Journal of Universal Computer Science, vol. 1, no. 7 (1995), 454-468
submitted: 15/12/94, accepted: 26/6/95, appeared: 28/7/95©Springer Pub. Co.

ESTIMATION OF ROUND-OFF ERRORS ON SEVERAL COMPUTERS ARCHITECTURES

Jalil Asserrhine
E-mail:asserrhine@masi.ibp.fr

Jean-Marie Chesneaux
E-mail:chesneaux@masi.ibp.fr

Jean-Luc Lamotte
E-mail:lamotte@masi.ibp.fr

Laboratoire MASI-IBP, URA-818 du CNRS
Université Pierre et Marie Curie
4 place Jussieu, 75252 Paris Cedex 05 FRANCE

Abstract: Numerical validation of computed results in scientific computation is always an essential problem as well on sequential architecture as on parallel architecture. The probabilistic approach is the only one that allows to estimate the round-off error propagation of the floating point arithmetic on computers. We begin by recalling the basics of the CESTAC method (*Contrôle et Estimation STochastique des Arrondis de Calculs*). Then, the use of the CADNA software (*Control of Accuracy and Debugging For Numerical Applications*) is presented for numerical validation on sequential architecture. On parallel architecture, we present two solutions for the control of round-off errors. The first one is the combination of CADNA and the PVM library. This solution allows to control round-off errors of parallel codes with the same architecture. It does not need more processors than the classical parallel code. The second solution is represented by the RAPP prototype. In this approach, the CESTAC method is directly parallelized. It works both on sequential and parallel programs. The essential difference is that this solution requires more processors than the classical codes. These different approaches are tested on sequential and parallel programs of multiplication of matrices.

1 Introduction

On computers, using the floating point arithmetic, each elementary operation creates a round-off error because of the limited coding of real numbers. Consequently, every computed result is affected by the round-off error propagation. Then, to validate numerical results on computers we must estimate the number of exact significant digits of every result. Exact significant digits mean the digits in common between the computed result and the mathematical result. The probabilistic approach is the only one which allows to estimate the round-off error of the floating point arithmetic.

On a sequential architecture, the numerical validation of computation can be done by using the CADNA software which is based on the CESTAC method. But, more and more computations are now performed on parallel architectures. A tool for the numerical validation for such kind of computations has become essential.

After recalling the basics of the CESTAC method and explaining the use of the CADNA software for sequential architectures, we present in this note two prototypes for the control of round-off error on parallel architectures.

The first one combines CADNA and the message passing library PVM. In this approach each processor validates its own results using CADNA. The passing of results between processors is performed by adding a specific extension to PVM because of the new stochastic types used by CADNA.

The second one is a direct parallelization of the CESTAC method - the RAPP prototype - available both on sequential and parallel architectures.

2 The CESTAC method

In the probabilistic approach, the round-off errors are modeled by independent identically distributed (iid) random variables [Hamming 70][Hull and Swenson 66]. Therefore, a computed result R is also modeled by a random variable. The number of significant digits of R is estimated from notions like the mean value and the standard deviation of R.

Based on the probabilistic approach, the CESTAC method has been developed by J. Vignes and M. La Porte [Vignes 90][Vignes 93][Vignes and La Porte 74] to estimate the round-off error propagation of the floating point arithmetic. At each step of the computation, the chosen rounding of any intermediate result is the upper or the lower rounding with the probability 0.5 . Practically this is realized by perturbing the lowest bit of the mantissa after each elementary operation. This technique is called the random arithmetic. Then the code is run N times with this new arithmetic. By this way, N different results R_i are obtained.

The computed result is taken as the average of the R_i's :

$$\overline{R} = \frac{1}{N} \cdot \sum_{i=1}^{N} R_i.$$

The number of exact significant digits of $\overline{R}$ is given by the formula :

$$C_{\overline{R}} = Log_{10}\left(\frac{\sqrt{N}.\left|\overline{R}\right|}{s.t_\beta}\right),$$

with $s^2 = \frac{1}{N-1} \cdot \sum_{i=1}^{N} \left(R_i - \overline{R}\right)^2$ and t_β the level of confidence of the t distribution for a probability $(1-\beta)$.

It has been shown [Chesneaux 90] that a computed result obtained with the random arithmetic may be modeled by

$$Z = r + \sum_{i=1}^{n} u_i(d).2^{-p} z_i \ ,$$

where r is the mathematical result, n is the number of elementary operations, the $u_i(d)$'s are coefficients depending only on the data and the algorithm, p is the length of the mantissa (the hidden bit included) and the z_i's are iid centered random variables.

