matematica computacional - SBC
Matematica: Alto desempenho
Biblioteca: Rotinas intervalares
Analise: Intervalos
CNPq 1.01.04.00-3

Technical Report

Revista de
Informática
Teórica e Aplicada

High Performance with High Accuracy Laboratory

Tiarajú A. Diverio; Philippe O. A. Navaux; Dalcidio M. Claudio; Carlos A. Hölbig; Úrsula A. L. Fernandes; Rafael L. Sagula.
Instituto de Informática da UFRGS
P.O.BOX: 15.064 – 91.501-970 – Porto Alegre – Brazil
Phone: (051) 316.68.46 – Fax: (051) 319.15.76
E-mails: {diverio, navaux}@inf.ufrgs.br

Abstract

In order to obtain high performance with high accuracy in the solution of scientific computational problems, a computational tool has been developed, called High Performance with High Accuracy Laboratory. In this paper we describe initially the high performance and then the high accuracy and the interval mathematics. After that, the tool is described, including two environments in which it has been developed, that is, the Cray Supercomputer vector environment and the parallel environment based on Transputers. The description summarizes the modules, the basic interval library, the high accuracy arithmetic kernel, the interval applied modules, especially the selint.p library. Finally, there are some comments about the performance.

Key Words

High Accuracy, High Performance, Interval Mathematics.

1. Introduction

Computations of large scale are numerical processes which require a large quantity of floating-point operations. They appear in the military and scientific areas, engineering, study and exploration of energy source, medicine, artificial intelligence, basic research, and economics. More specific examples are: the atmospheric forecast, modeling of oil field, aircraft projects, simulations of automobiles accidents to the determination for safer automobile and traffic control.

For the solution of these problems, powerful computers such as Mainframes and Supercomputers are necessary. They are modern, fast and with process capacity to a large quantity of operations and/or a large volume of data. To the Mainframes, it is attributed the feature of having high capacity for working with a large volume of data, and for storing it, as well as for transferring the data. An informal definition of supercomputer is not static. It is seen as the most potent machine in a determined moment. A supercomputer of yesterday may be today less potent than a simple PC 586 and a current supercomputer may be an obsolete machine tomorrow. The definition is associated to the performance. This one is measured according to the quantity of operations that it processes per second. These operations can be floating-point operations or instructions, resulting in units of measurement well-known as flops or mips, respectively.

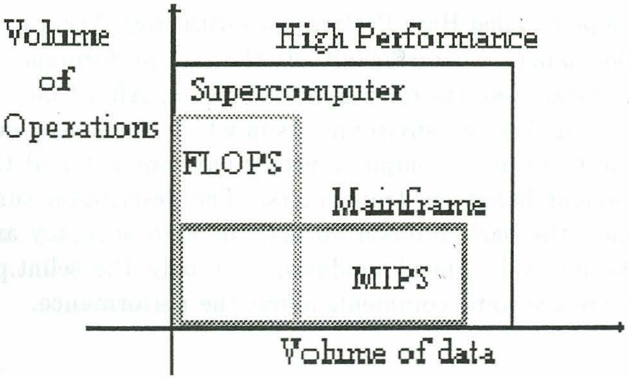


Figure 1: Machines of High Performance

Nowadays computers can carry out above 10^{11} floating-point operations in a second. It has just become possible by virtue of the development of integrated circuit technology. The advances in the computation technology suggest an attempt to render the computer arithmetically more powerful. Thus, a special attention must be given to the validity of the calculated results. A rapidly calculated but incorrect result is useless. The solution of numerical problems continues using the same arithmetic of the first computers. That is, it is based on the four basic floating-point operations: addition, subtraction, multiplication and division. They are still used, but presently many times faster and more frequently. Seemingly, we used the floating-point arithmetic with its well-known inefficiency, supposing it is a necessary evil.

The computer was invented to do the complicated work for people. The evident discrepancy between the computational power and the control of computational errors also suggests that the process of error estimation must be placed inside the computer. This new task has been made satisfactorily for practically all the basic problems of numerical analysis and many applications. In order to do it, the computer needs to be made arithmetically more powerful than the ordinary one.

