Stochastic arithmetic and verification of mathematical models

Jean-Marie Chesneaux, Fabienne Jézéquel,
Jean-Luc Lamotte, Jean Vignes

Laboratoire d'Informatique de Paris 6,

P. and M. Curie University, Paris, France

SMAI-2011

Overview

- IEEE-754
- The CESTAC method and the stochastic arithmetic
- The CADNA software
- Examples

IEEE-754 standard

The IEEE-754 standard defines, in particular, the **single precision** and **double precision** coding:

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 real numbers which can be coded exactly on computer : the set of floating point numbers.

Every real number which is not a floating point number is approximated by a floating point number $X \in \mathbb{F}$.

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

$$\forall x \in]X_{min}, X_{max}[\,,\ \exists \left\{X^-, X^+\right\} \in \mathbb{F}^2$$

such that

$$X^- < x < X^+ \text{ and } \left]X^-, X^+\right[\cap \mathbb{F} = \emptyset$$

To choose the rounding mode is to choose the algorithm that, according to x, gives X^- or X^+.

IEEE 754 standard : 4 rounding modes

A significant example - I

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

- **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

- **Rounding to zero**
  d = 0.
  The discriminant is null.
  The double real root is -.3500000E+01

- **Rounding to plus infinity**
  d = 3.81470E-06
  There are two different real roots.
  x1 = -.3500977E+01
  x2 = -.3499024E+01

- **Rounding to minus infinity**
  d = 0.
  The discriminant is null.
  The double real root is -.3500000E+01

Inconsistency of the floating point arithmetic

On computer, arithmetic operators are only approximations.

- commutativity
- no associativity
- no distributivity

On computer, order relationships are the same as in mathematics

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

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

Round-off error model

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

On computer, one obtains the result $R \in \mathbb{F}$ which is affected by round-off errors.

R can be modeled, at the first order with respect to 2^{-p}, by

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

p is the number of bits used for the representation including the hidden bit, $g_i(d)$ are coefficients depending only on data and α_i are the round-off errors.

Remark : we have assumed that exponents and signs of intermediate results do not depend on α_i.

A theorem on numerical accuracy

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

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

The last part corresponds to the accuracy which has been lost in the computation of R, we can note that it is independent of p.

Theorem

The loss of accuracy during a numerical computation on computer is independent of the precision used for the representation of floating point numbers.

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.

By running N times the code with the random arithmetic, one obtains a N-sample of the random variable modeled by

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

where the α_i's are modeled by independent identically distributed random variables. The common distribution of the α_i is uniform on $[-1, +1]$.

$\Rightarrow$ the mathematical expectation of R is the mathematical result r,

$\Rightarrow$ the distribution of R is a quasi-Gaussian distribution.

Implementation of the CESTAC method

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 $1-\beta$.

On the number of runs

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

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

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

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

Self-validation of the CESTAC method

The CESTAC method is based on a 1^{st} 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 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 small $\Longrightarrow$ a risk of infinite loop
ε too big $\Longrightarrow$ a too early termination.

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

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

This means that 0 belongs to the confidence interval.

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 computed results 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 computed results using the CESTAC method (N-sample), X is stochastically strictly greater than Y, noted $X \stackrel{s}{>} Y$, if and only if

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

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

A few properties

- $x = 0 \implies X = @.0$.
- $X \stackrel{s}{\neq} Y \implies x \neq y$.
- $X \stackrel{s}{>} Y \implies x > y$.
- $x \geq y \implies X \stackrel{s}{\geq} Y$.

The CADNA library - I

CADNA implements Discrete Stochastic Arithmetic
CADNA enables one to use new numerical types: the stochastic types.

- to estimate the error due to round-off error propagation,
- to detect numerical instabilities,
- to check the sequencing of the program (tests and branchings),
- to estimate the accuracy of all intermediate computations,
- a true numerical debugging.

CADNA is a library which can be used with Fortran or C++ programs and also with MPI parallel programs.

