Z AMM 76 (1996) S1, 115 - 118

AMODIO, P.; BRUGNANO, L.

Stable Parallel Solvers for General Tridiagonal Linear Systems

We analyze the problem of solving tridiagonal linear systems on parallel computers. In the case where the coefficient matrix is symmetric positive definite and/or diagonally dominant, this problem has been extensively discussed. More difficult is the case where the coefficient matrix does not satisfy the above requirements. Here we present two algorithms which are able to handle general tridiagonal matrices, and compare them on some test problems carried out on a distributed memory parallel computer.

1. Introduction

Let us consider the parallel solution of the linear system

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

where the coefficient matrix A is tridiagonal:

$$A = \begin{pmatrix} a_1 & c_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & b_{n-1} & a_n \end{pmatrix}. \tag{2}$$

For simplicity, in the following we will assume $n = kp - 1$, if p is the number of parallel processors to be used.

Problem (1) has been extensively analyzed in the last decades (see, for example, [1,3,4,5] and the references therein). In [1,3] it has been shown that very efficient parallel solvers can be derived by associating the ith processor with the following tridiagonal block:

$$M^{(i)} = \begin{pmatrix} 0 & c_0^{(i)}\mathbf{e}_1^T & 0 \\ b_0^{(i)}\mathbf{e}_1 & A^{(i)} & c_{k-1}^{(i)}\mathbf{e}_{k-1} \\ 0 & b_{k-1}^{(i)}\mathbf{e}_{k-1}^T & a_k^{(i)} \end{pmatrix}, \qquad i = 2, \ldots, p-1, \tag{3}$$

where $A^{(i)}$ is a square tridiagonal block of size $k-1$, $\mathbf{e}_1$ and $\mathbf{e}_{k-1}$ are respectively the first and last unit vectors in $\mathbb{R}^{k-1}$, and

$$a_j^{(i)} = a_{(i-1)k+j}, \qquad b_j^{(i)} = b_{(i-1)k+j}, \qquad c_j^{(i)} = c_{(i-1)k+j}.$$

The block corresponding to the first processor ($i = 1$) is obtained from (3) by deleting the first row and column, while that corresponding to the last processor ($i = p$) is obtained by deleting the last row and column.

Parallel solvers are obtained by considering suitable factorizations $N^{(i)}S^{(i)}$ of the block $A^{(i)}$, and the following factorization for the block $M^{(i)}$:

$$M^{(i)} = \begin{pmatrix} 1 & \mathbf{w}^{(i)T} & 0 \\ \mathbf{0} & N^{(i)} & \mathbf{0} \\ 0 & \mathbf{v}^{(i)T} & 1 \end{pmatrix} \begin{pmatrix} \alpha_1^{(i)} & \mathbf{0}^T & \gamma^{(i)} \\ \mathbf{0} & I_{k-1} & \mathbf{0} \\ \beta^{(i)} & \mathbf{0}^T & \alpha_2^{(i)} \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0}^T & 0 \\ \mathbf{z}^{(i)} & S^{(i)} & \mathbf{y}^{(i)} \\ 0 & \mathbf{0}^T & 1 \end{pmatrix}. \tag{4}$$

The vectors $\mathbf{v}^{(i)}, \mathbf{w}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)}$ are easily obtained by direct identification from (3) and (4), showing that the factorization (4) exists if and only if the blocks $A^{(i)}$ are nonsingular.

In this case, the parallel solver derived from (4) proceed as follows:

1. the linear sub-systems with the blocks $N^{(i)}$ are solved in parallel on each processor and then the right-hand side is updated in parallel by means of the vectors $\mathbf{v}^{(i)^T}$ and $\mathbf{w}^{(i)^T}$;

2. the second matrix in the factorization (4) corresponds to the solution of a *reduced system* with the coefficient matrix

$$T_p = \begin{pmatrix} (\alpha_2^{(1)} + \alpha_1^{(2)}) & \gamma^{(2)} & & \\ \beta^{(2)} & (\alpha_2^{(2)} + \alpha_1^{(3)}) & \ddots & \\ & \ddots & \ddots & \gamma^{(p-1)} \\ & & \beta^{(p-1)} & (\alpha_2^{(p-1)} + \alpha_1^{(p)}) \end{pmatrix}_{(p-1)\times(p-1)}$$

   The solution of this system, whose cost depends on the number of processors p and not on the size n of the problem, represents the only scalar part of the algorithm;

3. parallel updates of the right-hand side are made through the vectors $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ and finally the sub-systems with the blocks $S^{(i)}$ are solved in parallel.