Examples show that it is necessary and even obligatory to render the numerical methods and the computers more reliable than the computers and methods that use only the ordinary floating-point arithmetic. This development began about 25 years ago and has gained more importance up to now.

Some techniques have been developed in numerical analysis, which have enabled the computer itself to verify the validity of the results calculated for numerous problems and applications. The computer can also establish in this manner the existence and the uniqueness of the solution. It is from this point of view that the High Performance with High Accuracy Laboratory was projected and is being developed.

2. High Performance

High performance is associated with machines, computer architectures, and with the processing. In this work, it refers to the high performance in the processing of large scale processes.

Therefore, the emphasis is the use of high performance computers in order to do a large quantity of floating-point operations. In these computers, the type of processing is vectorial or parallel.

In vector processing, the computer has a hardware that makes possible the execution of scalar operations for vector operations, and this produces a decrease of the processing time. In parallel processing, the computer must have two or more units of processing or processors (CPUs) where the tasks can be executed at the same time.

In order to give an idea of the dimension of the number of operations and of the time spent for doing it, it will be considered a more specific example of atmospheric forecast, that is, in the multidimensional modeling of the atmosphere. The predictive modeling is made through extensive computational simulations of experiments, in which, generally, computations of large scale are involved in order to guarantee the desired accuracy and the minimization of the necessary time for the solution. Such numerical modeling needs a computer able to process with a speed of thousands of megaflops. Using a grid with 435 Km of a side, a forecast of 24 hours needs to fulfill about one hundred billion data operations. This forecast can be done in a computer of 100 megaflops in around 100 minutes, that is, one hour and forty minutes. The grid of 435 Km gives a forecast between São Paulo and Rio de Janeiro, approximately. Increasing a little this grid, in order to comprise a larger region, it would be necessary to have a faster machine or we would run a risk of spending more time of calculus for the forecast than the day that we want to foresee.

2.1 High Accuracy Arithmetic

The high accuracy arithmetic enables the calculations to be done with maximum accuracy. But it is necessary that the format or type of data, the arithmetic operations supported by the hardware or by the programming language satisfy the conditions of a semimorphism [KUL83]. All the operations constructed according to this definition will have maximum accuracy. That is, the calculated result differs from the accurate value to the utmost in one rounding.

The IEEE arithmetic standard has emerged with the purpose of setting the standard of the floating-point arithmetic in all the platforms. It has been an initial step for obtaining the high accuracy arithmetic.

The standard has not specified the roundings for complex variables, operations between vectors and matrices, and has not included the optimal scalar product, essential for the guarantee of the calculations. For this reason, GAMM/ IMACS have proposed another standard (see [IMA91]).

The use of high accuracy arithmetic associated with interval mathematics produce results with maximum accuracy, where the calculated result differs, by just one rounding, from the real value, as the intermediate calculations are done in special registers, in order to simulate the operations in the reals, and the rounding is done only at the end.

The requirements for having high accuracy arithmetic are: the use of directed roundings (downwards ∇x and upwards Δx); the four arithmetic operations with maximum accuracy; the interval mathematics and the optimal scalar product.

Directed roundings are monotone mappings (see [KUL83]) used to represent the real numbers in machine numbers. When the rounding function is applied to any element of the machine number set, it produces the machine number itself. In order to have implemented a high accuracy arithmetic, we must have the two types of directed rounding:

- Upward rounding (Δx) is the function that maps the real number x to the smallest machine number that is greater than or equal to it.
- Downward rounding (∇x) is the function that maps the real number x to the largest machine number that is less than or equal to it.

These roundings are necessary in order to guarantee calculation, being fundamental to the implementation of the machine interval arithmetic. Another type of rounding is the symmetrical rounding (or to the nearest machine number). It rounds the real number x to the nearest machine number. It is defined according to the previous roundings and produces a smaller error of approximation.

Arithmetic operations with maximum accuracy are defined in order that just one rounding is applied in the basic arithmetic operations, resulting that the calculated value and the accurate value are different only for one rounding. If $\circ$ is an arithmetic operation in the space R of the real numbers, then the corresponding operation in the computer $\boxed{\circ}$ into the set F of the machine numbers is defined by: $x \boxed{\circ} y := \square(x \circ y) \, \forall \, x, y \in F$. (where $\square$ is a rounding).

