ELSEVIER Linear Algebra and its Applications 332–334 (2001) 111–117

LINEAR ALGEBRA AND ITS APPLICATIONS

www.elsevier.com/locate/laa

A study of the performance of Neville elimination using two kinds of partitioning techniques

Pedro Alonso [a,*,1], Raquel Cortina [b,1], Vicente Hernández [c], José Ranilla [b,1]

[a] *Departamento de Matemáticas, Universidad de Oviedo, 33271 Gijón, Spain*
[b] *Centro de Inteligencia Artificial, Universidad de Oviedo, 33271 Gijón, Spain*
[c] *Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 46020 Valencia, Spain*

Received 17 December 1999; accepted 26 May 2000

Submitted by F. Puerta

Abstract

In this paper, a method to compute the solution of a system of linear equations by means of Neville elimination is described using two kinds of partitioning techniques: Block and Block-striped. This type of approach is especially suited to the case of totally positive linear systems, which is present in different fields of application. Although Neville elimination carried out more floating point operations than Gaussian elimination in some cases, in this study we confirm that these advantages disappear when we use multiprocessor systems. On the other hand, the overall parallel run time of Neville elimination is better than Gauss time as Neville elimination uses a lower cost communication model. © 2001 Elsevier Science Inc. All rights reserved.

Keywords: Neville elimination; Block; Striped; Performance

* Corresponding author. Tel.: +34-98-5182032; fax: +34-98-5182125.
E-mail address: palonso@etsiig.uniovi.es (P. Alonso).

[1] Partially supported by the University of Oviedo (NP-99-513-15).

PII: S 0 0 2 4 - 3 7 9 5 (0 0) 0 0 1 8 7 - 7

1. Introduction

Neville elimination is a method that creates zeros in a column of a matrix by adding to each row an appropriate multiple of the previous one. This kind of elimination has been used sporadically in a number of problems, but it has not been analyzed in depth until a sequence of recent papers (see [1–3,5–7]). In these papers, it has been confirmed that Neville elimination has advantages over Gaussian elimination when we work with certain kinds of matrices, in particular, totally positive matrices [10].

A matrix is said to be totally positive if and only if all its minors are non-negative. Ando has presented a good survey of this kind of matrix in [4]. Such matrices frequently appear in problems of statistics, theory of approximation and computer assisted geometric design, as well as in other fields.

Neville elimination provides very simple algorithmic characteristics of totally positive matrices and their subclasses. In general, the computational cost is the same for both Neville and Gauss eliminations. However, it is well known that the computational cost of Neville elimination can be lower than that of Gaussian elimination. This is the case of certain special matrices, such as the totally positive matrices that are a reverse version of a band matrix (see [6]).

At the same time, studies that we have carried out in parallel confirm the advantages of Neville elimination over the Gaussian method when we work with distributed parallel computers. The reason is that Neville uses a lower cost communication model (see [2]).

In the present paper, we study the performance of the Neville method from different points of view. First, we present the method and estimate its cost. Second, in Section 3, we describe a block-oriented version and introduce a parallel algorithm for this kind of partitioning technique. In both cases (sequential and parallel), we study their performance from the computational point of view. Finally, we present several parallel versions based on unidimensional data partitioning. Obviously, we also set bounds to its cost.

It is important to mention the fact that in the parallel algorithms we shall develop, we assume that the processors work synchronously, that is to say, at any given time, all processors work on the same iteration.

2. Neville elimination

Neville elimination is a procedure to make zeros in a column of a matrix by adding to each row a multiple of the previous one. Eventual reordering of the rows of the matrix may be necessary. This is very useful for some classes of matrices such as totally positive matrices. For a detailed introduction to this process, we refer the reader to [5]. Here we restrict ourselves to a brief description.

Let us consider the important case in which Neville elimination can be performed without exchanging rows; this happens, for example, when A is a nonsingular totally

positive matrix (see [5]). We assume Neville elimination to be carried out without row changes.

If A is a square nonsingular matrix of order n, this elimination procedure consists in $n-1$ successive major steps, resulting in a sequence of matrices of the following form:

$$A = A_1 \longrightarrow A_2 \longrightarrow \cdots \longrightarrow A_n = U,$$

where U is a upper triangular matrix.

