[CI-02]

Pais, J.C., Delgado, R.

"Solving linear equation systems using parallel processing"

International Conference on Lightweight Structures in Civil Engineering, Warsaw, 1995

SOLVING LINEAR EQUATION SYSTEMS USING PARALLEL PROCESSING

J. C. Pais[1] and R. M. Delgado[2]

[1]Professor, School of Engineering,
University of Minho, PORTUGAL
[2]Professor, Faculty of Engineering,
University of Porto, PORTUGAL

ABSTRACT: This paper shows the abilities of the parallel processing in the solution of linear equation systems. The solution of linear equation systems is one of the most time consuming task in the analysis of the structural problems in civil engineering. This is more evident in finite element analysis because the solving phase spends almost the whole time of the analysis. To solve this time consuming it is proposed the use of the parallel processing in the solution of the equation systems.
The Gaussian elimination method, the Cholesky factorization method and the Conjugate Gradient iterative method were chosen. For these methods it was analysed the sequential time, the parallel time, the speedup and the efficiency of the parallel algorithm relatively at the sequential algorithm. Parallel times are gotten for 2 to 16 processors because this work was developed in a parallel computer with 16 transputers IMS T800-20 every one with 2 MBytes of RAM.

1. INTRODUCTION

Problems of structural analysis imply, in general, the solution of linear equation systems of large dimensions which is time consuming.
A great deal of structural studies involves systems of great dimensions which often are not endured by the available calculation means.
The main goal of this work is to clarify the advantages that the parallel processing might provide to structural analysis, more precisely to the solution of linear equation systems. Therefore this work presents some methods of solution of equation systems to adapted to parallel processing.
The parallel processing machine used is included in the group of computers with several processors, each of them having a local memory. These machines processors are connected to each other by a communications net which allows the stored data in a processor's memory to be transfered to another.
This kind of computer known as multicomputer is made up of several processors which are called transputers.
The transputer is considered to be a computer in a single integrated circuit considering that the central processing unit is included in it as well as a certain amount of memory and a connections set to communicate with the exterior. Thus the meaning of the nomination "TRANSPUTER", transistor like a computer.
This work was developed using a computer composed of 16 IMS T800-20 processors, each one containing an arithmetical processor, 2 MBytes of memory and 4 connections to allow communication.

Speedup and efficiency. There is a general tendency to measure not the performance of the parallel computer but that of each parallel developed algorithm which is done through the speedup and efficiency of each algorithm.
The speedup, Sp, of a parallel algorithm "running through" p processors is achieved by the quotient between time, Tseq, spent by the most rapid sequential algorithm "running through" a single processor, and the time, Tp, spent by the parallel algorithm "running through" p processors simultaneously, that is,

$$Sp = Tseq / Tp \qquad (1)$$

The speedup defined by this way, measures as many times the parallel algorithm is more rapid than the sequential algorithm.
The optimum value for the speedup, Sp, should be equal to p, since this means that the parallel algorithm "running algorithm" p processor would take p times less time than the sequential.
The speedup may thus be interpreted as the number of processors working at 100% comparatively to the sequential algorithm.
One defines efficiency, Ep, of a parallel algorithm as the quotient between the speedup, Sp, and the number of processors p.

$$Ep = Sp / p \qquad (2)$$

Efficiency shows a percentual indication of the number of processors which were working at 100% during application comparatively to the time spent by the sequential algorithm.

Communications. In the sequential programs all the necessary data for the execution of the program are stored in the computer's memory, which is unique. This does not happen exactly the same way at parallel processing. Since these machines consist of several processors, each having a certain amount of memory, all data necessary to the problem are stored along the several processors' memory.
In this case to reach a specific datum of the problem it is first necessary to know in which processor it is stored, and only after it is possible to obtain that number through the corresponding processor.
As one can realize, the part related to data treatment in the parallel processing, becomes considerably more complicated.

Parallel programming. The processors are linked to one another by four available connections, this allows the definition of several configurations to the processors net. The parallel programming language allows the renaming of the whole set of junctions between processors in such a way as to allow a simplified usage. This resource permits defining the type of junction between processors more frequently used, the structure "farm", shown in Fig 1, which is based on bidirectional communication between master and slaves. This figure shows the case of the master controlling only two slaves.
In this configuration program A has got as subordinates programs B and C. Program A communicates with the exterior, with B and C. The slaves only communicate with A. Both of these neither communicate with the exterior nor between themselves. Whenever a slave intends to communicate with another slave, this communication must go through A.

