

# The CADNA software

Jean-Marie CHESNEAUX    Fabienne JÉZÉQUEL

Laboratoire d'Informatique de Paris 6  
Université Pierre et Marie Curie - Paris 6  
France

3rd MRS Network Workshop - Numerical Accuracy and Reliability  
15-16 January 2009

# Introduction (1)

A good knowledge of floating-point arithmetic should be required of all computer scientists!

For real time applications, the time step  $h$  may be  $10^{-1}$ .

Two possible implementations:

- do ...  
     $t = t + h$   
enddo
- do ...  
     $i = i + 1$   
     $t = h * i$   
enddo

$h$  is not a floating-point number  $\Rightarrow t = t + h$  generates an error after 100 hours, error  $\approx 0.34$ s

# Introduction (2)

It really happened!

During the 1st Gulf war (1991) in the control programs of Patriot missiles which had to intercept Scud missiles

At 1600 km/h, 0.34 s  $\Rightarrow$  500 m

The interception failed, 28 people were killed.

With the other implementation

do ...

$$i = i + 1$$

$$t = h * i$$

enddo

If no overflow occurs for  $i$ , with double precision IEEE arithmetic, the relative rounding error remains below  $10^{-15}$ .

# Representation of real numbers

In a floating-point arithmetic using the radix  $b$ ,

$$X = \varepsilon M b^E$$

is represented by:

- its sign  $\varepsilon$ , encoded on one digit (0 if  $X$  is positive, 1 if  $X$  is negative),
- its exponent  $E$ , a  $k$  digit integer,
- its mantissa  $M$ , encoded on  $p$  digits.

$$M = \sum_{i=0}^{p-1} a_i b^{-i} \text{ and } a_i \in \{0, \dots, b-1\}.$$

Floating-point numbers are usually normalized:

$a_0 \neq 0$ ,  $M \in [1, b)$  and zero has a special representation.

# The IEEE 754 standard

The IEEE 754 standard specifies the **single precision** format and **double precision** format, both using the radix 2.

1	2 ...	9	10	.....	32
s	$E + 2^7 - 1$	$a_1$	.....		$a_{23}$

## IEEE 754 single precision

1	2 ...	12	13	.....	64
s	$E + 2^{10} - 1$	$a_1$	.....		$a_{52}$

## IEEE 754 double precision

# Rounding mode

Let  $\mathbb{F}$  be the set of all floating-point numbers.

Let  $X_{min}$  (resp.  $X_{max}$ ) be the smallest (resp. the greatest) floating point number.

Let  $x$  be a real number which is not machine representable.

$$\text{If } x \in (X_{min}, X_{max}), \exists \{X^-, X^+\} \subset \mathbb{F}^2$$

such that

$$X^- < x < X^+ \text{ and } (X^-, X^+) \cap \mathbb{F} = \emptyset$$

A rounding mode is a rule which, from  $x$ , provides  $X^-$  or  $X^+$ .

This rounding occurs at each assignment and at each arithmetic operation.

# The 4 rounding modes defined by the IEEE 754 standard

- **Rounding towards  $+\infty$ :**  $x$  is represented by  $X^+$ .
- **Rounding towards  $-\infty$ :**  $x$  is represented by  $X^-$ .
- **Rounding towards 0:** if  $x$  is negative, then it is represented by  $X^+$ , if  $x$  is positive, then it is represented by  $X^-$
- **Rounding to the nearest:**  $x$  is represented by its nearest machine number

# A significant example - I

$$0.3 * x^2 + 2.1 * x + 3.675 = 0$$

- **Rounding towards plus infinity**

$$d = 3.81470E-06$$

There are two different real roots.

$$x1 = -.3500977E+01$$

$$x2 = -.3499024E+01$$

- **Rounding towards minus infinity**

$$d = 0.$$

The discriminant is null.

The double real root is  $-.3500000E+01$

## A significant example - II

$$0.3 * x^2 + 2.1 * x + 3.675 = 0$$

- **Rounding towards zero**

$d = 0$ .

The discriminant is null.

The double real root is  $-.3500000E+01$

- **Rounding to the nearest**

$d = -3.81470E-06$

There are two conjugate complex roots.

