Mathematica Balkanica

Mathematical Society of South-Eastern Europe
A quarterly published by
the Bulgarian Academy of Sciences – National Committee for Mathematics

Mathematica
Balkanica

New Series Vol. 9, 1995, Fasc. 2-3

Parallel Inversion of Block Tridiagonal Matrices

Dinko Gichev

Presented by P. Kenderov

A new algorithm for parallel inversion of a block tridiagonal matrix is proposed which is based on the Frobenius formulae. Estimates of the time and resources which are necessary for implemention of the algorithm are obtained.

1. Introduction

In this paper we shall consider the question about finding the inverse of a square $N \times N$ block tridiagonal matrix, i.e. a matrix of the following type:

$$(1) A = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & A_{32} & A_{33} & A_{34} & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & A_{n-1\ n-2} & A_{n-1\ n-1} & A_{n-1\ n} \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & A_{n\ n-1} & A_{n\ n} \end{bmatrix}$$

Here the blocks A_{ij}, $i,j = 1,2,\dots,n$ are square $k \times k$ matrices, i.e. $N = nk$. Let us suppose, for simplicity, that $n = 2^l$. To obtain the inverse matrix A^{-1} we shall apply a version of the Frobenius formulae [1]. As an auxiliary we need a method for finding the inverse of a square $k \times k$ matrix. Different methods can be applied - Gaussian elimination, LU - decomposition, Csanky's algorithm, *etc.* [2],[3].This is not of great importance for us and we shall consider this problem at the end of the paper.

We must point out that the proposed algorithm can be considered connection with those which are described in [5] and [8]. In [5] the autho have used the idea to divide the problem of dimension n into two parts, each dimension $n/2$. The same idea is used in our algorithm but, due to the speci structure of the matrices, we have avoided the necessity of inverting matric of dimension greater than k (k is the dimension of blocks). In [8] Frobeni formulae are applied to invert a dense matrix by using the "bordering method Only matrices of dimension 4 are inverted, but the algorithm is of linear orde In our case, due to the block tridiagonal structure of the matrix to be invertе the order of the algorithm is $\log n$.

2. Basic formulae and basic notations

Let $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ be a block matrix. When the Frobenius for mulaeare applied, the elements of the inverse matrix $A^{-1} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$ ar given by:

$$
\begin{aligned}
C_{11} &= A_{11}^{-1} + A_{11}^{-1} A_{12} D^{-1} A_{21} A_{11}^{-1} \\
C_{12} &= -A_{11}^{-1} A_{12} D^{-1} \\
C_{21} &= -D^{-1} A_{21} A_{11}^{-1} \\
C_{22} &= D^{-1}.
\end{aligned} \tag{2}
$$

where $D = A_{22} - A_{21} A_{11}^{-1} A_{12}$.

We shall apply a version of (2), namely

$$
\begin{aligned}
C_{11} &= (A_{11} - A_{12} A_{22}^{-1} A_{21})^{-1} \\
C_{21} &= -A_{22}^{-1} A_{21} C_{11} = -A_{22}^{-1} A_{21} (A_{11} - A_{12} A_{22}^{-1} A_{21})^{-1} \\
C_{22} &= -(A_{22} - A_{21} A_{11}^{-1} A_{12})^{-1} \\
C_{12} &= -A_{11}^{-1} A_{12} C_{22} = -A_{11}^{-1} A_{12} (A_{22} - A_{21} A_{11}^{-1} A_{12})^{-1}.
\end{aligned} \tag{3}
$$

The following notations will be used hereafter

$$
M = A_{11}^{-1} A_{12}, \qquad N = A_{22}^{-1} A_{21}. \tag{4}
$$

Then

$$
C_{11} = [A_{11}(I - A_{11}^{-1} A_{12} A_{22}^{-1} A_{21})]^{-1} = (I - MN)^{-1} A_{11}^{-1}
$$

and

$$C_{22} = [A_{22}(I - A_{22}^{-1}A_{21}A_{11}^{-1}A_{12})]^{-1} = (I - NM)^{-1}A_{22}^{-1}.$$

Finally:

$$(5) \qquad \begin{aligned} C_{11} &= (I - MN)^{-1}A_{11}^{-1} \\ C_1 &= -NC_{11} = -N(I - MN)^{-1}A_{11}^{-1} \\ C &= (I - NM)^{-1}A^{-1} \\ C_1 &= -MC = -M(I - NM)^{-1}A^{-1}. \end{aligned}$$

3. Description of the algorithm

At the first step of the algorithm the inverses of

$$(6) \qquad A_m^1 = \begin{bmatrix} A_{m-1\ m-1} & A_{m-1\ m} \\ A_{m\ m-1} & A_{m\ m} \end{bmatrix}$$

for $m = 1, 2, \ldots, n/2$ are computed (in parallel, if it is possible).Here A_m^1, $m = 1, 2, \ldots, n/2$ are the block 2×2 matrices on the main diagonal of A which blocks are $k \times k$ matrices. At this step formulae (2) can be applied. For the inverses of the $k\times k$ matrices $A_{m-1\ m-1}$ and $D_m = A_{m\ m} - A_{m\ m-1}(A_{m-1\ m-1})^{-1}A_{m-1\ m}$ to be find, the auxiliary method (Gaussian elimination, LU - decomposition, *etc.*) is applied.