That is, the operation in the computer must be made as if the accurate result were first calculated and then approximated by means of the chosen rounding.

Briefly, semimorph operations for real and complex numbers, vectors and matrices, as well as for real and complex intervals, interval vector and matrices, can be performed by fundamental arithmetic floating-point operations arithmetic: +, -, *, /, . (where . denotes an exact dot product), each one with the roundings upwards, downwards and to the nearest number.

2.2 Interval Arithmetic

Interval arithmetic treats data in the form of numerical intervals and has emerged with the purpose of automating the analysis of the computational error, bringing a new emphasis that allows a control of errors with sure limits, besides proofs of the existence or not of the solution of several equations.

Instead of approximating the real numbers by the floating-point numbers (through some of the approximation rules or the types of rounding), they are represented by intervals of floating-point numbers, that is, a real x is represented by an interval $X = [x_1, x_2]$, where the lower bound (x_1) and the upper bound (x_2) are machine numbers, so that $x_1 \leqslant x \leqslant x_2$. All the arithmetic operations, relational operators, as well as elementary functions are defined for interval arguments. Starting from the interval arithmetic, the concepts that compose interval mathematics and interval methods for the solution of numerical problems are developed. The table below shows some of the existent operations for floating-point interval sets.

Inverse additive:	$-X = [-x_2, -x_1]$
Pseudo inv. mult.:	$1/X = [1/x_2, 1/x_1]$ if $0 \notin X$
Addition:	$X + Y = [x_1 \nabla + y1, x2 \Delta + y2]$
Subtraction:	$X - Y = [x_1 \nabla - y_2, x_2 \Delta - y_1]$
Multiplication:	$X * Y = [min\{x_1 \nabla * y_1, x_1 \nabla * y_2, x_2 \nabla * y_1, x_2 \nabla * y_2\},$ $max\{x_1 \Delta * y_1, x_1 \Delta * y_2, x_2 \Delta * y_1, x_2 \Delta * y_2\}]$
Division:	$X/Y = [min\{x_1 \nabla / y_1, x_1 \nabla / y_2, x_2 \nabla / y_1, x_2 \nabla / y_2\},$ $max\{x_1 \Delta / y_1, x_1 \Delta / y_2, x_2 \Delta / y_1, x_2 \Delta / y_2\}]$
Intersection:	$X \cap Y = \{[max\{x_1, y_1\}, min\{x_2, y_2\}]$, if $x_1 \leq y_2$ and $y_1 \leq x_2$, $\emptyset$- other cases
Union:	$X \cup Y = \{[min\{x_1, y_1\}, max\{x_2, y_2\}]$, if $X \cap Y \neq \emptyset$, ERROR - other cases.
Distance:	$d(X, Y) = max\{\mid x_1 - y_1 \mid, \mid x_2 - y_2 \mid\}$
Absolute Value:	$\mid X \mid = d(X, [0, 0]) = max\{\mid x_1 \mid, \mid x_2 \mid\}$
Diameter:	$D(X) = x_2 - x_1$

Table 1: Basic Interval Operations

3. High Performance with High Accuracy Laboratory

The project of the High Performance with High Accuracy Laboratory has been projected in three environments, that is, on three platforms. The first platform is the environment of the PC 486 with Pascal XSC language. The second platform is the environment of the Cray Y-MP supercomputer with the Fortran 90 language, where vector processing is explored.

The Cray Supercomputer is located at the Supercomputer Center (CESUP-UFRGS). It belongs to the Y-MP family, with two vector processors, and with the following features: maximum total speed of 660 Mflops; word lenght of 64 bits; RAM of 256 Mbytes and 16 Gbytes of disk, and UNICOS operational system, compatible with UNIX system V.

The other platform comprises the parallel environment based on Transputer processors, as the B004 board available at CPGCC/UFRGS. The emphasis is on the exploration of the parallelism of the interval operations and the development of a project of a parallel tool.

