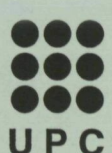
Design and Evaluation of Tridiagonal Solvers for Vector and Parallel Computers

Author: Josep Lluis Larriba Pey
Advisor: Juan José Navarro Guerrero

Barcelona, January 1995



Universitat Politècnica de Catalunya
Departament d'Arquitectura de Computadors

Chapter 6: Applications

In this chapter we analyze two applications that give rise to tridiagonal systems of equations. These are the computation of Natural Cubic Splines and B-Splines and, the use of domain decomposition techniques for the solution of neuron electrical models. In both cases we use the results of the previous sections to analyze the problems and propose new modifications of some existing techniques in order to make the methods work faster.

6.1 Introduction

In the introduction of this thesis we overviewed some applications that give rise to the tridiagonal systems of equations. The fields from which those applications come from are varied. To recall, let us mention, for example, the use of Natural Cubic Splines and B-Splines for the problem of curve fitting, the computation of photon statistics in lasers, and the solution of neuron models by domain decomposition techniques.

In this chapter we use the work developped in this thesis to analyze two practical applications. On one hand, we study the problem of Natural Cubic Spline and B-Spline curve fitting. We analyze the type of matrices that arise in this problem and propose a variation of a factorization that saves a considerable amount of work to the solution of the problem. Also, we evaluate the use of Divide and Conquer, *R*-Cyclic Reduction, the Overlapped Partitions Method and the Recursive Doubling algorithm when used to solve the systems that arise from the factorization.

On the other hand, we study the solution of neuron models by domain decomposition techniques. We analyze a method for the solution of such systems proposed in [Masc91] and use the background of this work to save work to the method. In particular we propose a formula to determine the amount of work we can save. This last work can be found in [LMJN95].

6.2 Natural Cubic Splines and B-Spline curve fitting

The solution of almost Toeplitz tridiagonal systems arises during the computation of Natural Cubic Splines and B-Splines. For the case of normalized splines, these systems are of the type:

$$\begin{bmatrix} c & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & & & \\ & & 1 & 4 & 1 \\ & & & 1 & c \end{bmatrix} \cdot \begin{bmatrix} t_{[1]} \\ \cdots \\ t_{[N-1]} \\ t_{[N]} \end{bmatrix} = \phi \cdot \begin{bmatrix} b_{[1]} \\ \cdots \\ b_{[N-1]} \\ b_{[N]} \end{bmatrix} \leftrightarrow At = \phi b \tag{6.1}$$

where the value of c depends on the imposed boundary conditions and the value of ϕ depends on the type of spline being used. In particular, for the case of Natural Cubic Splines, these values are 2 and 3 respectively [BaBB86] and for the case of B-Splines when the boundary points are doubled, these values are 5 and 6 respectively [ChSh92].

For the case of Natural Cubic Splines, the right hand side elements of the system are the componentwise difference between the value of the previous and the following points of each of the points to be interpolated $b_{[i]} = (x_{[i+1]} - x_{[i-1]}, y_{[i+1]} - y_{[i-1]}, z_{[i+1]} - z_{[i-1]})$ and the results $t_{[i]}$ are the derivatives of the polinomials at the points to be interpolated. For the case of B-Splines, the right hand side elements are the given set of points $b_{[i]} = (x_{[i]}, y_{[i]}, z_{[i]})$ and the results $t_{[i]}$ are control points of the fit.

So, every time a fit has to be computed, 2 systems have to be solved for the case of 2-dimensional problems and 3 systems have to be solved for 3-dimensional case. For more details on curve fitting and the specific problems of Natural Cubic Splines and B-splines, one can read [BaBB86] and [RoAd90].

In the computation of (7.1), matrix A is the same all the time and vector b changes for different fits. The best approach to this problem is to find an LU-type decomposition ($A = LU$), store it, and compute the solution of the bidiagonal systems $Lz = \phi b$ and $Ut = z$ every time that the solution of a new set of data is needed. This saves a considerable amount of work and, as a consequence, of computational time. Nevertheless, the classic LU decomposition applied to a Toeplitz matrix, does not lead to Toeplitz L and U matrices. This is a disadvantage as the properties of Toeplitz matrices cannot be exploited in the solution of the L and U systems and also, if different fitting techniques had to be used, a lot of data should be stored. Nevertheless, there is a fast LU-type decomposition for four-diagonal Toeplitz matrices proposed by Evans [Evan80] and revisited by Taha and Liaw (TJ decomposition) in [TaLi93] that leads to a Toeplitz decomposition as we said in chapter 1.

The TJ decomposition and procedure to solve the systems it factorizes is explained in the appendix of this work. Here, we propose a variation of the TJ decomposition for the almost Toeplitz matrices that arise in curve fitting that requires the solution of four Toeplitz bidiagonal systems. Also, we propose the use of an early termination strategy for two of those systems that helps to save a considerable amount of work.

For the solution of the bidiagonal systems arising from the method based on the TJ decomposition we analyze the pure Gaussian Elimination, 5-Cyclic Reduction, a version of the Divide and Conquer algorithm and the Overlapped Partitions Method. The use of these bidiagonal solvers lead to different versions of the method based on the TJ decomposition.

The different versions of the method proposed here are compared to the method proposed by Chung and Shen in [ChSh92]. This method is based on an inexact LDU decomposition. For this reason, this method is not accurate for the elements of the boundary of the solution as we show below. Also, as Recoursive Doubling is used to solve the upper and lower bidiagonal systems, the method by Chung and Shen is not competitive with the variants of the method proposed here on vector computers.

6.2.1 A method based on the TJ decomposition

For the matrix of the problem described above, it is possible to find a decomposition of the type:

$$(6.2)\qquad \begin{bmatrix} c & a & & & \\ a & d & a & & \\ & \ddots & \ddots & \ddots & \\ & & a & d & a \\ & & & a & c \end{bmatrix}_{N\times N} = \alpha TJ = \alpha \begin{bmatrix} \rho & 1 & & & \\ & \beta & 1 & & \\ & & \ddots & \ddots & \\ & & & \beta & 1 \\ & & & & \beta \;\; \sigma \end{bmatrix}_{N\times (N+1)} \begin{bmatrix} \gamma & & & & \\ 1 & \gamma & & & \\ & \ddots & \ddots & & \\ & & 1 & \gamma & \\ & & & 1 & \gamma \\ & & & & 1 \end{bmatrix}_{(N+1)\times N}$$

where the values of α, β, γ, ρ and σ are:

$$\alpha = \frac{d \pm \sqrt{d^2 - 4a^2}}{2} \qquad \beta = \gamma = \frac{a}{\alpha} \qquad \rho = (\frac{c}{\alpha} - 1)\frac{\alpha}{a} \qquad \sigma = c/\alpha - a^2/\alpha^2$$

For example, for the specific case of B-Splines where $a = 1$, $d = 4$, $c = 5$, we can choose

$$\alpha = 2 + \sqrt{3}, \qquad \beta = \gamma = (2 + \sqrt{3})^{-1}, \qquad \rho = \sigma = 3 - \sqrt{3}$$

α, β and γ are the same for the case of Natural Cubic Splines.

The number of floating point operations and the amount of storage capacity needed for this decomposition is $O(1)$. The solution of the system can be found by solving $At = \alpha TJt = \phi b$ in the steps explained in the appendix. Nevertheless, as the variation for the case we are studying is significatively different, we explain it below:

1. Solve $Ty = \phi b/\alpha$. This is performed by scaling b by ϕ/α ($b' = \phi b/\alpha$) and finding the solution of $Ty = b'$ assuming that y is an order n+1 vector, $y = \begin{bmatrix} y_{[0]} & y_{[1]} & \cdots & y_{[N-1]} & y_{[N]} \end{bmatrix}^T$. Given that the system is underdetermined, variable y_0 is considered a parameter. Thus, the system can be transformed into $T'y' = b' - y_{[0]}f$ where

$$T' = \begin{bmatrix} 1 & & & & \\ \beta & 1 & & & \\ & \cdots & & & \\ & & \beta & 1 & \\ & & & \beta & \sigma \end{bmatrix}_{N \times N}, \qquad y' = \begin{bmatrix} y_{[1]} \\ \cdots \\ y_{[N-1]} \\ y_{[N]} \end{bmatrix}, \qquad f = \begin{bmatrix} \rho \\ 0 \\ 0 \end{bmatrix}$$

The solution of this new system can be found by solving two bidiagonal systems $T'g = b'$ and $T'h = f$. Note that, in order to find the solution of the last variables of g and h ($g_{[n}$ and $h_{[n]}$), it is necessary to divide the respective right hand side elements in position N by σ at the end of this step.

Now, y' is a linear combination of the two solutions $y' = g - y_{[0]}h$.

2. Solve $Jt = y$. With the value of y' it is possible to see that $Jt = y = \begin{bmatrix} y_{[0]} \\ y' \end{bmatrix} = \begin{bmatrix} 0 \\ g \end{bmatrix} - y_{[0]} \begin{bmatrix} -1 \\ h \end{bmatrix}$.

Assuming that $t = t1 - y_{[0]}t2$, solve the following systems $J't1 = g$ and $J't2 = h$ where

$$J' = \begin{bmatrix} 1 & \gamma & & \\ & \ddots & & \\ & & 1 & \gamma \\ & & & 1 \end{bmatrix}_{n \times n}$$

Now, it is necessary to find the value of y_0. We know that $y_{[0]} = \gamma t_1$. We also know that $t_{[1]} = t1_{[1]} - y_{[0]} t2_{[1]}$. With these two equations we have that $y_{[0]} = \gamma \frac{t1_{[1]}}{1 + \gamma t2_{[1]}}$.

Finally, the solution of the system can be found by solving $t = t1 - y_{[0]} t2$.

6.2.2 An early termination of the *TJ* decomposition

From the method described above, it is easy to see that four bidiagonal systems have to be solved $T'g = b'$, $T'h = f$, $J't1 = g$ and $J't2 = h$. It is important to note here that those systems have two important characteristics. First, they are bidiagonal Toeplitz (the case of T' is almost Toeplitz but can be solved as a Toeplitz system and then divide the last equation by σ, as we said before). Second, they are strictly diagonal dominant with diagonal dominance $\delta = 2 + \sqrt{3}$.

Note that $T'h = f$ has only one element on top of its right hand side. For the case of this type of strictly diagonal dominant systems, the value of element $h_{[i]}$ of the solution decreases in absolute value to zero as i grows to N as we proved in lemma 2.1. The absolute value of $h_{[i]}$ can be quantified by $|h_{[i]}| \leq \delta^{-i+1} |\rho|$. From this equation it is easy to see that the number of elements of the solution vector that are larger in absolute value than a certain maximum error allowed ε are:

$$u = \left\lceil \frac{\log(\varepsilon / |\rho|)}{\log \delta^{-1}} \right\rceil$$

So, only the first u variables of $T'h = f$ have to be solved as the rest are smaller than ε. Also, $J't2 = h$ is an upper triangular system. Here, only equations u through 1 have to be solved because the $N - u$ last equations have a 0 in their right hand side. As a consequence, for the case of the axpy type operation $t = t1 - y_{[0]} t2$ only elements 1 through u have to be modified.

For the case of single precision ($\varepsilon = 10^{-7}$) $u = 14$ and for the case of double precision ($\varepsilon = 10^{-16}$) $u = 30$.

Systems $T'g = b'$ and $J't1 = g$ have all the right hand side elements different from 0 so they can be solved with any of the methods described in the previous chapters. As those systems are strictly diagonal

dominant, the early termination criteria proposed in chapter 2 can be applied to Divide and Conquer and *R*-Cyclic Reduction. Also, OPM can be used for the solution of those systems.

6.2.3 Description and analysis of the method by Chung and Shen

In this section we give a brief description of the method proposed by Chung and Shen in [ChSh92] and we analyze some features of the method. For more details about this method we suggest the reading of the referenced paper.

Let us suppose that we approximate the matrix of the problem A by the following matrix:

$$A' = \begin{bmatrix} 2+\sqrt{3} & 1 & & & \\ 1 & 4 & 1 & & \\ & & \cdots & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 4 \end{bmatrix}$$

It is possible to find an $L'D'U'$ decomposition of A' of the type:

$$A' = L'D'U' = \begin{bmatrix} 1 & & & \\ 2-\sqrt{3} & 1 & & \\ & \cdots & & \\ & & 2-\sqrt{3} & 1 \end{bmatrix} \begin{bmatrix} 2+\sqrt{3} & & & \\ & 2+\sqrt{3} & & \\ & & \cdots & \\ & & & 2+\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 2-\sqrt{3} & & \\ & 1 & 2-\sqrt{3} & \\ & & \cdots & \\ & & 1 & 2-\sqrt{3} \\ & & & 1 \end{bmatrix}$$

With this decomposition, it is possible to apply any of the methods described before to solve $L'y = 6B$ and $U'C = D'^{-1}y$. Here $D'^{-1}y$ is equivalent to the scaling of y given the characteristics of D'. Chung and Shen use the Recursive Doubling algorithm for the solution of $L'y = 6B$ and $U'C = D'^{-1}y$.

There is an important comment to make to this method. Given that matrix A is approximated by matrix A' the solutions to $A't = \phi b$ are incorrect in the upper boundary. A perturbation analysis that we make shows that, the first v solutions of the system are incorrect. For the case of $c = 5$, we have found that v is

$$v = \left\lceil -\log_2 \frac{12 \cdot \varepsilon}{\phi \cdot \max_j b_{[j]}} \right\rceil$$

where ε is the maximum error allowed. Thus the interpolation given by this method is not correct in the first $v = 22$ if the maximum absolute error allowed is $\varepsilon = 10^{-7}$ and $v = 52$ if the maximum absolute error allowed is $\varepsilon = 10^{-16}$ assuming $\phi = 3$ and $\max_j b_{[j]} = 1$.

