ISTITUTO NAZIONALE DI FISICA NUCLEARE
Laboratori Nazionali di Frascati

LNF-78/14(R)
7 Marzo 1978

D. Bini, A.U. Luccio and G.M. Piacentino: PARALLEL MATRIX INVERSION WITH A THREE-PROCESSOR COMPUTER.

Servizio Documentazione
dei Laboratori Nazionali di Frascati
Cas. Postale 13 - Frascati (Roma)

Laboratori Nazionali di Frascati
Servizio Documentazione

LNF-78/14(R)
7 Marzo 1978

D. Bini[*], A. U. Luccio[+] and G. M. Piacentino: PARALLEL MATRIX INVERSION WITH A THREE-PROCESSOR COMPUTER.

1. - INTRODUCTION. -

In many problems of numerical physics, such as magnet design(1), eddy currents distribution calculation(2), plasma simulation(3,4), particle accelerator optics design(5), the solution of large linear and non-linear systems of algebraic equations is necessary.

Such a solution can be obtained by means of several numerical methods, each characterized by its own computational complexity(6) (CC=total number of elementary operations, strictly related to the time needed to perform the alaboration). Frequently, such a problem involves the inversion of a large non-regular and dense matrix.

The solution of non-linear systems is generally obtained by means of an iterative method, like the Successive Substitutions or the Newton-Raphson methods(7), both consisting of a step-by-step linearization of the problem.

It is apparent that, if possible, the optimization of the CC of inverting a matrix produces a large saving in elaboration time. Such an inversion may be performed either sequentially(8) on a single high-speed computer, or in parallel(9) by using several microprocessors connected to a number of memories via a bus-multiplexer, thus forming an unique array-processor.

In this paper, we are concerned with the possibility of making the inversion of a matrix by means of an array-processor consisting of three processors and three memories connected through a cross bar switch, that settles conflicts and allows the contemporary access to each memory. Moreover, each microprocessor has a private memory composed of a few registers. We will compare the complexities of three different algorithms and suggest a method of computing the inverse of a matrix, in the case when the rank is a multiple of three.

(*) Univeristy of Pisa, Italy.
(+) On leave from the University of Pisa, Italy.

2. - GAUSS AND GAUSS-JORDAN METHODS. -

The solution of a linear system and, consequently, the inversion of a square matrix, can be performed by methods of successive eliminations like the Gauss and Gauss-Jordan methods(10). In the first method the original matrix is turned into a superior triangular, in the second into a diagonal matrix.

The total number of elementary operations (CC) needed to solve 'm' linear systems of equations by the Gauss method is(6)

$$C_{G,S}(m,n) = \frac{n}{3}(n^2-1)p + mn^2 p + \frac{n(n-1)(2n-1)}{6} s + \frac{m(n-1)(n+2)}{2} s \,, \qquad (1)$$

where 'p' denotes an elementary multiplication or division, and 's' an addition.

The inversion of an 'n x n' square matrix may be obtained solving, by Gauss method, 'n' systems of linear equations, where the column vector on r. h. s. is a vector of the basis (i. e. it has only one non-zero element equal to 1 and n-1 zero elements). The CC of such an inversion is computable replacing 'm' with 'n' in (1) and recalling the special character of the r. h. s. column vectors. We obtain

$$C_{G,I}(n) = n^3 p + \frac{n^2(n-1)}{2} s. \qquad (2)$$

The Gauss-Jordan method seems more attractive since it furnishes directly the elements of the inverse matrix. It involves however a larger CC that the Gauss method. We have in fact, for 'm' systems

$$C_{GJ,S}(m,n) = \frac{n}{2}(n^2-1)\, p + mn^2 p + n(n-1)\left(\frac{n-1}{2} + m\right) s. \qquad (3)$$

The CC of inversion is also larger here than before.
In any case, however, it is clear that the Gauss method in its simpler form cannot be utilized by parallel processing, because of the inherent sequentiality of the substitution processes involved.

3. - MODIFIED GAUSS METHOD. -

We will show that it is possible to apply a Gauss method to a matrix obtained by subdividing the original one into blocks of matrices of rank equal to the number of processors available, and inverting a matrix in a parallel fashion.