Programs have to be written in FORTRAN, C or C++
The cost of CADNA is about 4 for memory and 10 for run time.
CADNA can be downloaded from http://www.lip6.fr/cadna

One example proposed by S. Rump

$$f(x,y) = 9x^4 - y^4 + 2y^2$$

Computation of $f(10864, 18817)$ and $f(\frac{1}{3}, \frac{2}{3})$

```
      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)
      v1=9.d0*x**4
      v2=y**4
      v3=2.d0*y**2
```

One 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  (a-h,o-z)
rump = 9.d0*x**4 - y**4 + 2.d0*y**2
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  (a-h,o-z)
rump = 9.d0*x**4 - y**4 + 2.d0*y**2
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  (a-h,o-z)
rump = 9.d0*x**4 - y**4 + 2.d0*y**2
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  (a-h,o-z)
rump = 9.d0*x**4 - y**4 + 2.d0*y**2
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  (a-h,o-z)
rump = 9.d0*x**4 - y**4 + 2.d0*y**2
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)
rump = 9.d0*x**4 - y**4 + 2.d0*y**2
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)
rump = 9.d0*x**4 - y**4 + 2.d0*y**2
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)
rump = 9.d0*x**4 - y**4 + 2.d0*y**2
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) =   0.8024691358024 69E+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)
```

3 classes of algorithm

- finite numerical methods
- iterative methods:
  - the solution can be controlled
  - the solution can NOT be controlled
- approximation methods

Finite numerical methods

solving a linear system

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

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

Gaussian elimination method with partial pivoting is implemented.

```
| Solving a linear system using Gaussian elimination |
| with partial pivoting                              |
```

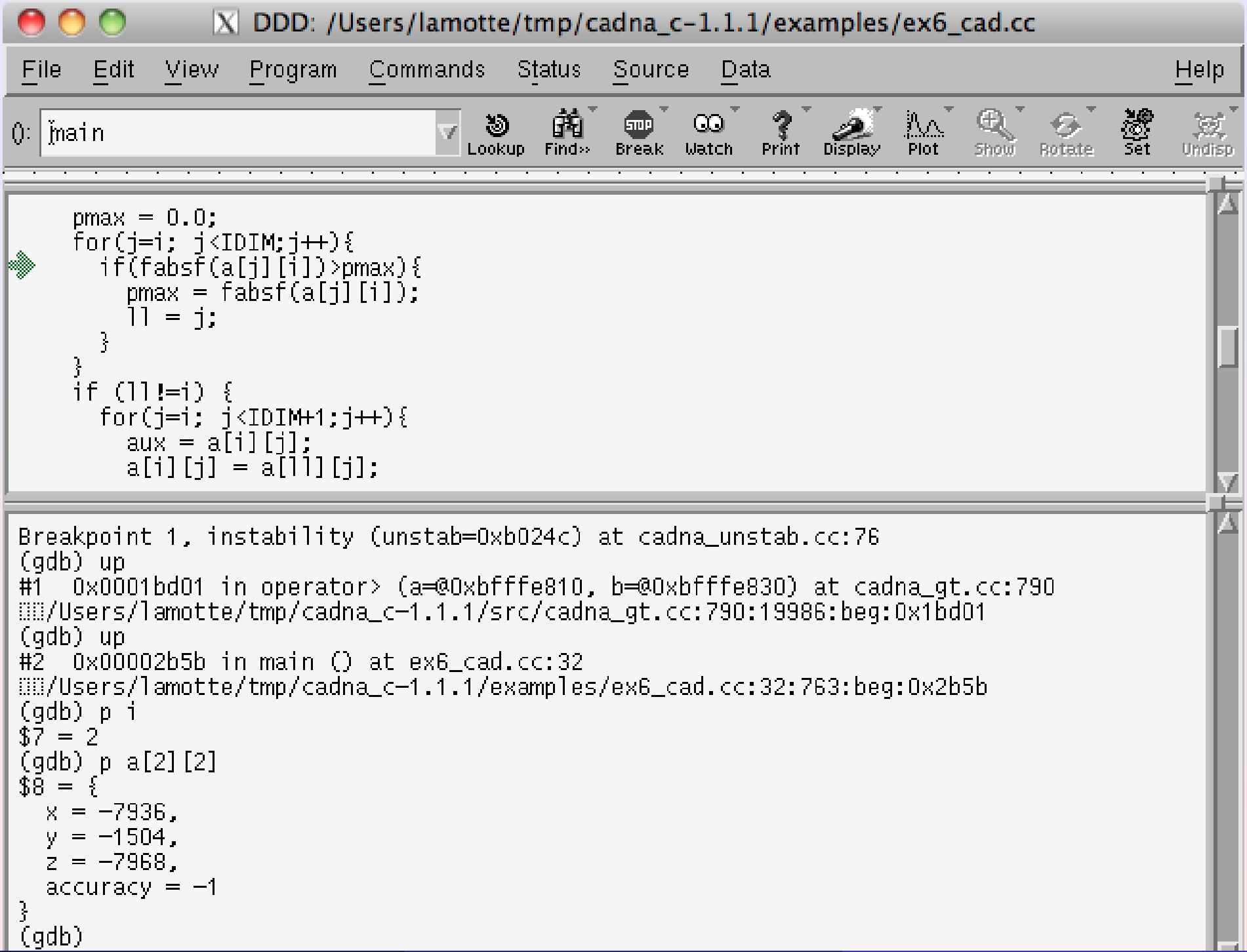
```
x_sol(0) = +6.261988e+01 (solution: xsol(0)= +1.000000e+00)
x_sol(1) = -8.953979e+00 (solution: xsol(1)= +1.000000e+00)
x_sol(2) = +0.000000e+00 (solution: xsol(2)= +1.000000e-08)
x_sol(3) = +9.999999e-01 (solution: xsol(3)= +1.000000e+00)
```