From the probabilistic point of view, the CESTAC method consists in applying Student's test on a sample of R (which is the R_i's). Then, an estimation of the mean value of R (in the model, it is the mathematical result) is obtained from a confidence interval.

The theoretical study has proved the validity of the CESTAC method on the model [Chesneaux 90]. Then the practical efficiency of the CESTAC method is based on the physical reality of the hypotheses and the approximations which have been assumed during the theoretical study.

Theses hypotheses are :

i) the signs and the exponents of the intermediate results are independent of the random arithmetic,

ii) the model of round-off errors by iid random variables is correct,

iii) the approximation of the R's distribution by the first order terms in 2^{-p} modeled by Z is correct,

iv) the approximation by the first four terms in the Edgeworth's development of the Z's distribution during the study of Student's test is correct,

v) the $u_i(d)$'s coefficients are *regular*, which means that none of them is of a greater order than the sum of the others.

In practice, the hypotheses i), ii), iv), v) are of little importance [Chesneaux 95].

The hypotheses iii) is the only one which may really make the CESTAC method fail. If it is not verified, the mathematical expectation of the terms of order greater than 2^{-2p} is not zero. Then, there could be a bias which is not of a smaller order than the standard deviation of Z. The mean value of R is not yet well modeled by the mathematical expectation of Z which is the mathematical result.

In fact, as the CESTAC method gives an estimation of the mean value of R, it is absolutely necessary that it is closed to the mathematical result according to the standard deviation of R, i.e., that the approximation at the first order is valid. If it is not true, there could be an overestimation of the accuracy of the computed result R.

Only multiplications between two non significant results (called stochastic zeroes) and divisions by non significant results may create preponderant terms of order higher than 2^{-2p} [Chesneaux 95]. Then, such operations must be detected at run-time and the user must be advised. It may also be shown that, for the CESTAC method to work efficently, a control of the accuracy of the operands during tests like $IF\ (A > B)\ THEN$ must be performed. The answer of the test must take the number of exact significant digits of each operand into account [Chesneaux 95].

All of this may be done very easily by using the synchronous implementation of the CESTAC method which allows to estimate the accuracy of any intermediate result at any time. It consists in performing in a complete parallel manner the N runs of a code using the random arithmetic [Vignes 90][Vignes 93]. This implementation allows to have a sample of N values for any variables at each step of the run and so to estimate the accuracy of any result. Then, it is possible to point out the denominators which are non significant, the unstable multiplications and other numerical unstabilities. It leads to a self-validation of the CESTAC method.

All the efficiency of the CADNA software is based on this self-validation.

3 The CADNA software for sequential architectures

CADNA [Chesneaux 92][Vignes 93] means *Control of Accuracy and Debugging for Numerical Applications*. The first goal of this software is the estimation of the round-off error in a scientific code using the floating point arithmetic. CADNA uses the synchronous implementation of the CESTAC method. It also implements and uses the definitions of the order and equality relations of the stochastic arithmetic [Chesneaux 95]. This enables to control the branching, that is, CADNA points out all the tests for which the computed and the real answer have opposite signs. This is the second goal of the software.

The third goal is the numerical debugging. With CADNA, users may detect numerical unstabilities that appear at run-time. We must emphazise that this kind of debugging does not deal with the logical validation of a code but with the ability of the computers to give correct results when the code is performed using the floating point arithmetic.

CADNA also includes all the control tests pointed out by the theoretical study to have an efficient use the CESTAC method. CADNA is copyrighted and marketed, it is the property of the Pierre and Marie Curie University.

CADNA works on codes written in FORTRAN 77 but requires a FORTRAN 90 compiler to generate executable codes. In practice, CADNA is a library which is used during the link. It implements three new numerical types - the stochastic types - and all the arithmetic operators for these new types. The control of round-off error propagation is only performed on variables of a stochastic type.

These types are :
- type (SINGLE_ST) : stochatic single precision;
- type (DOUBLE_ST) : stochastic double precision;
- type (COMPLEX_ST) : stochastic complex single precision.

Declarations of stochastic variables are of the same kind as for the classical numerical types. For instance,

TYPE (DOUBLE_ST) X, Y, Z

The estimation of the number of significant digits is available at any time on any stochastic variable. All the arithmetic operators and order relations have been overloaded for the stochastic types. Intrinsic mathematical functions of the FORTRAN 77 only exist by their generic name. In that way, it is very easy to use CADNA on old FORTRAN 77 codes almost without any modification.

In an arithmetic expression, stochastic types, floating types and integer types may be mixed. The classical rules of prevalence are applied with the prevalence of the stochastic types on the others.

In the writing procedures, only the significant digits are printed for the variables of stochastic types. In this way, it is very easy to see the accuracy of the results.

