J Math Chem (2012) 50:345–358
DOI 10.1007/s10910-010-9769-0

ORIGINAL PAPER

An efficient and scalable block parallel algorithm of Neville elimination as a tool for the CMB maps problem

P. Alonso · R. Cortina · J. Ranilla · A. M. Vidal

Received: 16 September 2010 / Accepted: 25 October 2010 / Published online: 11 November 2010

Abstract This paper analyses the performance of several versions of a block parallel algorithm in order to apply Neville elimination in a distributed memory parallel computer. Neville elimination is a procedure to transform a square matrix A into an upper triangular one. This analysis must take into account the algorithm behaviour as far as execution time, efficiency and scalability are concerned. Special attention has been paid to the study of the scalability of the algorithms trying to establish the relationship existing between the size of the block and the performance obtained in this metric. It is important to emphasize the high efficiency achieved in the studied cases and that the experimental results confirm the theoretical approximation. Therefore, we have obtained a high predicting ability tool of analysis. Finally, we will present the elimination of Neville as an efficient tool in detecting point sources in cosmic microwave background maps.

Keywords Neville elimination · Block parallel algorithms · Execution time · Efficiency · Scalability · Cosmic microwave background

This is one of several papers published in Journal of Mathematical Chemistry, Special Issue: CMMSE 2010, with invited editorial contribution by Prof. Jesus Vigo-Aguiar.

P. Alonso (✉)
Department of Mathematics, University of Oviedo, 33203 Gijón, Spain
e-mail: palonso@uniovi.es

R. Cortina · J. Ranilla
Department of Computer Science, University of Oviedo, 33203 Gijón, Spain

A. M. Vidal
Department of Information System and Computation, Universidad Politécnica de Valencia, 46022 Valencia, Spain

1 Introduction

This paper analyses the performance of several versions of a block parallel algorithm in order to apply Neville elimination to a matrix in a parallel computer using a message passing paradigm and identifying the values of the parameters that are necessary to obtain an optimal performance.

In our work, we propose an organization of the block Neville elimination algorithm for computers with the message passing model and we carry out a general analysis based on upper bounds for the three metrics: execution time, efficiency and scalability. We will concentrate on the most common block distributions and on two different representative machines of the message passing model: a network of workstations and a multicomputer. Special attention has been paid to the study of the scalability of the algorithms trying to establish the relationship existing between the size of the block and the performance obtained in this metric. It should be noted that previous works (see [1,2]) have only addressed the scalability for some particular cases. However, in this paper we analyze general cases obtaining general conclusions.

Finally, we will present the elimination of Neville as an efficient tool in detecting point sources in cosmic microwave background (CMB) maps. The CMB is a diffuse radiation which is contaminated by the radiation emitted by point sources. The detection of these point sources is vital for cleaning the radiation maps and also from the astrophysical point of view.

2 Preliminaries

In multicomputers, the physical environment used to share this information is the interconnection network, while the logical paradigm is, in general, so-called *Message Passing*. In message passing the information moves from its origin to its destination establishing communications in which the speakers cooperate actively. In our experiments we have used this kind of paradigm.

The evaluation of a parallel algorithm requires a minimum study of the characteristics of the systems and a theoretical model that could predict its behaviour. Several models have been already proposed and they keep becoming more complex and precise as well as having a more complex application. Together with several authors, this paper has adopted a model based on the compromise between the precision of the predictions and their simplicity of usage (see [8]).

The Neville elimination is an alternative procedure to that of Gauss to transform a square matrix A into an upper triangular one. Strictly speaking, Neville elimination makes zeros on an A column adding a multiple of the previous row to each row. This strategy has been proved to be specially useful when using certain types of matrices as is the case of those which are totally positive or sign-regular matrices (see [4,5]).

A real matrix is called totally positive if all its minors are non-negative. It is possible to come across this sort of matrices in many different branches of science such as Mathematics, Statistics, Economy (see [5]), or Computer Aided Geometric Design (see [9,11]). According to [6,7], Neville elimination is considered to be an interesting alternative to Gauss elimination for certain types of research. Furthermore, there are

other works (see [1–3]) that show the advantages of the foresaid procedure in the field of High Performance Computing.