```
CADNA_C 1.1.1 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
```

```
| Solving a linear system using Gaussian elimination |
| with partial pivoting                              |
```

```
x_sol(0) =  0.10E+001 ( solution: xsol(0)=  0.1000000E+001)
x_sol(1) =  0.999E+000 ( solution: xsol(1)=  0.1000000E+001)
x_sol(2) =  0.999999E-008 ( solution: xsol(2)=  0.1000000E-007)
x_sol(3) =  0.100000E+001 ( solution: xsol(3)=  0.1000000E+001)
```

```
CADNA_C 1.1.1 software --- University P. et M. Curie --- LIP6
There are 2 numerical instabilities
1 UNSTABLE BRANCHING(S)
1 LOSS OF ACCURACY DUE TO CANCELLATION(S)
```


DDD: /Users/lamotte/tmp/cadna_c-1.1.1/examples/ex6_cad.cc
File Edit View Program Commands Status Source Data Help
(): main
Lookup Find» Break Watch Print Display Plot Show Rotate Set Undisp
pmax = 0.0;
for(j=i; j<IDIM;j++){
if(fabsf(a[j][i])>pmax){
pmax = fabsf(a[j][i]);
ll = j;
}
}
if (ll!=i) {
for(j=i; j<IDIM+1;j++){
aux = a[i][j];
a[i][j] = a[ll][j];
Breakpoint 1, instability (unstab=0xb024c) at cadna_unstab.cc:76
(gdb) up
#1 0x0001bd01 in operator> (a=@0xbfffe810, b=@0xbfffe830) at cadna_gt.cc:790
/Users/lamotte/tmp/cadna_c-1.1.1/src/cadna_gt.cc:790:19986:beg:0x1bd01
(gdb) up
#2 0x00002b5b in main () at ex6_cad.cc:32
/Users/lamotte/tmp/cadna_c-1.1.1/examples/ex6_cad.cc:32:763:beg:0x2b5b
(gdb) p i
$7 = 2
(gdb) p a[2][2]
$8 = {
x = -7936,
y = -1504,
z = -7968,
accuracy = -1
}
(gdb)

Iterative methods: the solution can NOT be controlled

Examples: series, limits, ...

- with the IEEE arithmetic
  - test on the difference between 2 iterations
  - $||X_{k+1} - X_k|| \leq \varepsilon$
  - $||X_{k+1} - X_k|| \leq \varepsilon ||X_k||$
- with CADNA:
  - $||X_{k+1} - X_k|| = @.0$

Iterative methods: the solution can be controlled

General scheme:

- $x_{n+1} = f(x_n)$ with $n \in \mathbb{R}$
- $\exists \phi(x)$ such that $f(x^*) = 0$ and $\phi(x^*) = 0$

The computation must be stopped when

- $\phi(x_k) = 0$
- $||X_k - X_{k-1}|| = 0$, stationarity. It is necessary to test $\phi(x_k) = 0$
- $k > Nmax$ to avoid the diverging.

Iterative methods:

The Newton method to find a solution of:

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

$\epsilon = 1.e - 12$
Without CADNA

x(35) = +4.285714252078272e−01
x(36) = +4.285714252078272e−01

Iterative methods:

The Newton method to find a solution of:

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

With CADNA

```
x( 82) =  0.42857142E+000
x( 83) =  0.42857142E+000
```

```
CADNA_C 1.1.1 software --- University P. et M. Curie --- LIP6

CRITICAL WARNING: the self-validation detects major problem(s).
The results are NOT guaranteed.