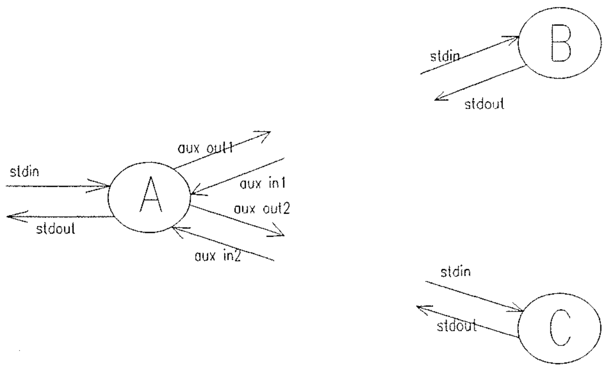


Fig. 1 - Structure "farm" of processors connection.

Extrapolation of values. Furthermore one observes that the parallel algorithms will "run through" several processors each of one storing part of the matrix of the coefficients. This means that in parallel processing one can store more data in the computer memory, which results in the solving of systems dimensions several times superior to what would be possible with one processor only. To study the quality of the parallel algorithm, comparatively to the sequential, it is necessary to extrapolate the time spent by sequential algorithm.
For all sequential algorithms the extrapolation was calculated based on the number of operations made by this algorithm.

2. THE GAUSS METHOD IN SEQUENTIAL PROCESSING

The Gauss method is, perhaps, the most used method in the solution of linear equation systems. This method permits to obtain the solution to any equation system, in an exact way.
The solving of a system of linear equations by the Gauss method involves two steps, elimination and back substitution.
During elimination, all the values of the matrix of the system below the main diagonal are linearly transformed in such a way that they are zero. With the knowledge of the coefficients in a determinated line, the elimination of the column is made. This process begins in the first line of the system with the elimination of all elements in the first column, this process being repeated until no non-zero elements below the main system diagonal exist.
After this step, back substitution starts by the last line of the matrix and the solution of the system corresponding to that line being obtained.

Implementation of the sequential algorithm. To implement the Gauss method it was considered a symmetrical matrix of coefficients and its semiband is stored under the form of a rectangular matrix.

Results. The times of elimination and of back substitution related to the number of equations of the system are depicted in Fig 2 and 3 respectively. These times were obtained by extrapolation of the times of systems up to 1500 equations, the maximum capacity of one processor.

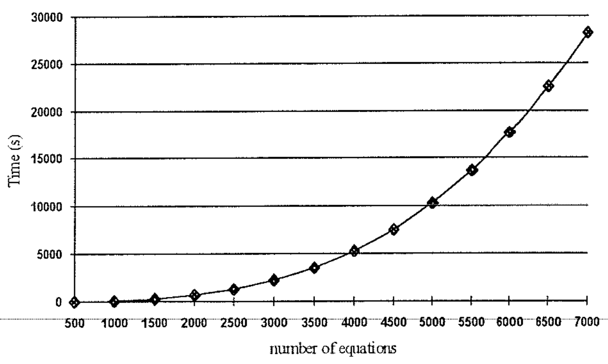


Fig. 2 - Gaussian sequential elimination time related to the number of equations.

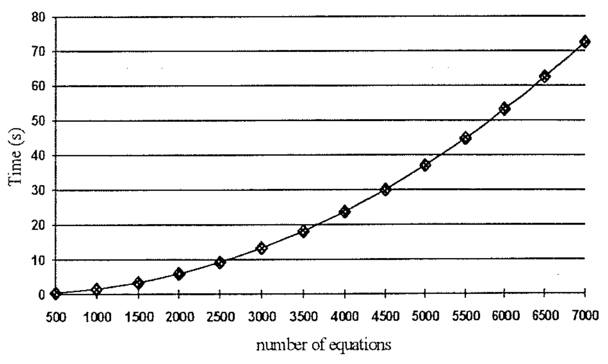


Fig. 3 - Gaussian sequential back substitution time related to the number of equations.

The time spent by the solution of the sequential algorithm which are presented in the Fig 2, 3 will be used to calculate the efficiency of the parallel algorithm proposed for the Gauss method.