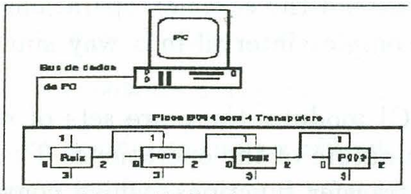
Environment	PC 486	Cray Y-MP	Transputers
Language	Pascal XSC	Fortran 90	C++ parallel
Basic Interval Library	Pascal XSC	`libavi.a`	libavip.h
Kernel High Accuracy	Pascal XSC	fpkernel.a	in progress
Linear Systems	selint.p	libselint.a	in progress
Integration	intnum.p	libintnum.a	in progress

Table 2: Laboratory environments

The INMOS Transputer is the first single-chip microprocessor that provides a high-speed processor, fast inter-processor communications, and an explicit support for multiple processes and multiple processor systems. The design aims were for a device that would be used in multiple processor message-passing systems, where each processor had its own physical memory, but with support for multiple shared-memory processes on each Transputer. What the Transputer does not provide, therefore, is any memory management on chip, or any support for off-chip memory management devices. The Transputer is a processor that does not control I/O operations directly. Such operations are required by the Transputer to the system's BIOS (Basic Input/ Output System) in which it resides. Presently, it has interfaces with IBM-PC and compatible computers, and with Sun workstations.

The T800 is a 32 bits RISC processor containing all the elements of a microcomputer on a single VLSI chip. It includes a 32 bit CPU, a 4 Kbytes on-chip static RAM, a 64 bit floating-point unit (FPU), a memory interface for external memory and four bi-directional serial links. These four links are the characteristic elements of Transputers. A further feature of the T800 is the support of multitasking which is essential for parallel programming. Each Transputer can easily manage several processes, and process switching is performed very fast by a hardware scheduler.

The development environment available at UFRGS consists of a B004 board with four T800 Transputers (one root Transputer and the other ones of work) connected to a PC 486 IBM-PC compatible.



PC - host computer where the Transputer's board is found;
Raiz (Root) - Transputer connected in the PC's data bus. It makes the communication with other network elements;
P001, P002, P003 - other Transputers in the board;
0, 1, 2, 3 - numeration of the communications channels (links) of each Transputer.

Figure 2: Environment of Transputer Board

3.1 Basic Interval Library

The standard used in the Basic Interval library was the same as that used in the Pascal XSC language, where intervals were available as intrinsic types of the language as well as all its arithmetic operations. In the vector environment, it was necessary to define this data type. Intervals were carried out in the `libavi.a` interval library.

`Libavi.a` is a library of interval routines that implements the high performance arithmetic joining the features of vector processing with the properties of interval mathematics. The `libavi.a` interval library was designed in order to make feasible the use of interval mathematics in supercomputers for the solution of physical and chemical problems and for the solution of engineering problems that need high accuracy. Thus, besides the interval vector arithmetic (operations, functions and expression evaluation), we had to provide libraries that would render the interval methods for the solution of problems available for these users.

`Libavi.a` library is composed of 290 interval routines organized in four modules. The BASICO module includes the file which contains the definition of complex and real intervals (`inter.inc`). In this module, all the operations among real interval are implemented. These operations are the basis of all the other modules.

The CI module contains routines for manipulation of data of complex interval type. Basically, it contains the same routines as the BASICO module rewritten for this kind of data and also specific routines for complex intervals (such as the conjugated number). From this module, routines that manipulate vectors and matrices of complex intervals are being implemented.

Thus, it is enough to extend the complex operations for each element of the vector or matrix of complex interval in a way similar to the one done with real intervals.

In the BASICO and CI modules there are sets of routines or functions grouped according to the nature of the operations. There are six sets. The first set is composed of transfer functions, which convert real or complex numbers into intervals or vice versa, as well as functions that calculate geometrical particularities of the intervals, such as diameter, radius, medium point and distance among intervals.

The second set is composed of relational functions, in which the result type is generally logical (Boolean). Among the relational functions are the equality, the difference, to be smaller, to be larger, to be contained in, to contain, inner inclusion relation and the pertinence relationship, that is, a real belonging to the interval.

