Parallel Algorithms for Elementary Tridiagonal Systems

Bogdan Dumitrescu
Department of Automatic Control
"POLITEHNICA" University of Bucharest
313 Splaiul Independenţei,
77206 Bucharest
ROMANIA

Abstract: Six methods that solve tridiagonal systems with one equation per processor are studied in this paper. A catalogue of their complexities is given. The main goal is to stress the importance of this elementary system when solving large tridiagonal systems.

Bogdan Dumitrescu was born in Bucharest, in 1962. He received the engineering degree in Automatic Control and Computer Science in 1987, and the Ph.D. degree in Automatic Systems, in 1993, both from the "POLITEHNICA" University of Bucharest. He has been an assistant professor there since 1990. In 1992, he worked for six months at the Laboratoire de Modélisations et Calculus, Grenoble, France. His main research interests are in parallel algorithms and numerical computation.

1. Introduction

This paper is concerned with algorithms capable of solving the tridiagonal system $Ax = d$ on a parallel computer, when the dimension n of A is equal to the number of processors p. Such a system is called *elementary* and the problem of solving it is denoted by $3DIAGEL(p)$. This problem is common to all known parallel algorithms: cyclic reduction of Hockney and Golub [6] in the implementation of Reale [10], Wang partition algorithm [14] in a variant presented in [4], or the divide and conquer algorithm of Sun, Zhang and Ni [13], Müller [9] and Bondeli [3].

Viewed within the context of the MIMD distributed memory computers, parallel architectures show simplicity, when they are ring, and accessibility,, when they are hypercube. There are many other architectures in which a ring can be embedded. There are p processors, denoted $P_0, P_1, \ldots, P_{p-1}$.

Let us denote by α the time required by a floating point operation and by $\sigma + m\beta$ the time necessary for the transmission of a message of length time and β the time during which a floating point element (and not a byte, as usual) is actually transmitted. A processor can simultaneously use all its ports (multiport model) and two communication models are available: full duplex (by default) and half duplex.This paper does not use others but known global communication algorithms. [11, 8, 12, 2].

2. Solving Elementary Tridiagonal Systems

In fact,the tridiagonal systems problem-solving algorithms (except Wang) are not so simple, since each processor has 1-2 equations. But processors which have two equations can eliminate one of them; for example, if a processor has the equations:

$$\begin{array}{ccccccc} b_i x_{i-1} & + & a_i x_i & + & c_i x_{i+1} & = & d_i \\ b_{i+1} x_i & + & a_{i+1} x_{i+1} & + & c_{i+1} x_{i+2} & = & d_{i+1} \end{array}$$

x_{i+1} will be expressed by the second equation and replaced in the first, in time 6α. After solving the elementary problem, the processor will compute x_{i+1} in time $5\alpha + (\sigma + \beta)$.

Now, we have to solve a $3DIAGEL(p)$ problem, processor P_k having the following equation (i.e. its coefficients):

$$b_k x_{k-1} + a_k x_k + c_k x_{k+1} = d_k$$

In the sequel are presented several ways of solving this problem. There are two points to be

considered: the basic algorithm and the communication technique; as all transmitted messages are very short, a pipeline will be not useful, nor the partition of a message into packets will be. The execution times for all algorithms are summarized in Table 1 for half duplex model and in Table 2 for full duplex.

A. Gaussian Elimination.

a. The well- known Gaussian elimination algorithm (without pivoting) can be implemented on a ring as follows. Processor P_k receives a_{k-1}, c_{k-1}, d_{k-1} coefficients from its left, then updates its coefficients (a_k, c_k, d_k) and sends them to the right; this is the elimination step; then it receives the solution x_{k+1} from its right, computes x_k and sends it to its left, completing the backsubstitution step. It can be noted that computation and communication are fully sequential, starting from P_0 and ending on P_0 computing x_0. Thus, the execution time is the same as that of the sequential Gaussian elimination (let us approximate it by $8p\alpha$, for simplicity) for arithmetic; communication complexity is $(p-1)(2\sigma+4\beta)$. The same execution time (line **Aa** in Tables) is valid for any architecture in which a ring can be embedded.