Also, the vector implementation of the Recursive Doubling algorithm is not consistent according to the definition by Lambiotte and Voigt [LaVo75], as we said. For this reason the method described in thid section is slower than other consisting methods like the decomposition proposed here.

6.2.4 Comparison of the methods

In this section we compare the variants of the method arising from the *TJ* decomposition and the method by Chung and Shen on a vector processor. The methods have been programmed in Fortran on one processor of the Convex C-3480. They have been run in multiple user mode and we have taken the smallest execution time out of a few executions for each size of matrix.

In terms of the vector implementation of the *TJ* decomposition, the scaling $b' = \phi b/\alpha$ can be performed by means of 1 scalar divide and 1 vector multiply. For the solution of $T'g = b'$ and $J't1 = g$ four algorithms have been tested, namely, Gaussian Elimination, the Divide and Conquer algorithm, the 5-Cyclic Reduction algorithm and the Overlapped Partitions Method. For the case of systems $T'h = f$ and $J't2 = h$, the use of special vector solvers is not worthy given the order that we are considering ($u = 14$ and $u = 30$) as we saw in chapter 4. So, a scalar Gaussian Elimination sweep is sufficient. For $t = t1 - y_{[0]}t2$, a vector axpy operation is performed on the u upper elements of the equation. The assembly code generated by the compiler has been modified in order to optimize the execution time of the bidiagonal solvers as explained in chapter 4.

For the case of the method by Chung and Shen, we have used the exact algorithm proposed in [ChSh92]. This method was programmed on a Cray and for this reason the only change we made was to substitute the vector directives of the Cray Fortran into directives of the Convex Fortran.

For the practical executions, double precision arithmetic has been used for all the methods although two different accuracies have been tested for the method proposed here. The accuracies tested are $\varepsilon = 10^{-7}$ and $\varepsilon = 10^{-16}$. Note that for the algorithm by Chung and Shen there is no choice for the accuracy required and the error is larger than for the method proposed here.

Figure 6.1 shows plots of the execution times of the methods for the case where we allow a maximum error $\varepsilon = 10^{-7}$. Figure 6.1.a shows plots for systems ranging from 50 to 4000. Figure 6.1.b shows a zoom of Figure 6.1.a for matrices of size 50 to 1000. In Figure 6.1.b, we can see that the method based on Gaussian Elimination is faster than the other methods for matrices smaller than approximately 180. In the rest of the cases, the method based on the Overlapped Partitions Method is clearly faster than the other methods.

Figure 6.2 shows plots of the execution times of the methods for the case where we allow a maximum error $\varepsilon = 10^{-16}$. Figure 6.2.a shows plots for systems ranging from 50 to 4000. Figure 6.2.b shows a zoom of Figure 6.2.a for matrices of size 50 to 1000. In Figure 6.2.b, we can see that the method based on Gaussian Elimination is faster than the other methods for matrices smaller than approximately 220. For the case of matrices in the range between 220 and 650, 5-CR is the fastest algorithm. For matrices larger than 650, the method based on the Overlapped Partitions Method is the fastest.

As it can be noticed in Figures 6.1 and 6.2, the method based on the Overlapped Partitions Method is the slowest for very small systems but as the systems get larger it turns out to be the fastest.

Note that the method by Chung and Shen is slower than the rest of the methods in general. In particular, for the case of small systems it is faster than the methods based on OPM, *R*-CR and DC. But at the same time, for those cases, it is slower than the method based on Gaussian Elimination.

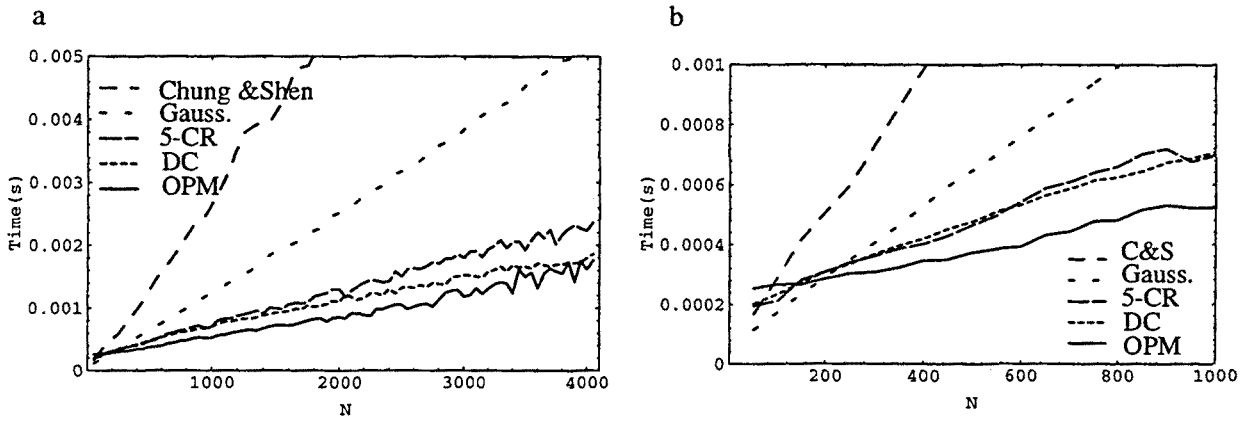


Figure 6.1. Execution time of the methods compared for $\varepsilon = 10^{-7}$. a) *N* ranging from 50 to 4000. b) *N* ranging from 50 to 1000.

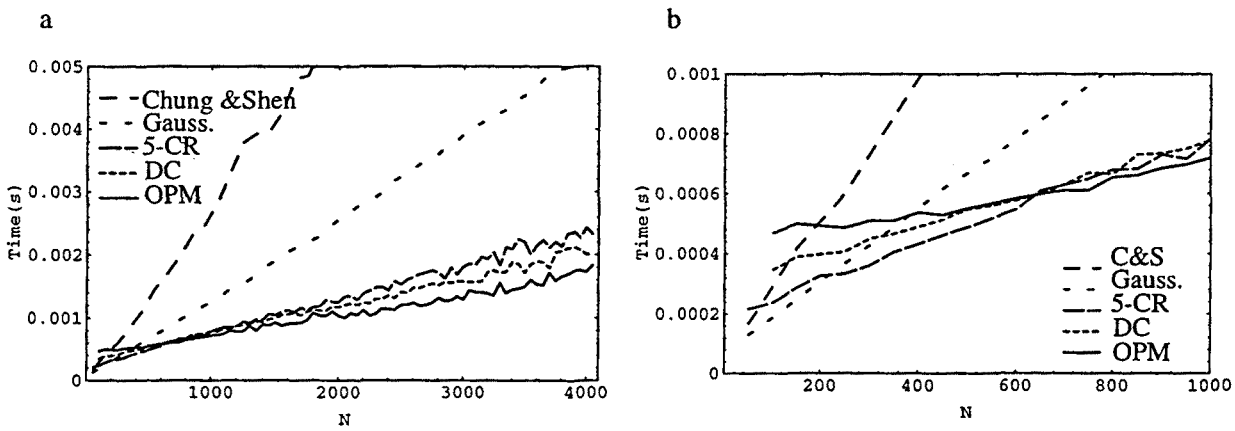


Figure 6.2. Execution time of the methods compared for $\varepsilon = 10^{-16}$. a) *N* ranging from 50 to 4000. b) *N* ranging from 50 to 1000.

6.3 Solution of neuron models by domain decomposition techniques

The anatomy of real neurons implies an extensively branched dendritic region that converges onto a single cell body. The cell body then emanates a lightly branched axon that ends in other heavily branched presynaptic regions. The electrical behaviour of such neural structures can be described by Partial Differential Equations models. The most famous of these models are the Hodgkin-Huxley parabolic equations that describe the electrical behaviour of the giant axon of a squid

$$C\frac{\partial^2 V}{\partial t} = \frac{a}{2R}\frac{\partial^2 V}{\partial x^2} - \bar{g}V \tag{6.3}$$

By scaling $x \rightarrow (2R)/a$ and $t \rightarrow t/C$ in the above equation one can obtain the cable equation, [Rall77]. If $\bar{g}$ is a constant, then this linear cable equation is a valid PDE model of the dendrites and a further change of variables in V reduces (6.3) to the heat equation. For the axons, $\bar{g} = \bar{g}(V, x, t, \ldots)$ is a nonlinear conductance and no such scaling in V is possible and (6.3) is a nonlinear PDE. Here, though, we suppose that all PDEs involved in our problem are linear.

The most popular methods for the solution of equation (6.3) are finite-difference methods [Masc89] although other methods like finite elements or spectral methods could be used. Among finite-difference methods, the use of implicit ones as opposed to explicit ones is more advisable due to stability reasons. The implicit methods require the solution of a linear or perhaps a nonlinear equation at each time step. Here, we assume that we can advance our implicit finite-difference solution one time step by solving a single linear system, as we said. For the discretization, we suppose that we use a backward-Euler time discretization although a backward Crank-Nicholson time discretization could also be used.

In general, the morphology of biological neurons requires that we consider heavily branched one-dimensional domains in which boundary conditions change in time. The linear system corresponding to an implicit finite-difference discretization of such domain is almost tridiagonal and has extra off diagonals that couple the different branches together at branch points. The solution to this system can be found by using a domain decomposition method as explained in [Masc91].

In this section we describe the use of the domain decomposition method described in [Masc91]. Then, we analyze the type of systems that arise from this domain decomposition method. With this analysis, we show how to save some work with the use of lemma 2.1 that we proved for bidiagonal solvers and we apply to tridiagonal solvers in this section. Finally, we prove the lemma used for tridiagonal solvers.

6.3.1 The domain decomposition method

The domain decomposition method proposed in [Masc91] is based on partitioning the domain of the entire neuron into subdomains. An example of the domain decomposition method used on a hypothetic neuron is shown in Figures 6.3 and 6.4. The domain with its discretization is shown in Figure 6.3 and the matrix arising from the numbering of this discretized domain is show in Figure 6.4.

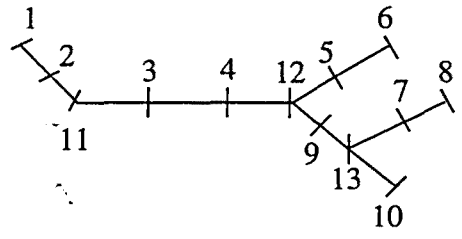


Figure 6.3. Discretized neuron example.

In Figure 6.3 we can see that the numbering of the discretization points for the example, has been chosen in such a way that the branching points are the last to be numbered. With this, the relation between the domains and the branching points (or boundaries) is found in the rightmost columns and botom rows of the discretization matrix of Figure 6.4. Each domain is independent from the rest of domains but it is linked to the branching points.

The system shown in Figure 6.4 can be solved in the following steps. The subsystems corresponding to each domain in Figure 6.4 can be diagonalized independently as shown in Figure 6.5.a. With this diagonalization, some fill in is introduced in the rightmost columns of the matrix. After this, it is necessary to eliminate the off-diagonal elements of the bottom rows of the matrix as shown in Figure 6.5.b. With this elimination, some fill in is introduced in the submatrix that represents the branching points as shown in Figure 6.5.b. This submatrix is the Schur complement matrix. The solution to the Schur complement system leads to the final solution of the entire system by scaling the rightmost columns by the solutions to each of the branching points and then substracting those scaled columns from the right hand side of the system.

$$
\begin{bmatrix}
a_{[1,1]} & a_{[1,2]} & & & & & & & & & & & & \\
a_{[2,1]} & a_{[2,2]} & & & & & & & & & a_{[2,11]} & & & \\
 & & a_{[3,3]} & a_{[3,4]} & & & & & & & a_{[3,11]} & & & \\
 & & a_{[4,3]} & a_{[4,4]} & & & & & & & & a_{[4,12]} & & \\
 & & & & a_{[5,5]} & a_{[5,6]} & & & & & & a_{[5,12]} & & \\
 & & & & a_{[6,5]} & a_{[6,6]} & & & & & & & & \\
 & & & & & & a_{[7,7]} & a_{[7,8]} & & & & & & a_{[7,13]} \\
 & & & & & & a_{[8,7]} & a_{[8,8]} & & & & & & \\
 & & & & & & & & a_{[9,9]} & & & a_{[9,12]} & a_{[9,13]} & \\
 & & & & & & & & & a_{[10,10]} & & & a_{[10,13]} & \\
 & a_{[11,2]} & a_{[11,3]} & & & & & & & & a_{[11,11]} & & & \\
 & & & a_{[12,4]} & a_{[12,5]} & & & & a_{[12,9]} & & & a_{[12,12]} & & \\
 & & & & & & a_{[13,7]} & & a_{[13,9]} & a_{[13,10]} & & & & a_{[13,13]}
\end{bmatrix}
$$

Figure 6.4. Matrix arising from the discretization of the neuron in Figure 6.3.