For the numerical debugging, CADNA detects at any time numerical unstabilities that appear at run-time and let a trace in a special file. With the symbolic debugger, the user may point out the operation which is responsible for the unstability.

The implementation of CADNA on sequential architecture consists in imbedding the computations in $\mathbb{R}^N$. This simple solution perfectly simulates the synchronous implementation of the CESTAC method. Each operation is performed N times. In that way and at any time, there exists a sample of N values for any stochastic variable for the control of accuracy, the branching and the numerical debugging. The use of CADNA multiplies the run time by a factor 3 or 5.

4 Interfacing of CADNA with PVM

Nowadays, parallel computers are more and more used for scientific softwares usually written in FORTRAN. It seems essential to develop a validation tool for numerical softwares on this kind of machine. As the existing parallel machines are very different in structure and programming techniques, we have intentionnally limited our tool to the parallel machine using the IEEE arithmetic for the floating point number and being programmed in MIMD (Multiple Instruction Multiple Data) mode. Then a program may be considered as a set of sequential processes allocated on the different processors of the computer and interacting together by message passing.

To create the numerical validation tool, we have proceeded in the following way : each sequential process uses the sequential CADNA library to locally validate its computations and exchanges by message passing variables of standard type or of stochastic type (to estimate the round-off error).

For the communications between the processors, the message passing library PVM (Parallel Virtual Machine) was chosen for several reasons. First, the concept of Virtual Machine is available on a large number of parallel computers and allows to create easily a parallel virtual machine from a network of sequential and heterogeneous machines. Secondly PVM has been welcome by the scientific community working on parallel computation. Finally it is a freeware software of easy access.

This section is made up of four paragraphs. First, we recall the principle of the foundation of PVM. Then, the different problems encountered to use FORTRAN 90, CADNA and PVM in a same program and the proposed solution are described. Starting with an example of a parallel program (multiplication of two matrices), we explain the modifications we have to perform on a FORTRAN source file to use CADNA library. Finally, the performance obtained on a Connection Machine 5 are presented and commented.

4.1 Presentation de PVM

P.V.M. (Parallel Virtual Machine) [Geist and al. 93] has been developed by the *Heterogeneous Network Project* which embodies research from the Oak Ridge National Laboratory and the universities of Tennessee and of Emory. The aim of this project was to do parallel computations on networks of heterogeneous machines which have different architectures and different representations for floating point numbers.

To interconnect machines, which may be of sequential, parallel or vectorial type, PVM is able to use different types of networks (Ethernet, FDDI, Token Ring, ...). Despite the global view of the computer and of the network, PVM allows the different processes to take the best advantage of the performance of the target machines.

To develop an application, the programmer may use of the C language, the FORTRAN 77 language and a message passing library of high level (synchronous and asynchronous sending and receiving, synchronisation of processes, broadcast of message passing, and concentration and diffusion of values (reduce)). For each message, the user's interface imposes to describe the data type that may be converted in another format for the target machine. This functionalities allow to exchange data between computers with very different architectures.

To create his own virtual machine, the user lists into a file, which is read in the initial phasis of PVM, the accessible machines on the network. Three modes of allocation are available to place the processes on the processors :

1. the transparent mode : each task is automatically located at the most appropriate site.
2. the architecture-dependent mode : the user may indicate the specific architecture on which particular tasks must be executed.
3. the machine-specific mode : a particular machine may be specified

4.2 Extension of PVM

The extensions brought to on PVM have their origin in the use of FORTRAN 90 and the creation of new types in CADNA. We begin by making a demonstration model of feasibility modifying only the main functions of PVM. The first difficulty we have found is due to the interface between FORTRAN 90 and PVM. For example, the subroutines PVMFPACK(type, data, ...) and PVMFUNPACK(type, data, ...) respectively allow to gather the data before sending them or to recover them after receiving them. They use 5 parameters including a pointer on the data and a constant that indicates their type. This technique works very well in FORTRAN 77 and in C because there is no type verification for the parameter of the subroutine and the functions. When these subroutines are used in a same program written in FORTRAN 90 to send or to recover data of different types, the compiler generates errors because it detects variables of different types for the second parameter. So it is necessary to overload PVMFPACK and PVMFUNPACK which leads to write as many subroutines (hidden type in the user's interface) as there are potential types. The problem is the same for the subroutine PVMFSEND (creation and sending of a message with one instruction) and PVMFRECV (receipt and recovering of data with one instruction). The subroutine PVMFREDUCE (concentration and diffusion) has not still been adapted for the new type of CADNA because it needs a development too important in the scope of a demonstration model of feasibility.

