Stochastic arithmetic in multiprecision

Stef Graillat

Joint work with Fabienne Jézéquel and Yuxiang Zhu

LIP6/PEQUAN - Université Pierre et Marie Curie (Paris 6) - CNRS

NSV3, Third International Workshop on Numerical Software Verification
Edinburgh, UK, July 15th, 2010

Outline of the talk

1. The CESTAC method
2. Overview of multiprecision
3. The CADNA and SAM libraries
4. Applications

The CESTAC method

The CESTAC method (Contrôle et Estimation Stochastique des Arrondis de Calculs) was proposed by M. La Porte and J. Vignes in 1974.

It consists in performing the same code several times with different round-off error propagations. Then, different results are obtained.

Briefly, the part that is common to all the different results is assumed to be reliable and the part that is different in the results is affected by round-off errors.

The random rounding mode

Let r be the result of an arithmetic operation : $R^- < r < R^+$.

The random rounding mode consists in rounding r to $-\infty$ or $+\infty$ with a probability 0.5.

Using this new rounding mode, the same code can be run with different round-off error propagations.

If round-off errors affect the result, even slightly, one obtains for N different runs, N different results on which a statistical test may be applied.

Implementation of the CESTAC method

- each arithmetical operation is performed N times using the random rounding mode
  $\Rightarrow$ for each arithmetical operation, N results R_i are computed.
- computed result : $\overline{R} = \frac{1}{N} \sum_{i=1}^{N} R_i$.
- the number $C_{\overline{R}}$ of exact significant digits is estimated by

$$C_{\overline{R}} = \log_{10} \left(\frac{\sqrt{N}\ |\overline{R}|}{s\ \tau_\beta} \right) \quad \text{with} \quad s^2 = \frac{1}{N-1} \sum_{i=1}^{N} \left(R_i - \overline{R} \right)^2$$

  τ_β being the value of the Student distribution for $N-1$ degrees of freedom and a probability level $(1-\beta)$.

  In practice, $N = 3$ and $\beta = 0.05$.

Inconsistency of the floating-point arithmetic

On a computer,

- arithmetic operators are only approximations
- order relations are the same as in mathematics

$\Longrightarrow$ it leads to a global inconsistent behaviour.

Let x (resp. y) be an exact result and X (resp. Y) the corresponding computed result.

$$X = Y \not\Rightarrow x = y \quad \text{and} \quad x = y \not\Rightarrow X = Y.$$
$$X \geq Y \not\Rightarrow x \geq y \quad \text{and} \quad x \geq y \not\Rightarrow X \geq Y.$$

The problem of stopping criteria

Let a general iterative algorithm be : $U_{n+1} = F(U_n)$, U_0 being a data.

WHILE (ABS(X-Y) $>$ EPSILON) DO
    X = Y
    Y = F(X)
ENDDO

ε too low $\Longrightarrow$ a risk of infinite loop

ε too high $\Longrightarrow$ a too early termination.

The optimal choice from the computer point of view :
$X - Y$ is a **non significant value**.

New methodologies for numerical algorithms may be developed.

The concept of computed zero

J. Vignes, 1986

Definition 1

Using the CESTAC method, a result R is a computed zero, denoted by @.0, if

$$\forall i, \quad R_i = 0 \quad \text{or} \quad C_{\overline{R}} \leq 0.$$

It means that R is a computed result which, because of round-off errors, cannot be distinguished from 0.

The stochastic relations

Definition 2

Let X and Y be two results computed using the CESTAC method.

- *Stochastically equality, denoted by $s=$, is defined as :*
  $X \ s= \ Y$ *if and only if* $X - Y = @.0$.
- *Stochastically inequalities, denoted by $s>$ and $s\geq$, are defined as :*
  $X \ s> \ Y$ *if and only if* $\overline{X} > \overline{Y}$ *and* $X \ s\neq \ Y$.
  $X \ s\geq \ Y$ *if and only if* $\overline{X} \geq \overline{Y}$ *and* $X \ s= \ Y$.

