not belong to the same scientific fields (computer science, number theory, numerical analysis, computer algebra...) and they have a different vocabulary. The aim of the Saint-Étienne Conference was to bring them together during this meeting, to establish some collaborations.

The very first problem with the manipulation of real numbers in computers is that the set of real numbers is not enumerable. As a consequence, it is not possible to represent each real number by a finite string of symbols taken from a finite alphabet. Depending on the application, one has to choose which finite or enumerable subset of the real numbers will be manipulated.
Even with enumerable subsets of the reals, there remain serious problems: probably the most important is that one cannot determine whether two computable real numbers[1] are equal.
Let us now examine some problems related to the discrete machine approximation of the continuous reals.

- In computational geometry, the main problem is to construct topologically consistent objects using (non-independent) numerical tests (e.g., signs of determinants). For instance, if we try to compute the distance (*a priori* null) between the intersection point of two straight lines and the straight lines separately, using floating-point arithmetic, it is almost certain that the answers will both be different and most likely non-null. For many such situations there is no obvious general treatment known.
- A report of the *United States General Accounting Office* (B-247094, Feb. 1992), explains that on February 1991 (during the war in the Gulf), a Patriot missile defense system failed to intercept an incoming Scud, that killed 28 people, due to an inaccurate tracking calculation.

[1] A real number x is computable if there is a machine that computes, for any given integer n, a rational number r_n that approximates x within error 2^{-n} (for instance, see Ker-I Ko, *Complexity Theory of Real Functions*, Birkhauser, 1991).

– If we try to compute the sequence (u_n) defined as

$$\begin{aligned} u_0 &= 2 \\ u_1 &= -4 \\ u_{n+1} &= 111 - \tfrac{1130}{u_n} + \tfrac{3000}{u_n u_{n-1}} \end{aligned}$$

using any bounded-precision arithmetic (such as floating-point arithmetic) on any computer, then 100 will seem to be the limit value of the sequence, while the correct limit value is 6.

– Define a sequence (x_n) as

$$\begin{aligned} x_0 &= 1.5100050721318 \\ x_{n+1} &= \tfrac{3x_n^4 - 20x_n^3 + 35x_n^2 - 24}{4x_n^3 - 30x_n^2 + 70x_n - 50} \end{aligned}$$

depending on your computer, the apparent limit value of x_n will be 1, 2, 3 or 4.

Those are archetypes of problems that happen in real-world computations (maybe especially during iterative calculations).

Many solutions have been proposed to cope with such problems:

– First, one may try to make the usual floating point arithmetic more reliable, and to entirely specify it, in order to be able to elaborate proofs and algorithms that use the specifications. For instance, the IEEE-754 and IEEE-854 standards for floating point computations[2] considerably helped to improve the quality and portability of programs, and to design multiple precision or interval arithmetic programs. The paper by Evgenija Popova (*On a Formally Correct Implementation of IEEE Computer Arithmetic*) is devoted to this topic. The specification of the arithmetic may also help to get *a priori* bounds on numerical errors for various computations. Raymond Pavec (*Some Algorithms Providing Rigorous Bounds for the Eigenvalues of a Matrix*) and Fabienne Jézéquel (*Round-Off Error Propagation in the Solution of the Heath Equation by Finite Differences*) obtained such bounds.

– It is not always possible to get realistic bounds on the numerical errors before the execution of a program, therefore it is most desirable to build tools that dynamically compute such bounds. Two possible ways to do this are the *interval arithmetic*, illustrated by the paper by Svetoslav Markov (*On Directed Interval Arithmetic and its Applications*) and the *perturbation methods*, illustrated by the paper by Jalil Asserhine, Jean-Marie Chesneaux and Jean-Luc Lamotte (*Estimation of Round-Off Errors on Several Computer Architectures*).

– A more drastic solution is to get rid of the usual floating-point arithmetic, and to build systems capable of computing with arbitrary accuracy. One may try to represent real numbers by flows of digits, as in *on-line arithmetic*. This is illustrated by Thomas Lynch and Michael Schulte in their

[2] *IEEE Standard 754-1985 for Binary Floating-Point Arithmetic*, IEEE. Reprinted in *SIGPLAN 22*, 2, pp. 9-25. People interested by this topic should read the paper by David Goldberg, *What Every Computer Scientist Should Know About Computer Arithmetic*, ACM Computing Surveys, Vol. 23 No 1, pp. 5-48

paper (*High-Radix OnLine Arithmetic for Credible and Accurate General Purpose Computing*). OnLine arithmetic was pioneered by one of the invited speakers at the Conference, Milos Ercegovac, professor at the *University of California at Los Angeles*. Another more general scheme is to represent a number by flows of coefficients, such as those of a continued-fraction expansion. This solution is explored by Peter Kornerup and David W. Matula (*LCF: A Lexicographic Binary Representation of the Rationals*, invited paper), by Asger Munk Nielsen and Peter Kornerup (*MSB-First Digit Serial Arithmetic*), and by D. Lester (*Exact Statistics and Continued Fractions*). Peter Kornerup, professor at *Odense University*, Denmark, was our second invited speaker at the Saint-Etienne Conference.

Other approaches, such as symbolic manipulation of numbers, are being explored, but they are not represented in this special issue.

Approximating the continuous real arithmetic as closely as possible with our inevitably discrete tools is attempting the impossible, but it is fascinating. There are still many things to be done in this domain, and we hope that the conference "Real Numbers and Computers No 2", that will be held in Marseille, France, in April 1996, will bring new solutions.

We would like to thank all the authors of submitted papers, including those authors of papers that could not be included in this special issue due to reviewer revision requests that could not be accommodated in our tight time frame for publication. Special thanks are due to the Editor-in-Chief of the *Journal of Universal Computer Science*, Hermann Maurer, for hosting this special issue.

Jean-Claude BAJARD, *Guest Editor*
Laboratoire LMI, Université de Provence
13453 Marseille Cedex 13, FRANCE

Dominique MICHELUCCI, *Guest Editor*
École des Mines de Saint-Étienne, SIMADE, 158 cours Fauriel
42023 Saint-Étienne Cedex 2, FRANCE

Jean-Michel MOREAU, *Guest Editor*
École des Mines de Saint-Étienne, SIMADE, 158 cours Fauriel
42023 Saint-Étienne Cedex 2, FRANCE

Jean-Michel MULLER, *Guest Editor*
CNRS, Laboratoire LIP, ENS Lyon, 46 Allée d'Italie
69364 Lyon Cedex 07, FRANCE