The second difficulty consists in integrating the new stochastic types SINGLE_ST , DOUBLE_ST, COMPLEX_ST, defined by CADNA in order to preserve the PVM principle of typed data. We must add the stochastic types to the intrisic types (BYTES, INTEGER2, INTEGER4, REAL4, REAL8, COMPLEX) predefined and used by PVMFPACK and PVMFUNPACK.

4.3 Difference of programmming

To explain in detail how the validation software for parallel computations works, we present an example of matrix multiplication intentionally simple and not optimized not to complicate the problem. The program realizes the multiplication on a computer using 4 processors in the following way : a process named *master* reads 2 matrices a and b. The matrix b is divided into four equal parts. The *master* processor sends to the four *slave* processors the matrix a and one quarter of the elements of b. Each one executes the multiplication between the matrix a and its part of b and sends back its result to the process *master* that builds the matrix solution. The program is written in FORTRAN.

Figure (1) presents in the left column the most interesting part of the source program of the process *slave* without using CADNA and in the right column the same source with the necessary modifications to use CADNA in bold-face. The printing subroutine of the matrix is not detailed. It is necessary to know that CADNA proposes a function $str(var)$ that takes a stochastic variable in parameter and, associated with a *print* or a *write* instruction, only displays the exact significant digits. If the value is no significant, @0 is displayed.

One may note that it is not necessary to do a lot of modifications on the original program to validate the numerical computation.

4.4 Results

To measure the performance of our model, two sequential versions and two parallel versions (each one with and without CADNA) have been written and run. The run times have been measured on a Connection Machine 5 (CM5) manufactured by the society Thinking Machine Corporation on full squared matrices of size 100x100, 200x200, 300x300 and are reported on table (1). For the measure of the run time only 4 processors (or nodes) have been used on a partition of 32. The vector unit associated to each node have been inhibited because it is impossible to change his working mode and therefore to modify their arithmetic. In the sequential versions, the run times represent only the run time of the matrix product subroutine. In the parallel versions, the run times are measured on the *master* processor. It includes the sending of the matrix a and b to the *slave* processors, the run time of the *slave* processors, the receipt of the partial matrix and the rebuilding of solution matrix.

Matrix	Sequential time without CADNA	Sequential time with CADNA	Parallel time without CADNA	parallel time with CADNA
100x100	4.46	16.10	2.26	6.50
200x200	38.20	134.00	14.63	45.23
300x300	153.20	483.00	51.00	147.00

Table 1: Computing time obtained on CM5 using one node for the sequential version and four nodes for the parallel versions.

To estimate the overhead generated by using CADNA, the sequential and parallel times obtained with CADNA are divided by the times obtained in the same conditions but without CADNA (see table (2)). CADNA induces a cost in time near a factor 3 which represents the price of the validation of the numerical results.

To conclude the study of the results, the efficiency of the parallelization is calculated. It is defined by the following ratio :

```
! mtid : address of the master process
integer mtid
double precicion a(100,100)
double precision b(100,25)




! receipt of a and b
call pvmfrecv (mtid,msgtype,info)
call pvmfunpack(REAL8,a,10000,1,info)
call pvmfunpack(REAL8,b,2500,1,info)

call pvmfunpack(REAL8,b,2500,1,info)

! r = a * b
call ProdMat(a,100,100,b,100,25,r,rlig,rcol)

! sending to the master of the result r
call pvmfinitsend(PVMDEFAULT, info)
call pvmfpack(REAL8,r,rlig*rcol,1,info)
call pvmfsend(mtid,msgtype)
end

Subroutine ProdMat(A,alig,acol,B,
        blig,bcol,MatP,rlig,rcol)
implicit none
integer i,j,k,alig,acol,blig,
         bcol,rlig,rcol
double precision A(alig,acol),
        B(blig,bcol),MatP(alig,bcol)
rlig = alig
rcol = bcol
do i=1,alig
    do j=1,bcol
         MatP(i,j)=0.D0
             do k=1,acol
             MatP(i,j)=MatP(i,j)+A(i,k)*B(k,j)
         end do
     end do
end do
return
end
```

```
!mtid : address of the master process
integer mtid
type (double_st) a(100,100)
type (double_st) b(100,25)

! inititialisation of the CADNA librairy
call cadna_init(0)

! receipt of a and b
call pvmfrecv (mtid,msgtype,info)
call pvmfunpack( DOUBLEST,a,10000,1,info)
call pvmfunpack( DOUBLEST,b,2500,1,info)

call pvmfunpack( DOUBLEST,b,2500,1,info)

! r = a * b
call ProdMat(a,100,100,b,100,25,r,rlig,rcol)

! sending to the master of the result r
call pvmfinitsend(PVMDEFAULT, info)
call pvmfpack( DOUBLEST,r,rlig*rcol,1,info)
call pvmfsend(mtid,msgtype)
end

Subroutine ProdMat(A,alig,acol,B,
         blig,bcol,MatP,rlig,rcol)
implicit none
integer i,j,k,alig,acol,blig,
         bcol,rlig,rcol
type (double_st) A(alig,acol),
         B(blig,bcol),MatP(alig,bcol)
rlig = alig
rcol = bcol
do i=1,alig
    do j=1,bcol
        MatP(i,j)=0.D0
            do k=1,acol
            MatP(i,j)=MatP(i,j)+A(i,k)*B(k,j)
        end do
    end do
end do
return
end
```

