Comments on recent mails concerning IEEE P1788

Interval arithmetic (IA) deals with closed and connected sets of real numbers. Originally it was developed for the set $\mathbb{IR}$ of bounded real intervals. On the computer it gets approximated by intervals of $\mathbb{IF}$, where $\mathbb{F}$ denotes the set of floating-point numbers. Since the early days of IA attempts were repeatedly made to extend IA to unbounded intervals. But inconsistencies occurred again and again since $-\infty$ and $+\infty$ were considered as elements of unbounded real intervals. This situation was ultimately solved in my book *Computer Arithmetic and Validity,* De Gruyter 2008, second edition 2013. Here $-\infty$ and $+\infty$ serve as bounds for the representation of unbounded real intervals but they are themselves not elements of these intervals. IEEE P1788 accepted this understanding.

50 years of extensive use of IA and development of several volumes of software packages with automatic result verification also in cooperation with computer manufacturers have never generated the need to extend the floating-point numbers by the exceptional values of the IEEE 754 arithmetic standard. These may have been useful for slow speed floating-point computations (Zuse 1940) or on early microprocessors around 1980. But taking IEEE 754 arithmetic with all its exceptions as basis for developing a standard for IA certainly is a tragic decision with the potential of killing IA. It puts an unnecessary heavy burden on the implementation and on the understanding of IA.

Taking together all the knowledge available today strongly recommends extending arithmetic for bounded and unbounded closed real intervals to bounded and unbounded open, and half-open real and floating-point intervals. The corresponding sets might be denoted by $\mathbb{JR}$ and $\mathbb{JF}$, respectively. Following the book *The end of Error*, CRC Press 2014 by John Gustafson, arithmetic of $\mathbb{JR}$ and $\mathbb{JF}$ is closed under addition, subtraction, multiplication, and division, also square root, powers, logarithm, exponential, and many other elementary functions needed for technical computing.

Let me finally make a remark on the exact dot product (EDP). Motion 9 required to compute the dot product of two floating-point vectors exactly. It was accepted by IEEE P1788 in 2009. Since Dec. 2013 we know that future processors by Intel and IBM (and others will follow) shall provide enough register memory on the arithmetic unit for computing the dot product of floating-point vectors exactly. This could have been a great achievement of the IEEE P1788 development. However, in the middle of 2013, Motion 9 was weakened to just recommending an EDP. A recommendation usually causes different behavior on different computer platforms.

Besides of increasing the accuracy, the EDP brings a considerable gain in computing speed. By pipelining it can be computed in the time the processor needs to read the data, i.e., it comes with ultimate speed. A VLSI implementation at the author's institute in 1994 computed the EDP in 1/4 of the time the main processor needed for a computation of the dot product in conventional floating-point arithmetic, although the main processor was using a more advanced technology. No slow intermediate access to the main memory is the reason for this speed increase. Frequently it was argued requiring an EDP has nothing to do with IA. This argument is very short-sighted. Just take two matrixes or vectors with interval components and try to compute the matrix or matrix-vector product. Computing the best possible answer requires the EDP, see section 4.6 in my book mentioned above. Of course, exact arithmetic allows comupting sharper bounds in applications.

The EDP also allows to compute long sums of floating-point numbers exactly and at high speed. The result is independent of the sequence in which the summands are added. So by the way the EDP brings associativity for floating-point and interval addition.

Ulrich Kulisch