The extrapolation of times for the elimination was done using Eqn 3 in which the elimination time is proportionate to the number of additions, S, and to the number of products, P.

$$t=\gamma_S * S+\gamma_P * P \tag{3}$$

being:

$$S=\frac{1}{2}N.lsb^2+\frac{1}{2}N.lsb-\frac{1}{3}lsb^3+\frac{1}{3}lsb-N \tag{4}$$

$$P=\frac{1}{2}N.lsb^2+\frac{3}{2}N.lsb-\frac{1}{3}lsb^3-\frac{1}{2}lsb^2+\frac{5}{6}lsb-2N \tag{5}$$

N, the number of equations in the system
lsb, the width of the semiband

Equation 6 was used in the extrapolation of times for back substitution.

$$t=\gamma*(S+P) \tag{6}$$

being:

$$S=N.lsb-\frac{1}{2}lsb^2-N-\frac{1}{2}lsb \tag{7}$$

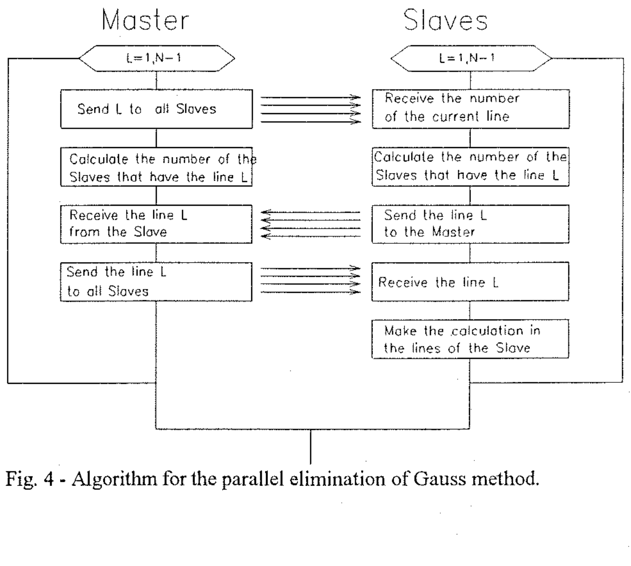
$$P=N.lsb-\frac{1}{2}lsb^2+\frac{1}{2}lsb \tag{8}$$

3. THE GAUSS METHOD IN PARALLEL PROCESSING

To define correctly the form of the parallel algorithm a detailed knowledge of the corresponding sequential algorithm is necessary. In order to solve a determined system each line of the coefficients is used as the basis to change the subsequent lines of the matrix. Each line is used to operate the lines below, those operations being necessary to the elimination of the elements below the main diagonal matrix.

Distribution of the matrix throughout the processors. The systems to be solved and which justify the usage of parallel processing are systems of high dimension which exceed the storage capacity of a sole processor. Thus it is essential that the systems coefficients are stored in more than one processor, the best method for this storage being the distribution of the systems coefficients by all the processors implied in the systems solution.
The cyclical distribution of the lines of the system by the several processors permits that all the processors work from the beginning of the solution till its end.

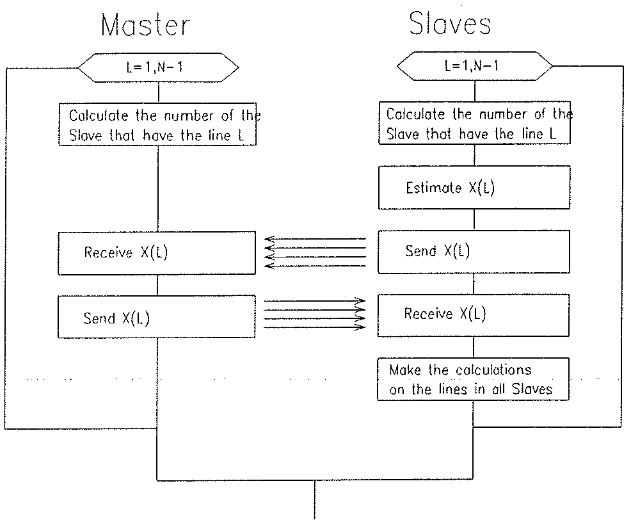
Parallel elimination. As was mentioned above the system coefficients matrix will be stored in several slaves, which with the use of structure "farm", can only communicate with master. Thus, all information exchange between the several slaves must be done through master. Figure 4 illustrates the algorithm proposed to the parallel elimination of Gauss method.



