Numerical validation using stochastic arithmetic

Stef Graillat, Fabienne Jézéquel, Jean-Luc Lamotte

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

Journées Développement Logiciel
de l'Enseignement Supérieur et de la Recherche

Ecole Polytechnique, 4-6 septembre 2013

Round-off error analysis

Several approaches

- Inverse analysis
  based on the “ Wilkinson principle”: the computed solution is assumed to be the exact solution of a nearby problem
  - provides error bounds for the computed results
- Interval arithmetic
  The result of an operation between two intervals contains all values that can be obtained by performing this operation on elements from each interval.
  - guaranteed bounds for each computed result
  - the error may be overestimated
  - specific algorithms
- Probabilistic approach
  - uses a random rounding mode
  - estimates the number of exact significant digits of any computed result

The CESTAC method

M. La Porte, J. Vignes, 1974

The implementation of the CESTAC method in a code providing a result R consists in:

- performing N times this code with the random rounding mode to obtain N samples R_i of R,
- choosing as the computed result the mean value $\overline{R}$ of R_i, $i = 1, ..., N$,
- estimating the number of exact significant decimal digits of $\overline{R}$ with

$$C_{\overline{R}} = \log_{10}\left(\frac{\sqrt{N}\,|\overline{R}|}{\sigma\tau_\beta}\right)$$

where

$$\overline{R} = \frac{1}{N}\sum_{i=1}^{N} R_i \quad \text{and} \quad \sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(R_i - \overline{R}\right)^2.$$

τ_β is the value of Student's distribution for $N-1$ degrees of freedom and a probability level β.

In pratice, $N = 3$ and $\beta = 95\%$.

Self-validation of the CESTAC method

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

- A multiplication of two insignificant results
- or a division by an insignificant 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.

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:
$X - Y$ **an insignificant value**.

New methods for numerical algorithms may be developed.

The concept of computed zero

J. Vignes, 1986

Definition

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

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

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

The stochastic definitions

Definition

Let X and Y be two results computed using the CESTAC method (N-sample), X is stochastically equal to Y, noted $X \stackrel{s}{=} Y$, if and only if

$$X - Y = @.0.$$

Definition

Let X and Y be two results computed using the CESTAC method (N-sample).

- X is stochastically strictly greater than Y, noted $X \stackrel{s}{>} Y$, if and only if

$$\overline{X} > \overline{Y} \quad \text{and} \quad X \stackrel{s}{\neq} Y$$

- X is stochastically greater than or equal to Y, noted $X \stackrel{s}{\geq} Y$, if and only if

$$\overline{X} \geq \overline{Y} \quad \text{or} \quad X \stackrel{s}{=} Y$$

Discrete Stochastic Arithmetic (DSA) is defined as the joint use of the CESTAC method, the computed zero and the stochastic relation definitions.

The CADNA library http://www.lip6.fr/cadna

The CADNA library implements Discrete Stochastic Arithmetic.

CADNA allows to estimate round-off error propagation in any scientific program written in Fortran or in C++.

More precisely, CADNA enables one to:

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

CADNA provides new numerical types, the stochastic types, which consist of:

- 3 floating point variables
- an integer variable to store the accuracy.

All operators and mathematical functions are overloaded for these types.

How to implement CADNA

The use of the CADNA library involves six steps:

- declaration of the CADNA library for the compiler,
- initialization of the CADNA library,
- substitution of the type REAL or DOUBLE PRECISION 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 CADNA library.

An example proposed by S. Rump (1)

Computation of $f(10864, 18817)$ and $f(\frac{1}{3}, \frac{2}{3})$ with $f(x, y) = 9x^4 - y^4 + 2y^2$

```
      program ex1
      implicit double precision (a-h,o-z)
      x = 10864.d0
      y = 18817.d0
      write(*,*)'P(10864,18817) = ', rump(x,y)
      x = 1.d0/3.d0
      y = 2.d0/3.d0
      write(6,100) rump(x,y)
100   format('P(1/3,2/3) = ',e24.15)
      end

      function rump(x,y)
      implicit double precision (a-h,o-z)
      a=9.d0*x*x*x*x
      b=y*y*y*y
      c=2.d0*y*y
      rump = a-b+c
      return
      end
```

An example proposed by S. Rump (2)

The results:

```
P(10864,18817) =     2.000000000000000
P(1/3,2/3) =     0.8024691358024690E+00
```

```
program ex1

implicit double precision  (a-h,o-z)

x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)

end

function rump(x,y)

implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

```
program ex1
use cadna
implicit double precision  (a-h,o-z)

x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)

end

function rump(x,y)
use cadna
implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

```
program ex1
use cadna
implicit double precision  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)

end

function rump(x,y)
use cadna
implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

```
program ex1
use cadna
implicit double precision  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
call cadna_end()
end

function rump(x,y)
use cadna
implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

```
program ex1
use cadna
implicit double precision  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
call cadna_end()
end

function rump(x,y)
use cadna
implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

```
program ex1
use cadna
implicit type(double_st)  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
call cadna_end()
end

function rump(x,y)
use cadna
implicit type(double_st)  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

```
program ex1
use cadna
implicit type(double_st)  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
call cadna_end()
end

function rump(x,y)
use cadna
implicit type(double_st)  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

```
program ex1
use cadna
implicit type(double_st)  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ',str(rump(x,y))
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ',str(rump(x,y))
call cadna_end()
end

function rump(x,y)
use cadna
implicit type(double_st)  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

The run with CADNA

CADNA software — University P. et M. Curie — LIP6
Self-validation detection: ON
Mathematical instabilities detection: ON
Branching instabilities detection: ON
Intrinsic instabilities detection: ON
Cancellation instabilities detection: ON

P(10864,18817) = @.0
P(1/3,2/3) = 0.80246913580246 9E+000

CADNA software — University P. et M. Curie — LIP6
There are 2 numerical instabilities
0 UNSTABLE DIVISION(S)
0 UNSTABLE POWER FUNCTION(S)
0 UNSTABLE MULTIPLICATION(S)
0 UNSTABLE BRANCHING(S)
0 UNSTABLE MATHEMATICAL FUNCTION(S)
0 UNSTABLE INTRINSIC FUNCTION(S)
2 UNSTABLE CANCELLATION(S)

Contributions of CADNA

- In direct methods:
  - estimate the numerical quality of the results
  - control branching statements
- In iterative methods:
  - optimize the number of iterations
  - check if the computed solution is satisfactory
- In approximation methods:
  - optimize the integration step

Conclusion

- ☺ can be used on real life applications
- ☹ difficult to understand the numerical instabilities in large codes
- ☹ time and memory consuming
- ☺ solution for parallel programs (MPI and GPU)
- ☹ difficult to use with the libraries (BLAS, LAPACK ...)

On the probability of the confidence interval

With $\beta = 95\%$ and $N = 3$,

- the probability of overestimating the number of exact significant digits of at least 1 is 0.054%
- the probability of underestimating the number of exact significant digits of at least 1 is 29%.

By choosing a confidence interval at 95%, we prefer to guarantee a minimal number of exact significant digits with high probability (99.946%), even if we are often pessimistic by 1 digit.