Stochastic Arithmetic in Multiprecision

Stef Graillat, Fabienne Jézéquel and Yuxiang Zhu

CNRS, UMR 7606, LIP6, University Pierre et Marie Curie, 4 place Jussieu, 75252
Paris cedex 05, France
`Stef.Graillat@lip6.fr`, `Fabienne.Jezequel@lip6.fr`, `Hareton.Zhu@gmail.com`

Abstract. The CADNA library [1] implements a stochastic arithmetic that makes it possible to estimate the propagation of rounding errors in scientific codes. Concretely, each variable is replaced by several floating-point numbers and the rounding mode is chosen randomly. The figures that differ in these numbers are due to rounding errors. The current version of CADNA only deals with single and double precision numbers. The MPFR library [2] makes it possible to define types of multiprecision numbers as well as arithmetic functions acting on multiprecision variables. The MPFR library also provides correctly rounded functions and operators in multiprecision.
In this paper, we present a library called SAM (Stochastic Arithmetic in Multiprecision). It is a multiprecision extension of the classic CADNA library using MPFR. In SAM (as in CADNA), the arithmetic and relational operators are overloaded in order to be able to deal with stochastic numbers. As a consequence, the use of SAM in a scientific code needs only few modifications. This new library SAM makes it possible to dynamically control the numerical methods used and more particularly to determine the optimal number of iterations in an iterative process. We present some applications of SAM in the numerical validation of chaotic systems modeled by the logistic map.

1 Introduction

The increasing power of current computers enables one to solve more and more complex problems. Then it is necessary to perform a high number of floating-point operations, each one leading to a round-off error. Because of round-off error propagation, some problems must be solved with a longer floating-point format. Therefore some versions of scientific software carry out multiprecision computation. For instance, XBLAS routines consist of a multiprecision version of basic linear algebra routines (BLAS). Moreover some libraries implement arbitrary precision arithmetic. MPFR [2] is an arbitrary precision library developed by INRIA in C language. Freely available on various platforms, MPFR uses very efficient algorithms. Based on MPFR, the MPFI library [3] provides arbitrary precision interval arithmetic. Several approaches exist to control round-off error propagation: interval arithmetic [4, 5], stochastic arithmetic [6, 7], abstract interpretation-based static analysis [8]. In this paper, we present the SAM (Stochastic Arithmetic in Multiprecision) library which is based on MPFR and extends the features of discrete stochastic arithmetic by enabling arbitrary precision computation.

In Section 2, we briefly describe Discrete Stochastic Arithmetic and the CADNA library, its implementation in single and double IEEE precision. In Section 3, we present the MPFR arbitrary precision library and we briefly describe the features of the SAM library. In Section 4, we analyse results of the logistic map obtained using SAM and MPFI. Conclusions and perspectives are given in Section 5.

2 Discrete Stochastic Arithmetic

Based on a probabilistic approach, the CESTAC method [7] allows the estimation of round-off error propagation which occurs with floating-point arithmetic. This method uses a random rounding mode: at each elementary operation, the result is rounded towards $-\infty$ or $+\infty$ with the probability 0.5. So one obtains, for N different runs, N different results on which a statistical test may be applied to estimate the round-off error on these results. Actually 2 or 3 runs are enough. It has been shown that $N = 3$ is the optimal number of runs [6]. With this method, one can know at any time during the execution of a scientific code the accuracy of any intermediate result. In any result, the number of exact significant digits, *i.e.* the number of significant digits not affected by round-off error, is estimated with the probability 95 %. From this point of view the concept of a computational zero introduced by J. Vignes [7] is essential. A computational zero is a computed result which has no exact significant digit or which is the mathematical zero. In practice, it is a result that the computer cannot distinguish from the null value because of round-off error propagation. From this new concept, a new theoretical arithmetic, called stochastic arithmetic [6, 7], has been developed. New definitions for order relations and equality relations have been proposed. All these definitions, unlike those used by the classical floating-point arithmetic, take into account the accuracy of the operands. For example, two computed results are stochastically equal if their difference is a computational zero.

Discrete Stochastic Arithmetic (DSA) [6, 9] is the use, on a computer, of the synchronous CESTAC method associated with the concepts of theoretical stochastic arithmetic. The CADNA software [6, 1] is a library which implements DSA in any code written in C++ or in Fortran and allows to use new numerical types: the stochastic types. The library contains the definition of all arithmetic operations and order relations for the stochastic types. The control of the accuracy is performed only on variables of stochastic type. When a stochastic variable is printed, only its exact significant digits appear. For a computational zero, the symbol @.0 is printed.