Examples of parallel solvers are obtained by considering the factorizations

$$A^{(i)} = L^{(i)} R^{(i)}, \qquad A^{(i)} = L^{(i)} U^{(i)} D^{(i)},$$

where $L^{(i)}$ is lower bidiagonal with unit diagonal entries, $R^{(i)}$ and $U^{(i)}$ are upper bidiagonal, $U^{(i)}$ with unit diagonal entries, and $D^{(i)}$ is diagonal (see [1,3] for more details). These factorizations essentially correspond to Johnsson's algorithm and Wang's algorithm, respectively. In [1,3] it is also introduced a very efficient parallel solver based on the cyclic reduction factorization which turns out to be the most effective parallel solver, among those previously considered.

All the above solvers are well-defined if and only if all the tridiagonal blocks $A^{(i)}$ in (3) are nonsingular. Typically, they can be used when the matrix (2) is symmetric positive definite or diagonally dominant. However, there exist tridiagonal linear systems which do not fit these requirements. As an example, consider the following problem, also used for the numerical tests:

$$\begin{pmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & 0 & 1 \\ & & -1 & 1 \end{pmatrix}_{n\times n} \mathbf{x} = \mathbf{b}. \tag{5}$$

It can be shown that $\kappa(A) \propto n$, so that the problem is quite well-conditioned when $n \ll \varepsilon^{-1}$, where ε is the machine precision. Nevertheless, the factorization (3) may be not defined since the blocks $A^{(i)}$ are singular when k is even.

2. Stable modified factorizations

When A is nonsingular but some of the blocks $A^{(i)}$ are singular, we obtain a stable parallel solver by considering a generalization of the above approach, consisting in a finer structuring of the factorization itself.

Suppose that the $(k-1)\times(k-1)$ block $A^{(i)}$ is singular. Then we proceed as follows: let $A_1^{(i)}$ be the largest nonsingular principal sub matrix of $A^{(i)}$, and let $j-1$ be its size. We define the following factorization of the block $M^{(i)}$:

$$M^{(i)} = L_1^{(i)} D_1^{(i)} U_1^{(i)},$$

where

$$L_1^{(i)} = \begin{pmatrix} 1 & \mathbf{w}_1^{(i)T} & 0 & & \\ 0 & N_1^{(i)} & 0 & & \\ 0 & \mathbf{v}_1^{(i)T} & 1 & 0^T & 0 \\ & & 0 & I_r & 0 \\ & & 0 & 0^T & 1 \end{pmatrix}, \qquad U_1^{(i)} = \begin{pmatrix} 1 & 0^T & 0 & & \\ \mathbf{z}_1^{(i)} & S_1^{(i)} & \mathbf{y}_1^{(i)} & & \\ 0 & 0^T & 1 & 0^T & 0 \\ & & 0 & I_r & 0 \\ & & 0 & 0^T & 1 \end{pmatrix},$$

$$D_1^{(i)} = \begin{pmatrix} \hat{\alpha}_1^{(i)} & 0^T & \gamma_1^{(i)} & & \\ 0 & I_{j-1} & 0 & & \\ \beta_1^{(i)} & 0^T & \hat{\alpha}_2^{(i)} & c_j^{(i)}\hat{\mathbf{e}}_1^T & \\ & & b_j^{(i)}\hat{\mathbf{e}}_1 & A_2^{(i)} & c_{k-1}^{(i)}\hat{\mathbf{e}}_{k-j-1} \\ & & & b_{k-1}^{(i)}\hat{\mathbf{e}}_{k-j-1}^T & a_k^{(i)} \end{pmatrix},$$