$z1 = -.3500000E+01 + i * 0.9765625E-03$

$z2 = -.3500000E+01 + i * -.9765625E-03$

Let  $r \in \mathbb{R}$  be the exact result of  $n$  arithmetic operations.

The computed result  $R \in \mathbb{F}$  can be modelled to the first order in  $2^{-p}$  as

$$R \approx r + \sum_{i=1}^n g_i(d) 2^{-p} \alpha_i$$

where  $p$  is the number of bits in the mantissa,  $\alpha_i$  are independent random variables uniformly distributed on  $[-1, 1]$  and  $g_i(d)$  are coefficients depending exclusively on the data and on the code.

# A theorem on numerical accuracy

The number of significant bits in common between  $R$  and  $r$  is

$$C_R \approx -\log_2 \left| \frac{R-r}{r} \right| = p - \log_2 \left| \sum_{i=1}^n g_i(d) \frac{\alpha_i}{r} \right|$$

The last term represents the loss of accuracy in the computation of  $R$ . It is independent of  $p$ .

## Theorem

*The loss of accuracy during a numerical computation is independent of the precision used.*

# 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.

By running  $N$  times the code, one obtains an  $N$ -sample of the random variable modelled by

$$R \approx r + \sum_{i=1}^n g_i(d) 2^{-p} \alpha_i$$

where the  $\alpha_i$ 's are modelled by independent random variables identically and uniformly distributed on  $[-1, +1]$ .

⇒ the mathematical expectation of  $R$  is the mathematical result  $r$ ,

⇒ the distribution of  $R$  is a quasi-Gaussian distribution.

# Application of Student test

$\forall \beta \in [0, 1], \exists \tau_\beta \in \mathbb{R}$  such that

$$P\left(r \in \left[\bar{R} - \frac{\tau_\beta s}{\sqrt{N}}, \bar{R} + \frac{\tau_\beta s}{\sqrt{N}}\right]\right) = P\left(|\bar{R} - r| \leq \frac{\tau_\beta s}{\sqrt{N}}\right) = \beta$$

with

$$\bar{R} = \frac{1}{N} \sum_{i=1}^N R_i \quad \text{and} \quad s^2 = \frac{1}{N-1} \sum_{i=1}^N (R_i - \bar{R})^2.$$

With a probability  $\beta$ , the number of exact significant digits of  $\bar{R}$  is undervalued by

$$C_{\bar{R}} = \log_{10} \left( \frac{\sqrt{N} |\bar{R}|}{s \tau_\beta} \right).$$

# Implementation of the CESTAC method

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

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

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

In practice,  $N = 2$  or  $N = 3$  and  $\beta = 0.05$ .

## On the number of runs

2 or 3 runs are enough. Increasing the number of runs is not necessary.

From the model, to increase by 1 the number of exact significant digits given by  $C_{\bar{R}}$ , we need to multiply the size of the sample by 100.

Such an increase will only point out the limit of the model without really improving the quality of the estimation.

It has been shown that  $N = 3$  is the optimal value.

# On the probability of the confidence interval

With  $\beta = 0.05$  and  $N = 3$ ,

- the probability of overestimating the number of exact significant digits of at least 1 is 0.00054.
- the probability of underestimating the number of exact significant digits of at least 1 is 0.29.

By choosing a confidence interval at 95%, we prefer to guarantee a minimal number of exact significant digits with a high probability (0.99946), even if we are often pessimistic by 1 digit.