The Neville elimination has been described in detail in [6]. Algorithm 1 shows the Neville iterations to solve a non-singular system of linear equations $Ax = b$. $A[i, j]$ denotes the (i, j)-element of the A matrix. When all the iterations finish, A becames upper triangular and x can be got via back substitution from A and b.

```
For j := 1 to n - 1 do
  For i := n down to j + 1 do
    A[i,j]:= A[i,j] / A[i-1,j];
    For r := j + 1 to n do
      A[i,r] := A[i,r] - A[i,j] x A[i-1,r];
    End for
    b[i]=b[i]- A[i, j] x b[i-1]
    A[i,j]=0
  End for
End for
```

Algorithm 1 Sequential Neville elimination

The sequential run time of this procedure is

$$T_s \approx \frac{2n^3}{3}\, t_c,$$

where t_c is the time spent in carrying out a floating-point operation.

3 Block parallel algorithm

In order to handle a matrix in parallel, we must divide it in such a way that the partitions can be assigned to the different processors. The distribution of the matrix data affects the performance of the parallel system as explained in the following sections. Therefore, determining the best distribution for each algorithm becomes a relevant issue. This section studies a type of generic partition where the matrix $A = (a_{ij})_{1\le i,j\le n}$ is divided into $n_0 \times n_1$ submatrices $A = (A_{ij})_{1\le i\le n_0, 1\le j\le n_1}$ of $m_0 \times m_1$ dimension, where $n_0 = n/m_0$ and $n_1 = n/m_1$. We can assume that n is divisible by m_0 and m_1.

Let us consider a rectangular mesh of $p_0 \times p_1$ processors where the processor in row s and column t is denoted as P_{st}, with $1 \le s \le p_0$ and $1 \le t \le p_1$. The submatrices will be distributed in a cyclical way among the processors so that each processor will contain different non-adjacent blocks. The number of submatrices assigned to each processor is the same: $h_0 \times h_1$ where h_0 and h_1 are obtained in the following way:

$$h_0 = \frac{n}{m_0\ p_0} \quad \text{and} \quad h_1 = \frac{n}{m_1\ p_1}.$$

In a jth stage it is necessary to nullify the elements $a_{nj}, a_{n-1,j}, \ldots, a_{j+1,j}$. So that, the following steps must be accomplished:

1. Each active processor P_{st} will send the a_{ik} elements, with $i, k \geq j$, of the last row of each of its submatrices to the immediately inferior processor; that is, $P_{s+1,t}$. The processors of the last row of the processors mesh will send the foresaid elements to the corresponding processors of row 1.
2. Those processors containing a_{ij} elements, with $i > j$, of matrix A will be the ones in charge of calculating the multipliers of the j stage. Let q be the column of the mesh of processors where these processors are placed then the operative processors P_{sq}, with $1 \leq s \leq p_0$, will calculate these multipliers.
3. The forementioned processors will communicate the calculated multipliers to the active processors placed in the same row of the mesh of processors. Therefore, the broadcast will take place from each active processor P_{sq}, with $1 \leq s \leq p_0$, to the active processors P_{sr} with $1 \leq r \leq p_1$.
4. Each active processor will update all the elements of matrix A which correspond to the rows and columns with index larger than j.

3.1 Computation time, communication time and efficiency

Taking into account the mentioned steps, we study the time required both to calculate the multipliers and to update the active part of the matrix (calculation time) and the time estimated for sending the last row of the corresponding submatrices together with the necessary multipliers for the updating process (communication time).

Working with a generic block model means that both the calculation and the communication time are not accurate so we have been in the need of obtaining their bounds. Therefore, provided that the final execution time of the algorithm is obtained by the adding the calculation time and the communication time then the result is an upper bound of the foresaid time.

The purpose of this paper is that of analysing the scalability and thus the efficiency of a general model showing the most practical approaches together with comparing the theoretical analysis with the empirical ones. Working with execution time bounds might distort the realistic vision of the obtained results. Due to this fact, we present the three most common models quoted in the bibliography. Firstly, we will deal with a bidimensional partition in which the coefficient matrix of the system is divided into square submatrices. As far as the classic unidimensional partitions are concerned, we will study those in which the data matrix is divided into complete consecutive rows or columns. In the three cases, the submatrices are cyclically distributed among the processors.

Square Distribution (SD): The matrix is divided into square submatrices which are distributed among the p processors of a square processor mesh ($m_0 \times m_1 = m \times m$).

Rectangular-Row Distribution (RRD): The matrix is divided into rectangular submatrices which contain m complete rows ($m_0 \times m_1 = m \times n$). These submatrices are distributed among the p processors of a rectangular mesh of $p \times 1$ processors.