In what follows, we will consider the case of parallel utilization of three processors, and hence the original matrix will be subdivided into a number of 3 x 3 matrices. It should be noted that the rank of a large number of matrices that occur in numerical physics problems is a multiple of three, since the matrices contain the components of vectors in the ordinary space (for instance, to compute the magnetization of an iron block, usually the magnetization vectors in a number of region of the material are considered(11)). On the other hand, were the dimension of the matrix not a multiple of three, we could always border it with one or two rows and columns of zeros, with 1's on the diagonal.

Let us write then

$$n = 3h, \qquad (4)$$

for the rank of the matrix to be inverted.

The Gauss elimination method consists in building up, at each i-th step, n-i multipliers obtained by multiplying one element in n-1 rows by the inverse of an element called the Pivot. Next step is to subtract the row containing the Pivot (P-row), multiplied in turn by the multipliers from each row under the P-row.

Formally, we can execute these operations on the 3 x 3 matrices in the same way as on the scalar elements of an ordinary matrix. However, in the former case, we have to operate the inversion of a 3 x 3 matrix before multiplying, while in the latter we have only to divide.

The CC is again given by (2), where we have to replace 'p' by 'p_3', product of 3 x 3 matrices, 's' by 's_3' sum of 3-matrices, and to add h inversions i_3, inversion of 3-matrices. The complexity of inverting a block matrix by means of the Gauss elimination method is thus given by

$$C_{GB,I}(n=3h) = h^3 p_3 + \frac{h^2(h-1)}{2} s_3 + h i_3. \tag{5}$$

3. - BORDERING METHOD. -

An alternative method for reducing the CC or the time of computation consists in building up, by successive bordering of an initial element of which we know the inverse, the matrix A_{n+1} and its inverse A^{-1}_{n+1}.

Let us write a matrix A_{n+1} as follows

$$A_{n+1} = \begin{pmatrix} A_n & U_n \\ & \\ V_n & a_{n+1,n+1} \end{pmatrix}, \tag{6}$$

where A_n is a n x n matrix, U_n a column vector, V_n a row vector, $a_{n+1,n+1}$ a scalar element, and where we assume that the inverse matrix A^{-1}_n is known.

We shall write the inverse of A_{n+1} in a form analogous to (6). Thus

$$A^{-1}_{n+1} = \begin{pmatrix} B_n & R_n \\ & \\ Q_n & \alpha^{-1}_{n+1,n+1} \end{pmatrix}. \tag{7}$$

Since it is

$$A_{n+1} A^{-1}_{n+1} = E_{n+1}, \tag{8}$$

where E_{n+1} is the identity matrix of rank n+1, from (8) we obtain the following relations among the elements of A_{n+1} and A^{-1}_{n+1}

$$\left| \begin{array}{l} A_n B_n + U_n Q_n = E_n \\ V_n B_n + a_{n+1,n+1} Q_n = 0 \\ A_n R_n + U_n \alpha^{-1}_{n+1,n+1} = 0 \\ V_n R_n + a_{n+1,n+1} \alpha^{-1}_{n+1,n+1} = E_1 \end{array} \right. \tag{9}$$

From the third of (9) we have

$$R_n = - A^{-1}_n U_n \alpha^{-1}_{n+1,n+1}, \tag{10}$$

and, by substituting into the fourth of (9), we obtain

$$\alpha_{n+1,n+1} = a_{n+1,n+1} - V_n A_n^{-1} U_n . \qquad (11)$$

Moreover, from the first of (9), it is

$$B_n = A_n^{-1} - A_n^{-1} U_n Q_n , \qquad (12)$$

and finally, from the second of (9) and from (11), we get

$$Q_n = - V_n A_n^{-1} \alpha_{n+1,n+1}^{-1} . \qquad (13)$$

Thus, the expressions (10) through (13) give us the elements of the inverse matrix A_{n+1}^{-1} in terms of the elements of A_{n+1} and of the matrix A_n^{-1}. They show how to build up the inverse of any matrix by means of an iterative bordering algorithm, starting from the first diagonal element.

As in the case of Gauss elimination, also in the method we have just examined, the expressions obtained hold in the case of a matrix built of blocks, where each block is a 3 x 3 matrix, since in (10) and (13) the commutativity of products between matrix elements has not been used. The bordering method can also be used therefore in the present case, where $a_{n+1,n+1}$ and $\alpha_{n+1,n+1}^{-1}$ are 3 x 3 matrices, like the elements of V, U, R, Q.