There are 387 numerical instabilities
59 UNSTABLE DIVISION(S)
59 UNSTABLE BRANCHING(S)
45 UNSTABLE INTRINSIC FUNCTION(S)
224 LOSS OF ACCURACY DUE TO CANCELLATION(S)
```

Iterative methods: the solution can be controlled

Application: a solver of linear system $A.X = B$ based on Jacobi's method

$$x_i^{(k+1)} = -\frac{1}{a_{ii}} \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} + \frac{b_i}{a_{ii}}$$

Stop when $\max_{i=1}^{n} |x_i^k - x_i^{k-1}| < \varepsilon$ without CADNA
check if the solution

$$\phi(X_k) = B - A.X_k$$

$eps = 1.e-3$

```
niter =           35
x_sol( 1)= 0.1699924E+01 (true: 0.1700000E+01), r( 1)= 0.3051758E-03
x_sol( 2)=-0.4746889E+04 (true:-0.4746890E+04), r( 2)= 0.1953125E-02
x_sol( 3)= 0.5023049E+02 (true: 0.5023000E+02), r( 3)= 0.1464844E-02
x_sol( 4)=-0.2453197E+03 (true:-0.2453200E+03), r( 4)=-0.7324219E-03
x_sol( 5)= 0.4778290E+04 (true: 0.4778290E+04), r( 5)=-0.4882812E-03
x_sol( 6)=-0.7572980E+02 (true:-0.7573000E+02), r( 6)= 0.9765625E-03
x_sol( 7)= 0.3495430E+04 (true: 0.3495430E+04), r( 7)= 0.3173828E-02
x_sol( 8)= 0.4350277E+01 (true: 0.4350000E+01), r( 8)= 0.0000000E+00
x_sol( 9)= 0.4529804E+03 (true: 0.4529800E+03), r( 9)= 0.9765625E-03
x_sol(10)=-0.2759901E+01 (true:-0.2760000E+01), r(10)= 0.9765625E-03
x_sol(11)= 0.8239241E+04 (true: 0.8239240E+04), r(11)= 0.7568359E-02
x_sol(12)= 0.3459919E+01 (true: 0.3460000E+01), r(12)=-0.4882812E-03
x_sol(13)= 0.1000000E+04 (true: 0.1000000E+04), r(13)= 0.9765625E-03
x_sol(14)=-0.4999743E+01 (true:-0.5000000E+01), r(14)= 0.1464844E-02
x_sol(15)= 0.3642400E+04 (true: 0.3642400E+04), r(15)=-0.1953125E-02
x_sol(16)= 0.7353594E+03 (true: 0.7353600E+03), r(16)=-0.3662109E-03
x_sol(17)= 0.1700038E+01 (true: 0.1700000E+01), r(17)= 0.1464844E-02
x_sol(18)=-0.2349171E+04 (true:-0.2349170E+04), r(18)= 0.1953125E-02
x_sol(19)=-0.8247521E+04 (true:-0.8247520E+04), r(19)=-0.8728027E-02
x_sol(20)= 0.9843570E+04 (true: 0.9843570E+04), r(20)= 0.0000000E+00
```

$eps = 1.e-4$