Master
Slaves
L=1,N-1
L=1,N-1
Send L to all Slaves
Receive the number of the current line
Calculate the number of the Slaves that have the line L
Calculate the number of the Slaves that have the line L
Receive the line L from the Slave
Send the line L to the Master
Send the line L to all Slaves
Receive the line L
Make the calculation in the lines of the Slave

Fig. 4 - Algorithm for the parallel elimination of Gauss method.

Parallel back substitution. Figure 5 illustrates the algorithm proposed for parallel back substitution of Gauss method. In this algorithm the amount of calculations that each slave has to do is very small. This way the communication time spent in the distribution of numbers between master and slaves is higher comparatively to the calculation time.



Master
Slaves
L=1,N-1
L=1,N-1
Calculate the number of the Slave that have the line L
Calculate the number of the Slave that have the line L
Estimate X(L)
Receive X(L)
Send X(L)
Send X(L)
Receive X(L)
Make the calculations on the lines in all Slaves

Fig. 5 - Algorithm for parallel back substitution of Gauss method

The Gauss elimination parallel time with several processors are shown in Fig 6. The speedup numbers are depicted in Fig 7. On can observe that the speedup for few processors, during simultaneous processing, come close to the optimum number. The speedup for many processors does not reach the optimum number since the amount of communications between master and the slaves is very big. In spite of this, one may conclude that the numbers are really high.
In Fig 8 the algorithm efficiency, which is very high for systems of great dimension, is represented. In fact speedups superior to 90% of efficiency can be reached.

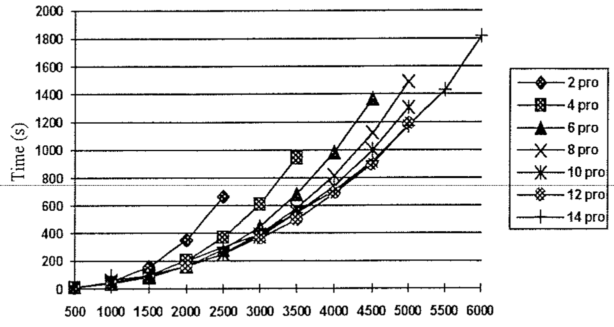


Fig. 6 - Parallel Gaussian elimination times to several processors.

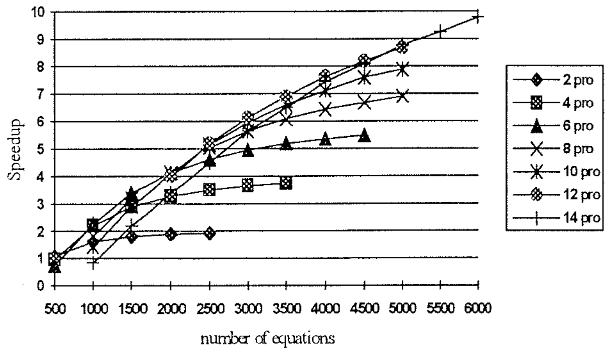


Fig. 7 - Speedup of parallel Gaussian elimination to several processors.

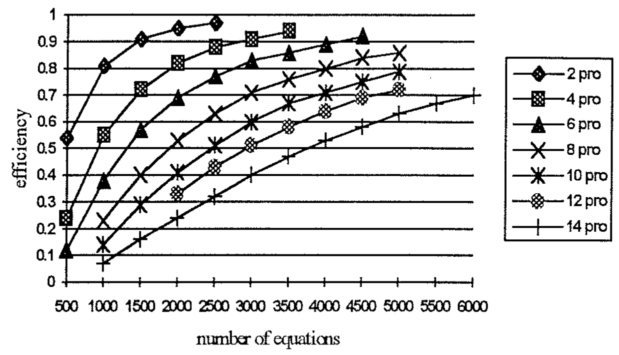


Fig. 8 - Efficiency of parallel Gaussian elimination to several processors.

Concerning back substitution results are not as brilliant as those of elimination. In relation to back substitution the amount of calculation is very small comparatively to the amount of information transferred between master and the slaves. Due to this, back substitution time increase with the number of processors working simultaneously as can be confirmed in Fig 9.