and $\hat{\mathbf{e}}_i$ is the ith unit vector in $\mathbb{R}^{k-j-1}$. This is equivalent to introduce the variable $x_j^{(i)} = x_{(i-1)k+j}$ in the reduced system. The same procedure is then recursively applied to the tridiagonal block $A_2^{(i)}$ of $D_2^{(i)}$, and so on. Therefore the factorization is always defined, eventually by increasing the size of the reduced system, that is, the scalar section of the corresponding parallel solver. Obviously, to obtain a stable solver, we also have to use a stable solver for the blocks $A^{(i)}$. This will be done in the next section.

We show now that the size of the reduced system is always $O(p)$, where p is the number of the parallel processors. In fact, since the matrix A is tridiagonal, then it is nonsingular if and only if the size of the null space of each block $A^{(i)}$ does not exceed two. As a consequence, the size of the reduced system can be at most tripled. Moreover, if we assume that the matrix A is irreducible, it is easily shown that the size of the null space of $A^{(i)}$ can not exceed one, so that the size of the reduced system is at most doubled.

3. Stable parallel solvers

Assume for simplicity that the blocks $A^{(i)}$ are nonsingular. We now present two stable parallel solvers for general tridiagonal matrices based on the QR factorization, and on the LU factorization with partial pivoting (PLU). We will denote the corresponding algorithms as PQR (Parallel QR) and PPLU (Parallel PLU), respectively.

The PQR algorithm is obtained by considering the following factorization of $M^{(i)}$:

$$M^{(i)} = \begin{pmatrix} 1 & \mathbf{w}^{(i)T} & 0 \\ 0 & Q^{(i)} & 0 \\ 0 & \mathbf{v}^{(i)T} & 1 \end{pmatrix} \begin{pmatrix} \alpha_1^{(i)} & 0^T & \gamma^{(i)} \\ 0 & I_{k-1} & 0 \\ \beta^{(i)} & 0^T & \alpha_2^{(i)} \end{pmatrix} \begin{pmatrix} 1 & 0^T & 0 \\ \mathbf{z}^{(i)} & R^{(i)} & \mathbf{y}^{(i)} \\ 0 & 0^T & 1 \end{pmatrix},$$

where $Q^{(i)}R^{(i)}$ is the QR factorization of the matrix $A^{(i)}$. In [2] this algorithm has been deeply analyzed, so that we will skip the remaining details. We only recall that it requires $(43+\eta)n$ parallel operations, where η is the ratio between the time to execute one square root, and the time to execute one of the four basic arithmetic operations.

The PPLU algorithm is obtained by factoring $M^{(i)}$ as follows:

$$M^{(i)} = \begin{pmatrix} 1 & \mathbf{w}^{(i)T} & 0 \\ 0 & P^{(i)T}L^{(i)} & 0 \\ 0 & \mathbf{v}^{(i)T} & 1 \end{pmatrix} \begin{pmatrix} \alpha_1^{(i)} & 0^T & \gamma^{(i)} \\ 0 & I_{k-1} & 0 \\ \beta^{(i)} & 0^T & \alpha_2^{(i)} \end{pmatrix} \begin{pmatrix} 1 & 0^T & 0 \\ \mathbf{z}^{(i)} & U^{(i)} & \mathbf{y}^{(i)} \\ 0 & 0^T & 1 \end{pmatrix}. \tag{6}$$

Here we have considered the factorization $P^{(i)}A^{(i)} = L^{(i)}U^{(i)}$, where $P^{(i)}$ is a permutation matrix, $L^{(i)}$ is lower bidiagonal with unit diagonal entries, and $U^{(i)}$ is lower triangular with at most three nonzero diagonals. Moreover,

from (3) and (6), one obtains that:

$$
\begin{aligned}
U^{(i)^T}\mathbf{w}^{(i)} &= c_0^{(i)}\mathbf{e}_1, & U^{(i)^T}\mathbf{v}^{(i)} &= b_{k-1}^{(i)}\mathbf{e}_{k-1},\\
P^{(i)^T}L^{(i)}\mathbf{z}^{(i)} &= b_0^{(i)}\mathbf{e}_1, & P^{(i)^T}L^{(i)}\mathbf{y}^{(i)} &= c_{k-1}^{(i)}\mathbf{e}_{k-1},\\
\alpha_1^{(i)} &= -\mathbf{w}^{(i)^T}\mathbf{z}^{(i)}, & \alpha_2^{(i)} &= a_k^{(i)} - \mathbf{v}^{(i)^T}\mathbf{y}^{(i)},\\
\beta^{(i)} &= -\mathbf{v}^{(i)^T}\mathbf{z}^{(i)}, & \gamma^{(i)} &= -\mathbf{w}^{(i)^T}\mathbf{y}^{(i)}.
\end{aligned}
\tag{7}
$$

It is easy to verify that only the vectors $\mathbf{w}^{(i)}$ and $\mathbf{z}^{(i)}$ are full vectors (*fill-in vectors*).

To solve problem (1), it is possible to derive from (6) and (7) a cost for the PPLU algorithm of about $22n$ parallel operations. Moreover, additional $O(p)$ operations are required for the solution of the reduced system, but this term is negligible if $k \gg p$. Since the scalar PLU algorithm requires about $11n$ operations the asymptotic speedup on p processors is $(11\,n)/(22\,n/p) \equiv p/2$.

4. Numerical results

In this section we report some numerical results obtained by applying the PQR and PPLU algorithms to the solution of problem (5) on $p = 4, 8, 16$ processors. The considered parallel computer is a network of transputers $T805$ with the Express parallel libraries [6]. In Table 1 we report the execution times expressed in *ticks* (1 tick = 64 μsec) for the PQR and the PPLU methods (the lower index denote the number of the used processors), along with the execution times of the scalar QR and PLU method on a single processor. As one can see, for large n the predicted asymptotic speedup of the PPLU method over the scalar PLU is reached. Moreover, one verifies that, when the number p of the processors is fixed, the solver PPLU is up to nearly 1.7 times faster than the solver PQR and hence it is the method of choice (among those considered) for the parallel solution of general tridiagonal linear systems.

Table 1: execution times

n	QR	PQR_4	PQR_8	PQR_{16}	PLU	$PPLU_4$	$PPLU_8$	$PPLU_{16}$
1000	1093	471	300	244	506	308	230	211
2000	2187	893	509	344	1011	556	356	268
4000	4376	1740	899	555	2022	1051	605	399
8000	8574	3432	1779	979	4045	2050	1097	642
16000	17518	6819	3471	1825	8088	4043	2092	1141
32000	35030	13590	6855	3520	16178	8027	4083	2135
64000	70060	27138	13628	6904	32356	15998	8064	4128
128000	140120	–	27126	13675	64712	–	16039	8115
256000	280240	–	–	27224	129424	–	–	16085

5. References

1 Amodio, P., Brugnano, L. : Parallel factorizations and parallel solvers for tridiagonal linear systems; Linear Algebra Appl. **172** (1992), 347–364.

2 Amodio, P., Brugnano, L.: The parallel QR factorization algorithm for tridiagonal linear systems; Parallel Comput. (1995) (to appear).

3 Amodio, P., Brugnano, L., Politi, T. : Parallel factorizations for tridiagonal matrices; SIAM J. Numer. Anal. **30** (1993), 813–823.

4 Johnsson, S. L.: Solving tridiagonal systems on ensemble architectures; SIAM J. Sci. Statist. Comput. **8** (1987), 354–392.

5 Wang, H. H.: A parallel methods for tridiagonal equations; ACM Trans. Math. Software **7** (1981), 170–183.

6 Express fortran user's guide; ParaSoft Corporation, Pasadena, 1990.

Addresses: Pierluigi Amodio, Dipartimento di Matematica, Via Orabona 4, I-70125 Bari (Italy)
E-mail: amodio@max.uniba.it
Luigi Brugnano, Dipartimento di Energetica, Via Lombroso 6/17, I-50134 Firenze (Italy)
E-mail: na.brugnano@na-net.ornl.gov