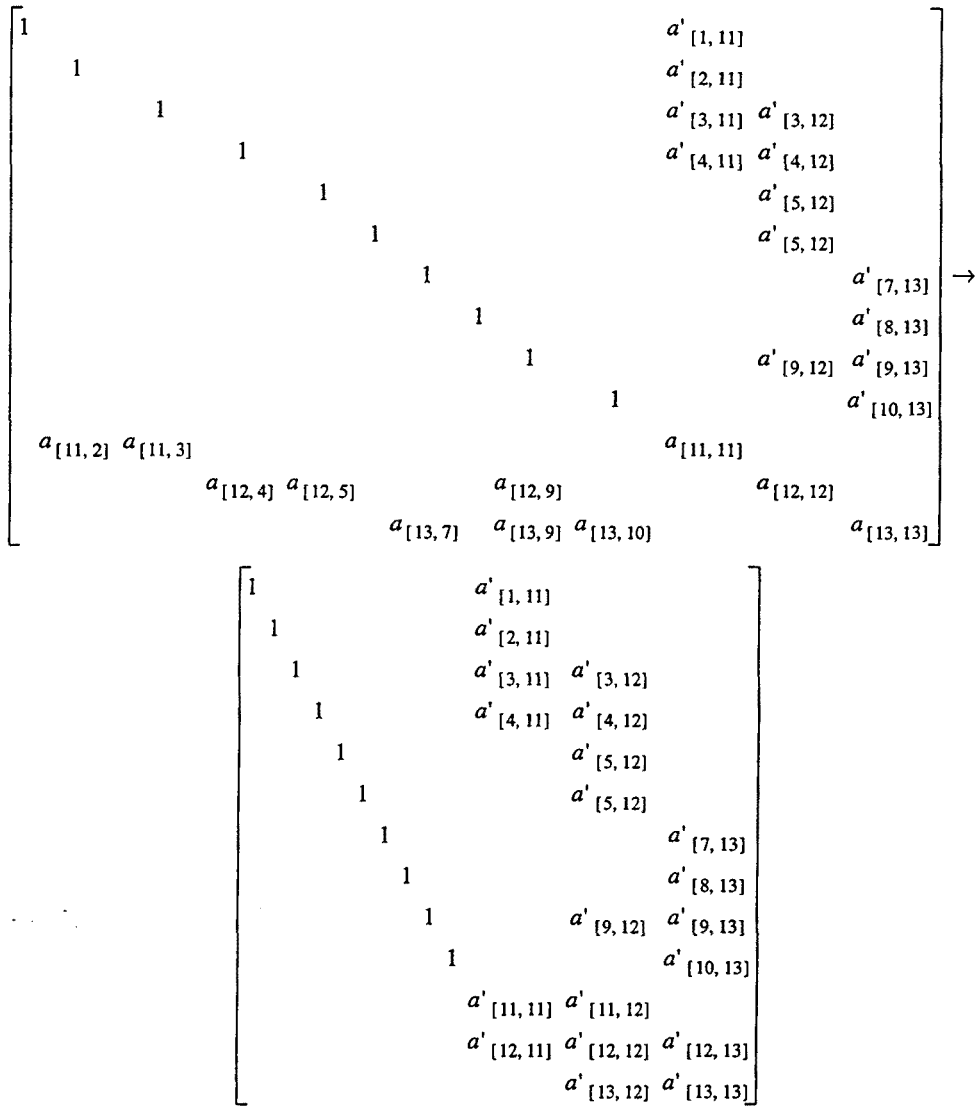


Figure 6.5. a) Independent diagonalization of the domains. b) Elimination of the bottom rows of the matrix.

Now, we are going to describe the solution to the system with the domain decomposition method in a more general way. Here, we use the notation used for the description of DC in chapter 1. The system can be solved by decomposing the domain into K unbranched domains of size n_k for $1 \le k \le K$. The K domains are connected by means of L branching points where $L < K$. The system for the general decomposition at step i is shown in Figure 6.6. There, $E_2^{(i)}$ are tridiagonal matrices that represent the K unbranched domains and, vectors $b_k^{(i)}$ and $v_k^{(i)}$ represent the relation between the domains and their branching points $p_t^{(i)}$ and $p_r^{(i)}$. Note that vectors $b_k^{(i)}$ and $v_k^{(i)}$ have one only element different from 0 in positions 1 and n_k respectively. $x_k^{(i)}$ are the solution vectors of each domain at step i and $w_l^{(i)}$ are the solution to each branching point at step i.

$$
\begin{bmatrix}
E_1^{(i)} & & & & & & & b_1^{(i)} & & & & & & \\
& E_2^{(i)} & & & & & & b_2^{(i)} & v_2^{(i)} & & & & & \\
& & E_3^{(i)} & & & & & & b_3^{(i)} & v_3^{(i)} & & & & \\
& & & E_4^{(i)} & & & & & b_4^{(i)} & & & & & \\
& & & & \cdots & & & & & & \cdots & & & \\
& & & & & E_K^{(i)} & & & & & & b_K^{(i)} & & \\
b_1^{(i)T} & & & & & & & p_1^{(i)} & & & & & & \\
& b_2^{(i)T} & & & & & & & p_2^{(i)} & & & & & \\
& v_2^{(i)T} & b_3^{(i)T} & b_4^{(i)T} & & & & & & p_3^{(i)} & & & & \\
& & v_3^{(i)T} & & & & & & & & p_4^{(i)} & & & \\
& & & & \cdots & & & & & & & \cdots & & \\
& & & & & b_K^{(i)T} & & & & & & p_L^{(i)} & &
\end{bmatrix}
\begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ x_3^{(i)} \\ \cdots \\ x_K^{(i)} \\ w_1^{(i)} \\ w_2^{(i)} \\ \cdots \\ w_L^{(i)} \end{bmatrix}
=
\begin{bmatrix} x_1^{(i-1)} \\ x_2^{(i-1)} \\ x_3^{(i-1)} \\ \cdots \\ x_K^{(i-1)} \\ w_1^{(i-1)} \\ w_2^{(i-1)} \\ \cdots \\ w_L^{(i-1)} \end{bmatrix}
$$

Figure 6.6. System arising from the discretization of a general neuron using domain decomposition.

The solution to the linear system with the domain decomposition method is found by performing the following three phases at each time step *i*:

1) This phase is divided into two steps.

1.a) Solve the K to $2K$ independent tridiagonal systems $E_k^{(i)} y_k^{(i)} = b_k^{(i)}$ and $E_k^{(i)} z_k^{(i)} = v_k^{(i)}$. Each domain has one or two of those systems if it is connected to one or two branch points. These systems can be solved independently. This is equivalent to the transformation of Figure 6.4.b for the example.

1.b) Solve K independent tridiagonal systems $E_k^{(i)} d_k^{(i)} = x_k^{(i-1)}$. These systems correspond to the K unbranched domains and can be solved independently, too. This are the transformations that have to be done to the right hand side of the system and have not been explained explicitely in the example.

2) Using part of the solutions of step 1, eliminate the L botom off diagonal rows of the system. With this, matrix $P^{(i)}$ is built from the modifications of elements $p_l^{(i)}$ and some fill in that is introduced around those branching points. The solution to the sparse system for the branch points $P^{(i)} x_b^{(i)} = x_b^{(i-1)}$ leads to $x_b^{(i)}$ which is the solution vector to the branch points at step *i*, that is $x_b^{(i)} = (w_1^{(i)}, w_2^{(i)}, \ldots, w_L^{(i)})$.

3) Perform K general axpy operations $x_k^{(i)} = d_k^{(i)} + w_r^{(i)} y_k^{(i)} + w_l^{(i)} z_k^{(i)}$ where $w_r^{(i)}$ and $w_l^{(i)}$ are the

solution to branch points r and l adjacent to branch k. These K axpy operations are independent. At the end of this phase, the solution at time step i has been found.

Matrices $E_k^{(i)}$ have to be assembled anew at each time step i as the boundary conditions vary along the time. For non-changing boundary conditions, phase 1.a could be performed as an initialization before the iteration in time starts. This would save execution time.

6.3.2 Analysis of the discretization matrices

Depending on the boundary conditions of the problem and the characteristics of the specific branching region (axon, dendrites or presynaptic regions), each tridiagonal matrix of those mentioned above varies in characteristics. Thus they may vary in size, diagonal dominance and in having Toeplitz structure or not. Now, we briefly review the features of those matrices.

A common feature of the matrices analyzed here is that they are all strictly diagonal dominant due to the parabollic nature of the problem. We can define the typical equation of the tridiagonal systems arising from the discretization of (6.3) by $f_{[i]}x_{[i-1]} + g_{[i]}x_{[i]} + h_{[i]}x_{[i+1]} = e_{[i]}$ where $f_{[i]} = h_{[i]} = -\lambda_{[i]}$ and $g_{[i]} = 1 + 2\lambda_{[i]} + \gamma_{[i]}$. With this we have that the diagonal dominance of the system is $\delta = \min_i(|1 + 2\lambda_{[i]} + \gamma_{[i]}|/|2\lambda_{[i]}|)$. In general, we will have $\delta = 1 + \min_i|\lambda_{[i]}^{-1}| > 1$.

The axons are large in size and their characteristics often vary along their structure. Their discretizations lead to large non-Toeplitz systems.

The matrices in all of the non-tapered internal dendrites can be made Toeplitz and are of moderate size. On the contrary, the terminal dendrites are small in size and the matrices they give rise to are non-Toeplitz due to their tapered structure. Something similar happens to the internal and external presynaptic regions.

In the following sections we describe and analyze some strategies to optimize the execution time of phases 1, 2 and 3 assuming that the tridiagonal matrices described here are non-Toeplitz.

6.3.3 Phases 1 and 3

As we said, it is necessary to solve up to two systems per branching domain during phase 1.a. Those systems are $E_k^{(i)}y_k^{(i)} = b_k^{(i)}$ and $E_k^{(i)}z_k^{(i)} = v_k^{(i)}$. The solutions to those systems are used during phase 3 to obtain the final solution to each domain for step i with $x_k^{(i)} = d_k^{(i)} + w_r^{(i)}y_k^{(i)} + w_l^{(i)}z_k^{(i)}$.

Matrices $E_k^{(i)}$ are strictly diagonal dominant and vectors $b_k^{(i)}$ and $v_k^{(i)}$ have one only nonzero value in position 1 and n_k respectively ($b_{k[1]}^{(i)} = \lambda_{k[1]}$ and $v_{k[n_k]}^{(i)} = \lambda_{k[n_k]}$). Under those circumstances, the solution vector of such system decreases towards position n_k for $E_k^{(i)}y_k^{(i)} = b_k^{(i)}$ and towards position 1

for $E_k^{(i)} z_k^{(i)} = v_k^{(i)}$. For tridiagonal systems, only the first m_k elements of the solution vector of $E_k^{(i)} y_k^{(i)} = b_k^{(i)}$ and the last m_k elements of the solution vector of $E_k^{(i)} z_k^{(i)} = v_k^{(i)}$ are larger than a certain predetermined value ε as shown by the following expression

$$m_k = \left\lceil \frac{\log\varepsilon\,(1-\delta^{-2})\left|\dfrac{E_{k[1,1]}^{(i)}}{b_{k[1]}^{(i)}}\right|}{\log\delta^{-1}} + 1 \right\rceil \tag{6.4}$$

for $E_k^{(i)} y_k^{(i)} = b_k^{(i)}$ and by the following expression

$$m_k = \left\lceil \frac{\log\varepsilon\,(1-\delta^{-2})\left|\dfrac{E_{k[n_k,n_k]}^{(i)}}{v_{k[n_k]}^{(i)}}\right|}{\log\delta^{-1}} + 1 \right\rceil \tag{6.5}$$

for $E_k^{(i)} z_k^{(i)} = v_k^{(i)}$. These formulae are proved in section 6.3.5.

With this, it is possible to see that during phase 1.a it is possible to save some work for all those systems with $m_k < n_k$. Also, during phase 3, the axpy operations have to be performed only on m_k sets of data if $m_k < n_k$ saving a considerable amount of work.

Table 6.1 shows a few examples of number of equations m_k to be solved for different values of ε and λ assuming $\gamma = 0$. We can see that large values of λ induce weak diagonal dominances and little savings. On the other hand, small values of λ may induce large savings. Note that the value of γ has been assumed zero because it is small in practice and has little influence on the diagonal dominance of the problem.

The physiological meaning of the optimizations discussed in this subsection is that, for certain cable domains the influence of an electric potential at one end of the domain has very little influence on the other end of the domain in a sufficiently small period of time (time step). This is intuitive for large axon cables. Nevertheless, there are other characteristics of the problem that also influence this like the membrane capacitance or the axial resistivity. Those characteristics, together with the time and space discretization, influence greatly the diagonal dominance of the systems [Hine84].

	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-16}$
$\lambda = 1, \delta = 1.5$	22	39	90
$\lambda = 5, \delta = 1.1$	107	180	397
$\lambda = 10, \delta = 1.05$	223	364	789

Table 6.1. Number of elements m_k larger than ε for different values of λ with $\gamma = 0$.

6.3.4 Phase 2

After performing phase 1, it is necessary to build matrix $P^{(i)}$ and solve system $P^{(i)}x_b^{(i)} = x_b^{(i-1)}$ as we said. $P^{(i)}$ is formed with the help of the solutions to the elements that connect the domains with the branching points and have been found in phase 1. Thus a branching point will be connected to another branching point in matrix $P^{(i)}$ if there is a domain between them.

Nevertheless, the optimizations of the previous section can be taken into account when building matrix $P^{(i)}$. This can be done when a certain domain induces sufficiently strictly diagonal dominant systems. In those cases, the two branching points connected to the domain do not have any significative influence on each other and can be considered independent.

With this, matrix $P^{(i)}$ has sets of disconnected branching points. Thus instead of one system of equations $P^{(i)}x_b^{(i)} = x_b^{(i-1)}$, there are a set of smaller independent sparse systems of equations. This allows for the solution of $P^{(i)}x_b^{(i)} = x_b^{(i-1)}$ with a certain degree of natural parallelism.