Figure 1: Source of the slave program. Left column : standard version using PVM, right column : version using PVM and CADNA.

Matrix	Ratio of times with and without CADNA	
	for the sequential program	for the parallel program
100x100	3.61	2.87
200x200	3.50	3.09
300x300	3.15	2.88

Table 2: Overhead due to the use of CADNA

$$eff = \frac{TpsSeq}{NbProc.TpsPara}$$

with :

eff : efficiency of the parallelism,
$NbProc$: number of processors,
$TpsPara$: run time of the parallel program,
$TpsSeq$: run time of the sequential program.

Matrice	efficiency without CADNA	efficiency with CADNA
100x100	49.33	61.92
200x200	65.18	74.06
300x300	75.10	82.14

Table 3: efficiency of the parallel program

The efficiency of 100 % is obtained only in the case where all the processors are independent and do not exchange any data. Table (3) presents the obtained result on the matrix multiplication. Two behaviours must be noticed.

First, the efficiency is less for the small size matrix. When the message is short, the time of initialization of a communication between two processes becomes more predominant. It would be better to send large size messages.

Secondly, with an equivalent matrix size, the efficiency is superior when CADNA is used. This increase is simple to explain : the ratio $\frac{computation}{exchange}$ is more important.

5 Parallelization of the CESTAC method

The previous system (CADNA + PVM) validates the result of a parallel program without modifying the architecture used. Here we present a prototype of direct parallelization of the CESTAC method which brings about an increase of the processors number.

5.1 Extraction of the parallelism of CESTAC

Let us consider an algebraic procedure PA composed of a finite sequence of arithmetical operations. Practically, the CESTAC method consists in executing N ($N = 2$ ou 3) copies of the program of the procedure PA with the random arithmetic in order to be able to estimate the accuracy of all intermediate or final computed results. The N copies constitute N independent tasks of computation. At the time of their execution, it is necessary to know the accuracy of the intermediate computed result in order to carry out certain operations (conditional splitting, division). Then, the tasks communicate their results to a control task that returns the mean of the N representative values with the number of exact significant digits.

The implementation of the CESTAC method on a sequential machine multiplies the run time of a program by a factor depending in the average on the number of image programs. These programs being independent, their implementation on a parallel machine should allow to decrease the run time. Figure (2) presents the scheduling of the tasks in the CESTAC method. A prototype of feasibility named RAPP (Random Arithmetic Parallel Prototype) has been developed on a distributed and reconfigurable machine based on transputers with the OCCAM2 language of INMOS [OCCAM 88].

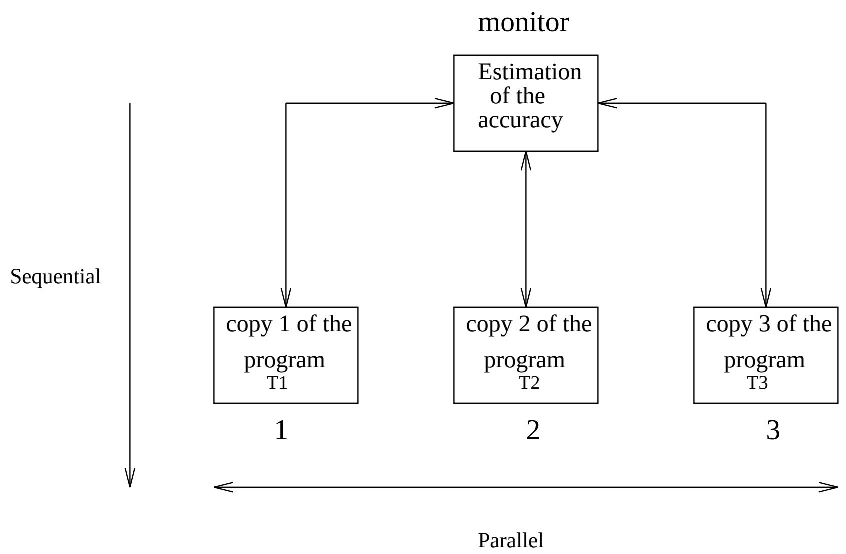

Figure 2: Elementary module of computation for the random arithmetic

5.2 Presentation of RAPP

In the scope of RAPP, we have chosen to use three tasks of computation T_i ($i = 1, 2, 3$) (image program of PA). The T_i are allocated to the processors 1,2 and 3 (see figure (2)) and are run in parallel. The task of accuracy estimation is allocated to a monitor processor which is also used for the communication between the host processor and the rest of the network.