Rectangular-Column Distribution (RCD): The matrix is divided into rectangular submatrices with m complete columns ($m_0 \times m_1 = n \times m$). These submatrices are distributed among the p processors of a rectangular mesh of $1 \times p$ processors. The approximated execution times of the three cases are shown in Table 1 where t_s denote

Table 1 Computation and communication time

Distribution	Calculation	Communication
SD	$\left(\frac{2}{3}\frac{n^3}{p}+\frac{1}{3}n\,m^2+\frac{n^2\,m}{\sqrt{p}}\right)t_c$	$n\log\sqrt{p}\,t_s+\left(\frac{1}{3}\frac{n^3}{m\,p}+\frac{1}{2}\frac{n^2}{\sqrt{p}}\log\sqrt{p}\right)t_w$
RRD	$\left(\frac{2}{3}\frac{n^3}{p}+\frac{1}{2}n^2\,m+\frac{1}{2}n\,m^2\right)t_c$	$n\,t_s+\left(\frac{1}{3}\frac{n^3}{m\,p}+\frac{1}{4}n^2\right)t_w$
RCD	$\left(\frac{2}{3}\frac{n^3}{p}+\frac{1}{2}n^2\,m+\frac{1}{2}n\,m^2\right)t_c$	$n\log\,p\,t_s+\frac{1}{2}n^2\,\log\,p\,t_w$

the startup time (or latency) and t_w the transmission time per floating point number. Table 2 shows the asymptotic values obtained of the efficiency when the expressions of Table 1 are considered.

3.2 Analysing of the scalability

This section analyses the scalability of the distribution focusing on foresaid models. We will deal with this analysis through the isoefficiency function (see [8]). In order to do this, we will start from the overhead function of the communication of each partition. Next, we will determine the term of T_o which needs the highest increase in W with respect to p in order to keep a fixed efficiency. This term will be the one determining the total isoefficiency function of the parallel system.

Square distribution

Let m be the block size a function of n and p defined as $m(n,\,p) := n/p^s$ where s is a constant value. We start from the overhead function corresponding to this distribution; that is, from

$$T_{0-SD-COM} \approx \left(n\;p+n\;p\;\log\;\sqrt{p}\right)t_s+\left(\frac{1}{3}\frac{n^3}{m}+\frac{1}{2}n^2\sqrt{p}+\frac{1}{2}n^2\sqrt{p}\log\;\sqrt{p}\right)t_w.$$

We will calculate the isoefficiency function that results from each of the individual terms of the former expression, studying the increasing rate W with respect to p.

Table 2 Efficiency

Distribution	Efficiency
SD and RRD	$\frac{1}{1+\frac{t_w}{2m\,t_c}}$
RCD	1

The total isoefficiency is the maximum of the different isoefficiency functions; that is,

$$\max\{p^{3s},\ p^{\frac{3}{2}}\ \log^3\ \sqrt{p}\}.$$

The maximum of these two functions is established by the value of s. We conclude that

$$\text{for } s \leq 0.75\ \rightarrow\ \max\{p^{3s},\ p^{\frac{3}{2}}\ \log^3\ \sqrt{p}\} = p^{\frac{3}{2}}\ \log^3\ \sqrt{p}$$

and

$$\text{for } s > 0.75\ \rightarrow\ \max\{p^{3s},\ p^{\frac{3}{2}}\ \log^3\ \sqrt{p}\} = p^{3s}.$$

Having studied the isoefficiency function we can conclude that the block-size (m) influences the scalability since we have considered m to be an n and p function and the value of s determines the total isoefficiency.

Apart from the theoretical approach carried out to obtain the isoefficiency function, the study of scalability has been widened by analysing the behaviour of different scenarios when n increases as a p function. The foresaid analysis has been made by increasing n as $p^{\frac{1}{2}}\ \log\ p$, as p or as p^2, considering the constant block size ($m = 1,\ 32$ and 128) or variant ($m(n, p) := n/p^s$, with $s = 0.125,\ 0.5,\ 0.8,\ 1,\ 2$). The estimated execution time and efficiency values have been calculated in all cases. Furthermore, we have considered that $t_c = t_s = t_w$ which has allowed us to make an approach independently of the different experimental environments. Bearing in mind the results obtained for the foresaid model, we can conclude that:

- The size of the block does not influence the scalability but it does influence the efficiency for constant block sizes. The efficiency remains constant if n increases as $p^{\frac{1}{2}}\ \log\ p$, as p or as p^2. In the simulations made in these three cases, the efficiency that remains constant is 0.67 for $m = 1,\ 0.98$ in the case of $m = 32$ and 1 for $m = 128$.
- For variable block sizes, block size does influence the scalability since despite s is a constant it appears as an exponent of p. The result obtained in the isoefficiency function is confirmed since up to $s = 0.8$ it is possible to keep a constant efficiency increasing n as $p^{\frac{1}{2}}\ \log\ p$. For $s = 1$ or $s = 2$ the increasing rate is not enough being the drop more outstanding in the case of $s = 2$. If we make a simulation of the behaviour of the algorithm increasing n with p, then this increasing is sufficient when the block-size is n/p while insufficient for n/p^2; that is, $s = 2$. If we increase n with p^2 then the efficiency is constant if $s = 2$. On the other hand, when efficiency remains constant it always ranges from 0.99 to 1.

Rectangular-row distribution

Let m be the block size a function of n and p defined in the following way: $m(n, p) := n/(sp)$. Let the following expression show the overhead function related to communications

$$T_{0-RRD-COM} \approx n\ p\ t_s + \left(\frac{1}{3}\frac{n^3}{m} + \frac{1}{4}n^2\ p\right) t_w.$$

Having calculated the isoefficiency function of each of the terms of the last expression, the total isoefficiency function will be the maximum of the isoefficiency functions obtained, that is, p^3.

Following the same model as in the former case, we have been able to confirm that neither constant nor variable block size alter scalability:

- In dealing with constant block size, the block size does not alter the scalability but it does alter the obtained efficiency. The efficiency remains constant whether n or p increases as $10p$ or as $100p$. In the simulation of these three cases, we can see that the efficiency that remains constant is 0.67 for $m = 1$, 0.98 if $m = 32$ and 1 if $m = 128$.
- For variable block size, the block size does not influence the scalability since s is a multiplicative constant.

Rectangular-column distribution

Let m have the same characteristics as in the former distribution and starting from the following expression of the overloading function of communications:

$$T_{0-RCD-COM} \approx n\ p\ \log\ p\ t_s + \frac{1}{2}n^2\ p\ \log\ p\ t_w,$$

then, the total isoefficiency function is $p^3\ \log^3\ p$. Therefore, the block size (m) does not influence the scalability. The simulations we have made confirm that

- For constant block-size, the size of the block does not influence the scalability but it does influence the resulted efficiency. Note that the constant efficiency is 1 for $m = 1$, it ranges from 0.99 to 1 for $m = 32$ and from 0.95 to 0.99 for $m = 128$.
- For variable block size, the size of the block does not alter the scalability. However, changes in the block size mean a rather sharp variation of the efficiency reaching values which range from 0.52 to 0.93.

4 Experimental results: comparing the theoretical model with the empirical one

Once finished the theoretical studies, we will compare the theoretical model with the empirical values as far as the efficiency and the scalability are concerned.

4.1 Experimental conditions

The experimental results showed in this section have been obtained from two variants of the distributed memory model: an IBM SP2, subcategory MPP, and a network of

workstations (a commodity cluster). The programming paradigm used for the communications is message passing through the library MPICH.[1]

On the one hand, the IBM SP2 counts with 28 Thin2 processors (POWER2 architecture) and with a network topology similar to an Omega network. On the other, the network of workstation is formed by 20 elements (nodes) each of which has an Intel mono-core processor and a high latency Network Card.

To make the comparison between the theoretical model and the empirical one it is necessary to have estimations of the constants of the performance model used (see Sect. 2). In [1] we have obtained these estimations for our experimental environments using standard benchmarks.

In Sect. 3.1 three data distributions have been defined. For these three distributions, both the calculation and the communication time depend on several variables. One of the most relevant variables is the size of the block: $m \times m$ for the SD case, $m \times n$ for RRD and $n \times m$ for RCD. Our analysis is focused on the cases in which the maximum efficiency is obtained. For this purpose, once n and p are fixed, we will get a m value which minimizes the execution time of Neville elimination (m_{optimum}).

4.2 Efficiency

Bearing in mind the described size of the block we have compared the theoretical efficiency with the empirical one for the three forementioned data distribution. The following figures show the results obtained depending on the different n and p values, the order of the matrix, and the number of processors respectively.