6.3.5 Proofs for formulae (6.4) and (6.5)

In this section we prove formulae (6.4) and (6.5) that determine the number of elements to be computed from systems $E_k^{(i)}y_k^{(i)} = b_k^{(i)}$ and $E_k^{(i)}z_k^{(i)} = v_k^{(i)}$ in order to have a maximum absolute error ε in the solution. With this aim, we first prove a lemma on the dacay of the solution of a tridiagonal system that has one only element in the right hand side vector of the system and then we give two corollaries that proof expressions (6.4) and (6.5).

Lemma 6.1. Let $Ax = b$ be an N x N linear system with the following characteristics:

i) A is a tridiagonal matrix with $a_{[i,i]} \neq 0, \quad \forall i$.

ii) $\min_i \left(|a_{[i,i]}| / (|a_{[i,i-1]}| + |a_{[i,i+1]}|)\right) = \delta > 1$.

iii) $b_{[i]} = 0, \quad \forall i > 1$.

Then, it is possible to verify for a general tridiagonal system of equations that:

$$|x_{[i]}| \leq \frac{|b_{[1]}|}{|a_{[1,1]}|} \frac{\delta^{-i+1}}{1-\delta^{-2}} \quad , \; i = 1, 2, \ldots, n. \tag{6.6}$$

Proof. Let us first decompose $A = D + B$, where $D = \text{diag}(A)$. Therefore, $A = D[I + D^{-1}B]$ and this means that $A^{-1} = [I+M]^{-1}D^{-1}$, where $M = D^{-1}B$.

So, $x = A^{-1}b = [I+M]^{-1}D^{-1}b = d - Md + M^2d - M^3d\ldots$, where $d = D^{-1}b$ and the series $(I+M)^{-1} = I - M + M^2 - M^3\ldots$ is convergent due to the fact that $\|M\|_\infty = \delta^{-1} < 1$.

Due to the special structure of A for the case of tridiagonal systems of equations, we have that the first term in the series $x = d - Md + M^2d - M^3d\ldots$ that affects $x_{[1]}$ comes from $M^{i-1}d$. Note that term M^id does not contribute to the value of $x_{[1]}$, but term $M^{i+1}d$ does. This happens for all i and also when $i > N$. This can be observed in the following two products:

$$Md = \begin{bmatrix} 0 & a_{[1,2]} & & \\ a_{[2,1]} & 0 & \cdots & \\ & \cdots & \cdots & a_{[N-1,N]} \\ & & a_{[N,N-1]} & 0 \end{bmatrix} \begin{bmatrix} d_{[1]} \\ 0 \\ \cdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ d_{[1]}a_{[2,1]} \\ \cdots \\ 0 \end{bmatrix}$$

$$Md = \begin{bmatrix} 0 & a_{[1,2]} & & \\ a_{[2,1]} & 0 & \cdots & \\ & \cdots & \cdots & a_{[N-1,N]} \\ & & a_{[N,N-1]} & 0 \end{bmatrix} \begin{bmatrix} 0 \\ d_{[1]}a_{[2,1]} \\ \cdots \\ 0 \end{bmatrix} = \begin{bmatrix} d_{[1]}a_{[2,1]}a_{[1,2]} \\ 0 \\ d_{[1]}a_{[2,1]}a_{[3,2]} \\ 0 \\ \cdots \end{bmatrix}$$

As a consequence:

$$\begin{aligned} |x_{[i]}| &\le \|M^{i-1}d\|_\infty + \|M^{i+1}d\|_\infty + \|M^{i+3}d\|_\infty + \ldots \le \\ &\le \|M\|_\infty^{i-1}\|d\|_\infty + \|M\|_\infty^{i+1}\|d\|_\infty + \|M\|_\infty^{i+3}\|d\|_\infty + \ldots \le \\ &\le \delta^{-i+1}(1 + \delta^{-2} + \delta^{-4} + \ldots)\|d\|_\infty = \frac{\delta^{-i+1}}{1-\delta^{-2}}\|d\|_\infty \end{aligned}$$

And using $\|d\|_\infty = |d_{[1]}| = \dfrac{|b_{[1]}|}{|a_{[1,1]}|}$ we obtain the proposition.□

Corollary 6.1. Given expression (6.6), it is sufficient to compute the first m_k elements of $E_k^{(i)} y_k^{(i)} = b_k^{(i)}$ to solve the system with an absolute error less than ε, where m_k is equivalent to

$$m_k = \left\lceil \frac{\log \varepsilon \, (1 - \delta^{-2}) \left| \frac{E_{k\,[1,\,1]}^{(i)}}{b_{k\,[1]}^{(i)}} \right|}{\log \delta^{-1}} + 1 \right\rceil$$

Proof. The proof follows from the explicit expression of (6.6) and the substitution of $b_{[1]}$ and $a_{[1,\,1]}$ by the corresponding values in $E_k^{(i)} y_k^{(i)} = b_k^{(i)}$.

Corollary 6.2. This corollary is equivalent to corollary 6.1 for $E_k^{(i)} z_k^{(i)} = v_k^{(i)}$.

Conclusions and future work

In this thesis we have dealt with the vector and parallel solution of recurrences and, in particular, with those that arise from the decomposition of tridiagonal systems of equations. The work has a theorecical and a practical component. From the theoretical point of view, we have proposed a new solver and we have unified two families of existing parallel methods for the solution of such recurrences. From the practical point of view, we have modelled and analyzed those methods for their optimum solution on vector and parallel computers and also, we have performed an analysis of two applications related to the problem.

In the following paragraphs we sum up and comment the different contributions of this work. Finally, we shortly describe the future research induced by the present thesis.

Theoretical aspects of the contributions

1) The *R*-Cyclic Reduction and the Divide and Conquer families of algorithms are two classic methods for the vector and parallel solution of recurrences that obtain parallelism at the cost of incrementing the amount of work compared to the sequential Gaussian elimination. Those methods have always been regarded as totally different approaches to the solution of such systems.

A unified algorithm for the *R*-Cyclic Reduction and the Divide and Conquer families of algorithms has been proposed (chapter 2). With the unified algorithm, it has been shown that the Divide and Conquer family of algorithms is a particular case of the *R*-Cyclic Reduction family.

One important remark is that the understanding of *R*-CR and DC given by the unification of the algorithms has influenced their optimization and analysis on vector and parallel computers. In particular, the tecniques used for the optimization of *R*-CR and DC on vector computers and the proposal of a variant of *R*-CR for parallel computers have been directly influenced by this unification.

Part of this contribution can be found in [LNRJ94].

2) For the case of strictly diagonal dominant bidiagonal systems of equations, it is possible to save some work to the unified algorithm at the cost of introducing some controllable error in the solutions. This reduction in the amount of work is called early termination of the algorithm.

A bound of the error of the early terminated unified algorithm has been proposed and proved (chapter 2). This bound depends on the diagonal dominance of the system δ and its order N. From this bound, it is possible to find criteria for the early termination of *R*-Cyclic Reduction and Divide Conquer that depend on δ, N and the maximum absolute error allowed to the solution, ε. The early termination criterion of *R*-CR is called S_{min}, and is the minimum number of steps to be performed by the algorithm to have a maximum absolute error ε in the solution. The early termination criterion of DC is called R_{min}, and is the minimum

size of the partitions that allows to avoid the solution of the reduced system and have a maximum absolute error ε in the solution.

An analysis of the early terminated algorithms shows that, while the early termination of DC pays off depending on δ, N and ε; the early termination of R-CR does not pay off unless the diagonal dominance of the system δ is extremely large and also a permisive absolute error ε is allowed.

The bound of the error and its analysis can be found in [LaJN95] and some preliminary work can be found in [LaJN93b].

3) The Overlapped Partitions Method (OPM) is proposed for the vector and parallel solution of strictly diagonal dominant recurrences (chapter 3). OPM is based on the independent solution of an arbitrary set of partitions that overlap. The final solution to the system is found by choosing some of the solutions to the equations of the different independent partitions. The parallelism of the method is achieved by replicating data.

OPM solves the recurrences with a certain amount of inaccuracy that can be quantified. This is done by a bound of the amount of error that is proposed and proved here and depends on the diagonal dominance of the system δ and its order N. From this bound, it is possible to find a formula to determine the amount of overlapping equations m necessary to have an absolute error smaller than ε in the solution .

Some preliminary work on OPM can be found in [Larr90] and [LaJN93].

Practical aspects of the contributions

1) Divide and Conquer, R-Cyclic Reduction and the Overlapped Partitions Method have been analyzed for their optimum use on vector computers. The early and non-early terminated versions of DC and R-CR have been evaluated (chapter 4).

The vector assembly versions of the algorithms have been analyzed and different algorithm transformations have been used to minimize their amount of traffic with memory.

Five different models of vector processors have been evaluated for the implementation of the algorithms. The difference between the five types of vector processors is in the quantity and quality of paths between memory and vector registers. This feature turns out to be very important for the problem studied. So, optimum schedulings of the algorithms have been found for the different vector processors.

A general execution time model has been built for each optimized algorithm. These models adapt to each of the five different architectures studied. Some parameters of those models have been optimized with classic minimization techniques. One of the architectures has the same characteristics as the Convex C-3480 and both the theoretical and practical execution times have been compared. The error of the execution time models with respect to the actual executions on the C-3480 is smaller than an 8% which shows that the models are accurate.

Finally, the execution times given by the models of the algorithms have been compared for each of the

five vector processor types. Figure 1 shows a qualitative approximation of the behaviour of the algorithms for the different vector processor types. Two different plots are shown, one for the case of small ε and the other for the case of large ε. The plots have to be seen as a tendency of the algorithms rather than as an exact plot of their behaviour. Some general conclusions can be summed up with the help of figure 1

a) Gaussian elimination is the fastest algorithm for small systems, i.e. $N < 150$ equations.

b) The optimum versions of *R*-CR and DC behave similarly for medium sized systems of equations, i.e. $150 < N < 1000$. It is important to say, though, that the optimization of *R*-CR can be performed easily than the optimization of the two versions of DC. For this reason it is advisable to use *R*-CR.

c) The optimum version of *R*-CR is the fastest algorithm for non-strictly diagonal dominant systems ($\delta = 1$) and for strictly diagonal dominant systems with weak diagonal dominances (small δ). This method is the fastest for large systems, i.e. $N > 1000$.

d) OPM and sometimes the early terminated versions of DC are the fastest algorithms for systems with moderate to high diagonal dominance δ and moderate to large order *N*. It is important to say that OPM is easier to be implemented and it behaves considerably better than the early terminated versions of DC on most of the architectures analyzed.

e) The behaviour of the early terminated methods and OPM has a strong dependence on the absolute error allowed to the solutions ε. Permisive errors give smaller values of $R_{min} = m$ than restrictive errors as it could be imagined. This means that the amount of work to be performed by the methods is larger for large values of $R_{min} = m$ and for this reason the early terminated DC and OPM compare worser to the non-early terminated versions. This can be seen comparing figure 1.a to figure 1.b.

The implementation of the methods for non-strictly diagonal dominant systems on vector computers has been described in [LNRJ94].

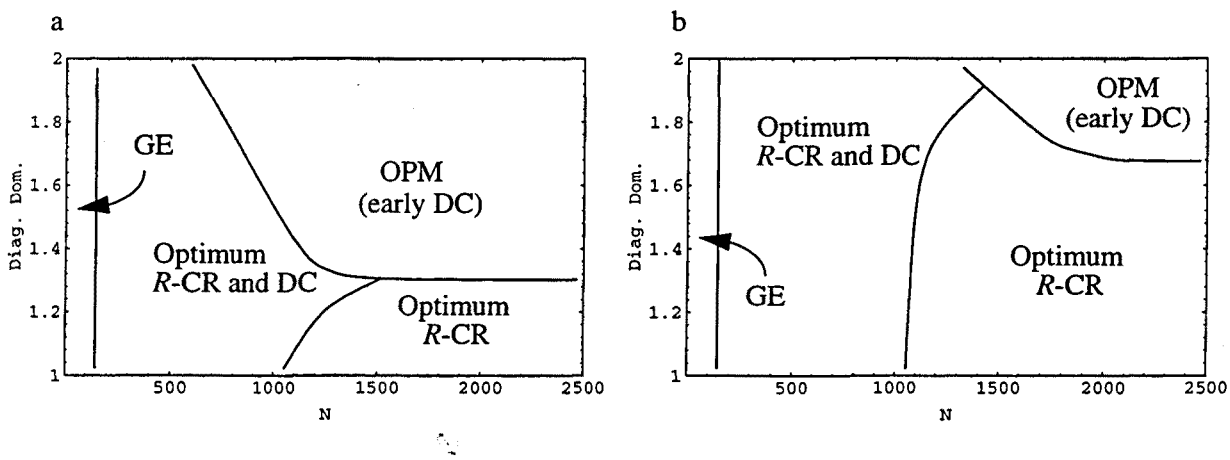


*Figure 1.*Qualitative behaviour of the methods on vector processors. a) Permissive absolute error allowed ε » , b) Restrictive absolute error allowed ε « .

2) Divide and Conquer, *R*-Cyclic Reduction and the Overlapped Partitions Method have been analyzed for their optimum use on parallel computers (chapter 5). The study has been focussed on the influence of the computation and communicaction times on the global execution time of the algorithms. Both the early and non-early terminated versions of DC and *R*-CR have been analyzed.

Parallel computers with distributed memory that communicate by means of message passing are used as target architecture. The algorithms have been analyzed for this type of parallel computers and, for the case of *R*-CR, a parallel variation has been proposed which has proved to be successful. General execution time models have been built for each implementation. The theoretical optimum number of processors has been found for each algorithm with the help of the models using classic minimization techniques.