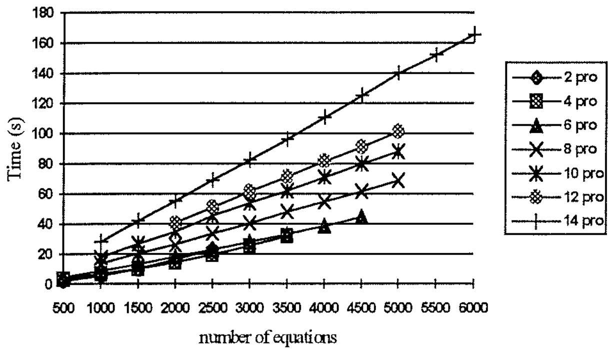


Fig. 9 - Gaussian back substitution parallel times with several processors.

4. THE CHOLESKI METHOD

The great advantage of this method comparatively to the studied above, is obtained where one intends to solve several equations systems with the same matrix of coefficients and only the vectors containing independent terms are different. In this situation the decomposition methods are more efficient than the elimination methods.
In engineering these systems happen frequently, showing up in all problems which present several loading cases for the same structure.
Consider an equations system written in the matricial form by:

$$[A]\{X\} = \{B\} \qquad (9)$$

In this method the matrix [A] is decomposed in two matrixes [L] and [U] one being an inferior triangle and the other being a superior triangle, one transposed of the other.

$$[A] = [L].[L]^t \qquad (10)$$

The solution of a system of equations through this method involves, besides the former decomposition, a substitution and a back substitution. The decomposition of matrix [A] in $[L][L]^t$ by the Choleski method can be implemented by the Eqn 11,12.

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2} \qquad (11)$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} (l_{jk} . l_{ik})}{l_{jj}} \qquad (12)$$

The application of this method is done by calculating the main diagonal element of the current column and, with the help of this number, calculate the remaining elements of the respective column. To calculate these, there is also a need for the current line elements and the line corresponding to the main diagonal.
It is worthy of notice that all the already calculated elements belonging to the matrix L are stored in the position where the matrix coefficients elements can be found, in view of the fact that these are used only once during the whole calculation process.
The capacity of a processor made the solution of systems up to 800 equations possible. With the obtained solution the time spent was extrapolated up to 3200 equations applying the Eqn 13. The obtained values are represented in Fig 10.

$$t = \gamma_S * S + \gamma_P * P \qquad (13)$$

being:

$$S = N^3 + \frac{5}{2}N^2 + 3N \qquad (14)$$

$$P = \frac{N^3}{6} + \frac{N^2}{2} + \frac{N}{3} \qquad (15)$$

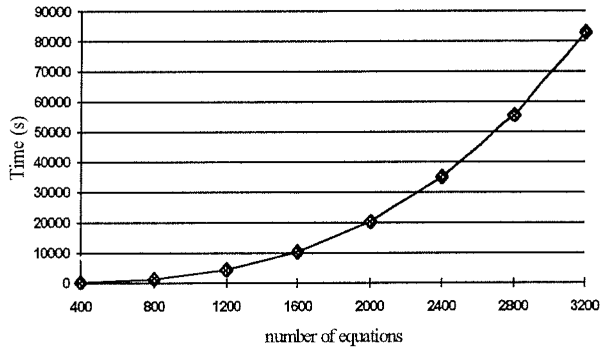


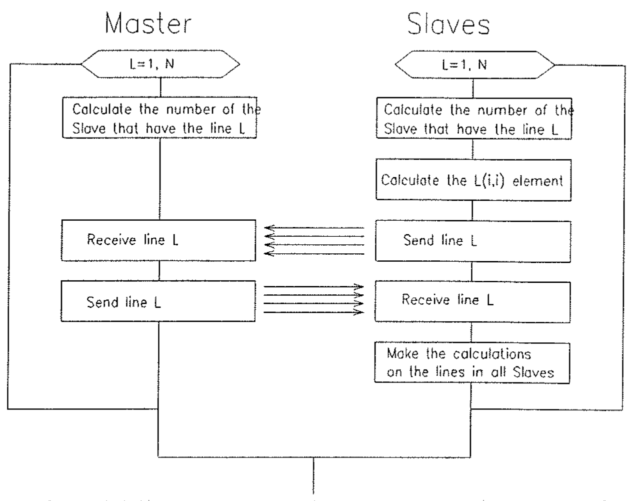
Fig. 10 - Times to the Choleski method for symmetrical matrixes.