The matrix $A_k = (a_{ij}^{(k)})_{1\leqslant i,j\leqslant n}$ has zeros below its main diagonal in the first $k-1$ columns, also one column has

$$a_{ik}^{(k)} = 0, \;\; i \geqslant k \quad \Longrightarrow \quad a_{hk} = 0 \;\; \forall h \geqslant i. \tag{1}$$

To get A_{k+1} from A_k, we produce zeros in the column k below the main diagonal by subtracting a multiple of the ith row from the $(i+1)$th for $i = n-1, n-2, \ldots, k$, according to the expression

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{i-1,k}^{(k)}} a_{i-1,j}^{(k)} & \text{if } k+1 \leqslant i \leqslant n, \; a_{i-1,k}^{(k)} \neq 0, \; k \leqslant j \leqslant n, \\ a_{ij}^{(k)} & \text{elsewhere.} \end{cases} \tag{2}$$

The element $p_{ij} = a_{ij}^{(j)}$, with $1 \leqslant i, j \leqslant n$, is called the (i, j) pivot of Neville elimination of A and the number

$$m_{ij} = \begin{cases} \dfrac{a_{ij}^{(j)}}{a_{i-1,j}^{(j)}} & \text{if } a_{i-1,j}^{(j)} \neq 0, \\ 0 & \text{if } a_{i-1,j}^{(j)} = 0 \; (\Longrightarrow a_{ij}^{(j)} = 0), \end{cases}$$

the (i, j) multiplier.

Now, let $Ax = b$ a system of linear equations with A, a nonsingular matrix and where Neville elimination is to be carried out without row changes. From the computational point of view, we can calculate the number of operations in float point that are carried out to compute the solution. The total cost is

$$T \;\; \textit{sequential} \in \theta(n^3 \, t_c), \tag{3}$$

where t_c is the time spent to carry out one operation in float point. This cost coincides with the cost of sequential Gaussian elimination (see [8]).

3. Block partitioning

Block Neville elimination has been studied in depth by Alonso and Peña in [3]. In this section, we shall briefly describe this strategy and present a parallel version on a computer in which each processor is capable of executing a different program independently of the other processors multiple instruction/multiple data (MIMD). We shall observe that the computational cost of the sequential version is greater than

that of the Gaussian one, although the upper bound is the same, and that the parallel Neville algorithm has an equal or lower cost than the Gaussian one.

If $A = (A_{ij})_{1\leqslant i,j\leqslant p}$, and A_{ij} is a $q \times q$ submatrix of A for $1 \leqslant i, j \leqslant p$, with $pq = n$, block Neville elimination consists of $p - 1$ successive major steps, resulting in a sequence of matrices $A^{(k)}$, with $k = 1, 2, \ldots, p$, where $A^{(p)} = U$ is a block upper triangular matrix.

The matrix $A^{(k)} = (A_{ij}^{(k)})_{1\leqslant i,j\leqslant p}$ $(1 \leqslant k \leqslant p)$ has zero blocks below its main diagonal in the first $k - 1$ columns. Therefore the form of $A^{(k)}$ is

$$A^{(k)} = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} & \cdots & \cdots & \cdots & \cdots & A_{1p}^{(k)} \\ 0 & A_{22}^{(k)} & \cdots & \cdots & \cdots & \cdots & A_{2p}^{(k)} \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & A_{k-1,k-1}^{(k)} & A_{k-1,k}^{(k)} & \cdots & A_{k-1,p}^{(k)} \\ \vdots & \vdots & \vdots & 0 & A_{kk}^{(k)} & \cdots & A_{kp}^{(k)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & A_{pk}^{(k)} & \cdots & A_{pp}^{(k)} \end{pmatrix},$$

where 0 will denote a zero matrix.

In order to perform step k of block Neville elimination, we must have

$$\det(A_{ik}^{(k)}) = 0, \quad i \geqslant k \quad \Longrightarrow \quad A_{hk}^{(k)} = 0 \ \ \forall h > i. \tag{4}$$

To get $A^{(k+1)}$ from $A^{(k)}$ taking into account the fact that if $k + 1 \leqslant i \leqslant p$, $\det(A_{i-1,k}^{(k)}) \neq 0$, and $k \leqslant j \leqslant p$, then

$$A_{ij}^{(k+1)} = A_{ij}^{(k)} - A_{ik}^{(k)}[A_{i-1,k}^{(k)}]^{-1}A_{i-1,j}^{(k)}, \tag{5}$$

where $A_{ij}^{(k+1)} = A_{ij}^{(k)}$ elsewhere.

From the computational point of view, block algorithms are important because they are rich in matrix multiplication. Let us recall that matrix operations are basic to many high performance computer architectures.