To compute the elements of A_{n+1}^{-1}, we can procede as follows: first, compute the products

$$A_n^{-1} U_n \text{ and } V_n A_n^{-1} , \qquad (14)$$

then compute $\alpha_{n+1,n+1}$ and invert it, to get $\alpha_{n+1,n+1}^{-1}$.

Next , obtain R_n and Q_n from (10) and (13) and finally, multiplying the first of (14) by Q_n, obtain B_n from (12).

The CC of this process can be evaluated in the following way:
The products (14) require

$$2 \times (n^2 p_3 + n(n-1)s_3).$$

To compute $\alpha_{n+1,n+1}$ we need

$$np_3 + (n-1+1)s_3 = np_3 + ns_3 .$$

To invert $\alpha_{n+1,n+1}$ and obtain $\alpha_{n+1,n+1}^{-1}$ a single inversion is needed

$$1 \times i_3.$$

To compute R_n and Q_n require

$$2 \times np_3 ,$$

and for B_n we need

$$n^2 p_2 + n^2 s_3 .$$

Going from 'n' to 'n+1' the complexity is thus

$$3n(n+1)p_3 + n(3n-1)s_3 + i_3 . \qquad (15)$$

We obtain finally the total CC for inverting A_{3h} by adding one inversion to the summation of expressions like (15), for 'n' running from 1 to h-1

$$C_{BB,I}(n=3h)=3p_3 \sum_1^{h-1} n(n+1)+s_3 \sum_1^{h-1} n(3n-1)+\sum_1^{h} i_3 = (h^3-h)p_3 + \frac{h(h-1)(2h-1)}{2} s_3+hi_3 . \qquad (16)$$

If we compare this expression with (5), that gives the CC of the inversion by means of a Gauss block-elimination, we can see that both the numbers of products and sums are reduced.

Thus, the block-bordering method appears cheaper in computing time than the Gauss method.

4. - ALGORITHMS FOR SUMMING, MULTIPLYING AND INVERTING 3 x 3 MATRICES. -

The expressions for the CC to invert n x n matrices contain the following parameters: p_3, complexity of multiplying 3 x3 matrices, s_3, complexity of summing and i_3, complexity of inverting 3-matrices. We have now to write expressions for p_3, s_3, and i_3 in terms of elementary operations, such as commutative products p and sums of scalar elements of matrices.

Since the row-by-column product between two n x n matrices needs n^3 elementary products, and $n^2(n-1)$ elementary sums, we have

$$p_3 = n^3 p + n^2(n-1)s \quad \left\{ n^3 = 27, \; n^2(n-1) = 18 \right\} . \tag{17}$$

The CC given by (17) may be reduced, by the use of special algorithms, as the Strassen or the Winograd's(12). In our case, an algorithm equivalent to Winogrd's is obtained, when we observe that each element of the product matrix C=A. B can be written as

$$C_{ij} = \sum_{k=1}^{3} a_{ik} b_{kj} = (a_{i1}+b_{2j})(a_{i2}+b_{1j}) - a_{i1}a_{i2} - b_{1j}b_{2j} + a_{i3}b_{3j} \; , \tag{18}$$

by making use of the commutativity of the product between scalars.

It is clear that, once performed the six products $a_{i1}\,a_{i2}$ and $b_{1j}\,b_{2j}$, each c_{ij} needs two products and five sums. The complexity of the algorithm is thus

$$p_3 = (6+18)p + 5 \times 9 \; s = 24 \; p + 45 \; s. \tag{19}$$

The computing time (18) is certainly less than (16), because sums are much faster than products.

The sum of 3-matrices needs nine elementary sums

$$s_3 = n^3 s \; , \quad n^3 = 9 \; . \tag{20}$$

The CC of the inversion of a matrix is given, by the Gauss method, by

$$i_{3G} = 27 \; p + 36 \; s, \tag{21}$$

and with the bordering method, by

$$i_{3B} = 24 \; p + 15 \; s \; . \tag{22}$$

To our purposes, the most important feature of an algorithm to perform products, sums and inversions is that of being implementable for parallel computation on a three-processor computer.