The divisions and the conditionnal splitting synchronize automatically the tasks T_i at the moment of the accuracy estimation. When a division occurs, the values of the denominator are sent to the monitor processor and the trace of a possible unstability is generated into a file and may be consulted after the end of the computation. In the same way, for a conditionnal splitting, a precision control is carried out before the comparison of two variables. The precision estimation is possible for all the other results, but it is not automatically done to preserve the performance of the prototype.

5.3 Communication between the nodes of the RAPP network

The language OCCAM2 gives the possibility to define new protocols of communication between nodes of a network in order to adapt the communication to the type of exchanged data.

The RAPP prototype uses 3 communication protocols to gain in performance and to preserve for the programmer the standard Input/Output, the system call that work according to the standard protocol SP.

The first prototype allows the communications of the tasks $T_i (i = 1, 2, 3)$ towards the task of precision estimation and limits the communication to a table of real numbers coded on 64 bits. The number of elements of this table is variable.

The second protocol allows the task of precision estimation to send back to the task T_i the response to the precision request. A table of real numbers coded on 64 bits and a table of integer numbers form the response. The two tables are of variable size.

Finally, the protocol of standard communication SP is used by the tasks 1,2 and 3 so that every communication (input/output, system call, ...) with the host processor should still be possible for the programmer. It is then necessary to introduce two new processors $mux1$ and $mux2$ (see figure (3)) which are only used for the multiplexing of the communication channel. In the case of a parallel program, the addition of these processors concern only the processor directly linked to the host processors.

On figure (3) the processors $P1$, $P2$, $P3$ and $monitor$ realize the computation of RAPP. The processors $mux1$, $mux2$ and aux are used to solve the problem due to the little number of communication channels of the transputer.

5.4 Application to a sequential program

Let us consider a computer program P giving a single result r and that is executed with the computer's classical arithmetic. For the program to be run with RAPP, it is sufficient :

- to perturb every arithmetic operation and every assignment. (for example to replace $x := y + z$ by $x := p(y + z)$, p being a perturbation function of the tool box of RAPP);
- to use a specific output that takes into account the significant digits of the result r;

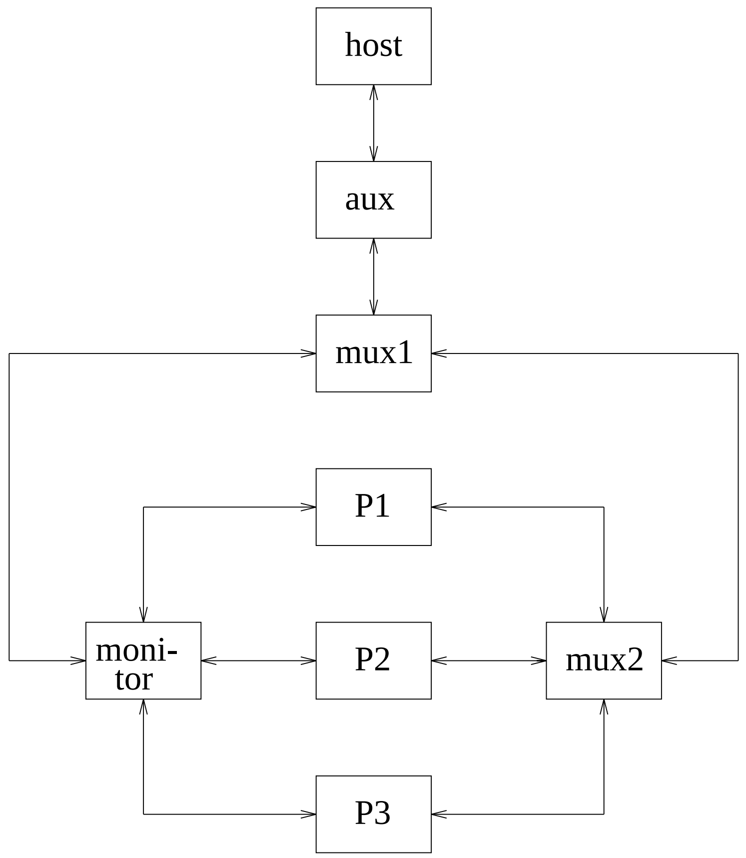

Figure 3: RAPP network for a sequential program

– to add at the beginning of the program P the statement of declaration of the communication channel with the task of precision estimation.