# Inconsistency of the floating-point arithmetic

On a computer,

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

⇒ 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
```

$\varepsilon$  too low  $\implies$  a risk of infinite loop

$\varepsilon$  too high  $\implies$  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

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

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

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

## Definition

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  $\bar{X} > \bar{Y}$  and  $X s\neq Y$ .  
 $X s\geq Y$  if and only if  $\bar{X} \geq \bar{Y}$  and  $X s= Y$ .

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

# A few properties

- $x = 0 \implies X = @.0$ .
- $X \text{ s}\neq Y \implies x \neq y$ .
- $X \text{ s}\> Y \implies x > y$ .
- $x \geq y \implies X \text{ s}\geq Y$ .
- The relation  $\text{s}\>$  is transitive.
- The relation  $\text{s}=\text{}$  is reflexive, symmetric but not transitive.
- The relation  $\text{s}\geq$  is reflexive, antisymmetric but not transitive.

# 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.

# 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 and also with MPI parallel programs.

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

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.

# 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.

# Declaration of the CADNA library

The `use CADNA` pseudo-statement must take place before any declaration of stochastic variables.

As usual in a Fortran 90 source code this statement must be added:

- after one among the following lines
  - **PROGRAM** that begins an application
  - **MODULE** that begins a module
  - **SUBROUTINE** if it begins an “isolated subroutine”, i.e. a subroutine that is not declared within the scope of a program or a module declaration.
  - **FUNCTION** if it begins an “isolated function”, i.e. a function that is not declared within the scope of a program or a module declaration
- before any declaration.

# Initialization of the CADNA library

The call to the `cadna_init` subroutine must be added just after the main program declaration statements to initialize the library.

```
cadna_init (numb_instability, cadna_instability,  
           cancel_level, init_random)
```

- if `numb_instability = -1` : all the instabilities will be detected
- if `numb_instability = 0` : no instability will be detected
- if `numb_instability = M` : (strictly positive M), the M first instabilities will be detected.

The other arguments are optional.

## Initialization of the CADNA library (2)

`cadna_instability` : describes the instabilities which are disabled

- `cadna_branching`, `cadna_mathematic`, `cadna_intrinsic`, `cadna_cancellation`
- `cadna_division`, `cadna_power`, `cadna_multiplication`
- `cadna_all`

`cancel_level` : When one loses more than `cancel_level` significant digits in one addition or subtraction, CADNA considers that a catastrophic cancellation has been detected. Default value is 4.

`init_random` : the seed of the random generator used by the CADNA library.

The call to the `cadna_end` subroutine must finish the source code. It should be the last statement.

The `cadna_end` subroutine writes on the standard output a report on the numerical stability of the run.

# Changes in the type of variables

To control the numerical quality of a variable, one changes its standard type into the associated stochastic type.

```
real, float*4            $\implies$  type(single_st)  
double precision, double, float*8  $\implies$  type(double_st)
```

Example :

```
type (single_st), dimension(6) :: a,b,c  
type (double_st) :: e,f,g  
implicit type (single_st) (A-H,O-Z)
```

# Operator Modifications

All arithmetic operators on floating-point variables are overloaded. Arithmetic expressions without functions do not have to be modified.

Expressions may contain a mixture of stochastic types, classical types and integer types.

In Fortran 90, some arithmetic operators and intrinsic functions are generalized to act on arrays. In this version of CADNA, only arithmetic operators and the `abs`, `min`, `max`, `sqrt` functions are overloaded for stochastic arrays.

# Changes in printing statements

Before printing each stochastic variable, it must be transformed in a string by the `str` function. The required length is 14 for a `single_st` variable and 23 for a `double_st` variable. Therefore formats should be modified.

Initial Fortran statements	Modified statements for CADNA
<pre>real :: x  ... write(6,100) x 100 format(1x,'x=',f8.3)</pre>	<pre><b>use cadna</b> <b>type (single_st) :: x</b> <b>call cadna_init(-1)</b>  ... write(6,100) <b>str(x)</b> 100 format(1x,'x=',<b>a14</b>)</pre>

# Changes in reading statements

The generic function `read` is adapted to classical floating-point variables, which must be transformed into stochastic variables.

Example :

Initial Fortran statements	Modified statements for CADNA
<pre>real :: x  ..... read (5,*) x</pre>	<pre><b>use cadna</b> <b>real :: xaux</b> <b>type (single_st) :: x</b> <b>call cadna_init(-1)</b>  ..... read (5,*) <b>xaux</b> <b>x=xaux</b></pre>

## the `data_st` function

It allows us to take into account error on data by perturbing the samples of a stochastic variable  $X$ .

- `data_st (X)` : perturbation of the last bit of the mantissa.
- `data_st (X, ERX, 0)` : relative error

$$X_i = X_i * (1 + ERX * ALEA)$$

- `data_st (X, ERX, 1)` : absolute error

$$X_i = X_i + (ERX * ALEA)$$

*ALEA* is a random variable uniformly distributed on  $[-1, 1]$ .