3 MPFR and SAM

In order to improve the accuracy of numerical results, the working precision can be increased using extended floating-point formats, multiprecision libraries or arbitrary precision libraries. In arbitray precision, the precision is only limited by memory storage, whereas in multiprecision the mantissa length is a fixed multiple of the mantissa length in IEEE double precision. The MPFR [2] arbitray precision library, written in the ISO C language, is based on the GNU MP library [10] (GMP for short). The internal representation of a floating-point number x by MPFR is a mantissa m, a sign s and a signed exponent e. If the precision of x is p, then the mantissa m has p significant bits. The mantissa m is represented by an array of GMP unsigned machine-integer type. MPFR provides the four IEEE rounding modes and correct rounding for all the operations and mathematical functions it implements. The semantic in MPFR is as follows: for each instruction $a = f(b, c)$ the variables may have different precisions. In MPFR, the data b and c are considered with their full precision and a correct rounding to the full precision of a is computed. Applications using MPFR inherit the same properties as programs using the IEEE 754 standard (portability, well-defined semantics, possibility to design robust programs and prove their correctness) with no significant slowdown on average with respect to arbitrary precision libraries with ill-defined semantics.

The SAM library implements in arbitrary precision the features of DSA: the stochastic types, the concept of computational zero and the stochastic operators. The particularity of SAM (compared to CADNA) is the arbitrary precision of stochastic variables. In SAM, the number of exact significant digits of any stochastic variable is estimated with the prabability 95 %, whatever its precision. Like in CADNA, the arithmetic and relational operators in SAM take into account round-off error propagation. All numerical instabilities which occur at run time are detected. Such instabilities are usually generated by an operation involving a computational zero. The SAM library is written in C++ and is based on MPFR. In the SAM library, all operators are overloaded. Consequently for a program written in C++ to be used with SAM, only a few modifications are needed: mainly changes in type declarations. Classical variables have to be replaced by stochastic variables (consisting of three variables of MPFR type). In SAM, for each stochastic operation, three MPFR operations are performed using different rounding modes and the numerical instability that may be generated is detected.

4 Application of SAM for chaotic dynamical systems

Let us consider the logistic iteration [11] defined by $x_{n+1} = ax_n(1 - x_n)$ with $a > 0$ and $0 < x_0 < 1$.

- When $a < 3$, this sequence converges to a unique fixed point, whatever the initial condition x_0 is.
- When $3.0 \leq a \leq 3.57$ this sequence is periodic, whatever the initial condition x_0 is, the periodicity depending only on a. Furthermore the periodicity is multiplied by 2 for some values of a called "bifurcations".
- When $3.57 < a < 4$ this sequence is usually chaotic, but there are certain isolated values of a that appear to show periodic behavior.

– Beyond $a = 4$, the values eventually leave the interval [0,1] and diverge for almost all initial values.

The logistic map has been computed with $x_0 = 0.6$ using SAM and MPFI. In stochastic arithmetic, iterations have been performed until the current iterate is a computational zero, *i.e.* all its digits are affected by round-off errors. In interval arithmetic, iterations have been performed until the two bounds of the interval have no common significant digit. In Tables 1 and 2, we report the number N of iterations performed for two ways of computing the logistic map.

In Table 1, we have computed the logistic map using the formula

$$x_{n+1} = ax_n(1 - x_n) \quad \text{with} \quad x_0 = 0.6. \tag{1}$$

During the computation performed using SAM with $a = 3.57$, no computational zero has been detected. The program has been stopped after one million iterations. Whereas using MPFI with the same value for a, at a certain iteration the stopping criterion is satisfied: the bounds of the last interval have no common digit. If $a = 3.575$, $a = 3.6$ or $a = 3.7$, more iterations are performed with SAM than with MPFI for the stopping criterion to be satisfied. However it must be pointed out that SAM is based on an estimation of round-off erros. Results obtained with MPFI are more pessimistic, but are guaranteed. If $a = 10$, the sequence diverges. Similar results are obtained with SAM and MPFI.

In Table 2, we have computed the logistic map using the formula

$$x_{n+1} = -a(x_n - \frac{1}{2})^2 + \frac{a}{4} \quad \text{with} \quad x_0 = 0.6. \tag{2}$$

Concerning SAM, the results are very similar to those in Table 1 (similar N). Nevertheless, the results are better with MPFI compared to Table 1. In some cases, MPFI provides better results than SAM. This can be explained by the so-called *dependency problem* which is a major drawback of interval arithmetic. Indeed, if an interval occurs several times in an expression, each occurrence is taken independently and then can lead to an unwanted over-estimation of the resulting interval. This is the case in equation (1) where the variable x_n appears twice whereas x_n appears only once in equation (2).

Although SAM has been designed for arbitrary precision, it can be compared with CADNA if the chosen precision is 24 bits or 53 bits. The results are, in general, similar with CADNA in double precision (53 bits) and SAM with 53 bits. It is normal since the operations are correctly rounded with the same precision in both cases. When there is a difference, it is due to the fact that the exponent of the floating-point numbers is not limited in SAM, whereas it is limited to 1023 in double precision. The same explanation applies for single precision and SAM with 24 bits (here the exponent in single precision is limited to 127).