b. The sequential nature of Gaussian elimination is often mentioned; however, there is one way to parallel it, namely *twisted factorization* in [1]. Gaussian elimination proceeds as schematically presented in Figure 1a, where Arrow 1 shows the order in which subdiagonal elements are eliminated and Arrow 2 the order in which the unknowns are computed. But other way is also possible, as in Figure 1b: in parallel one eliminates subdiagonal elements for the first half of the system and superdiagonal elements for the second half. Then, also in parallel, the unknowns are computed, beginning with the center equations, towards the first and the last, respectively. The execution time will be half that of the algorithm **Aa**.

c. The idea of [13]; each P_k processor sends its coefficients (a_k, b_k, c_k, d_k) to all the other processors; this is an all-to-all broadcast; then, all processors solve, in parallel, the whole system. Thus, the arithmetic complexity is $8p\alpha$ and the communication complexity will depend on the architecture and on the communication model. Execution times specified in Tables are the best known times for all-to-all broadcast, short messages.

For example, on a ring, half duplex model, each processor sends a message to the right (starting with its message), receiving other message from its left, until the message arrives to the processor from its left, i.e. after $(p-1)$ steps. Communication time is $(p-1)(\sigma+4\beta)$.

In a full duplex model, each processor can send its message to both neighbours, receiving at once their messages. This reduces the communication time by approx. 50 %.

On a hypercube, algorithms are complicated, and it is not the goal of this paper to present them.

d. Other idea, not yet common to the literature, would be to let a single processor solve the system. Each processor sends its coefficients to a processor P_c (this is a gathering), which solves the system, while the other processors are idle; then, P_c sends to each processor the solution it needs (this is a scattering).

Gathering and scattering have the same complexity, being dual operations. On a ring, P_c sends messages on both channels, beginning with those meant for the farthest processors; all the other processors re-transmit messages until they receive the message addressed to them. On gathering, the same algorithm is used, but in reverse order. Communication takes $p(\sigma+\frac{5}{2}\beta)$, as 4 elements are transmitted to the gathering and 1 to the scattering.

B. Fully Parallel Cyclic Reduction.

Cyclic reduction is a well- known algorithm, so it will not be reminded here. Usually, at a first step reduction is recursively applied, halving each time the number of equations; then, solutions are found in a substitution step. When applied to $3DIAGEL(p)$, some processors are idle most of the time and only one 'central' processor works all the time.

The idea is to use reduction from the point of view of each processor. Let us denote by $e_k = (a_k, b_k, c_k, d_k)$ the vector of coefficients of an equation and consider $e_j = (1,0,0,0)$ for $j<0$ or $j>p-1$ (adding $x_j = 0$ to the system equations does not change the solution); the evolution of the computation is suggested in Figure 3; e_k^l denotes the coefficient vector belonging to P_k at the recursion level l. At the first recursion level, P_k communicates with its immediate neighbours on a ring; at the next levels,

P_k communicates with processors P_j, the 'distance' $|j-k|$ between the processors doubling each time. On a hypercube will only immediate neighbours communicate. There are $\log p$ recursion levels.

Arithmetic complexity does not depend on topology and is always $12(\log p)\alpha$. On a hypercube, communication complexity is $(\log p)(\sigma + 4\beta)$, while on a ring: $\sum_{l=0}^{\log p-1} 2^l(\sigma+4\beta) \approx p(\sigma+4\beta)$. For the half duplex model, communication time is twice longer.

C. Parallel Prefix Method.

Another algorithm with $O(\log p)$ for arithmetic is proposed in [5]. In a matricial form, the equation of P_k can be written like that:

$$\begin{bmatrix} x_{k+1} \\ x_k \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_k & \beta_k & \gamma_k \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \\ 1 \end{bmatrix}$$

$$\Longleftrightarrow X_{k+1} = B_k X_k$$

with $x_{-1} = x_p = 0$. Then X_k can be expressed as: $X_k = B_{k-1}B_{k-2}\cdots B_1 B_0 X_0$.

Let $C_i^j = B_j B_{j-1}\cdots B_i$. If all the products C_0^j, for $0 \le i \le p-1$, and X_0, are computed, then solutions can be computed. The problem of computing the products C_0^j is a classical one in parallel algorithmic - prefix sum, see [7]. Computation proceeds like in Figure 2.