The substitution and back substitution in the Choleski method is absolutely identical to the one used in the Gauss method, one should point out that this process spent twice the time of the back substitution of the method Gauss.

5. THE CHOLESKI METHOD IN PARALLEL PROCESSING

It is important to keep in mind that the Choleski method in sequential processing was based on a column to column calculation, and for each column the knowledge of the calculated elements of the line being studied and the line equal to the column is definitely relevant.
The distribution of the coefficients matrix through the several processors is done taking into consideration the way the method of calculation chosen is processed. Factorization is developed column by column, this implies that, as the calculation develops, there are lines in the system which do not come forth in the calculation. Thus one needs to resort to cyclical distribution, otherwise, as the calculation would develop, there are processors without coincident use and an assynchronism of the whole system would appear.

Parallel algorithm. The master program and the slave program is similar to the one already used by the Gauss method. The master is in charge of the information management and the slaves are responsible for the execution of calculations which lead to the solution of the system. To implement parallel factorization, where the coefficient matrix is divided through the several processors, each line of the current matrix has to be sent to all processors in such a way that these may proceed to the calculations of the column which is being studied.
In Fig 11 the enchainment of master with slaves is reproduced.
In Fig 12 the Choleski factorization in relation to the number of processors for symmetrical matrixes are shown and in Fig 13 the corresponding efficiencies.
Comparing this method to the Gauss it presents considerably superior efficiencies, and it is worth noticing that for the majority of the systems solved the efficiency is superior to 90%. One also needs to point out that, for a certain number of equations, the efficiency is less dependent on the number of processors than in the Gauss method.



Master
Slaves
L=1, N
L=1, N
Calculate the number of the Slave that have the line L
Calculate the number of the Slave that have the line L
Calculate the L(i,i) element
Receive line L
Send line L
Send line L
Receive line L
Make the calculations on the lines in all Slaves

Fig.11 - Algorithm for parallel factorization of the Choleski method.

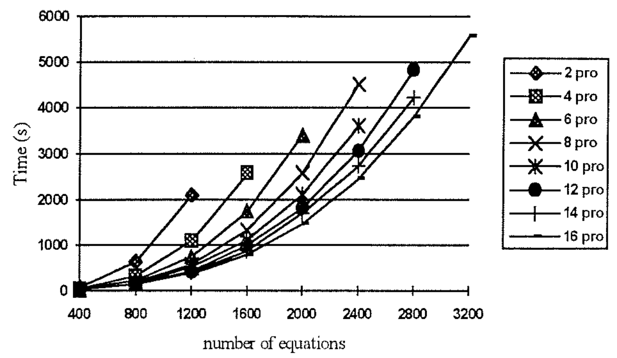


Fig. 12 - Choleski factorization time in relation to the number of processors for the case of symmetrical matrixes.

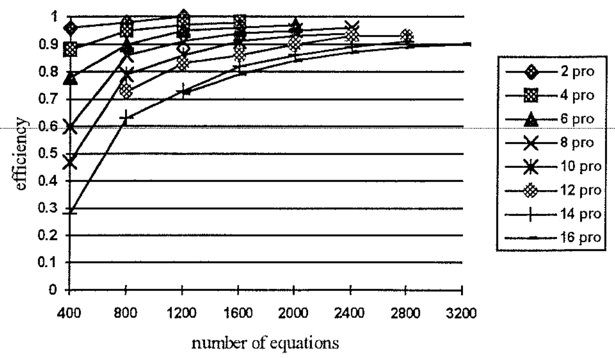


Fig. 13 - Parallel algorithm efficiency for the Choleski method in relation to the number of processors.