```
niter =          1000
x_sol( 1)= 0.1699924E+01 (true: 0.1700000E+01), r( 1)= 0.1831055E-03
x_sol( 2)=-0.4746890E+04 (true:-0.4746890E+04), r( 2)=-0.4882812E-03
x_sol( 3)= 0.5022963E+02 (true: 0.5023000E+02), r( 3)=-0.9765625E-03
x_sol( 4)=-0.2453193E+03 (true:-0.2453200E+03), r( 4)= 0.1464844E-02
x_sol( 5)= 0.4778290E+04 (true: 0.4778290E+04), r( 5)=-0.1464844E-02
x_sol( 6)=-0.7573022E+02 (true:-0.7573000E+02), r( 6)=-0.1953125E-02
x_sol( 7)= 0.3495430E+04 (true: 0.3495430E+04), r( 7)= 0.5126953E-02
x_sol( 8)= 0.4350277E+01 (true: 0.4350000E+01), r( 8)=-0.4882812E-03
x_sol( 9)= 0.4529798E+03 (true: 0.4529800E+03), r( 9)=-0.9765625E-03
x_sol(10)=-0.2760255E+01 (true:-0.2760000E+01), r(10)=-0.1953125E-02
x_sol(11)= 0.8239240E+04 (true: 0.8239240E+04), r(11)= 0.3173828E-02
x_sol(12)= 0.3459731E+01 (true: 0.3460000E+01), r(12)=-0.1464844E-02
x_sol(13)= 0.1000000E+04 (true: 0.1000000E+04), r(13)=-0.1953125E-02
x_sol(14)=-0.4999743E+01 (true:-0.5000000E+01), r(14)= 0.1953125E-02
x_sol(15)= 0.3642400E+04 (true: 0.3642400E+04), r(15)= 0.0000000E+00
x_sol(16)= 0.7353599E+03 (true: 0.7353600E+03), r(16)=-0.7324219E-03
x_sol(17)= 0.1699763E+01 (true: 0.1700000E+01), r(17)=-0.4882812E-03
x_sol(18)=-0.2349171E+04 (true:-0.2349170E+04), r(18)= 0.0000000E+00
x_sol(19)=-0.8247520E+04 (true:-0.8247520E+04), r(19)=-0.9155273E-03
x_sol(20)= 0.9843570E+04 (true: 0.9843570E+04), r(20)=-0.3906250E-02
```

With CADNA

```
 niter =            29
x_sol( 1)= 0.170E+01      (true: 0.1699999E+01), r( 1)=@.0
x_sol( 2)=-0.4746888E+04 (true:-0.4746888E+04), r( 2)=@.0
x_sol( 3)= 0.5023E+02     (true: 0.5022998E+02), r( 3)=@.0
x_sol( 4)=-0.24532E+03    (true:-0.2453199E+03), r( 4)=@.0
x_sol( 5)= 0.4778287E+04 (true: 0.4778287E+04), r( 5)=@.0
x_sol( 6)=-0.75729E+02    (true:-0.7572999E+02), r( 6)=@.0
x_sol( 7)= 0.349543E+04   (true: 0.3495428E+04), r( 7)=@.0
x_sol( 8)= 0.435E+01      (true: 0.4349999E+01), r( 8)=@.0
x_sol( 9)= 0.45298E+03    (true: 0.4529798E+03), r( 9)=@.0
x_sol(10)=-0.276E+01      (true:-0.2759999E+01), r(10)=@.0
x_sol(11)= 0.823923E+04   (true: 0.8239236E+04), r(11)=@.0
x_sol(12)= 0.346E+01      (true: 0.3459999E+01), r(12)=@.0
x_sol(13)= 0.10000E+04    (true: 0.9999996E+03), r(13)=@.0
x_sol(14)=-0.5001E+01     (true:-0.4999999E+01), r(14)=@.0
x_sol(15)= 0.364239E+04   (true: 0.3642398E+04), r(15)=@.0
x_sol(16)= 0.73536E+03    (true: 0.7353597E+03), r(16)=@.0
x_sol(17)= 0.170E+01      (true: 0.1699999E+01), r(17)=@.0
x_sol(18)=-0.234917E+04   (true:-0.2349169E+04), r(18)=@.0
x_sol(19)=-0.8247515E+04 (true:-0.8247515E+04), r(19)=@.0
x_sol(20)= 0.984356E+04   (true: 0.9843565E+04), r(20)=@.0
```

Conclusion

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

On the probability of the confidence interval

Up to one digit, $\frac{\sqrt{N}\left|\overline{R}\right|}{10st_\beta} \leq \left|\overline{R} - r\right| \leq \frac{10\sqrt{N}\left|\overline{R}\right|}{st_\beta}$

with $\beta = 0.95$ and $N = 3$,

The probability of overestimating the number of exact significant digit of at least 1 is 0.00054.

The probability of underestimating the number of exact significant digit 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 high probability (0.99946), even if we are often pessimistic of 1 digit.

Explanation

```
>> a=9*x*x*x*x
a =
     1.253722838223421e+17
>> b=-y*y*y*y
b =
    -1.253722845305011e+17
>> a+b
ans =
  -708158976
>> c=2*y*y
c =
   708158978
>> a+b+c
ans =
     2
```