The third set of routines deals with operations among sets. Among these operations are the intersection, the union and the convex union.

The fourth set of routines implements the arithmetic, that is, the arithmetic operations of addition, subtraction, multiplication, and division. Some arithmetic operations among different types of data are included. The elementary functions such as absolute value, square root, square, powers, exponential, logarithm and trigonometric functions constitute the fifth set of routines. Finally, there are the routines of input and output, `sread` and `swrite` for the new data types.

The MVI module includes the BASICO module and it is included into the module of applications APLIC. This module implements all the operations among interval vectors, interval matrices and routines of different types of data with interval vectors and matrices. The module of the interval matrices and vectors (MVI) has been organized in three parts: real interval vectors and matrices and the part that contains arithmetic operations of different types of data with intervals, real intervals vectors and matrices. Each one of them contains different groups of routines. These groups are: transfer functions, relational operations and set operations, arithmetic operations, transposition, basic functions and input and output routines.

The last module is the application module, called APLIC. In this part, there are some routines that implement composite arithmetic operations existent in other libraries.

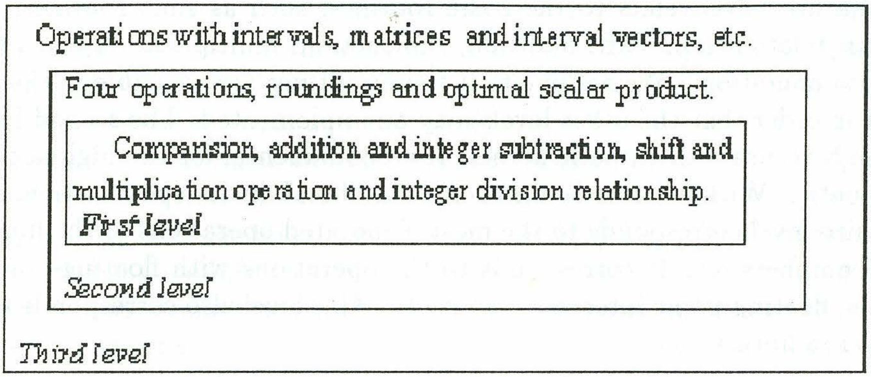

Operations with intervals, matrices and interval vectors, etc.
Four operations, roundings and optimal scalar product.
Comparision, addition and integer subtraction, shift and multiplication operation and integer division relationship.
First level
Second level
Third level

Figure 3: Levels of the Kernel

In order to choose these routines, some libraries have been analyzed, such as NUMERALS, from Burroughs, BLAS, and Pascal XSC. The APLIC module uses the BASICO and MVI modules.

In a parallel environment, it is necessary to implement the interval, interval vector and matrix types together with the arithmetic operations between these data types. In the parallel environment, the interval library is being implemented in parallel C++ and it has been called `libavip.a`.

3.2 High Accuracy Arithmetic Kernel

In a scalar environment, the high accuracy arithmetic is one of the characteristics of the Pascal XSC language. In the other environments, it was necessary to implement routines with this quality, creating the High Accuracy Arithmetic Kernel.

In order to obtain high accuracy arithmetic, a large gap has to be filled. Thus, a high accuracy arithmetic kernel is being implemented and going to be incorporated into the `libavi.a` (developed in FORTRAN 90), including the directed roundings, the four operations with maximum accuracy and the optimal scalar product. The kernel is the basic module of the `libavi.a` library. It is being entirely implemented by software. There have been foreseen three levels of implementation for high accuracy arithmetic. The picture below illustrates the disposition of the levels of implementation.

The first level refers to the basic routines, such as shift routines; comparison relationship; shift, addition, subtraction, multiplication and integer division operations. We assume that these routines are available in the computer in order that the other levels may be implemented. The second level is the high accuracy arithmetic kernel. It is fundamental for the high accuracy arithmetic. Without this level, there is a "hole" in its specification. And the third level corresponds to the most elaborated operations of the floating-point numbers set. It corresponds to the operations with floating-point intervals, floating-point interval vectors, etc. This level also corresponds to the `libavi.a` library.

