Numerical reproducibility in HPC: issues in interval arithmetic

Nathalie Revol, Philippe Théveny

Title: Numerical reproducibility in HPC: issues in interval arithmetic

Authors:
Nathalie Revol, INRIA – AriC team, LIP, ENS de Lyon, France
Philippe Théveny, ENS de Lyon – AriC team, LIP, ENS de Lyon, France
`Nathalie.Revol@ens-lyon.fr`, `Philippe.Theveny@ens-lyon.fr`

Abstract. The problem of numerical reproducibility is the problem of getting the same result when a numerical computation is run several times, whether on the same machine (and then the word *repeatability* is often preferred) or on different machines. This phenomenon is best exemplified by the sum of n floating-point numbers: as the addition of floating-point numbers is not associative, the result of the sum depends on the order used to perform the additions. For instance, let us consider the computation of $-L + L + 1$, where L is so large that $L + 1$ is rounded as L, then $(-L + L) + 1$ gives 1 in floating-point arithmetic, whereas $-L + (L + 1)$ returns 0. If several cores are available, the addends can be split among the cores and the sum is split into sub-sums which are computed in parallel and then added together. Depending on the number of cores available at run time, the sum can be split into a variable number of sub-sums. Furthermore, the order used for the final reduction (the final sum of the sub-sums) is not deterministic. Thus the result may depend on the run.

Numerical reproducibility thus means *getting exactly the same result, bit for bit, independently on the execution: architecture, execution environment...*. In [4], several algorithms are proposed to compute the sum of floating-point numbers in a reproducible way. The differences between the proposed algorithms concern the accuracy of the result.

The accuracy of the result is indeed a different issue: what can be said regarding the (relative) error of the possible computational result? Techniques proposed by He and Ding in [3] or Bailey in [1] consist in increasing the resulting accuracy, by increasing the accuracy of each intermediate result, but without any clear statement about the accuracy of the result, apart from worst case upper bounds.

These two questions of numerical reproducibility and of numerical accuracy can be reconciled if one requires the computed result to be the correct rounding of the exact result, *i.e.*, if the computed result is obtained as if the result were computed exactly and then rounded to the floating-point format. On some examples, such as a LHChome computation [2], this requirement has been met. However, in most cases it is difficult to achieve. We stress the point that, although using this requirement gives a well-defined semantics to numerical reproducibility, these two concerns of reproducibility and of accuracy are distinct.

As far as interval arithmetic is concerned, we feel that the relevant issue is the inclusion property, *i.e.*, the guarantee that the exact result belongs to the computed resulting interval. This property is supposed to be guaranteed by interval arithmetic. The lack of reproducibility in this context may even be con-

sidered as an asset for interval computations: as the exact result always belongs to any computed result, it belongs to the intersection of all the computed results and one could hope for an improved accuracy by getting different results and taking their intersection. (Of course, as reproducibility is a property that many users depend on either for debugging or testing on many codes, both floating-point and interval ones, the lack of reproducibility remains troublesome.)

Nevertheless, implementation issues may invalidate the inclusion property. We will illustrate what could prevent the implementation of interval arithmetic to satisfy the inclusion property on architectures and libraries for HPC (High-Performance Computing). We will present how we circumvented these difficulties, on the example of the interval matrix multiplication.

References

[1] David Bailey. High-Precision Computation and Reproducibility. In *Reproducibility in Computational and Experimental Mathematics*, ICERM, Providence, Rhode Island, USA, December 2012.

[2] Florent de Dinechin, Eric McIntosh, and Franck Schmidt. Massive tracking on heterogeneous platforms. In *9th International Computational Accelerator Physics Conference (ICAP)*, 2006.

[3] Yun He and Chris H.Q. Ding. Using Accurate Arithmetics to Improve Numerical Reproducibility and Stability in Parallel Applications. *The Journal of Supercomputing*, 18:259–277, 2001.

[4] Hong Diep Nguyen and James Demmel. Fast Reproducible Floating-Point Summation. In *ARITH'21: 21st IEEE Symposium on Computer Arithmetic*, Austin, Texas, USA, April 2013.