6. METHOD OF CONJUGATED GRADIENTS IN SEQUENTIAL PROCESSING

The iteractive methods have always played an important role in the solution of many engineering problems given the speed and simplicity needed. Both direct methods and factorization in the solution of large linear equations systems require a long time only to obtain the solution of a system. The results obtained by these methods may present, in certain situations, considerable mistakes, essentially due to the accumulation of round-off errors resulting from the amount of calculations performed.
Using iteractive methods it is possible to obtain solutions for the same problems with far less calculation that would prove necessary using a direct

method. From the above it is easy to realise that for systems of large dimensions, associated to iteractive methods, there are always reduced periods of time in the solution of linear equation systems.
The implementation of the conjugated gradients methods involves, for each interaction, a matrix by a vector product which is brought up to date in each interaction. This result is responsible for about 95% of the total period of each interaction. So, only the product of one matrix by a single vector will be analysed. Due to its easy understanding a large reference will not be made.

Results. To this method, the case in which the system coefficients matrix is symmetrical and in band was studied. Figure 14 shows the time spent for the product of the system coefficients matrix by the vector already mentioned for each interaction. The time spent for the solution of systems superior to 1500 equations was extrapolated using Eqn 16.

$$t = \alpha * P \tag{16}$$

being:

$$P = (3lsb - 2) * (lsb - 1) + (2lsb - 1) * (n - 2lsb + 2) \tag{17}$$

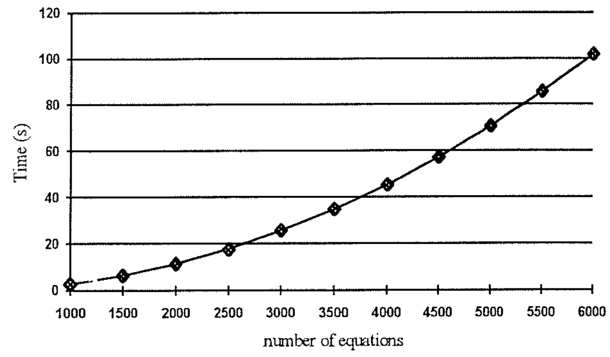


Fig. 14 - Time spent for the product matrix by vector, for the method of conjugated gradients.

7. METHOD OF CONJUGATE GRADIENTS IN PARALLEL PROCESSING

In the method of conjugated gradients, in sequential processing, the most time spent is due to the product of a matrix by a vector. In the tests performed this operation is responsible for about 95% of the time spent on each interaction and, therefore, on the solution of the system.
As is understandable the parallel implementation of the conjugated gradients method should mainly fall upon this result. Thus only this product will be implemented.

Distribution of the matrix through processors. The distribution of the coefficients matrix is done taking into account the way the chosen calculation method is processed. The product of a matrix by a vector in parallel processing, in case a cyclical distribution or one by sub-matrixes is done in a very similar way. For symmetrical and banded matrixes each processor only needs part of the vector to perform the operations with its stored lines.
In Fig 15 is presented an example in which one multiplies the matrix by a vector being the matrix stored in the slaves by blocks and the vector is sent from the master to the slaves in each interaction. This figure allows to conclude that each slave needs only part of the vector. As such, master does not need to send all the elements of the vector to all the slaves.

In Fig 16 one can find the time spent by each interaction of the conjugated gradients method and in Fig 17 for several systems of different dimensions their respective efficiencies. The presented results refer to symmetrical and banded matrixes with a band about 10% of the dimension of the system.

Comparatively to the other methods, the conjugated gradients methods is the one which presents the most stable efficiencies both for a fixed number of processors and for a certain number of equations.

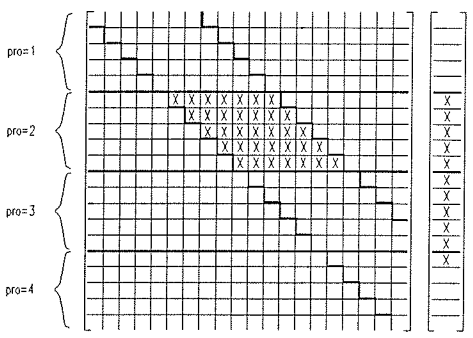


Fig. 15 - The distribution of the coefficients by blocks through the several slaves.

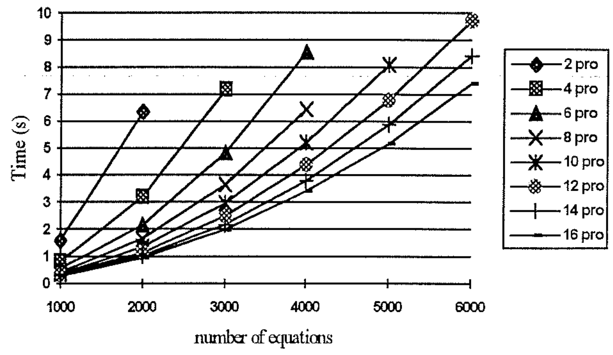


Fig. 16 - Time of each interaction of the method of conjugated gradients in parallel processing for symmetrical banded matrixes.