3.3 Interval Applications Modules

One of the purposes of High Performance with High Accuracy Laboratory is the elaboration of computational tools that use interval arithmetic in the solution of problems of scientific computation and of computation in several areas of research. For example, problems that involve the solution of system of linear equations (see [HOL96]). The module of applications is composed of operations that correspond to the three levels of the BLAS library and of routines that make easier the development of interval methods for solution of linear systems.

The modules of applications were projected for the three environments as shown in table 1. They were, initially, implemented in PASCAL XSC like in the case of linear systems solutions. In vector environment, the modules are being implemented in FORTRAN 90, and in parallel environment, they are in project stage.

3.4 Interval Applied Library for Linear Systems

Throughout the study of interval methods for the solution of systems of linear equations ([HOL96]), it has been observed the need of defining applied libraries which may concern chapters of numerical mathematics. The first one of these interval applied libraries is the one that concerns the solution of systems of linear equations.

The principal objectives in developing this interval applied library are the following: the use of these libraries for the diffusion of the study of interval methods near academic environments and near people who may be interested in these methods and the use of these libraries in the solution of computational problems in different areas of research.

The Linear System library has been developed in modules. Each one corresponds to a methodology of developing interval methods. These modules are: `dirint` module, including the methods based on interval algebraic operations and interval properties (they are also known as interval direct methods); `refint` module, including the methods based on inclusions or interval refinements of the solution and of the error (they are also called hybrid methods, since we can use the initial solution calculated by punctual methods or by punctual inverse matrix); `itrint` module, including the interval iterative methods, also known as relaxation methods (they are also based on monotone inclusions). Besides these three modules, it has been defined a fourth one, `equalg`, which contains routines for the particular case of linear system of order one, that is, algebraic solution of equations by interval methods, like the interval versions of Newton's method (see [HOL96]).

`Selint.p` library has been developed in a personal computer environment (PC type) and implemented in a PC-486, making use of Pascal-XSC compiler ([KLA92]). Finally, through this library, there were made comparisons among the results obtained (punctual and interval results) in order to do an analysis of quantitative performance. For this comparison we used the `libselint.a` interval library, which has been developed for Cray Y-MP supercomputer of CESUP/UFRGS environment, implemented in FORTRAN 90 and making use of `libavi.a` interval library ([DIV95]). This library belongs to the Interval Vector Arithmetic project of GMC/UFRGS, in which there are also the `libavi.a` interval library and the `libselint.a` interval library.

For the use of the `libselint.a` library, the user must also include in his applied program the `inter.inc` file, which contains the definitions of real and complex intervals and the `libavi.a` library, since the necessary routines of manipulation of intervals, vectors and intervals matrices have been implemented there. It may still be necessary to include the `libsci.a` library, for some of the APLIC module use routines from BLAS library, included in it.

The results obtained through implemented interval methods have proved to be of good quality in their major part. Through the comparison of those results with the punctual results of the tested systems, it has been verified the same magnitude of accuracy, and that the punctual solution has been contained in the solution interval. The order of the solution interval diameter has been of 10^{-15}. This has guaranteed an accuracy around 14 correct significant digits.

In the implemented and tested direct methods, the only method that has not presented satisfactory quality was a version of Hansen's method, described by hansen4, which produced interval results with a very large diameter (in the order of 10^2) that render the solution useless to the purposes of the libraries.

In general, the methods based on refinement have been considered as the methods that produce the best results. In some systems, they produced a result which is considered arithmetically accurate. Those methods have also used the least processing time for solving linear systems. Some of the obtained data are presented and compared in table 3 and figure 4.

order/method	hansen2	hansen3	hansen5	refgeral	refdensa
10	0,330	0,275	0,439	0,165	0,165
20	1,813	1,758	2,692	0,769	3,296
30	5,328	5,273	8,679	2,032	8,459
40	11,975	11,700	20,434	4,010	18,402
50	23,016	23,181	37,738	6,921	35,705
60	40,100	41,308	66,467	10,931	58,666
70	60,864	62,182	106,237	16,369	87,340
80	89,318	90,142	160,399	23,565	135,844