After the computation of C_0^{p-1} by P_{p-1}, equation $X_p = C_0^{p-1}X_0$ yields X_0 . Then P_{p-1} broadcast X_0 to all other processors, which compute the appropriate solution.

Arithmetic complexity is $20(\log p)\alpha$; at each communication step 6 elements of a matrix C_i^j are transmitted, the last row remaining unchanged on multiplication. On a hypercube, communication only takes place between neighbours. On a ring, execution time is computed the same way as for the cyclic reduction. See Tables for the results.

3. Comparisons and Conclusions

Execution times of the six algorithms which solve $3DIAGEL1(p)$ are listed in Table 1 and Table 2. Among many possible conclusions, our selection keeps the following:

- on ring topology, for the Gaussian elimination, the best algorithm is the twisted factorization algorithm **b**; on hypercube, for the half duplex model the new algorithm **d** will be the best; for full duplex, one can choose between **d** and **c**;
- fully parallel cyclic reduction is always better than prefix method;
- on hypercube, cyclic reduction is obviously the best algorithm;
- on ring, however, the twisted factorization is usually better than cyclic reduction. For a half duplex model, the relation:

$$4p\alpha + (p-1)(\sigma+2\beta) < 12(\log p)\alpha + p(2\sigma+8\beta)$$

$$\Longleftrightarrow 4 - \frac{12\log p}{p} < \frac{\sigma+6\beta}{\alpha}$$

is obvious; for a full duplex model, the right term of the unequality will be $\frac{2\beta}{\alpha}$, and, usually, $\beta > 2\alpha$ (transmission is slower than computation).

Of course, the list of algorithms solving $3DIAGEL(p)$ can be enriched. The main point is, when choosing an algorithm for solving tridiagonal systems, to use the appropriate solver of the elementary problem that eventually comes up.

Table 1: **Execution Times for Solving $3DIAGEL(p)$, Half Duplex Model**

Method	Ring	Hypercube
Aa	$8p\alpha + (p-1)(2\sigma + 4\beta)$	$8p\alpha + (p-1)(2\sigma + 4\beta)$
Ab	$4p\alpha + (p-1)(\sigma + 2\beta)$	$4p\alpha + (p-1)(\sigma + 2\beta)$
Ac	$8p\alpha + (p-1)(\sigma + 4\beta)$	$8p\alpha + \left\lceil \frac{p-1}{\log p} \right\rceil (2\sigma + 8\beta)$
Ad	$8p\alpha + p(\sigma + \frac{5}{2}\beta)$	$8p\alpha + 2(\log p)\sigma + 5 \left\lceil \frac{p-1}{\log p} \right\rceil \beta$
B	$12(\log p)\alpha + p(2\sigma + 8\beta)$	$12(\log p)\alpha + (\log p)(2\sigma + 8\beta)$
C	$20(\log p)\alpha + p\left(\frac{3}{2}\sigma + \frac{13}{2}\beta\right)$	$20(\log p)\alpha + (\log p)(2\sigma + 7\beta)$

Table 2: **Execution Times for Solving $3DIAGEL(p)$, Full Duplex Model**

Method	Ring	Hypercube
Aa	$8p\alpha + (p-1)(2\sigma + 4\beta)$	$8p\alpha + (p-1)(2\sigma + 4\beta)$
Ab	$4p\alpha + (p-1)(\sigma + 2\beta)$	$4p\alpha + (p-1)(\sigma + 2\beta)$
Ac	$8p\alpha + p(\frac{1}{2}\sigma + 2\beta)$	$8p\alpha + \left\lceil \frac{p-1}{\log p} \right\rceil (\sigma + 4\beta)$
Ad	$8p\alpha + p(\sigma + \frac{5}{2}\beta)$	$8p\alpha + 2(\log p)\sigma + 5 \left\lceil \frac{p-1}{\log p} \right\rceil \beta$
B	$12(\log p)\alpha + p(\sigma + 4\beta)$	$12(\log p)\alpha + (\log p)(\sigma + 4\beta)$
C	$20(\log p)\alpha + p\left(\frac{3}{2}\sigma + \frac{13}{2}\beta\right)$	$20(\log p)\alpha + (\log p)(2\sigma + 7\beta)$