The RAPP user elaborates only a single program but in the reality three identical programs are executed on the RAPP nodes.

By analogy with the synchronous programmation on sequential architectures of the CESTAC method (imbedding in $\mathbb{R}^N$), the RAPP prototype defines a structure. The elements of this structure are distribued on different processors.

To test the performance of RAPP on a sequential program, we considered the example of two square matrices multiplication presented in the previous section. Table (4) shows the results obtained on a machine based on transputer T800.

RAPP use 3 processors for the execution of the image programs and a processor for the precision estimation. So it should be normal to obtain a time ratio with and without near of 1. Therefore according to table (4), this ratio is contained between 4.5 and 5. This increase is principally due to the perturbation of the elementary operation in the image program. Table (5) shows the cost of the perturbation in relation to the arithmetic function.

Matrix	Sequential time without RAPP (in sec.)	Sequential time with RAPP (in sec)	time ratio with and without RAPP
40x40	3.81	19.00	5.09
80x80	30.21	142.00	4.70
100x100	58.81	270.46	4.60
160x160	240.14	1074.75	4.48
200x200	468.37	2091.00	4.46

Table 4: Run time obtained on transputers T800.

operation arithmetic	Cost of the operation without perturbation (μsec)	Cost of the operation with perturbation (μsec)	time ratio
addition	20.30	111.15	5.47
substraction	21.30	105.96	4.97
multiplication	30.59	120.07	3.92
division	35.23	124.7	3.53

Table 5: Perturbation cost in relation to the arithmetic operations

Let us recall that this work deals with a study of feasability. With an optimized perturbation (as in CADNA), we may hope that the time ratio whould be in the order of 1.5.

5.5 Application to a parallel program

Let us consider a computer program P composed of k sequential tasks T_i running in parallel on k processors $Proc_i$. The application of RAPP to the program P consists in replacing each k processors by a subnetwork of figure (2). Thus, each task T_i is replaced with three images plus a task of precision estimation. The interconnection of all the tasks is done in the following way : In the program P, if a task T_i communicates with a task T_j then, in the program working with RAPP, the image tasks $T_{i,1}$, $T_{i,2}$,$T_{i,3}$ communicate respectively with the tasks $T_{j,1}$, $T_{j,2}$,$T_{j,3}$ (see figure (4)).

Here again, we shall use the previous example of matrix multiplication with the same structure of parallel program (see table (6)).

The increase of time that RAPP generates in the case of a parallel program is less important than in the case of a sequential program because RAPP increases only the time corresponding to the computation and not the time corresponding to the communication of the classical parallel program (i.e. without RAPP).

Matrix	Parallel time without RAPP (in s.)	Parallel time with RAPP(in s.)	Parallel time ratio without RAPP and with RAPP
40x40	1.43	5.50	3.85
80x80	11.07	37.98	3.43
100x100	21.45	71.77	3.35
160x160	86.89	278.46	3.20
200x200	169.08	540.62	3.20

Table 6: Run time obtained on transputers T800.