Table 3: Processing speed in `selint.p` program in seconds

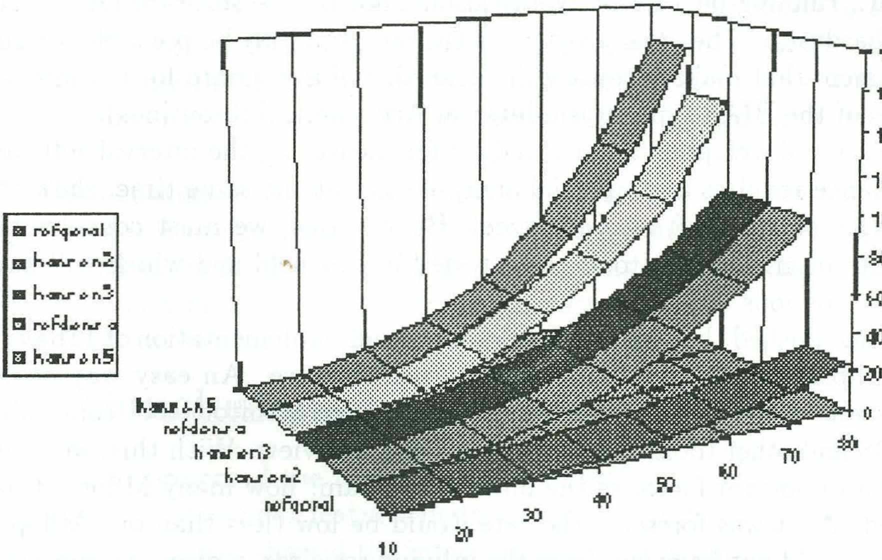


Figure 4: Comparison of the processing speed in `selint.p` program.

Tests done in a PC 486 DX2 100 MHz, using the Hilbert matrix.

4. Interval Software Performance

The measure of performance of these libraries requires a previous study of the best technique to be applied in order that common errors may be avoided such as those described by Jain [JAI91], rendering all the work useless. These errors may result from a biased goal, from a poor research, from an analysis without a deep knowledge of the problem, from a wrong measure of the data, as well as by the erroneous presentation of the obtained results.

A difficulty in the comparison of performances is that in each one the existing libraries use different environments, not only hardware but also software. The `libavi.a` library, for example, has been developed in the Cray Y-MP Supercomputer using the Cray's Fortran 90 compiler, trying to use to the utmost what the machine offers. Consequently, programs that use it will probably run faster than those that use libraries such as PROFIL, INTLIB, or PXSC, running on PCs or Workstations, not by the software factor, but by the hardware. The ideal program is the one that may be portable for any environment that make necessary the definition of a standard for the interval libraries of the BIAS type (Basic Interval Arithmetic Subroutines).

Thus, the development of a standard for measuring the interval software performance requires an impartial analysis and, at the same time, show the advantages of each software analyzed. Besides this, we must consider the possibility of arising new tools to be tested in this field and which will have fit in the previous method.

Having finished the phase of development and implementation of `libavi.a` interval library, we begin to measure its performance. An easy way of doing this was with the `htm` (Hardware Performance Monitor) tool containing UNICOS and other tools such as perfview and profview. With this, we could define an important factor of the interval program: how many Mflops it has achieved. As it was foreseen, the rate would be low (less than one Mflops), because we did not have yet, from the inlining resource, a directive compiler, in the version available of the Fortran 90 in the Cray Y-MP2E of CESUP. Without this resource it has been almost impossible to use all the advantages of a vector processor and program restrict to a scalar code.

For a person that does not know the Cray Y-MP2E supercomputer, this value can be high, but when compared with the potential of this machine (peak of 330 Mflops per CPU), we verify that this value is insignificant. Indeed, just in order to test the performance, it was made a manual inline of the function of interval matrices sum and it was measured the performance again: from 20 to 127 Mflops.

Another doubt concerning the data presented above is that the program was used to make the measurement. Corliss' work was very useful is this sense: he defined a workload to be used in performance tests of interval packages. The work of Corliss together with each developer was impartial and also showed all the potential of this product.