First, we consider the SD distribution. As it is shown in Fig. 1 (left) both the theoretical and the empirical model have the same tendency from the beginning and their proximity is clearly higher as the order of the matrix increases.

In Fig. 1 (right), the results obtained through the experiments are close to the theoretical estimations. However, the distance between them is longer than in Fig. 1 (left) specially for 16 processors.

In the case of RRD distribution, in Fig. 2 it is pointed out that the IBM SP2 empirical values do not exceed the theoretical expectations in contrast to the results in the network of PCs.

When comparing the two figures, the efficiencies yielded in the IBM SP2 differ from those yielded in the network of PCs in up to 0.12. This is clearly due to the fact that the constant t_w has a lower value in the IBM SP2 (see [1]). Up to a certain extent, this difference is counteracted by the difference between the two calculation constants in both environments. However, this counteraction is not enough. If the calculation constants were the same, the difference would be even higher.

It is important to point out that as far as RCD is concerned, the optimal block size is 1 in all cases. In dealing with the agreement between the theoretical expectations and the reality in machines (see Fig. 3), the coincidence rate is very high ranging from

[1] www.mcs.anl.gov/research/projects/mpich2.

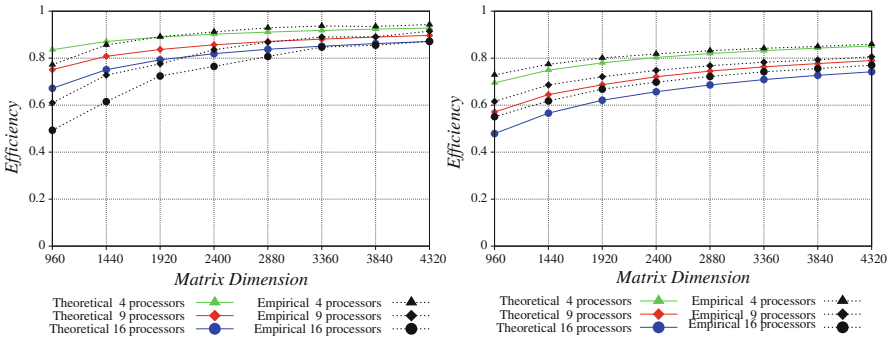

Fig. 1 SD: Theoretical and empirical efficiencies in the IBM SP2 (*letf*) and network of PCs (*right*)

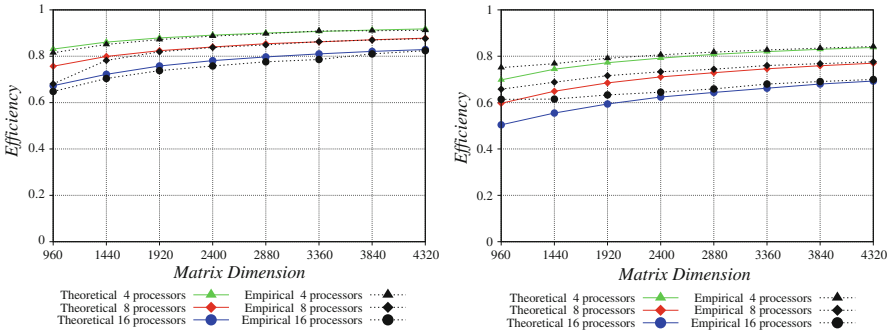

Fig. 2 RRD: Theoretical and empirical efficiencies in the IBM SP2 (*left*) and network of PCs (*right*)

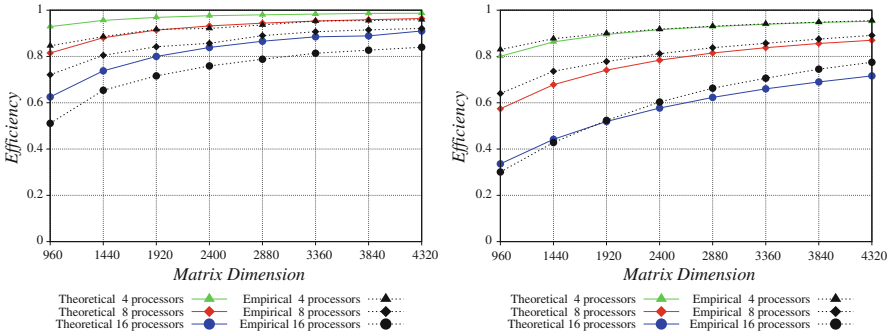

Fig. 3 RCD: Theoretical and empirical efficiencies in the IBM SP2 (*left*) and network of PCs (*right*)

97 to a 92% for a 4320 matrix and from 99.6 to 92% in the IBM SP2 and the network of PCs respectively.