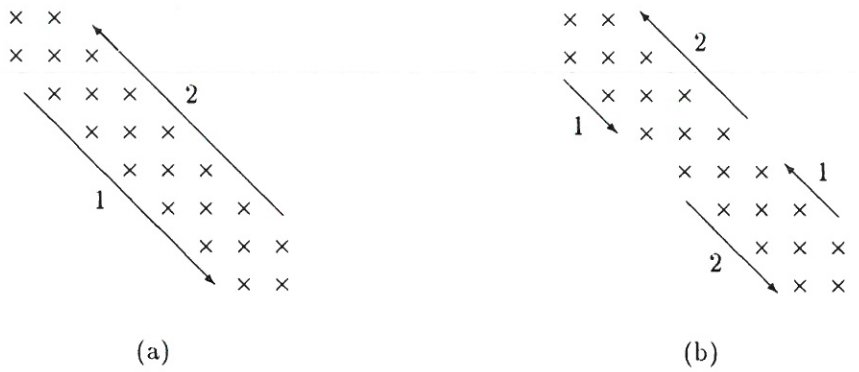

Figure 1: **Computing Schemes for:** (a) Classical Gaussian Elimination (b)Twisted Factorization

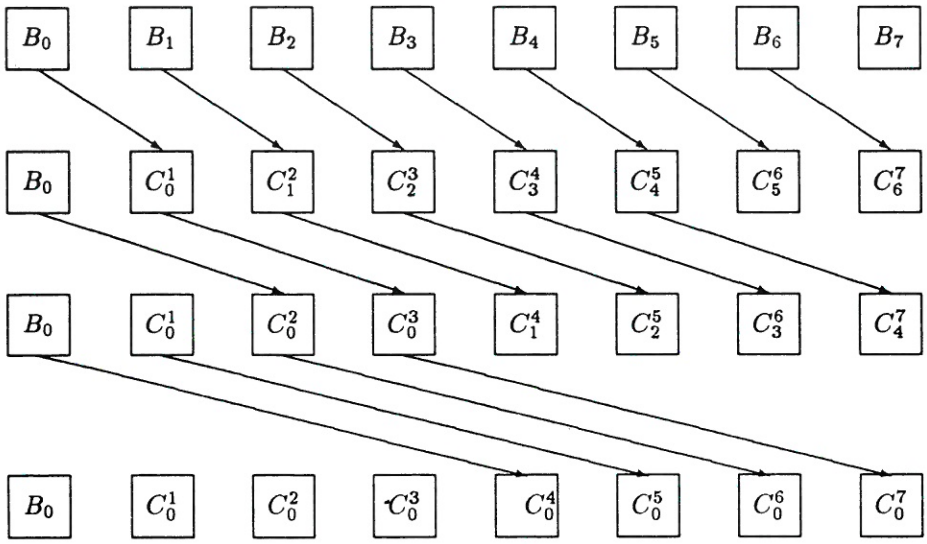

Figure 2: A Scheme for the Computation of C_0^j in the Prefix Method

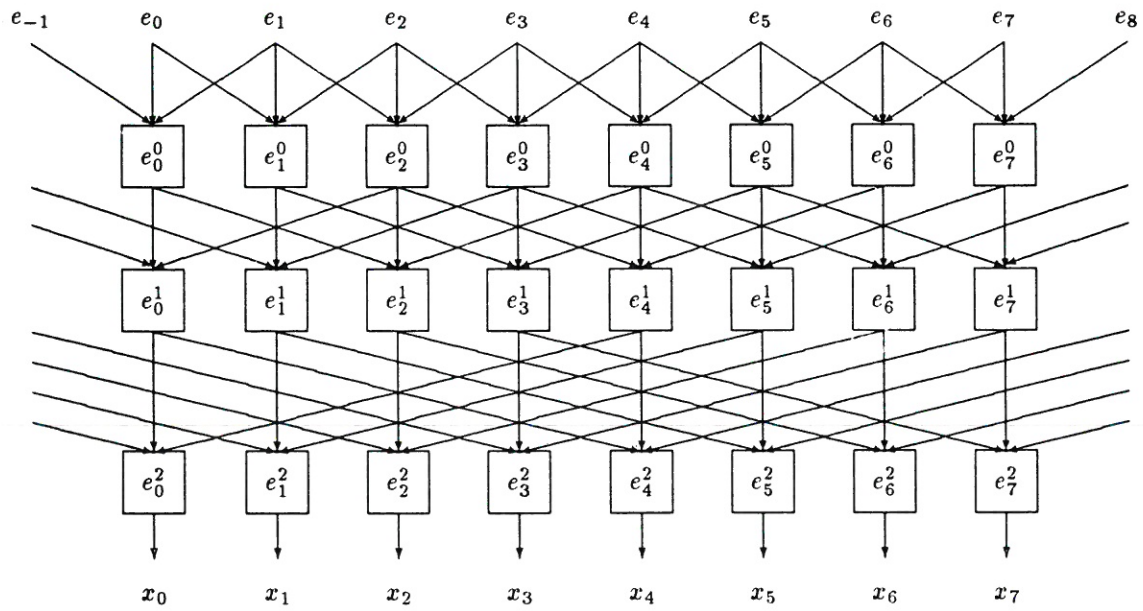

Figure 3: Computation Scheme for the Fully Parallel Cyclic Reduction, for $p = 8$

REFERENCES

1. BABUSKA,I., **Numerical Stability in Problems of Linear Algebra**, SIAM J. NUMER. ANAL., 9(1),1972, pp. 53–77.
2. BERTSEKAS, D.P., ÖZVEREN, C., STAMOULIS, G.D., TSENG, P. and TSITSIKLIS, J.N., **Optimal Communication Algorithms for Hypercubes**, J. PAR. DISTRIB. COMPUT., 11, 1991, pp.263–275.
3. BONDELI, S. **Divide and Conquer: A New Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations**, Lecture Notes in Computer Science 457, CONPAR 90, SPRINGER-VERLAG, 1990, pp. 108–119.
4. DUMITRESCU, B., **Improvements for Wang Partition Method on MIMD Architectures, 9-th Internat. Conf. on Control Systems and Computer Science**, Bucharest, 1993.
5. EGECIOGLU, O., KOC, C.K. and LAUB, A.J., **A Recursive Doubling Algorithm for Solution of Tridiagonal Systems on Hypercube Multiprocessors**, J. COMPUT. APPL. MATH., 27, 1989.