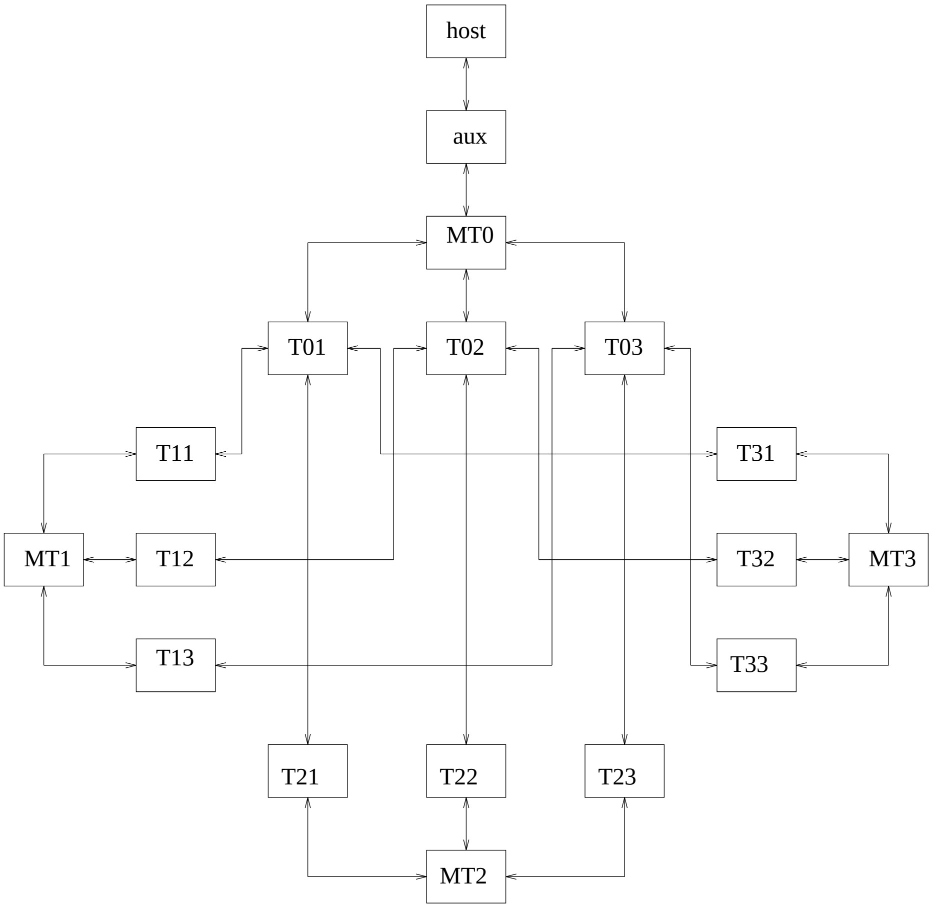

Figure 4: Application of RAPP to the parallel program P. $MT_i (i = 1..4)$: processors monitor

6 Conclusion

This work has shown that it is now possible to study the round-off errors propagation on results provided by sequential or parallel codes written in FORTRAN 77. It is easily done using the CADNA software on sequential architectures and on parallel architectures if the message passing library PVM is used. It was very important to show the feasibility of estimating the accuracy on parallel architectures which play an important role in the scientific computers today. We have seen that the solution CADNA + PVM is very simple and does not modify the number of processors required for running the initial parallel program. But the run time is multiplied in the same way than with the use of CADNA on sequential architecture.

The CESTAC method seems to be well-suited to a direct parallelization. The problem was to find out if the cost of message passing between processors (an absolute necessity) was not too important compared to the running time. The study of the RAPP prototype has shown that it is very satisfactory. Such an approach has a strong future, but there is not a useful FORTRAN tool yet.

We did not mention the numerical validation on vectorial architectures which are very often combined with the parallel architectures. The solution with CADNA or RAPP cut off the vectorisation and the problem is still open.

Aknowledgements : This work has been supported by the team *Arithmétique et précision* of the pool PRC-PRS of the CNRS and by the *Centre National de Calcul Parallèle en Sciences de la Terre.* (CNCPST)

References

[Chesneaux 90] J.-M. Chesneaux, J.-M. : "Study of the computing accuracy by using probabilistic approach"; Contribution to Computer Arithmetic and Self-Validating Numerical Methods, ed. C. Ulrich, (J.C. Baltzer) (1990), 19-30.

[Chesneaux 92] Chesneaux, J.-M. : "Descriptif d'utilisation du logiciel CADNA_ F"; MASI Report, n° 92-32 (1992).

[Chesneaux 95] Chesneaux, J.-M. : "L'arithmétique stochastique et le logiciel CADNA"; Habilitation à diriger des recherches, to appear.

[Hamming 70] R.W. Hamming, R. W. : "On the distribution of numbers"; The Bell System Technical Journal (1970), 1609-1625.

[Hull and Swenson 66] Hull, T.E., Swenson, J. R. : "Test of probabilistic models for propagation of round-off errors"; Communication of A.C.M., vol.9, n° 2 (1966), 108-113.

[Vignes 90] Vignes, J. : "Estimation de la précision des résultats de logiciels nu mé ri ques"; La Vie des Sciences, Comptes Rendus, série générale, 7 (1990), 93-145.

[Vignes 93] Vignes, J. : "A stochastic arithmetic for reliable scientific computation"; Math. Comp. Simul., 35 (1993), 233-261.

[Vignes and La Porte 74] Vignes, J., La Porte, M. : "Error analysis in computing"; Information Processing 74, North-Holland (1974).

[Geist and al. 93] Geist, G. A. and al. : "PVM 3 User's Guide and Reference Manual"; ORNL (Oak Ridge National Laboratory), TM-12187 May (1993)

[OCCAM 88] "OCCAM 2 Reference Manual"; Prentice Hall, International Series in Computer Sciences (1988).