Accuracy Optimization using Automatic Compensation in Floating-Point Arithmetic

Laurent Thévenoux, Matthieu Martel and Philippe Langlois

1 Univ. Perpignan Via Domitia, Digits, Architectures et Logiciels Informatiques, F-66860, Perpignan, France
2 Univ. Montpellier II, Laboratoire d'Informatique Robotique et de Microélectronique de Montpellier, UMR 5506, F-34095, Montpellier, France
3 CNRS, Laboratoire d'Informatique Robotique et de Microélectronique de Montpellier, UMR 5506, F-34095, Montpellier, France
`firstname.lastname@univ-perp.fr`

Keywords: accuracy, floating-point compensation, code transformation, instruction level parallelism, program optimization.

Algorithms using IEEE-754 floating-point arithmetic [1] suffer from accuracy problems caused by rounding and by the fact that floating-point numbers are approximations of real numbers. Since accuracy is a critical matter in scientific computing as well as for critical embedded systems, news techniques have been developed to improve the accuracy of numerical algorithms, like compensation [2], error-free transformations [3] and others. Research on this subject is very rich and since several decades, new research work is written annually. Solutions exist but they are for specialists and program optimizations must be implemented manually by experts.

We tempt to automatize this process by applying automatically compensations in floating-point computations at compile-time. We have implemented a tool able to parse C codes and to apply compensations on each floating-point operations. Compensated terms are computed and accumulated in parallel to the original operations. We generate a new C code by replacing in the original code, floating-point operations $(+, -, \times)$ by their respective compensated algorithms, (see TwoSum, TwoProd, etc. in [3]). The compensated terms of each instructions are accumulated and added to the final result of the evaluation.

We applied our approach on several cases. For example, by reproducing automatically what experts have done manually in [4]. The authors propose a compensated algorithm for Horner polynomial evaluation. They evaluate Horner's form of the polynomial $p(x) = (0.75 - x)^5(1 - x)^{11}$ close to its multiple roots.

They show that compensation is necessary to reach the expected accuracy. We obtain the same results with our automatic method.

In Figure 1 we show the results that we obtain on the previous example by computing the polynomial $p(x)$ with Horner's algorithm. The graph shows the the value of $p(x)$ close to one of its roots, without and with compensation. Without compensation the results are very inaccurate but with compensations results are better and we can observe a smoother drawing of the function. Our contribution is that this result is obtain automatically and can be done by a non-expert user, quickly and easily.

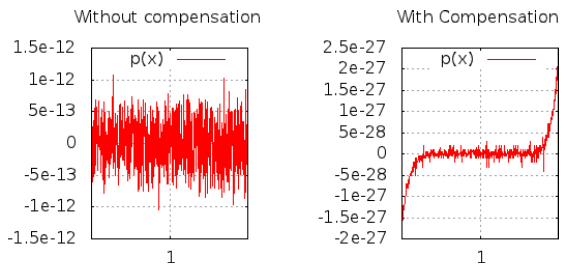

Figure 1: The leftmost graph shows the results of $p(x)$ around his root 1 computed in double precision. The rightmost graph shows the results of our generated code.

This example is just a study case and we want now to prove in which cases our approach is validated. We propose also to add a second criteria in our optimization process, the execution time. In modern architectures instruction level parallelism can be exploited to save execution time (as well as newer instructions like the FMA). Because in our approach compensated terms are computed in parallel to the original arithmetic expressions, we can use instruction level parallelism to obtain a fast execution. In future work we want to combine accuracy and time criteria to propose trade-offs to the user.

References:

[1] *IEEE Standard for Binary Floating-point Arithmetic*, Revision of Std 754-1985, 2008.

[2] Philippe Langlois, *Compensated Algorithms in Floating Point Arithmetic*, SCAN06, Duisburg, Germany, 2006.

[3] Takeshi Ogita, Siegfried M. Rump, Shin'ichi Oishi, *Accurate Sum and Dot Product*, SIAM J. Sci. Comput, vol. 26, 2005.

[4] Stef Graillat, Philippe Langlois, Nicolas Louvet, *Algorithms for accurate, validated and fast polynomial evaluation*, Japan Journal of Industrial and Applied Mathematics, vol. 26, pp. 191–214, 2009.