```
type (single_st) :: b
b=-2.1
call data_st(b, 0.1, 0)
```

The 3-sample becomes:

```
-2.309487      -1.980967      -2.100000
```

- 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

## In direct methods - Example 1

$$0.3x^2 - 2.1x + 3.675 = 0$$

Without CADNA, in single precision with rounding to the nearest:

$d = -3.8146972E-06$

Two complex roots

$z1 = 0.3499999E+01 + i * 0.9765625E-03$

$z2 = 0.3499999E+01 + i * -.9765625E-03$

With CADNA:

$d = @.0$

The discriminant is null

The double real root is  $0.3500000E+01$

## In direct methods - Example 2

Computation of the determinant of Hilbert matrix of dimension 11 using Gaussian elimination

without CADNA	with CADNA
p6 = <b>0.1431549050</b> 48182E-05	p6 = 0.14315490506E-005
p7 = <b>0.9009749236</b> 43140E-07	p7 = 0.9009749273E-007
p8 = <b>0.5659970607</b> 16175E-08	p8 = 0.56599705E-008
p9 = <b>0.3551362553</b> 32890E-09	p9 = 0.3551346E-009
p10= <b>0.2226656943</b> 06967E-10	p10= 0.22264E-010
p11= <b>0.1398301799</b> 86415E-11	p11= 0.140E-011
D = <b>0.3026439382</b> 718219E-64	D = 0.302E-064

# In iterative methods

The equation

$$1.47x^3 + 1.19x^2 - 1.83x + 0.45 = 0$$

is solved using Newton's method with  $x_0 = 0.5$ .

The stopping criterion is  $\|x_n - x_{n+1}\| \leq 10^{-20}$ .

rounding to the nearest:	$x(24) = 0.4285714332352813$
rounding towards zero	$x(24) = 0.4285714343480436$
rounding towards $-\infty$ :	$x(24) = 0.4285714343480436$
rounding towards $+\infty$ :	$x(30) = 0.4285714326546251$
with CADNA :	$x(25) = 0.428571433$

With CADNA the stopping criterion is **x.eq.y**

# Which strategy to adopt?

- problems with a solution that can be controlled:  
the solution  $x_s$  satisfies  $\Psi(x_s) = 0$ .  
The optimal stopping criterion should be used

IF ( $\Psi(x(k)).eq.0$ ) THEN

- problems with a solution that cannot be controlled (sequence computation):  
The following stopping criterion should be used

IF ( $x(k).eq.x(k + 1)$ ) THEN

How to estimate the optimal step ?

If  $h$  decreases,  $X(h)$  : 

s	exponent	mantissa
---	----------	----------

  
 $e_m(h) \longrightarrow$   
 $\longleftarrow e_c(h)$

If  $e_c(h) < e_m(h)$ , decreasing  $h$  brings reliable information.

Computation should stop when  $e_c(h) \approx e_m(h)$

# Approximation of integrals

$I = \int_a^b f(x)dx$  is computed using a quadrature method (trapezoidal rule, Simpson's rule, ...)

Let  $I_n$  be the approximation computed with step  $h = \frac{b-a}{2^n}$ .

The computation stops when  $I_n - I_{n+1} = \epsilon$ .

```
DO WHILE (integold .NE. integ)
```

```
  integold = integ
```

```
  h=h/2
```

```
  ...
```

```
  integ = h * ( ... )
```

```
ENDDO
```

Using this strategy, the significant digits of the result which are not affected by round-off errors are in common with  $I$ , up to one.

A demo ?

# Solving a linear system $AX = B$

$$A = \begin{bmatrix} 21.0 & 130.0 & 0.0 & 2.1 \\ 13.0 & 80.0 & 4.74e+8 & 752.0 \\ 0.0 & -0.4 & 3.9816e+8 & 4.2 \\ 0.0 & 0.0 & 1.7 & 9.0e-9 \end{bmatrix} \quad B = \begin{bmatrix} 153.1 \\ 849.74 \\ 7.7816 \\ 2.6e-8 \end{bmatrix}$$

$$\text{The solution is } = \begin{bmatrix} 1. \\ 1. \\ 1.e-8 \\ 1. \end{bmatrix}$$

Gaussian elimination method with total pivoting is implemented.