Finally, we would like to point out that the efficiency achieved is very high in all cases, specially in RCD and SD where it is close to 1 in the IBM SP2.

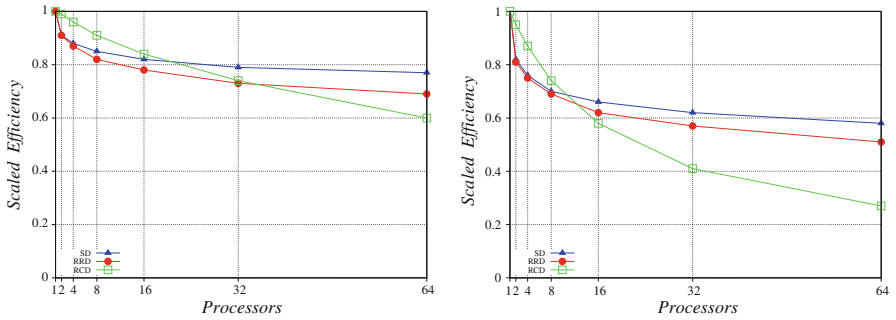

Fig. 4 IBM SP2 (*left*) and network of PCs (*right*) scaled efficiency

4.3 Scaled efficiency

A thorough study of the algorithms scalability has been presented in Sect. 3.2 taking into account the isoefficiency function (see [8]). In this section we will study the scaled efficiency which has been considered to be the most useful tool from a practical point of view (see [13]).

As far as the ability of our first distribution (SD) to keep a constant efficiency is concerned, Fig. 4 shows that when increasing n and W equally the efficiency is slightly weakened in the IBM SP2 while this weakening is stronger in the network of PCs. For instance, in the IBM SP2 the efficiency ranges from 0.91 for $(n, p) = (1210, 2)$ to 0.77 for $n = 3841$ and $p = 64$. Thus, there has been a 0.14 loss of efficiency. The efficiency loss is 0.24 in the network of PCs as it drops from 0.81 to 0.57.

Figure 4 shows the evolution of the scaled efficiency as the number of processors and W increase equally in the second distribution (RRD). It is obvious that efficiency does not remain constant being its degradation lower in the IBM SP2 than in the network of PCs. Figure 4 shows the scaled efficiency that corresponds to the RCD. We can also observe that the efficiency is reduced to 0.60 and to 0.27 in the IBM SP2 and the network of PCs respectively. In both environments the drop of the efficiency is slightly sharper than in the other distributions.

5 A case of study: detection of point sources in cosmic microwave background maps

The cosmic microwave background (CMB) is a radiation which comes from the beginning of the Universe, carrying relevant information about its origin, evolution and structure. Since its discovery in 1964 by Penzias and Wilson (see [12]), who received the Nobel Prize in 1978, the CMB has been measured by instruments aboard balloons and satellites such as the NASA COBE satellite (1992) [14], whose principal investigators, Mather and Smoot, received the Nobel Prize in 2006 and the NASA WMAP satellite (2003), which was able to determine the cosmological parameters with unprecedented accuracy (see [15]). In 2009 the ESA Planck satellite was launched and nowadays it is gathering CMB data in order to improve the knowledge about our Universe (see [16]).

The CMB is a diffuse radiation which is contaminated by the radiation emitted by point sources. The detection of these point sources is vital for cleaning the radiation maps and also from the astrophysical point of view [10]. From now on, we will present a typical method of point source detection in CMB maps.

In a region of the celestial sphere, we suppose to have a certain number n of radio sources that can be considered as point-like objects if compared to the angular resolution of our instruments. This means that their actual size is smaller than our smallest resolution cell. The emission of these sources is superimposed to the radiation $f(x, y)$. In our particular case this radiation is the CMB. A model for the emission as a function of the position (x, y) is:

$$\tilde{d}(x, y) = f(x, y) + \sum_{\alpha=1}^{n} a_\alpha \, \delta(x - x_\alpha, y - y_\alpha)$$

where $\delta(x, y)$ is the 2D Dirac delta function, the pairs are the locations of the point sources in our region of the celestial sphere, and a_α are their intensities. We observe this radiation through an instrument, with beam pattern $b(x, y)$, and a sensor that adds a random noise $n(x, y)$ to the signal measured. Again, as a function of the position, the output of our instrument is:

$$d(x, y) = \sum_{\alpha=1}^{n} a_\alpha \, b(x - x_\alpha, y - y_\alpha) + (f * b)(x, y) + n(x, y) \tag{1}$$

where the point sources and the diffuse radiation have been convolved with the beam. In our application, we are interested in extracting the locations and the intensities of the point sources. We thus assume that the intensities of the point sources are sufficiently above the level of the rest of the signal, and consider the latter as just a disturbance superimposed to the useful signal. If $c(x, y)$ is the signal which does not come from the point sources, model (1) becomes

$$d(x, y) = \sum_{\alpha=1}^{n} a_\alpha \, b(x - x_\alpha, y - y_\alpha) + c(x, y). \tag{2}$$

If our data set is a discrete map of N pixels, the above equation can easily be rewritten in vector form, by letting d be the lexicographically ordered version of the discrete map $d(x, y)$, a be the n-vector containing the positive source intensities a_α, c the lexicographically ordered version of the discrete map, $c(x, y)$, and ϕ be an $N \times n$ matrix whose columns are the lexicographically ordered versions of n replicas of the map $b(x, y)$, each shifted on one of the source locations. Equation (2) thus becomes

$$d = \phi \, a + c. \tag{3}$$

Looking at Eqs. (2, 3), we see that, if the goal is to find locations and intensities of the point sources, our unknowns are the number n, the list of locations (x_α, y_α), with $\alpha = 1, \ldots, n$ and the vector a. It is apparent that, once n and (x_α, y_α) are known, matrix ϕ is perfectly determined. Let us then denote the list of source locations by the $n \times 2$ matrix R, containing all their coordinates. For the CMB we can assume that c is

a Gaussian random field with zero mean and known covariance ξ. Thus the likelihood function is

$$p(d|n, R, a) = \exp\left(-(d - \phi\, a)'\xi^{-1}(d - \phi\, a)/2\right). \tag{4}$$

If we define $M = \phi'\xi^{-1}\phi$, $e = \phi'\xi^{-1}\, d$, the maximization of (4) leads us to the linear system

$$Ma = e \tag{5}$$

where M is a $n \times n$ matrix. The solution of this system will yield the maximum likelihood estimator of the source intensities.

One problem with this approach is that, in principle, we know neither the number n of point sources nor their positions. One standard way of dealing with this difficulty is considering (5) the local maxima of e and selecting as source positions these local maxima above a certain threshold. A 5σ threshold, with σ the standard deviation of e, is typically used, since such high fluctuations rarely have their origin in the CMB or the noise.

Finally, we have to find suitable methods to solve the system shown in (5), taking into account that the number of sources can range from tens to thousands depending on the size of the region studied and also on the frequency analyzed. One of those methods is the Neville elimination one presented in this paper.

Indeed, the behaviour of Neville elimination observed through the experimental results allow us to state that the Neville elimination is an efficient tool in the resolution of (5). First, we can use it to obtain M and e without calculating the inverse of ξ. The calculate of the inverse of a matrix usually involves a loss of accuracy in the resolution of the problem.

Thus, the matrix M can be obtained as $M = \phi' Z$, where Z is the solution of a system with multiple independent terms: $\xi^{-1}\phi = Z \rightarrow \xi Z = \phi$. On the other hand, the vector e is obtained as $e = \phi' z$, where z is the solution of the system $\xi z = d$. The matrix Z and the vector e can be obtained using Neville elimination.

Once calculated M and e, it is necessary to identify which components of e are above the threshold. These components set the equations to take into account in the resolution of the system, which will take place through the Neville elimination.

Finally, it should be noted that considering the high number of data that is necessary to use in the resolution of the problem, the use of multiprocessor strategies is particularly suitable.

6 Conclusions

In multicomputers, it is possible to deal with Neville elimination using algorithms with a different block distribution of the matrix. In this paper we have discussed both the theoretical and experimental performance of the three more common approximations distributions: SD, RRD and RCD.

Since it is possible to subdivide the matrix in several different ways, the performance obtained is also different. Consequently, the potential algorithms will be more or less adequate depending on the available hardware and its usage.

The experimental results confirm the theoretical approximation obtaining a tool of analysis of high predicting ability. For any parallel system liable to be modelled by the message passing paradigm we can predict the experimental results obtained in the foresaid system quite precisely.