a		# bits	N
3.57	SAM	24	$> 10^6$
	SAM	53	$> 10^6$
	SAM	100	$> 10^6$
	SAM	200	$> 10^6$
	SAM	2000	$> 10^6$
	MPFI	24	10
	MPFI	53	26
	MPFI	100	52
	MPFI	200	107
	MPFI	2000	1087
3.575	SAM	24	141
	SAM	53	389
	SAM	100	801
	SAM	200	1541
	SAM	2000	15789
	MPFI	24	11
	MPFI	53	26
	MPFI	100	52
	MPFI	200	107
	MPFI	2000	1086
3.6	SAM	24	63
	SAM	53	157
	SAM	100	327
	SAM	200	731
	MPFI	24	11
	MPFI	53	26
	MPFI	100	52
	MPFI	200	106
3.7	SAM	24	40
	SAM	53	107
	SAM	100	205
	SAM	200	388
	MPFI	24	10
	MPFI	53	26
	MPFI	100	51
	MPFI	200	104
10	SAM	24	21
	SAM	53	28
	SAM	100	28
	SAM	200	28
	MPFI	24	28
	MPFI	53	28
	MPFI	100	28
	MPFI	200	28

Table 1. Logistic map: number N of iterations performed with SAM and MPFI, $x_{n+1} = ax_n(1 - x_n)$ with $x_0 = 0.6$.

a		# bits	N
3.57	SAM	24	$> 10^6$
	SAM	53	$> 10^6$
	SAM	100	$> 10^6$
	SAM	200	$> 10^6$
	SAM	2000	$> 10^6$
	MPFI	24	372
	MPFI	53	1966
	MPFI	100	5020
	MPFI	200	11292
	MPFI	2000	123196
3.575	SAM	24	141
	SAM	53	389
	SAM	100	801
	SAM	200	1581
	SAM	2000	15821
	MPFI	24	92
	MPFI	53	302
	MPFI	100	706
	MPFI	200	1516
	MPFI	2000	15864
3.6	SAM	24	63
	SAM	53	157
	SAM	100	327
	SAM	200	725
	MPFI	24	48
	MPFI	53	142
	MPFI	100	328
	MPFI	200	712
3.7	SAM	24	40
	SAM	53	107
	SAM	100	205
	SAM	200	386
	MPFI	24	35
	MPFI	53	97
	MPFI	100	191
	MPFI	200	385
10	SAM	24	21
	SAM	53	28
	SAM	100	28
	SAM	200	28
	MPFI	24	28
	MPFI	53	28
	MPFI	100	28
	MPFI	200	28

Table 2. Logistic map: number N of iterations performed with SAM and MPFI, $x_{n+1} = -a(x_n - \frac{1}{2})^2 + \frac{a}{4}$ with $x_0 = 0.6$.

5 Conclusion and future work

In the article, we have demonstrated that SAM can be very useful to study the behavior of chaotic dynamical system. We have only studied the behavior of the logistic map. We plan to do a similar work with other systems like the Hénon map [12] or the Lorenz attractor [13].

In a future work, we will also compare in terms of efficiency and computing times our SAM library with MPFI. As mentioned previously, SAM and MPFI do not give the same answer since MPFI leads to a certified answer whereas SAM gives an answer true within a given probability. Anyway, it is sometimes sufficient to know the answer only with high probability.

Another perspective is the adaptive refinement of the precision in programs using SAM when the accuracy of the results is not satisfactory. Such a strategy would be based on an appropriate threshold for the number of exact significant digits lost during the computation.

References

1. Université Pierre et Marie Curie, Paris, F.: CADNA: Control of Accuracy and Debugging for Numerical Applications (2010) `http://www.lip6.fr/cadna`.
2. Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., Zimmermann, P.: MPFR: A multiple-precision binary floating-point library with correct rounding. ACM Trans. Math. Softw. **33**(2) (2007) 13 `http://www.mpfr.org`.
3. Revol, N., Rouillier, F.: MPFI (Multiple Precision Floating-point Interval library). (2009) Available at `http://gforge.inria.fr/projects/mpfi`.
4. Moore, R.: Interval analysis. Prentice Hall (1966)
5. Alefeld, G., Herzberger, J.: Introduction to interval analysis. Academic Press (1983)
6. Chesneaux, J.M.: L'arithmétique stochastique et le logiciel CADNA. Habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris (1995)
7. Vignes, J.: A stochastic arithmetic for reliable scientific computation. Math. Comput. Simulation **35** (1993) 233–261
8. Goubault, E., Putot, S., Baufreton, P., Gassino, J.: Static analysis of the accuracy in control systems: Principles and experiments. In: Proceedings of Formal Methods in Industrial Critical Systems, LNCS 4916, Springer-Verlag (2007)
9. Vignes, J.: Discrete stochastic arithmetic for validating results of numerical software. Num. Algo. **37**(1–4) (2004) 377–390
10. Grandlund, T.: GNU MP: The GNU Multiple Precision Arithmetic Library (2010) `http://gmplib.org`.
11. Devaney, R.L.: An introduction to chaotic dynamical systems. Second edn. Addison-Wesley Studies in Nonlinearity. Addison-Wesley Publishing Company Advanced Book Program, Redwood City, CA (1989)
12. Pichat, M., Vignes, J.: The numerical study of chaotic systems - future and past. In: 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Lausanne, Switzerland (2000)
13. Yao, L.S.: Computed chaos or numerical errors. Nonlinear Analysis: Modelling and Control **15**(1) (2010) 109–126