DSA (Discrete Stochastic Arithmetic) is the joint use of the CESTAC method, the computed zero and the stochastic relations.

Self-validation of the CESTAC method

The CESTAC method is based on a 1st order model.

- A multiplication of two non-significant results
- or a division by a non-significant result

may invalidate the 1st order approximation.

Therefore the CESTAC method requires a dynamical control of multiplications and divisions, during the execution of the code.

Multiprecision with MPFR

MPFR library is written in C language based on the GNU MP library

The internal representation of a floating-point number x by MPFR is

- a mantissa m ;
- a sign s ;
- 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 and is interpreted as $1/2 \leq m < 1$.

$\rightarrow$ makes it possible to deal with arbitrary precision numbers

The CADNA library - I

The CADNA library allows to estimate round-off error propagation in any scientific program.

More precisely, CADNA enables one to :

- estimate the numerical quality of any result
- control branching statements
- perform a dynamical numerical debugging
- take into account uncertainty on data.

CADNA is a library which can be used with Fortran or C++ programs.

CADNA can be downloaded from http ://www.lip6.fr/cadna

The CADNA library - II

CADNA implements Discrete Stochastic Arithmetic

CADNA enables one to use new numerical types : the stochastic types.

With CADNA, all arithmetic operators and mathematical functions are overloaded. This overloading has been optimized for array operations.

The cost of CADNA is about :

- 3.5 for memory
- 10 for run time.

The stochastic types

CADNA provides two new numerical types, the stochastic types :

- `type (single_st)` for stochastic variables in single precision stochastic type associated with real.
- `type (double_st)` for stochastic variables in double precision stochastic type associated with double precision.

Complex numbers are not implemented in this CADNA version.

The SAM library - I

The SAM library implements in arbitrary precision the features of DSA :

- the stochastic types
- the concept of computational zero
- the stochastic operators

The particularity of SAM (compared to CADNA) is the arbitrary precision of stochastic variables

Like in CADNA, the arithmetic and relational operators in SAM take into account round-off error propagation

The SAM library - II

- The SAM library is written in C++ and is based on MPFR
- All operators are overloaded $\rightarrow$ 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) : `type (mp_st)`

How to implement SAM

The use of the SAM library involves six steps :

- declaration of the SAM library for the compiler,
- initialization of the SAM library,
- substitution of the type `float` or `double` by stochastic types in variable declarations,
- possible changes in the input data if perturbation is desired, to take into account uncertainty in initial values,
- change of output statements to print stochastic results with their accuracy,
- termination of the SAM library.

Example of SAM code

```
#include "sam.h"
#include <stdio.h>

int main() {
   sam_init(-1);
   mpfr_set_default_prec(200);
   mp_st x = 77617.; mp_st y = 33096.; mp_st res;
   res=333.75*y*y*y*y*y*y*y+x*x*(11*x*x*y*y-y*y*y*y*y*y*y
      -121*y*y*y*y-2.0) +5.5*y*y*y*y*y*y*y*y+x/(2*y);
   printf("res=%s\n",strp(res));
   sam_end();
}
```

Application of SAM for chaotic dynamical systems

Logistic iteration :

$$x_{n+1} = a x_n (1 - x_n) \text{ with } a > 0 \text{ and } 0 < x_0 < 1$$

- 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.

Logistic map

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.

Comparison of SAM and MPFI - I

Number N of iterations performed with SAM and MPFI, for $x_{n+1} = ax_n(1 - x_n)$ with $x_0 = 0.6$.

a		# bits	N
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

Comparison of SAM and MPFI - II

Number N of iterations performed with SAM and MPFI, $x_{n+1} = -a(x_n - \frac{1}{2})^2 + \frac{a}{4}$

a		# bits	N
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

Conclusion and future work

Conclusion :

- An efficient library for validation of scientific code
- Possibility to choose any working precision

Future work :

- Lorenz attractor
- multiple roots of polynomial
- computation of integrals

Thank you for your attention