A comparison of the methods has been done by means of the execution time models of the methods. This comparison has been done for three different models of parallel computers. The three models of parallel computer have processing nodes with the same supposed time to execute instructions. Their difference is in the time spent to start-up a communication and the communication speed.

Figure 2 shows a qualitative approximation of the behaviour of the non-early and early terminated parallel algorithms. Plot 2.a shows the case of architectures with small communication start-ups and small communication time per element and plot 2.b shows the case in which the communication start-ups are significatively higher than the computation time. With these plots it is intend to give an idea of the tendency of the algorithms for two extreme cases although intermediate cases can be extrapolated by the reader. Note that the plots obviate the maximum error allowed to the solution of the systems ε which was considered for the conclusions for vector computers. Here, we consider the case of large values of ε and the case of smaller values would change the plots in the same way as for vector processors. Some conclussions can be summed up for this contribution with the help of figure 2

a) For the case of computers with small communication start-up (figure 2.a), *R*-CR is the fastest algorithm for systems of equations with weak diagonal dominance or small size. Also, OPM is the fastest algorithm for moderate to large systems with large diagonal dominance.

b) For computers with large communication start-ups (figure 2.b), Gaussian elimination is faster than *R*-CR, OPM and the early terminated DC for small systems of equations. The reason for this is that the time to perform one communication on such computers is larger than the time to perform a large number of memory and arthimetic instructions of Gaussian elimination.

c) For computers with large communication start-ups OPM is faster than *R*-CR in more cases than for computers with small communication start-ups. The reason for this is that OPM and the early terminated DC only require the equivalent to one communication during their execution while *R*-CR requires a number of communications larger than one.

d) For computers with large communication start-ups and large communication time per element, the early terminated DC is faster than OPM. The reason for this is that the amount of data sent by the only communication performed by OPM is larger than the amount of data sent by the only communication by the

early terminated DC.

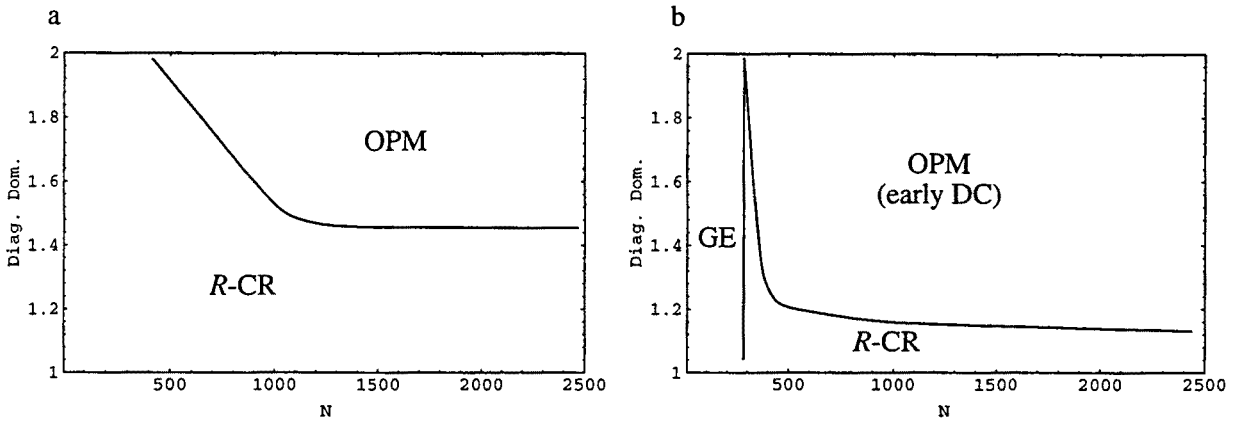


*Figure 2.*Behaviour of the methods on parallel processors. a) Small communication start-up, b) Large communication start-up. Between brackets, large communication time per element.

3) An analysis of two different applications has been performed (chapter 6). Those applications require the solution of tridiagonal systems of equations. We use some results of the analysis performed in the thesis to minimize the execution time of those applications.

The first application analyzed is the computation of Natural Cubic Splines and B-Splines. The systems arising from that problem are tridiagonal, almost Toeplitz and strictly diagonal dominant with diagonal dominance $\delta = 2 + \sqrt{3}$ (larger than that shown in figures 1 and 2). A method has been proposed to reduce the amount of work for the solution of that application. The method is based on an existing method that decomposes a Toeplitz tridiagonal system into four Toeplitz recurrences. The proposed variation is based on the observation that an incomplete solution of two of those systems is sufficient to solve the problem with a maximum absolute error ε. The two remaining systems have to be solved completely with the methods analyzed in the thesis.

As a conclusion for the analysis of this problem it is possible to say that it compares very favourably to a previous existing decomposition method [ChSh92] both from the point of view of the accuracy and from the point of view of the execution time. Another conclusion is that for the solution of the recurrences arising from the method proposed it is possible to use the methods analyzed here very successfully. For instance, OPM is the fastest method for systems of order larger than 200 when the maximum absolute error allowed is $\varepsilon = 10^{-7}$ while GE is the fastest method for smaller systems.

The results for the computation of Natural Cubic Splines and B-Splines curve fitting have been described in [LaNJ94].

The second application analyzed is the simulation of the electrical behaviour of neurons by domain

decomposition. The sparse systems given by the discretization of the problem using domain decomposition techniques imply the solution of a large quantity of tridiagonal systems of equations. Those tridiagonal systems are strictly diagonal dominant and some of them can be solved incompletely. This has a double advantage for the solution of the global system of equations. On one side, the savings for the solution of each tridiagonal system can be significative depending on its diagonal dominance. On the other side, the Schur complement matrix that has to be solved can be partitioned into sets of smaller independent matrices. This gives a lot of parallelism to the Schur complement solution and as a consequence, a reduction of the global execution time. Finally, some bounds of the achievable amount of savings have been quantified.

The results for the simulation of the electric behaviour of neurons have been described in [LMJN95].

Future work

Some work will be developped in the future as a spin-off of this thesis. We will study the use of OPM as a preconditioner for Conjugate Gradient type methods. Although some work has been already done on the topic of using overlapped preconditioners, it would be interesting to formalize the use of OPM as preconditioner. Also, we will study the use of the algorithms for tridiagonal systems from two different perspectives. On one side, we will unify DC and *R*-CR and will try to generalyze the bounds for the absolute error of those methods. On the other side, we will perform a study of the register requirements of OPM, DC and *R*-CR for superscalar and vector processors. Finally, in the field of applications we will implement the domain decomposition solver for the models of the electrical behaviour of neurons. This solver will be analyzed for parallel computers and compared to other existing solvers.

Appendix

In this appendix, we analyze some LU-type decompositions that can be used to factorize tridiagonal matrices. In the analysis, we compare the amount of computations caused by the decompositions and the amount of work to solve the different systems that arise from those decompositions with Gaussian elimination. For one of the factorizations analyzed here, we propose a method to save some work during the solution of the system.

1 Basic *LU*-type decompositions

Here, we describe a few *LU*-type decompositions that can be used to factorize the tridiagonal matrices arising from the applications analyzed in this thesis.

The types of decompositions described in this section are the classic *LU* decomposition and the *LDU* decomposition both for general and Toeplitz matrices. The LDL^T and the Cholesky factorizations for general and Toeplitz symmetric matrices. The decomposition by Evans and Taha and Liaw [Evan80, TaLi93] for Toeplitz matrices and an extension of this decomposition that we propose for the case of strictly diagonal dominant matrices. Finally, the modification of the classic *LU* decomposition proposed by Malcolm and Palmer for Toeplitz symmetric strictly diagonal dominant systems [MaPa74].

In the following, we describe the decompositions mentioned in the introduction and the procedure to find the final solution to a tridiagonal system through them. Also, we count the number of arithmetic, load and store operations per equation required to compute the decomposition and the final solution assuming that we use Gaussian elimination. In order to minimize the count of number of operations per equation, we assume that we reuse data through registers to reduce the amount of loads as explained in chapter 4.

In section 2 we compare the different decompositions from the point of view of the amount of work to compute them and the amount of work to compute the solution to the systems they give rise to. We do this both for general and Toeplitz systems.

_LU_ decomposition for general systems

The *LU* decomposition of a general tridiagonal matrix A can be described by the following recursions

$$
\begin{aligned}
u_{[1]} &= d_{[1]} \\
l_{[i]} &= \frac{e_{[i]}}{u_{[i-1]}} \quad , \quad 2 \le i \le N \\
u_{[i]} &= d_{[i]} - c_{[i-1]} \frac{e_{[i]}}{u_{[i-1]}} \quad , \quad 2 \le i \le N
\end{aligned}
\tag{1}
$$

With this, we have that the resulting factorization is

$$
A = \begin{bmatrix}
d_{[1]} & c_{[1]} & & & 0 \\
e_{[2]} & d_{[2]} & c_{[2]} & & \\
 & e_{[3]} & d_{[3]} & c_{[3]} & \\
 & & \cdots & \cdots & \cdots \\
0 & & & e_{[N]} & d_{[N]}
\end{bmatrix}
= LU =
\begin{bmatrix}
1 & & & & \\
l_{[2]} & 1 & & & \\
 & l_{[3]} & 1 & & \\
 & & \cdots & \cdots & \\
 & & & l_{[N]} & 1
\end{bmatrix}
\begin{bmatrix}
u_{[1]} & c_{[1]} & & & \\
 & u_{[2]} & c_{[2]} & & \\
 & & u_{[3]} & \cdots & \\
 & & & \cdots & c_{[N-1]} \\
 & & & & u_{[N]}
\end{bmatrix}
$$

We can rewrite (1) into

$$
\begin{aligned}
u_{[1]} &= 1/d_{[1]} \\
l_{[i]} &= e_{[i]} u_{[i-1]} \quad , \quad 2 \le i \le N \\
u_{[i]} &= 1/(d_{[i]} - c_{[i-1]} l_{[i]}), \quad 2 \le i \le N
\end{aligned} \tag{2}
$$

in such a way that elements $u_{[i]}$ store the inverse of the diagonal of matrix U saving one division during the solution of recurrences $Lz = b$ and $Ux = z$.

The amount of work per equation to compute the LU decomposition is $1\mathrm{D} + 2\mathrm{P} + 1\mathrm{A} + 3\mathrm{L} + 2\mathrm{S}$ where D, P, A, L and S stand for Division, Product, Add-like operations (addition, substraction, change of sign,...), Load and Store respectively. The amount of work per equation to solve recurrences $Lz = b$ and $Ux = z$ with Gaussian elimination is $3\mathrm{P} + 2\mathrm{A} + 5\mathrm{L} + 2\mathrm{S}$.

It is important to note that this decomposition does not keep the Toeplitz nature of the original matrix. For Toeplitz matrices, the amount of work per equation to compute the LU decomposition is $1\mathrm{D} + 2\mathrm{P} + 1\mathrm{A} + 2\mathrm{S}$ and to solve recurrences $Lz = b$ and $Ux = z$ is $3\mathrm{P} + 2\mathrm{A} + 4\mathrm{L} + 2\mathrm{S}$.

***LDU* decomposition**

The recursions that define the LDU decomposition of a general tridiagonal system are

$$
\begin{aligned}
t_{[1]} &= 1/d_{[1]} \\
u_{[i]} &= c_{[i]} t_{[i]} \quad , \quad 1 \le i \le N-1 \\
l_{[i]} &= e_{[i]} t_{[i-1]} \quad , \quad 2 \le i \le N \\
t_{[i]} &= 1/(d_{[i]} - l_{[i]} c_{[i-1]}), \quad 2 \le i \le N
\end{aligned} \tag{3}
$$

With this, we have that the resulting matrices are

$$
LDU = \begin{bmatrix} 1 & & & \\ l_{[2]} & 1 & & \\ & \cdots & \cdots & \\ & & l_{[N]} & 1 \end{bmatrix} \cdot \begin{bmatrix} t_{[1]} & & & \\ & t_{[2]} & & \\ & & \cdots & \\ & & & t_{[N]} \end{bmatrix} \cdot \begin{bmatrix} 1 & u_{[1]} & & \\ & 1 & \cdots & \\ & & \cdots & u_{[N-1]} \\ & & & 1 \end{bmatrix}
$$

in such a way that elements $t_{[i]}$ keep the inverse of the diagonal matrix D saving one division during the solution of $Lz = b$, $Dy = z$ and $Ux = y$.

The amount of work per equation to compute the LDU decomposition is $1\mathrm{D} + 3\mathrm{P} + 1\mathrm{A} + 3\mathrm{L} + 3\mathrm{S}$. The amount of work per equation to solve recurrences $Lz = b$, $Dy = z$ and $Ux = y$ is $3\mathrm{P} + 2\mathrm{A} + 5\mathrm{L} + 2\mathrm{S}$ supposing that $Lz = b$ and $Dy = z$ are solved together to save one load per equation.

For Toeplitz systems, the amount of work per equation to compute the LDU decomposition is $1\mathrm{D} + 3\mathrm{P} + 1\mathrm{A} + 3\mathrm{S}$and to solve recurrences $Lz = b$, $Dy = z$ and $Ux = y$ is $3\mathrm{P} + 2\mathrm{A} + 5\mathrm{L} + 2\mathrm{S}$.