As seen before, an interval benchmark must have some special features, such as: to be relative to system potential about what is running and show impartially all potential of the software tested. On the other hand, we will have absolute measures of performance that do not serve much to show the reality. The use of BIAS (Basic Interval Arithmetic Subroutines) is also essential in order that the test programs can be written and translated without ambiguities, besides encouraging the use of this standard in other applications.

The benchmark development must not be an isolated effort of the user interested in knowing which is the best software available in the market, but it must have the participation of the person that develops the system, that wants to show all the potential of this product.

5. Conclusions

This paper presented the high performance with high accuracy laboratory, and it described its main features in its three environments: sequential, vector and parallel. It also showed the importance of having quality in the floating-point operations, for in computations of large scale the errors accumulate and they can produce wrong results.

The `libavi.a` library has been developed with the purpose of exploring the Cray high performance with the use of Interval Mathematics.

The results obtained are contained in intervals, which produce a certain reliability, but they may carry with them an unnecessary quantity of information, since the calculations are not done with maximum accuracy, neither is there the optimal scalar product. With the implementation of the high accuracy arithmetic kernel, the intervals that contain the solution will be the smallest possible and we will have the automatic result verification by the computer itself. Thus, we will finally have the high accuracy and high performance arithmetic available to the users of Supercomputer Center of UFRGS.

Most of the results obtained from the interval methods implemented in libraries were good when compared to the punctual results systems tested (see [HOL96]). The methods based on refinement, in general, were considered as methods that had produced the best results and, in some systems, they produced the result considered arithmetically exact.

The comparison among the libraries implemented in this work has demonstrated that the results obtained in these libraries were approximate, but the results obtained in the `selint.p` library have better accuracy due to the characteristics available in the Pascal-XSC compiler, which are not all available in the `libselint.a` library. One limitation of the `selint.p` library is that the solution of large systems may be impracticable as it needs a large scale of computation and a large amount of memory to store data.

The project in the parallel environment that has been developed at UFRGS is in its initial stage. The basic interval routines have already been implemented in the three environments. Today, we are studying the performance speed optimization of them through its parallel processing in the hardware based on available Transputers. The most delicate phase is choosing the level of their parallel processing, or which operations will be executed in parallel: the most primitive, the intermediate ones, the most elaborated (with manipulation of vectors and interval matrices), or a combination of the former ones?

After the phases of definition and implementation, it will be possible to compare the results obtained from interval libraries in the three environments. Then, we will be able to evaluate the performance of the combination between parallel algorithms and interval operations library with rounding errors verification.

References

[COR93] CORLISS, G. F. **Comparing Software Packages for Interval Arithmetic.** Preprint presented at SCAN'93, Vienna, 1993.

[DIV95] DIVERIO, T. A. **Uso efetivo da matemática intervalar em supercomputadores vetoriais.** Porto Alegre: CPGCC da UFRGS, 1995. Tese de doutorado.

[HAM93] HAMMER, R. et al. **Numerical Toolbox for Verified Computing I: basic numerical problems.** Berlin: Springer-Verlag, 1993. 337p.

[HOL96] HÖLBIG, C.A. **Métodos Intervalares para a Resolução de Sistemas de Equações Lineares.** Porto Alegre: CPGCC da UFRGS, 1996. Dissertação de mestrado.

[IMA91] IMACS, GAMM. **Resolution on computer arithmetic.** In: Computer Arithmetic, Scientific Computation and Mathematical Modeling. KAUCHER, E.; MARKOV, S. M.; MAYER, G.(Eds.).Basel: J.C.Baltzer, 1991. v.12. p.477-479.

[JAI91] JAIN, R. **The art of computer systems performance analysis.** Littleton, Massachussets, Digital Equipment Corporation, 1991.

[KLA 92] KLATTE, R. et al. **PASCAL-XSC Language Reference with Examples.** Berlin: Springer-Verlag, 1992.

[KUL83] KULISCH, U.; MIRANKER, W. L. **A new approach to scientific computation.** New York: Academic Press, 1983. 384p.

[KUL93] KULISCH,U.; RALL, L. B. **Numerics with automatic result verification.** Mathematics and Computers in Simulation. (North Holland). v.35, p.435-450, 1993.