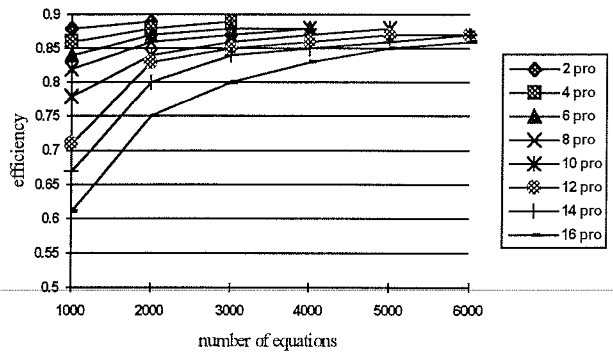


Fig. 17 - Efficiency of the method of conjugated gradients in parallel processing for symmetrical and banded matrixes taking into account the number of processors.

8. CONCLUSIONS

The main goal was to obtain parallel algorithms which would lead to high levels of efficiencies for the most part of possible dimensions to be resolved in the machine available for parallel processing.
In spite of the great difficulty underlying parallel algorithms, the developed algorithms present a quite simple structure.
The algorithms obtained displayed high levels of efficiencies even taking into account the lack of efficiency regarding back substitution.
Accounting for the range of systems tested and for the several methods studied it was concluded that the efficiency to parallel algorithms is considerably high, achieving numbers such as 90 to 95% in the case of processors working with the all of the available memory.
As expected, efficiency decreases with the increase of the number of processors since it implies an increase of communications between processors, because these operations are very time consuming in this type of machine. It was equally observed that for a determined number of processors the efficiency increases with the increase in the number of

equations. This happens due to the fact that the increase of equations enlarges the amount of calculation to be performed by slaves, the only processors truely working in parallel, seeing that master only coordinates and distributes tasks throughout the slaves.
The methods applied to symmetrical matrixes present better efficiencies than those applied to symmetrical and banded matrixes seeing that the latter involved more operations to be executed. The conjugated gradients method was the one that led to smaller solution times in the analysed systems. It is worthy of notice that this being an iteractive method, the final time is very dependent on the type and size of the system, this is why a direct comparison, regarding the time spent by this method with the previous ones, should be made.
In what concerns efficiency the conjugated gradients method is the one that, in spite of not showing, very high efficiencies presents them as very stable both in relation to the number of processors and in relation to the number of equations. In what concerns the other two methods, the Choleski presents solution times inferior to the Gauss and superior efficiencies, consequently its use in parallel processing should not be disregarded.

9. BIBLIOGRAPHY

1. Alvaro Azevedo e Joaquim Barros, Análise Comparativa de Métodos Directos e Iterativos na Resolução de Grandes Sistemas de Equações Lineares, JPEE, Tema A, LNEC, 1990.
2. Alves Filho, The Use of Transputers Based Computers in Finite Element Calculation, Ph D Thesis, University College of Swansea, 1989.
3. Armando R. C. Almeida, Distriduted Continuos Simulation of Large-Scale Systems, Ph D Thesis, Joanesburgo, 1990.
4. Dimitri P. Bertsekas, Parallel and Distributed Computation, Numerical Methods, Prentice-Hall, 1989.
5. Harold Cohen, Mathematics for Scientists and Engineers, Prentice-Hall, 1992.
6. Hojjat Adeli e Marcel Dekker, Parallel Processing in Computation Mechanics, Dekker Inc, 1992.
7. Humberto L. Soriano, Sistemas de Equações Algébricas Lineares em Problemas Estruturais, Seminário 280, LNEC, 1981.
8. Maria Raquel Valença, Métodos Numéricos, INIC, 1990.
9. Numerical Recipes, The art of Scientific Computing, William H. Press, Cambridge University Press, 1986.
10. Yves Robert, The Impact of Vector and Parallel Architectures on the Gaussian Elimination Algorithm, Halsted Press, Manchester University Press, 1990.