LDL^T decomposition for symmetric systems

The recursions that define the LDL^T decomposition of a symmetric tridiagonal system are

$$t_{[1]} = 1/d_{[1]}$$
$$l_{[i]} = e_{[i]} t_{[i-1]} \quad , \quad 2 \le i \le N$$
$$t_{[i]} = 1/(d_{[i]} - e_{[i]}^2 t_{[i-1]}), \quad 2 \le i \le N$$

in such a way that the resulting matrices are

$$A = \begin{bmatrix} d_{[1]} & e_{[2]} & & \\ e_{[2]} & d_{[2]} & e_{[3]} & \\ & \cdots & \cdots & \cdots \\ 0 & & e_{[N]} & d_{[N]} \end{bmatrix} = LDL^T = \begin{bmatrix} 1 & & & \\ l_{[2]} & 1 & & \\ & \cdots & \cdots & \\ & & l_{[N]} & 1 \end{bmatrix} \cdot \begin{bmatrix} t_{[1]} & & & \\ & t_{[2]} & & \\ & & \cdots & \\ & & & t_{[N]} \end{bmatrix} \cdot \begin{bmatrix} 1 & l_{[2]} & & \\ & 1 & \cdots & \\ & & \cdots & l_{[N]} \\ & & & 1 \end{bmatrix}$$

with $t_{[i]}$ storing the inverse of D.

The solution to system $Ax = b$ is equivalent to solve the following three recurrences $Lz = b$, $Dy = z$ and $L^T x = y$. The amount of work per equation to compute the LDL^T decomposition is $1\mathrm{D} + 2\mathrm{P} + 1\mathrm{A} + 2\mathrm{L} + 2\mathrm{S}$ and to solve the above recurrences is $3\mathrm{P} + 2\mathrm{A} + 5\mathrm{L} + 2\mathrm{S}$ supposing that $Lz = b$ and $Dy = z$ are solved at the same time to save one load per equation.

Also in this case, this decomposition does not keep the Toeplitz nature of the original matrix. The amount of work per equation to compute the LDL^T decomposition of a Toeplitz matrix is $1\mathrm{D} + 2\mathrm{P} + 1\mathrm{A} + 2\mathrm{S}$ and to solve recurrences $Lz = b$, $Dy = z$ and $L^T x = y$ is $3\mathrm{P} + 2\mathrm{A} + 5\mathrm{L} + 2\mathrm{S}$ supposing that $Lz = b$ and $Dy = z$ are solved at the same time to save one load per equation.

Cholesky factorization

The recursions that define the Cholesky factorization of a symmetric tridiagonal system are

$$t_{[1]} = 1/\sqrt{d_{[1]}}$$
$$g_{[i]} = e_{[i]} t_{[i-1]} \quad , \quad 2 \le i \le N$$
$$t_{[i]} = \sqrt{(t_{[i-1]} d_{[i]} - g_{[i]}^2)/t_{[i-1]}} \quad , \quad 2 \le i \le N$$

in such a way that the resulting matrices are

$$A = GG^T = \begin{bmatrix} t_{[1]} & & & \\ g_{[2]} & t_{[2]} & & \\ & \cdots & \cdots & \\ & & g_{[N]} & t_{[N]} \end{bmatrix} \cdot \begin{bmatrix} t_{[1]} & g_{[2]} & & \\ & t_{[2]} & \cdots & \\ & & \cdots & g_{[N]} \\ & & & t_{[N]} \end{bmatrix}$$

where $t_{[i]}$ stores the inverse of the main diagonal.

The amount of work per equation to compute the Cholesky factorization ($A = GG^T$) is $1D + 2P + 1A + 1SR + 2L + 2S$ where SR stands for Square Root. The amount of work per equation to compute $Gy = b$ and $G^T x = y$ is $4P + 2A + 6L + 2S$.

For the case of Toeplitz matrices, the amount of work per equation to compute the Cholesky factorization ($A = GG^T$) is $1D + 2P + 1A + 1SR + 2S$ and the amount of work to solve a Toeplitz system with this decomposition is the same as for non-Toeplitz systems.

Taha and Liaw variant of Evans decomposition for Toeplitz systems

Taha and Liaw proposed an *LU*-type decomposition for 4 diagonal Toeplitz matrices in 1993 [TaLi93]. This decomposition is similar to that proposed for tridiagonal matrices by Evans [Evan80]. Here, we describe the tridiagonal version of this factorization that we call *TJ* decomposition and count the number of operations per equation needed to perform it.

The decomposition of matrix A into matrices *T* and *J* preserves the Toeplitz nature of the original matrix at the cost of solving two more systems. These two systems can be solved while performing the decomposition of the matrix which means that no extra cost is incurred by this method to the solution of the decomposed system.

In addition to the description of the *TJ* decomposition, we propose a way to save some operations to the solution of the system for the strictly diagonal dominant case. A variant of this decomposition for almost Toeplitz strictly diagonal dominant systems is described in chapter 6.

Let us suppose that we have a Toeplitz tridiagonal matrix. It is possible to find the following Toeplitz factorization

$$(4)\qquad \begin{bmatrix} d & c & & & \\ e & d & c & & \\ & \ddots & \ddots & \ddots & \\ & & e & d & c \\ & & & e & d \end{bmatrix}_{N\times N} = \alpha TJ = \alpha \begin{bmatrix} \beta & 1 & & & \\ & \beta & 1 & & \\ & & \ddots & \ddots & \\ & & & \beta & 1 \\ & & & & \beta \end{bmatrix}_{N\times (N+1)} \begin{bmatrix} \gamma & & & & \\ 1 & \gamma & & & \\ & \ddots & \ddots & & \\ & & 1 & \gamma & \\ & & & 1 & \end{bmatrix}_{(N+1)\times N}$$

where the values of α, β and γ are:

$$(5)\qquad \alpha = (d \pm \sqrt{d^2 - 4ec})/2 \qquad \beta = e/\alpha \qquad \gamma = c/\alpha$$

The solution of the system can be found by solving $Ax = \alpha TJt = b$ in the following steps:

1. Solve $Ty = b/\alpha$. This is performed by scaling b by $1/\alpha$ ($b' = b/\alpha$) and finding the solution of

$Ty = b'$ assuming that y is an order N+1 vector, $y = \left[y_0\, y_1 \ldots y_{n-1}\, y_n\right]^T$. Given that the system is underdetermined, variable y_0 is considered a parameter. Thus, the system can be transformed into $T'y' = b' - y_0 f$ where

$$T' = \begin{bmatrix} 1 & & & \\ \beta & 1 & & \\ & \cdots & \cdots & \\ & & \beta & 1 \end{bmatrix}_{n \times n}, \quad y' = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}, \quad f = \begin{bmatrix} \beta \\ 0 \\ \\ 0 \end{bmatrix}$$

The solution of this new system can be found by solving two bidiagonal systems $T'g = b'$ and $T'h = f$. Note that $T'h = f$ can be solved during the decomposition of matrix A because it does not vary for different right hand sides.

Now, y' is a linear combination of the two solutions $y' = g - y_0 h$.

2. Solve $Jt = y$. With the value of y' it is possible to see that $Jt = y = \begin{bmatrix} y_0 \\ y' \end{bmatrix} = \begin{bmatrix} 0 \\ g \end{bmatrix} - y_0 \begin{bmatrix} -1 \\ h \end{bmatrix}$.

Assuming that $t = t1 - y_0 t2$, solve the following systems $J't1 = g$ and $J't2 = h$ where

$$J' = \begin{bmatrix} 1 & \gamma & & \\ & 1 & \gamma & \\ & & \cdots & \\ & & & 1 \quad \gamma \\ & & & 1 \end{bmatrix}_{n \times n}.$$

Note that $J't2 = h$ can be solved during the decomposition of matrix A for the same reason as $T'h = f$.

Now, it is necessary to find the value of y_0. We know that $y_0 = \gamma t_1$. We also know that $t_1 = t1_1 - y_0 t2_1$. With these two equations we have that $y_0 = \gamma \dfrac{t1_1}{1 + \gamma t2_1}$.

Finally, the solution of the system can be found by solving $t = t1 - y_0 t2$.

The amount of work per equation to perform the *TJ* decomposition is negligible. Nevertheless, during the decomposition we can solve systems $T'h = f$ and $J't2 = h$, as we said. In case that we do so, the amount of work per equation for the decomposition goes up to 2P + 1A + 1CS + 1L + 2S, where CS stands for Change of Sign. During the solution of a system with this decomposition we have to perform the following amount of work per equation 4P + 3A + 3L + 2S. This amount is the addition of work for the

scaling of b $b' = b/\alpha$, the solution of $T'g = b'$ and $J't1 = g$ and the update performed in $t = t1 - y_0 t2$. For this operation count we suppose that we solve $b' = b/\alpha$ and $T'g = b'$ together and, $J't1 = g$ and $t = t1 - y_0 t2$ together.

TJ decomposition for strictly diagonal dominant systems

In the case of stricly diagonal dominant matrices, we can save some work to the solution of a system by TJ decomposition. The decomposition described up to here leads to strictly diagonal dominant factors if the original system is strictly diagonal dominant. In order to prove this, we propose the following lemma.

Lemma 1. If $|e| + |c| < |d|$ then the TJ decomposition gives $|\beta| < 1$ and $|\gamma| < 1$.

Proof. This is easy to be proved by using the definitions of β and γ in (5) and by assuming that we choose the additive deffinition of α, that is $\alpha = (d + \sqrt{d^2 - 4ec})/2$. □

Now, we show how some work can be saved under those circumstances. Note that the right hand side of system $T'h = f$ has one only element in position 1. So, we can apply lemma 2.1 and say that only k_{TJ} solutions are necessary if we want the error to the solution of $T'h = f$ be lower than ε

$$k_{TJ} \geq \left\lceil \frac{\log \varepsilon}{\log |\beta|} \right\rceil. \tag{6}$$

For the solution of $J't2 = h$ we only have to work on the k_{TJ} nonzero elements of h. Finally, during the computation of $t = t1 - y_0 t2$, only k_{TJ} elements of $t1$ have to be updated by $-y_0 t2$.

So, for the case of strictly diagonal dominant systems, the amount of work per equation for the decomposition of the first k_{TJ} equations of the system is $2P + 1A + 1CS + 1L + 2S$ and it is null for the rest. During the solution of a system with this decomposition we have to perform $4P + 3A + 3L + 2S$ for the first k_{TJ} equations of the system and $3P + 2A + 2L + 2S$ for the rest of equations.

Malcolm and Palmer decomposition for Toeplitz symmetric systems

The decomposition proposed by Malcolm and Palmer in [MaPa74] is based on the classic LU decomposition. The authors showed that some work can be saved when solving symmetric strictly diagonal dominant Toeplitz systems. The description of the method is as follows. Consider the following matrix

$$\begin{bmatrix} d & e & & & \\ e & d & e & & \\ & \ddots & \ddots & \ddots & \\ & & e & d & e \\ & & & e & d \end{bmatrix} = \frac{1}{e} \begin{bmatrix} \alpha & 1 & & & \\ 1 & \alpha & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & \alpha & 1 \\ & & & 1 & \alpha \end{bmatrix} \tag{7}$$

Then, the LU decomposition of this matrix can be computed in the form shown in the previous sections

and gives the following factors:

$$L = \begin{bmatrix} 1 & & & & \\ l_{[2]} & 1 & & & \\ & \cdots & \cdots & & \\ & & l_{[N-1]} & 1 & \\ & & & l_{[N]} & 1 \end{bmatrix}, U = \begin{bmatrix} u_{[1]} & 1 & & & \\ & u_{[2]} & 1 & & \\ & & \cdots & \cdots & \\ & & & u_{[N-1]} & 1 \\ & & & & u_{[N]} \end{bmatrix}$$

Malcolm and Palmer showed that for $|\alpha| \geq 2$ the sequences $u_{[i]}$ and $l_{[i]}$ converge to u and l. For the case of $u_{[i]}$, they show that the sequence converges to u where

$$u = \frac{1}{2}\left[\alpha + \text{sgn}(\alpha)\,(\alpha^2 - 4)^{\frac{1}{2}}\right].$$

Also, they show that for $|\alpha| > 2$ (which means strict diagonal dominance or $\delta > 1$ with our notation), a given floating point radix β and a certain number of exact digits required t for the decomposition, the number of terms of $u_{[i]}$ to be computed until its value converges to u has an upper bound in

$$k_{MP} \leq \left\lceil \frac{(t-1)}{2\log_\beta u} \right\rceil \tag{8}$$

which can be transformed into

$$k_{MP} \leq \left\lceil \frac{-\log\varepsilon}{2\log u} \right\rceil \tag{9}$$

if we assume that a computer with a t-digit base β arithmetic has precision β^{1-t} [Bois91].

With this, the amount of work per equation to perform the LU decomposition is $1\text{D} + 2\text{P} + 1\text{A} + 2\text{S}$ for the first k_{MP} equations of the matrix. The amount of work per equation to solve $Lz = b$ and $Ux = z$ is $3\text{P} + 2\text{A} + 4\text{L} + 2\text{S}$ for the first k_{MP} elements of the recurrences and $3\text{P} + 2\text{A} + 2\text{L} + 2\text{S}$ for the rest.

2 Comparison of the *LU*-type decompositions

Here we compare the amount of work performed by the different decompositions described above. First, we count the amount of work per equation to solve a tridiagonal system $Ax = b$ with Gaussian elimination. If the system is general, then the number of operations per equation is $1\text{D} + 4\text{P} + 3\text{A} + 7\text{L} + 3\text{S}$. For the case of general symmetric systems, the amount of work per equation reduces to $1\text{D} + 4\text{P} + 3\text{A} + 6\text{L} + 3\text{S}$ operations. In case that the system is Toeplitz (either symmetric or non-symmetric), the amount of operations per equation to solve system $Ax = b$ is $1\text{D} + 4\text{P} + 3\text{A} + 3\text{L} + 3\text{S}$.