The product p_3, with the complexity given by (19), is easily parallelizable because it needs only products between different elements. The same we can say about the sum s_3.
With evident notation, we can write

$$p_3 = 3 \times (8p+15s), \quad s_3 = 3 \times 3 \; s. \tag{23}$$

To perform a product, between 3 x 3 matrices we need therefore only the time required to execute 8p and 15s, and for a sum we need only the time equivalent to 3s.

In the case of an inversion, the problem of parallelization has to be explicitly faced. We try, to build an algorithm and show that it can be executed by three processors connected with three independent memories.

Let us consider three processors, called α, β, γ, each with a private memory of a few registers, and connected by means of a cross-bar switch to three independent common memories, that we shall denote by the symbols Γ, Δ, Θ, as it is shown in Fig. 1.

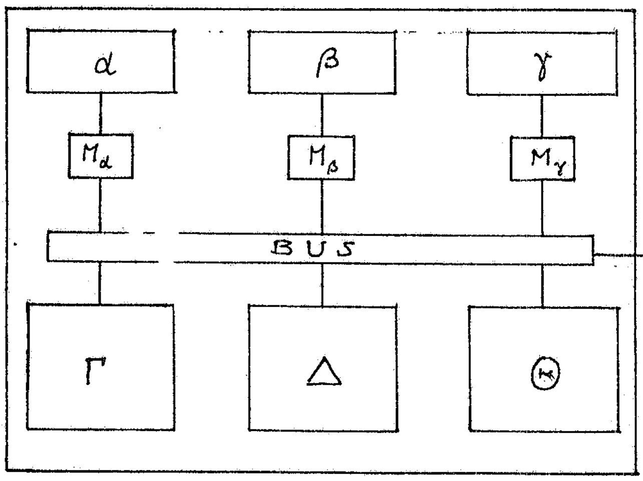

FIG. 1- Diagram of a system of three processors and of three memories for parallel inversion of 3 x 3 matrices.
α, β, γ, microprocessors.
M_α, M_β, M_γ, private memories.
Γ, Δ, Θ, common memories.

Let A be the matrix to be inverted

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} , \qquad (24)$$

where a_{ij} are scalars.

Let us memorize the elements column by column in the three common memories, as follows

$$\begin{array}{ccc} \Gamma & \Delta & \Theta \\ \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} & \begin{pmatrix} a_{12} \\ a_{22} \\ a_{32} \end{pmatrix} & \begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix} \end{array} \qquad (25)$$

At the t_1 clock time, the processor α enters the Γ memory and transfers a_{21} into its private memory M_α. At the same time β puts a_{32} into M_β and γ puts a_{13} into M_γ.

At the successive time t_2, α gets a_{32} into M_α, β gets a_{13} into M_β, and γ a_{21} into M_γ. As a next step, each processor performs a multiplication between the elements which are in its private memory and puts the resulting product into a common memory: explicitly, α writes into Δ, β into Θ, and γ into Γ.

After the time t_2 the configuration will be the following

private memories	M_α	M_β	M_γ
t_1	a_{21}	a_{32}	a_{13}
t_2	a_{32}	a_{13}	a_{21}
	⋮	⋮	⋮

(26)

common memories	Γ	Δ	Θ
	a_{11}	a_{12}	a_{13}
	a_{21}	a_{22}	a_{23}
	a_{31}	a_{32}	a_{33}
t_2	$a_{13}a_{21}$	$a_{21}a_{32}$	$a_{32}a_{13}$
	$\vdots$	$\vdots$	$\vdots$

(26)

The process goes on as follows, α reads cyclically the elements of the last two rows of the matrix, while β reads from the first and the third row and γ from the first two rows.

At the t_7 time the common memories configuration will be

	Γ	Δ	Θ
	$\vdots$	$\vdots$	$\vdots$
t_2	$a_{13}a_{21}$	$a_{21}a_{32}$	$a_{32}a_{13}$
t_3	$a_{12}a_{23}$	$a_{23}a_{31}$	$a_{31}a_{12}$
t_4	$a_{11}a_{22}$	$a_{22}a_{33}$	$a_{33}a_{11}$
t_5	$a_{12}a_{21}$	$a_{23}a_{32}$	$a_{31}a_{13}$
t_6	$a_{13}a_{22}$	$a_{21}a_{33}$	$a_{32}a_{11}$
t_7	$a_{11}a_{23}$	$a_{22}a_{31}$	$a_{33}a_{12}$.