The capability of the algorithms has been analysed by using three different metrics: execution time, efficiency and scalability.

As far as execution time is concerned, the SD and RRD often obtain good results. Nevertheless, in dealing with this metric, the best distribution is the column-block oriented one since its communication time does not depend on block size. If the focus of the analysis were the efficiency, conclusions would be similar to the forementioned in relation to execution time.

It is also important to point out that the best distribution is the one based on the SD, as far as scalability is concerned. In this case, the advantage of the RCD disappears since the size of the block does not influence the scalability.

In the present study, we have also discussed the relationship among block size and scalability. Therefore, we can state that the fact of making the block size depend on the dimension of the matrix and on the number of processors influences the scalability of the SD. However, it does not influence the scalability of the RRD and the RCD.

Furthermore, scalability is also considerably influenced by the characteristics of the parameters of communication. Note that scaled efficiency is always better in the IBM SP2 (the multicomputer with the fastest subsystem of communications) for all distributions.

The application of Neville elimination to the detecting point sources in CMB maps shows the usefulness of the proposed algorithm when it is applied to real problems. Moreover, considering the good results obtained on the scalability of our algorithms, in the event that the region studied and frequency analyzed generate M and e of sizes very high, an increase in resources to use can ensure a high performance.

Acknowledgments This work has been partially supported by the Spanish Research Grants TIN2007-61273, TIN2008-06570-C04-02 and TIN2010-14971, and by Valencia Regional Government Grant PROMETEO/2009/013. Also, we would like to give special thanks to Professor Francisco Argüeso of University of Oviedo for his help.

References

1. P. Alonso, R. Cortina, I. Díaz, J. Ranilla, Analyzing scalability of Neville elimination. J. Math. Chem. **40**(1), 49 (2006)
2. P. Alonso, R. Cortina, I. Díaz, J. Ranilla, Scalability of Neville elimination using checkerboard partitioning. Int. J. Comput. Math. **85**(3–4), 309 (2008)
3. P. Alonso, R. Cortina, I. Díaz, J. Ranilla, Blocking Neville elimination algorithm for exploiting cache memories. Appl. Math. Comput. **209**, 2 (2009)
4. T. Ando, Totally positive matrices. Linear Algebra Appl. **90**, 165 (1987)
5. M. Gasca, C.A. Michelli, *Total Positivity and its Applications* (Kluwer, Dordrecht, 1996)
6. M. Gasca, J.M. Peña, Total positivity and Neville elimination. Linear Algebra Appl. **165**, 25 (1992)
7. L. Gemignani, Neville elimination for rank-structured matrices. Linear Algebra Appl. **428**(4), 978 (2008)
8. A. Grama, A. Gupta, G. Karypis, V. Kumar, *Introduction to Parallel Computing* (Pearson Education Limited, London, 2003)

9. H. Lin, H. Bao, G. Wang, Totally positive bases and progressive iteration approximation. Comput. Math. Appl. **50**, 575 (2005)
10. M. Lopez-Caniego et al., Comparison of filters for the detection of point sources in Planck simulations. Mon. Not. Roy. Astron. Soc. **370**, 2047 (2006)
11. J.M. Peña, *Shape Preserving Representations in Computer Aided–Geometric Design* (Nova Science Publishers, New York, 1999)
12. A.A. Penzias, R.W. Wilson, A measurement of excess antenna temperature at 4080 Mc/s. Astrophys. J. **142**, 419 (1965)
13. M. Prieto, R.S. Montero, I.M. Llorente, F. Tirado, A parallel multigrid solver for viscous flows on anisotropic structured grids. Parallel Comput. **29**, 907 (2003)
14. G. Smoot et al., Structure in the COBE differential microwave radiometer first-year maps. Astrophys. J. **396**, L1 (1992)
15. D.N. Spergel et al., First-year Wilkinson microwave anisotropy probe (WMAP) observations: determination of cosmological parameters. Astrophys. J. Suppl. **148**, 175 (2003)
16. J.A. Tauber, The Planck mission, in *New Cosmological Data and the Values of the Fundamental Parameters. Proceedings of IAU Symposium* vol. 201 (2005), p. 86, eds. by A. Lasenby, A. Wilkinson
17. L. Toffolatti et al., Extragalactic source counts and contributions to the anisotropies of the cosmic microwave background: predictions for the Planck Surveyor mission. Mon. Not. Roy. Astron. Soc. **297**, 117 (1998)