In general, the decomposition and further solution of a system compares always favourably in front of the use of Gaussian elimination directly. For this reason, although we include the operation counts per equation in the following Tables, we do not mention them in the text.

Table 1 shows a summary of the amount of work per equation performed by the *LU* and *LDU* decompositions for general non-symmetric systems. In this case, both the *LU* and *LDU* decompositions lead to the same operation count for the solution of the system. Nevertheless, if we turn to the amount of work for the decomposition, *LU* compares favourably.

	LU	*LDU*	No decomp.
Dec.	1D + 2P + 1A + 3L + 2S	1D + 3P + 1A + 3L + 3S	-
Solve	3P + 2A + 5L + 2S	3P + 2A + 5L + 2S	1D + 4P + 3A + 7L + 3S

Table 1. General non-symmetric systems: Amount of work per equation for the work generated by the *LU*-type decompositions and the solution of the tridiagonal system with Gaussian elimination.

Table 2 shows the amount of work per equation performed by *LU*, Cholesky and LDL^T decompositions for general symmetric systems. There, we can see that although Cholesky factorization was specially designed for symmetric systems, it has a higher operation count. Also, LDL^T decomposition has the same operation count as *LU* for the solution of the system. In this case, the amount of work for the decomposition is lower for LDL^T.

	LU	GG^T	LDL^T	No decomp.
Dec.	1D + 2P 1A + 3L + 2S	1D + 2P + 1A 1SR + 2L + 2S	1D + 2P 1A + 2L + 2S	-
Solve	3P + 2A 5L + 2S	4P + 2A 6L + 2S	3P + 2A 5L + 2S	1D + 4P + 3A 6L + 3S

Table 2. General symmetric systems: Amount of work per equation for the work generated by the *LU*-type decompositions and the solution of the tridiagonal system with Gaussian elimination.

Table 3 shows the amount of work per equation performed by the *LU*, LDL^T and *TJ* decompositions for Toeplitz systems. Note that the LDL^T decomposition can only be used for symmetric systems. For non-symmetric systems, it is important to note that depending on the specific cost of the load operations in front of the additions and products on a certain computer one of the two decompositions (*LU* or *TJ*) may have a lower operation count.

Finally, Table 4 shows the amount of work per equation performed by the *LU*, LDL^T, *TJ* or *MP* decompositions for strictly diagonal dominant Toeplitz systems. In this case, we have to note that for non-symmetric systems, the *TJ* decomposition has the lowest operation count depending on the cost of the load

and arithmetic operations. For the case of symmetric systems, the decomposition by Malcolm and Palmer can work better for large diagonal dominances, i.e. large values of k_{MP} (note that $k_{MP} = k_{TJ}/2$). Nevertheless, for large diagonal dominances, i.e. small values of k_{MP} the differences shorten and can be unnoticeable. Note that (6) and (9) imply that $k_{MP} \approx k_{TJ}/2$ which means a considerable difference for large values of k_{TJ}.

	LU	LDL^T only symmetric	*TJ*	No decomp.
Dec.	1D + 2P 1A + 2S	1D + 2P 1A + 2S	2P + 1A 1CS + 1L + 2S	-
Solve	3P + 2A 4L + 2S	3P + 2A 5L + 2S	4P + 3A 3L + 2S	1D + 4P + 3A 3L + 3S

Table 3. Toeplitz systems (symmetric and non-symmetric): Amount of work per equation for the work generated by the *LU*-type decompositions and the solution of the tridiagonal system with Gaussian elimination.

	LU	LDL^T only symmetric	*TJ*	*MP* only symmetric	No decomp.
Dec.	1D + 2P 1A + 2S	1D + 2P 1A + 2S	2P + 1A 1CS + 1L + 2S for k_{TJ}	1D + 2P 1A + 2S for k_{MP} equations	-
Solve	3P + 2A 4L + 2S	3P + 2A 5L + 2S	4P + 3A 3L + 2S for k_{TJ} equations and 3P + 2A 2L + 2S for the rest	3P + 2A 4L + 2S for k_{MP} equations and 3P + 2A 2L + 2S for the rest	1D + 4P + 3A 3L + 3S

Table 4. Toeplitz systems (strictly diagonal dominant): Amount of work per equation for the work generated by the *LU*-type decompositions and the solution of the tridiagonal system with Gaussian elimination.

References

[Abu-85] I. K. Abu-Shumays. *Comparison of methods and algorithms for tridiagonal systems and for vectorization of diffusion computations*, Supercomputer Applications, R. Numrich ed., Plenum Press, New York & London, 1985.

[AxEi86] O. Axelsson and V. Eijkhout. *A note on the vectorization of scalar recurrences.* Parallel Computing 3 (1986), pp. 73-83.

[BaBB86] R. Bartels, J. Beatty and B. Barsky. *An introduction to Splines for use in computer graphics and geometric modeling.* Morgan Kaufmann Publishers, inc., Los Altos (CA) 1986.

[Babu72] J. Babuska. *Numerical stability in problems of linear algebra.* SIAM j. Num. Anal. 9 (1972), pp.53-77.

[Bar-87] I. Bar-On. *A practical parallel algorithm for solving band symmetric positive definite systems of linear equations.* ACM TOMS, Vol. 13, No. 4, December 1987, pp. 323-332.

[BeEv93] M. P. Bekakos and D. J. Evans. *Parallel cyclic odd-even reduction algorithms for solving Toeplitz tridiagonal equations on MIMD computers.* Parallel Computing, 19 (1993), pp. 545-561.

[BeTs89] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation. Numerical methods.* Prentice Hall, Englewood Cliffs, 1989.

[BlCZ92] G. Blelloch, S. Chatterjee and M. Zagha. *Solving Linear Recurrences with Loop Raking*, Intl. Parallel Proc. Symposium, 1992, pp. 416-424.

[Bois91] R. Boisvert. *Algorithms for special tridiagonal systems*, SIAM J. Sci. Stat. Comput., 12 (1991), pp423-442.

[Bond91] S. Bondelli. *Divide and conquer: a parallel algorithm for the solution of a tridiagonal linear system of equations*, Parallel Computing, 17 (1991), pp. 419-434.

[BuGN70] B. Buzbee, G. Golub and C. Nielson. *On direct methods for solving Poisson's equations.* SIAM J. Num. Anal. Vol. 7 No. 4 December 1970, pp. 627-656.

[BrCM93] R. Bru, C. Corral and J. Mas. Overlapping Multisplitting Preconditioners for the Conjugate Gradient Method. SIAM Int. Conf. on Par. Proc. for Sci. Comp., Norfolk, 1993, pp. 660-663.

[Brug91] L. Brugnano. *A parallel solver for tridiagonal linear systems for distributed memory parallel computers.* Parallel Computing 17 (1991), pp. 1017-1023.

[ChSh92] K. Chung and L. Shen. *Vectorized Algorithm for B-Spline curve Fitting on Cray X-MP EA/16se.* Proceedings of the Supercomputing'92 conference. pp. 166-169.

[CoKn91] C. Cox and J. Knisely. *A tridiagonal system solver for distributed memory parallel processors with vector nodes.* Parallel and Distributed Computing, Vol. 13, 1991, pp. 325-331.

[Conr89] J. M. Conroy. *Parallel algorithms for the solution of narrow banded systems.* Applied Numerical Mathematics, 5 (1989), pp. 409-421.

[Conv91] Convex Computer Corp. *Convex C-3400 Supercomputer System Overview.* Richardson, Texas, 1991.

[deGr91] P.P.N. de Groen. *Base-p-cyclic reduction for tridiagonal systems of equations.* Appl. Num. Math., 8 (1991), pp 117-125.

[DMBS79] J. J. Dongarra, C. B. Moler, J. R. Bunch and G. W. Stewart. *LINPACK User's Guide.* SIAM, Philadelphia, 1979.

[DoGK84] J. J. Dongarra, F. G. Gustavson and A. Karp. *Implementing linear algebra algorithms for dense matrices on a vector pipeline machine.* SIAM Review, Vol. 26 No. 1 January 1984, pp. 91-112.

[DoJo87] J. J. Dongarra and L. Johnsson. *Solving banded systems on a parallel processor.* Parallel Computing, Vol. 5, 1987, pp. 219-246.

[DoLe92] D. Dodson and S. Levin. *A tricyclic tridiagonal equation solver*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1246-1254.

[DuRo77] P. Dubois and G. Rodrigue. *An analysis of the Recursive Doubling algorithm.* In High Speed Computer and Algorithm organization, Kuck, Lawrie and Sameh eds. Academic Press, New York, 1977.

[Evan72] D. J. Evans. *An algorithm for the solution of certain tridiagonal systems of linear equations.* The Comp. Journal, Vol. 15, No. 4, 1972, pp. 356-359

[Evan80] D. J. Evans. *On the solution of certain Toeplitz tridiagonal linear systems.* SIAM J. on Numer. Anal., Vol 17, No. 5, October 1980, pp. 675-680.

[EvHa75] D. J. Evans and M. Hatzopoulos. *The solution of certain banded systems of linear equations using the folding algorithm,* The Computer Journal, 19 (1975), pp. 184-187.

[EvMe89] D. J. Evans and G. M. Megson. *Fast triangularization of a symmetric tridiagonal matrix.* J. of Par. and Distr. Comput., Vol. 6, 1989, pp.663-678.

[EvYo94] D. J. Evans and W. S. Yousif. *The solution of unsymmetric tridiagonal Toeplitz systems by the strides reduction algorithm.* Parallel Computing 20 (1994), pp. 787-798.

[GaSa89] E. Gallopoulos and Y. Saad. *A parallel block cyclic reduction algorithm for the fast solution of elliptic equations.* Parallel Computing, 10 (1989), pp. 143-159.

[Gava84] D. Gannon and J. van Rosendale. *On the impact of Communication Complexity on the Design of Parallel Numerical Algorithms.* IEEE Trans on Computers, Vol C-33, N.12, December 1984, pp. 1180-1194.

[GBDJ94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck and V. Sunderam. *PVM3 User's Guide and Reference Manual.* Oak Ridge Natl. Labs. May 1994.

[GoVa89] G. H. Golub and C. F. Van Loan. *Matrix Computations,* second edition. Johns Hopkins University Press, Baltimore and London, 1989.

[Guit94] J. Guitart. Personal communication. February, 1994.

[GuRu93] J. Guitart and S. Ruiz-Moreno. *Strict calculation of the light statistics at the output of a travelling wave optical amplifier,* Electronics Letters, 29 (1993), pp. 1589-1590.

[HäSc90] H. Häfner and W. Schönauer. *Investigation of different algorithms for the first order recurrence,* Supercomputer, 40 (1990), pp. 34-41.

[Hatz82] M. Hatzopoulos. *Parallel linear system solvers for tridiagonal systems,* Parallel Processing Systems, an advanced course, D. J. Evans ed., Cambridge Univ. Press, 1982.

[Hegl91] M. Hegland. *On the parallel solution of tridiagonal systems by wrap-around partitioning and incomplete LU factorization,* Numer. Math. 59 (1991), pp. 453-472.

[Hell74] D. Heller. *Some Aspects of the Cyclic Reduction Algorithm for Block Tridiagonal Linear Systems,* SIAM J. Numer. Anal. 13 (1976), pp. 484-496.

[Hines84] M. Hines. *Efficient computation of branched nerve equations.* Int. J. Bio-Medical Computing, 15 (1984), pp. 69-76.

[Hock65] R. W. Hockney. *A fast direct solution of Poisson's equation using Fourier analysis.* J. ACM, 12 (1965), pp. 95-113.

[Hock93] R. W. Hockney. *Performance parameters and benchmarking of supercomputers.* in Computer Benchmarks, J.J. Dongarra and W. Gentxsch (eds.) Elsevier Science Publishers B.V., Amsterdam 1993.

[HoPo90] W. Hoffmann and K. Potma. *Experiments with basic linear algebra algorithms on a meiko computing surface,* Tech. Report CS-90-02, Dept. of Comp. Systems, Univ. of Amsterdam, 1990.

[HoJe81] R. W. Hockney and C. R. Jesshope. *Parallel Computers.* Adam Hilger, Bristol, 1981.

[HoJe88] R. W. Hockney and C. R. Jesshope. *Parallel Computers 2,* Adam Hilger, Bristol and Philadelphia, 1988.

[John85] S. L. Johnsson. *Solving Narrow Banded Systems on Ensemble Architectures.* ACM TOMS, Vol. 11, No. 3, September 1985, pp. 271-288.

[John86] S. L. Johnsson. *Band Matrix Systems Solvers on Ensemble Architecture.* In, Supercomputers: Algorithms, Architecture and Scientific Computing. F. A. Matsen and T. Tagima (eds.). Univ. of Texas Press, Austin, 1986.

[John87] S. L. Johnsson. *Solving tridiagonal systems on ensemble architectures,* SIAM J. Sci. Stat. Comput., 8 (1987), pp. 354-392.

[JoHo87] S. L. Johnsson and C. Ho. *Multiple tridiagonal systems, the alternating direction methods and boolean cube configured multiprocessors,* Yale Univ., Dept. of Computer Sci., Tech. Report YALEU/DCS/TR-532, 1987.

[JoLN92] Àngel Jorba, Josep-L. Larriba-Pey and Juan J. Navarro. *A proof for the accuracy of OPM.* Research Report RR-92/10, Centre Europeu de Paral.lelisme de Barcelona, Universitat Politècnica de Catalunya, Spain, 1992.