6. HOCKNEY, R.W., **A Fast Direct Solution of Poisson's Equation Using Fourier Analysis**, J.ACM, 12, 1965, pp.95–113.
7. KARP, R.M. and RAMACHANDRAN, V., **Parallel Algorithms for Shared - Memory Machines**, in J. van Leeuwen (Ed.) Handbook of Theoretical Computer Science, ELSEVIER Science Publishers, 1990.
8. LENNART JOHNSSON, S. and HO, C.T., **Optimum Broadcasting and Personalized Communication in Hypercubes**, IEEE TRANS. COMPUT., 38(9), 1989, pp.1249–1268.
9. MÜLLER, S.M., **A Method to Parallelize Tridiagonal Solvers**, 5th Distributed Memory Computing Conference, 1990, pp. 340–345.
10. REALE, F., **A Tridiagonal Solver for Massively Parallel Computer Systems**, PARALLEL COMPUTING, 16, 1990, pp.361–368.
11. SAAD, Y. and SCHULTZ, M.H., **Data Communication in Parallel Architectures**, PARALLEL COMPUTING , 11, 1989, pp.131–150.
12. STOUT, Q.F. and WAGAR, B., **Intensive Hypercube Communications - Prearranged Communication in Link-Bound Machines**, J. PAR. DISTRIB. COMPUT., 10, 1990, pp.167–181.
13. SUN, X.H., ZHANG, H. and NI, L.M., **Efficient Tridiagonal Solvers on Multicomputers**, IEEE TRANS. COMPUT., 41(3), 1992, pp.286–296.
14. WANG, H.H., **A Parallel Method for Tridiagonal Equations**, ACM TRANS. MATH. SOFTWARE, 7(2), 1981, pp.170–183.