If we analyze the algorithm of block Neville elimination using only one processor, we can observe that the only difference in relation to block Gauss method is that in the iteration k Neville calculates $p - k$ reverses of matrices of order q, whereas Gauss computes only one reverse. The total cost of this process over $p - 1$ iterations is approximately $(n^3/p)\ t_c$ for Neville and $(n^3/p^2)\ t_c$ for Gauss. However, the sequential run time of both methods is asymptotically $n^3\ t_c$.

Let us consider an MIMD computer, where the p^2 processors form a mesh so that processor P_{ij} stores block (i, j) of the matrix $A^{(k)}$, that is $A_{ij}^{(k)}$ with $1 \leqslant i, j \leqslant p$.

We have the following steps to obtain matrices $A_{ij}^{(k+1)}$ $(1 \leqslant k \leqslant p - 1)$ for $k + 1 \leqslant i \leqslant p$ and $k \leqslant j \leqslant p$:

Step 1. Calculate in P_{ik} the matrices $[A_{ik}^{(k-1)}]^{-1}$, with $i = p-1, p-2, \ldots, k$. The time spent in this computation is $\theta(q^3\, t_c)$.

Step 2. Send the matrix $[A_{i-1,k}^{(k)}]^{-1}$ from $P_{i-1,k}$ to P_{ik} and $A_{i-1,j}^{(k)}$ from $P_{i-1,j}$ to P_{ij} for $i = p, p-1, \ldots, k+1$ and $j = k+1, \ldots, p$. The time for the transmission of a message of q^2 floating point numbers between two directly connected processors is $t_s + q^2\, t_w$ (see [11]), where t_s denotes the startup time and t_w is the transmission time of a floating point number.

Step 3. Calculate in P_{ik} the matrix $A_{ik}^{(k)}[A_{i-1,k}^{(k)}]^{-1}$, with $i = p, p-1, \ldots, k+1$. The cost of this calculation is $\theta(q^3\, t_c)$.

Step 4. One communication that requires a one-to-all broadcast of the matrix $A_{ik}^{(k)}[A_{i-1,k}^{(k)}]^{-1}$ along the active part of the ith row for $i = p, p-1, \ldots, k+1$. This communication step takes $(t_s + q^2\, t_w)\log p$ time on a mesh.

Step 5. Computation of the matrices $A_{ij}^{(k+1)}$ with $k \leqslant i \leqslant p$ and $k \leqslant j \leqslant p$. The time of this computation step is $\theta(q^3\, t_c)$.

Hence, the total time spent during the computation steps in the parallel implementation of block Neville elimination is $\theta((n^3/p^2)\, t_c)$, and the total communication time over all iterations is $\theta(p \log p\; t_s) + \theta((n^2/p)\log p\; t_w)$. The total parallel run time is

$$T\,block(n, p^2) \in \theta\left(\frac{n^3}{p^2}\, t_c\right) + \theta(p \log p\; t_s) + \theta\left(\frac{n^2}{p} \log p\; t_w\right). \tag{6}$$

The computation time is the same for block Neville and block Gauss. However, the communication time for Neville is lower than for Gauss (see [8]). This is due to the fact that some of the communications of Neville are between neighboring processors, whereas the communications of Gauss require a single processor to send identical data to all other processors or to a subset of them.

4. Block-striped partitioning

This section discusses parallel formulations of the Neville elimination method explained in Section 2. We thus consider the resolution of a system of linear equations using two kinds of data partitioning, that is to say, two ways to partition matrices among the processors. We shall see that data partitioning significantly affects the performance of a parallel system.

We shall consider a system of linear equations of order n, in matrix notation this system is written as $Ax = b$. Here $A = (a_{ij})_{1\leqslant i,j\leqslant n}$ is the matrix of coefficients and $b = (b_i)_{1\leqslant i\leqslant n}$ is the vector of independent terms. The matrix A is nonsingular and we assume the elimination process to be carried out without row changes.

Let us consider an MIMD computer with r processors on a linear array, mesh or hypercube.

4.1. Rowwise block striping

We consider a parallel implementation of the algorithm in which the coefficient matrix is rowwise striped partitioning among the processors. In this kind of partitioning each processor is assigned a block of complete and contiguous rows of the matrix A. We assume that we have r processors and each processor has n/r contiguous rows. Then the processor i contains rows with indices $(n/r)(i-1)+1$, $(n/r)(i-1)+2, \ldots, (n/r)i$.

The algorithm proceeds in two steps to obtain $a_{ij}^{(k+1)}$, $k+1 \leqslant i \leqslant n$, $k \leqslant j \leqslant n$:

Step 1. Send the last row of each active processor to the next processor. This communication requires the transmission of a message of $n-k$ floating point numbers between two directly connected processors and takes $t_s + (n-k)\, t_w$ time (see [11]).