[JoSS87] S. L. Johnsson, Y. Saad and M. Shultz. *Alternating Direction Methods on Multiprocessors.* SIAM J. Sci. Stat. Comp. Vol. 8, No. 5, September 1987. pp. 686-700.

[KaBr84] R. N. Kapur and J. C. Browne. *Block tridiagonal system solution on reconfigurable array computers.* Proc. Intl. Conf. on Parallel Processing, ICPP'81, pp. 92-99.

[KaBr84] R. N. Kapur and J. C. Browne. *Techniques for solving block tridiagonal systems on reconfigurable array computers.* SIAM J. Sci. Stat. Comp. Vol. 3, No. 3, September 1984, pp. 701-719.

[Kers82] D. Kershaw. *Solution of single tridiagonal linear systems and vectorization of the ICCG algorithm on the Cray-1,* Parallel Computations, G. Rodrigue ed., Academic Press, New York, 1982.

[KiAz92] B. Kirk and Y. Azmy. *An iterative algorithm for solving the multidimensional neutron diffusion nodal method equations on parallel computers.* Nuclear Science and Engineering, No. 111, 1992, pp. 57-65.

[KiLe90] H. J. Kim and J. G. Lee. *A parallel algorithm solving a tridiagonal Toeplitz linear system.* Parallel Computing, 13 (1990), pp. 289-294.

[KrPS89] A. Krechel, H. Plum ans K. Stüben. *Parallel solution of tridiagonal linear systems,* 1st European Workshop on Hypercube and Distributed Computers, F. Andre and J. P. Verjus eds., North-Holland, Amsterdam, 1989.

[KrPS89] A. Krechel, H. Plum ans K. Stüben. *Parallelization and vectorization aspects of the solution of tridiagonal linear systems.* Parallel Computing, Vol. 14, 1990, pp.31-40.

[Kuma89] S. Kumar. *Solving tridiagonal linear systems on the Buttrefly parallel computer.* The Intl. J. on Supercomputer Applications, Vol. 3, N. 1, Spring 1989, pp. 75-81.

[LaJN93] Josep-L. Larriba-Pey, Àngel Jorba and Juan J. Navarro. *A Parallel Tridiagonal Solver for Vector Uniprocessors.* SIAM Int. Conf. on Par. Proc. for Sci. Comp., Norfolk, 1993, pp. 590-597.

[LaJN93b] Josep-L. Larriba-Pey, Àngel Jorba and Juan J. Navarro. *Spike Algorithm with savings for strictly diagonal dominant tridiagonal systems,* Microprocessing and Microprogramming, 39 (1993) pp. 125-128. North Holland.

[LaJN95] Josep-L. Larriba-Pey, Àngel Jorba and Juan J. Navarro. *A generalized criterion for the early termination of R-Cyclic Reduction and Divide and Conquer for recurrences.* To be published in the Proc. of the 7th SIAM Conf. on Par. Proc. for Sci. Comp. San Francisco, Feb. 1995.

[LaNJ94] Josep-L. Larriba-Pey, Juan J. Navarro and Àngel Jorba. *Vectorized algorithms for Natural Cubic Spline and B-Spline Curve Fitting.* CEPBA Research Report RR-94/15, Centre Europeu de Paral.lelisme de Barcelona, Universitat Politècnica de Catalunya, Spain, 1994.

[Larr90] Josep-L. Larriba-Pey. *On the Improvement of First Solution Vectors for Iterative Methods.* Proceedings of the Mini and Microcomputers and their Applications Conference 1990. pp. 154-157. ACTA Press.

[Larr93] Josep-L. Larriba-Pey. *Tridiagonal systems on parallel computers.* Invited talk. 3rd Mons Workshop on Parallel Computing. Mons (Belgium), September 23-24 1993.

[LaSa84] D. Lawrie and A. Sameh. *The Computation and Communication Complexity of a Parallel Banded System Solver*, ACM TOMS, 10 (1984), pp. 185-195.

[LaVo75] J. J. Lambiotte and R. G. Voigt. *The solution of tridiagonal linear systems on the CDC STAR-100 computer*. ACM TOMS, 1 (1975), pp. 308-329.

[Levi89] C. Levit. *Parallel Solution of Pentadiagonal Systems Using Generalized Odd-Even Elimination*. Proceedings of the ICS'89. pp. 333-336.

[LMJN95] Josep-L. Larriba-Pey, Michael. Mascagni, Àngel Jorba and Juan J. Navarro. *An analysis of the parallel computation of arbitrarily branched cable neuron models*. To be published in the Proc. of the 7th SIAM Conf. on Par. Proc. for Sci. Comp. San Francisco, Feb. 1995.

[LNRJ94] Josep-L. Larriba-Pey, Juan J. Navarro, O. Roig and Àngel Jorba. *A generalized vision of some parallel bidiagonal systems solvers*. Proc. of the Intl. Conf. on Supercomputing 1994, ACM Press, pp. 404-411.

[Lope94] J. López Gómez. *Arquitectura unificada para sistemas tridiagonales*. PhD dissertation, Universidad de Málaga, 1994.

[LoZa94] J. López Gómez and E. L. Zapata. *Unified Architecture for Divide and Conquer based tridiagonal system solvers*. IEEE Trans. on Computers. To appear.

[MaCh90] S. Ma and A. Chronopoulos. *Implementation of Iterative Methods for Large Sparse Nonsymmetric Linear Systems on a Parallel Vector Machine*. The Intl. J. on Superc. Appl. Vol 4 No. 4 1990, pp. 9-24.

[MaPa74] M. A. Malcolm and J. Palmer. *A fast method for solving a class of tridiagonal linear systems*, Comm. ACM, 17 (1974), pp. 14-17.

[Masc89] M. Mascagni. *Numerical methods for neuronal modeling*. Methods in Neuronal Modeling: From Synapse to Networks, C. Koch and I. Segev (Eds.) MIT Press, Cambridge, MA, pp. 439-484.

[Masc91] M. Mascagni. *A parallelizing algorithm for computing solutions to arbitrarily branched cable neurons*, J. of Neuroscience Methods, Elsevier Sci. Publishers 36 (1991), pp. 105-114.

[MaRo77] N. K. Madsen and G. H. Rodrigue. *Odd-Even Reduction for Pentadiagonal Matrices*. in Parallel Computers-Parallel Mathematics. M. Feilmeier (ed.) Intl. Assoc. for Mathematics and Computers in Simulation (1977) pp.103-106.

[Mehr93] V. Mehrmann. *Divide and Conquer for block tridiagonal systems*. Parallel Computing 19 (1993), pp. 257-279.

[Meie85] U. Meier, *A Parallel Partition Method for Solving Banded Systems of Linear Equations*, Parallel Computing. 2 (1985), pp. 33-43.

[MüSc91] S. Müller and D. Scheerer. *A method to parallelize tridiagonal solvers*. Parallel Computing, 17 (1991), pp. 181-188.

[Munt89] H. Munthe-Kaas. *Topics in Linear Algebra for Vector and Parallel Computers*. Dr. Ing. Thesis. The Norwegian Institute of Technology. Trodheim. 1989

[O'Br92] O'Brien et al. *Advanced Compiler Technology for the RISC System/6000 Architecture*. In IBM Risc System/6000 Technology. IBM Corp, Mechanicsburg Penn. Publication SA23-2619.

[O'LWh85] D. O'Leary and R. White. *Multi-Splittings of Matrices and Parallel Solution of Linear Equations*. SIAM J. on Alg. and Discrete Methods, Vol. 6 n. 4, Oct. 1985. pp. 630-640.

[OrVo85] J. M. Ortega and R. G. Voigt. *Solution of Partial Differential Equations on Vector and Parallel Computers*. SIAM, Philadelphia, 1985.

[Over90] R. Overill. *On the efficient vectorization of the general first-order linear recurrence relation*. Supercomputer, 42 (1990), pp. 31-36.

[Pisk92] Pl.Iv. Piskoulijski. *Error analysis of a parallel algorithm for the solution of a tridiagonal Toeplitz linear system of equations*. Parallel Computing, 18 (1992), pp. 431-438.

[Rall77] W. Rall. *Core conductor theory and cable properties of neurons*. Handbook of Physiology: The Nervous System, Vol.1, Kandel, E. R. Brookhardt, J. M. and V. B. Mountcastle (Eds.), Williams and Wilkins, Co., Baltimore, MD, pp. 39-98.

[RaRo88] G. Radicati di Brozzolo and Y. Robert. *Parallel and Vector Conjugate Gradient like Algorithms for Sparse non Symmetric linear Systems.* Proceedings of ICS'88. pp.478-487. ACM Press.

[Real90] F. Reale. *A tridiagonal solver for massively parallel computer systems,* Parallel Computing, 16 (1990), pp. 361-368.

[ReRo84] E. Reiter and G. Rodrigue. An incomplete-Choleski factorization by a matrix partition algorithm. Proceedings of the second symposium on elliptic problem solvers, Vol. II. Ac. Press, 1984.

[Reut88] R. Reuter. *Solving tridiagonal systems of linear equations on the IBM 3090 VF,* Parallel Computing, 8 (1988), pp. 371-376.

[RoAd90] D.F. Rogers and J.A. Adams. *Mathematical Elements for Computer Graphics.* McGraw- Hill, New York, 1990.

[SaKu78] A. Sameh and D. Kuck. *On Stable Parallel Linear System Solvers.* J.ACM, 25 (1978), pp. 81-91.

[Schö87] W. Schönauer. *Scientific Computing on Vector Computers.* 2, Special topics in Supercomputing. North-Holland.Amsterdam, 1987.

[SpEv93] G. Spaletta and D. J. Evans. The parallel recursive decoupling algorithm for solving tridiagonal linear systems. Parallel Computing 19 (1993), pp. 563-576.

[Ston73] H. S. Stone. *An efficient parallel algorithm for the solution of a tridiagonal linear system of equations.* J. ACM, 20 (1973), pp. 27-38.

[Ston75] H. S. Stone. *Parallel Tridiagonal Equation Solvers.* ACM TOMS, Vol. 1 No. 4 December 1975, pp. 289-307.

[Swar74] P. N. Swarztrauber. *A Direct Method for the Discrete Solution of Separable Elliptic Equations.* SIAM J. Num. Anal, Vol. 11 No 6, December 1974, pp. 1136-1150.

[Swar79] P. N. Swarztrauber. *A Parallel Algorithm for Solving General Tridiagonal Equations.* Mathematics of Computation, Vol. 33, No. 145, January 1979, pp. 185-199.

[Swee74] R. Sweet. *A generalized Cyclic Reduction algorithm.* SIAM J. Num. Anal, Vol. 11 No 3, June 1974, pp. 506-520.

[Swee77] R. Sweet. *A Cyclic Reduction Algorithm for Solving Block Tridiagonal Systems of Arbitrary Dimension.* SIAM J. Numer. Anal., Vol 14, No4, September 1974, pp. 706-720.

[SuSN89] X. Sun, H. Sun and L. Ni. *Parallel algorithms for solution of tridiagonal systems on multiprocessors,* Proc. of the 3rd Intl. Conf. on Supercomputing, Crete, Greece, ACM Press, 1989.

[SuZN92] X. Sun, H. Zhang and L. Ni. *Efficient tridiagonal solvers on multicomputers,* IEEE Trans. on Comput., 41 (1992), pp. 286-296.

[TaLi93] T. Taha and J. Liaw. *An Algorithm for solving a 4-Diagonal Toeplitz Linear System of Equations on Vector Computers.* Proceedings of the Sixth SIAM Conf. on Parall. Process. for Scientific Computing. pp. 510-514. Norfolk, Va.

[Tran94] Transtech Ltd. Paramid architecture reference manual. 1994.

[Trans94a] Transtech Ltd. PVM reference manual. 1994

[Tsao94] N.-K. Tsao. *An Error Model for Swarztrauber's Parallel Tridiagonal Equation Solver.* SIAM J. Matrix Anal. and Appl. Vol 15, No 2, April 1994, pp. 692-713.

[VaDe89] H. A. Van der Vorst and K. Dekker. *Vectorization of linear recurrence relations,* SIAM J. Sci. Stat. Comput., 10 (1989), pp. 27-35.

[VaLA92] M. Valero, T. Lang and E Ayguadé. *Conflict-Free Access of Vectors with Power-of-Two Strides.* 6th ACM ICS Conference, Washington D.C., USA, July 1992.

[Vand87] H. A. Van der Vorst. *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers.* Parallel Computing 5 (1987), pp. 45-54.

[Vand87b] H. A. Van der Vorst. *Analysis of a parallel solution method for tridiagonal linear systems.* Parallel Computing 5 (1987), pp. 303-311.

[Vand88] H. A. Van der Vorst. *Parallel solution of bidiagonal systems coming from discretized PDE's.* Trans. Magnetics, 24 (1988), pp.286-290.

[Vand89] H. A. Van der Vorst. *Practical aspects of Parallel Scientific Computing*, Future Generation Computer Systems, 4(1988/1989), pp. 285-291.

[WaMo91] X. Wang and Z.G. Mou. *A Divide-and-Conquer method of solving tridiagonal systems on hypercube massively parallel computers.* Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing, Dallas, pp. 810-817.

[Wang81] H. H. Wang. *A Parallel Method for Tridiagonal Equations*, ACM TOMS, 7 (1981), pp. 170-183.

[ZaLA94] E. L. Zapata, J. López and F. Argüello. Embedding r-ary tree based algorithms in constant geometry multiprocessors. To appear in IEEE Trans. on Parallel and Distributed Systems.

[Zhan91] H. Zhang. *On the accuracy of the parallel diagonal dominant algorithm.* Parallel Computing (1991), pp. 265-272.