(27)

Then the nine minor determinants are computed: α works in Δ evaluating the differences $a_{22}a_{33}-a_{23}a_{32}$, $a_{23}a_{31}-a_{21}a_{33}$, and $a_{21}a_{32}-a_{22}a_{31}$, and writing into Θ, Γ, Δ, cyclically. At the same time β works in Θ writing into Γ, Δ, Θ and γ in Γ, writing into Δ, Θ, Γ.

At the time t_{10}, the configuration will be

	Γ	Δ	Θ
	$\vdots$	$\vdots$	$\vdots$
t_8	$a_{32}a_{13}-a_{12}a_{33}$	$a_{12}a_{23}-a_{13}a_{22}$	$a_{22}a_{33}-a_{23}a_{32}$
t_9	$a_{13}a_{31}-a_{21}a_{33}$	$a_{11}a_{33}-a_{13}a_{31}$	$a_{13}a_{21}-a_{11}a_{23}$
t_{10}	$a_{11}a_{22}-a_{12}a_{21}$	$a_{21}a_{32}-a_{22}a_{31}$	$a_{12}a_{31}-a_{11}a_{32}$.

(28)

Next, the determinant δ is evaluated three times, as follows

$$\alpha : k_1 = a_{11}(a_{22}a_{33}-a_{23}a_{32})$$
working in Γ and Θ and writing into Δ

$$\beta : k_2 = a_{12}(a_{23}a_{31}-a_{21}a_{33})$$
working in Δ and Γ and writing into Θ

$$\gamma : k_3 = a_{13}(a_{21}a_{32}-a_{22}a_{31})$$
working in Θ and Δ and writing into Γ,

(29)

and then

$$\begin{aligned} \alpha &: \delta = k_1 + k_2 + k_3 \text{, write into } \Theta \\ \beta &: \delta = k_2 + k_3 + k_1 \text{, write into } \Gamma \\ \gamma &: \delta = k_3 + k_1 + k_2 \text{, write into } \Delta. \end{aligned} \tag{30}$$

Finally, with three divisions, α working in the Γ memory, β in Δ, and γ in Θ, we obain the elements of A^{-1}

The whole process needs

3 x 6p	for step (27)
3 x 3s	for step (28)
3 x 1p	for step (29)
3 x 2s	for step (30)
3 x $3p^+$	for the last step.

(+ divisions).

As a total

$$i_3 = 3 \times (10p + 5s) \quad . \tag{31}$$

5. - CONCLUSION. -

The above shows that, by the processor system depicted in Fig. 1, it is convenient to invert numerical matrices. The parallel process has been derived by simple extentions of classical algorithms and should therefore be quite stable.

It is useful to consider the possibility of employing a number of processors larger than three. To deal with physical problems, that number should be a power of three.

REFERENCES.

(1) A. U. Luccio, Proc. Fifth Intern. Conf. on Magnet Technology (MT-5). Roma April 21 (1975)p. 183.
(2) A. U. Luccio, Contr. to the 4th Work Meeting on ERA. IPP, Garching, February 5 (1971)p. 63.
(3) Proc. of the Fourth Conf. on Numerical Simulation of Plasmas, Naval Res. Lab. , Washington, Nov. 2-3 (1970).
(4) J. Killeen (ed.), Methods in Computational Physics (Academic Press, 1976), Vol. 16.
(5) J. S. Colonias, Particle Accelerator Design: Computer Programs (Academic Press, 1974).
(6) E. Isaacson and H. B. Keller, Analysis of Numerical Methods (Wiley, 1966).
(7) J. M. Ortega and W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables (Academic Press, 1970).
(8) IBM - Scientific Subroutine Package, Programs and references therein.
(9) D. J. Kuck and K. Maruyama, The parallel Evaluation of Arithmetic Expressions, Tech. Report University of California, Berkeley (1975).
(10) D. K. Faddeev and V. N. Faddeeva, Computational Methods of Linear Algebra (Freeman, 1963).
(11) M. J. Newman, J. Simkin, C. W. Trowbridge and L. R. Turner, GFUN User's Guide, Rutherford Lab. Report RHEL/R 244, (1972).
(12) L. Csanky, Ph. D. Thesis 1973, University of California, Dept. of Engineering, Berkeley.