Step 2. Computation

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{i-1,k}^{(k)}} a_{i-1,j}^{(k)},\ k+1 \leqslant i \leqslant n,\ a_{i-1,k}^{(k)} \neq 0,\ k \leqslant j \leqslant n.$$

The computation step requires n/r divisions for calculating the multipliers belonging to a block of rows and $(n/r)(n-k)$ multiplications and subtractions. The total time spent on computation is $(n/r)(2n-2k+1)\, t_c$.

The overall parallel run time of this algorithm is

$$T\ \textit{block-row}(n,r) \in \theta\left(\frac{n^3}{r}\, t_c\right) + \theta(n\, t_s) + \theta(n^2\, t_w). \tag{7}$$

Note that in the $(n/r)-1$ last iterations when P_r is performing the computations in its part of the matrix, the remaining $r-1$ processors are idle.

4.2. Columnwise block striping

We now describe another parallel implementation of Neville elimination in which the matrix A is divided into groups of complete and contiguous columns. The way to obtain $a_{ij}^{(k+1)}$, for $k+1 \leqslant i \leqslant n$ and $k \leqslant j \leqslant n$, is implemented in three steps:

Step 1. Computation of $a_{ik}^{(k)}/a_{i-1,k}^{(k)}$ for $k+1 \leqslant i \leqslant n$ in the first active processor. This computation needs $n-k$ divisions in the kth iteration.

Step 2. The communication requires a one-to-all broadcast of $a_{ik}^{(k)}/a_{i-1,k}^{(k)}$ to the remaining active processors. This step takes $(t_s + (n-k)\, t_w)\log r$ time (see [11]).

Step 3. Update the terms $a_{ij}^{(k+1)}$ for $k+1 \leqslant i \leqslant n$ and $k \leqslant j \leqslant n$. This step involves approximately $(n/r)(n-k)$ subtractions and multiplications.

The computation time over all iterations is $\theta((n^3/r)\, t_c)$ and the communication time is $\theta(n \log r\ t_s) + \theta(n^2 \log r\ t_w)$. So the total time is

$$T\ \textit{block-column}(n,r) \in \theta\left(\frac{n^3}{r}\, t_c\right) + \theta(n \log r\ t_s) + \theta(n^2 \log r\ t_w). \tag{8}$$

Note that the communications are asymptotically higher than the communications of rowwise block striping.

As in the parallel algorithm of the preceding section, the time of computations of rowwise block striping is the same for Neville and Gauss, whereas the cost of communications for Neville is lower than for Gauss. In the case of columnwise block striping, the results obtained are similar to the costs achieved by other authors (see [9]) for the Gaussian method.

Take into account expressions (6), (7) and (8), we can observe that the different algorithms have asymptotically the same times of computation but that there are differences in the communication times.

References

[1] P. Alonso, M. Gasca, J.M. Peña, Backward error analysis of Neville elimination, Appl. Numer. Math. 23 (1997) 193–204.
[2] P. Alonso, R. Cortina, J. Ranilla, Block-striped partitioning and Neville elimination, in: P. Amestoy, P. Berger et al. (Eds.), Euro-Par'99 Parallel Processing, Lecture Notes in Computer Science, vol. 1685, Springer, Berlin, 1999.
[3] P. Alonso, J.M. Peña, Development of block and partitioned Neville elimination, C.R. Acad. Sci. Paris, t. 329, Série I (1999) 1091–1096.
[4] T. Ando, Totally positive matrices, Linear Algebra Appl. 90 (1987) 165–219.
[5] M. Gasca, J.M. Peña, Total positivity and Neville elimination, Linear Algebra Appl. 165 (1992) 25–44.
[6] M. Gasca, J.M. Peña, A matricial description of Neville elimination with applications to total positivity, Linear Algebra Appl. 202 (1994) 33–45.
[7] M. Gasca, C.A. Michelli, Total Positivity and its Applications, Kluwer Academic Publishers, Boston, 1996.
[8] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, London, 1989.
[9] I.C. Ipsen, Y. Saad, M.H. Schulz, Complexity of dense-linear-system solution on a multiprocessor ring, Linear Algebra Appl. 77 (1986) 205–239.
[10] S. Karlin, Total Positivity, Stanford University Press, Stanford, 1968.
[11] V. Kumar, A. Grama et al., Introduction to Parallel Computing. Design and Analysis of Algorithms, Benjamin/Cummings, Menlo Park, CA, 1994.