In the case when the initial matrix A is not necessary to be kept then the derived matrices $(A_m^1)^{-1}$, $m = 1, 2, \ldots, n/2$ can be disposed upon the matrices A_m^1, $m = 1, 2, \ldots, n/2$.

At each of the next $(l - 1)$ steps, formulae (4) and (5) are applied. Let us describe the step with ordinary number S ($S \in \{2, 3, \ldots, l\}$).

At the S-th step the inverse matrices of the diagonal 2×2 block matrices

$$(7) \qquad A_m^s = \begin{bmatrix} A_{m-1\ m-1}^{s-1} & A_{m-1\ m}^{s-1} \\ A_{m\ m-1}^{s-1} & A_{m\ m}^{s-1} \end{bmatrix}$$

for $m = 1, 2, \ldots, n/2^s$ are computed. Here A_{ij}^{s-1} is $2^{s-1}k\times2^{s-1}k$ matrix ($2^{s-1}\times 2^{s-1}$ block matrix which blocks are $k \times k$ matrices). At the previous ($(S-1)$-th) step of the algorithm the inverse matrices $(A_{m-1\ m-1}^{s-1})^{-1}$ and $(A_{m\ m}^{s-1})^{-1}$ are obtained and as was said before,if the initial matrix A is not necessary to be kept, on the main diagonal of A exactly these matrices are disposed. The other

2 blocks of A^s_m have specific structure (non-zero $k \times k$ submatrices are marked with $\star$ and zero ones - with 0), namely:

$$(8)\quad A^{s-1}_{2m-1\ 2m} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & 0 \\ \star & 0 & \dots & 0 \end{bmatrix}, \qquad A^{s-1}_{2m\ 2m-1} = \begin{bmatrix} 0 & 0 & \dots & \star \\ 0 & 0 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

For different $m = 1, 2, \dots, n/2^s$ operations are independent and can be done simultaneously. We compute

$$(9)\qquad \begin{aligned} M^s_m &= (A^{s-1}_{m-1\ m-1})^{-1} A^{s-1}_{m-1\ m} \\ N^s_m &= (A^{s-1}_{m\ m})^{-1} A^{s-1}_{m\ m-1}\,. \end{aligned}$$

Matrices $(A^{s-1}_{m-1\ m-1})^{-1}$ and $(A^{s-1}_{m\ m})^{-1}$ are known from the previous ($(S-1)$-th) step of the algorithm. From (8) it is clear that M^s_m and N^s_m have the following structure ($\star$ means non-zero $k \times k$ block, 0 means zero $k \times k$ block):

$$(10)\qquad M^s_m = \begin{bmatrix} \star & 0 & \dots & 0 \\ \star & 0 & \dots & 0 \\ \vdots & & & \\ \star & 0 & \dots & 0 \end{bmatrix}; \qquad N^s_m = \begin{bmatrix} 0 & \dots & 0 & \star \\ 0 & \dots & 0 & \star \\ \vdots & & & \\ 0 & \dots & 0 & \star \end{bmatrix}.$$

Hence

$$(11)\quad M^s_m N^s_m = \begin{bmatrix} 0 & \dots & 0 & \star \\ 0 & \dots & 0 & \star \\ \vdots & & & \\ 0 & \dots & 0 & \star \end{bmatrix}; \qquad N^s_m M^s_m = \begin{bmatrix} \star & 0 & \dots & 0 \\ \star & 0 & \dots & 0 \\ \vdots & & & \\ \star & 0 & \dots & 0 \end{bmatrix}$$

and

$$(12)\ I_{2^{s-1}} - M^s_m N^s_m = \begin{bmatrix} I_k\,0 \dots 0\,\star \\ 0\,I_k \dots 0\,\star \\ \vdots \\ 0\,0 \dots I_k\,\star \\ 0\,0 \dots 0\,\star \end{bmatrix}; \ I_{2^{s-1}} - N^s_m M^s_m = \begin{bmatrix} \star\,0 \dots 0\,0 \\ \star\,I_k \dots 0\,0 \\ \vdots \\ \star\,0 \dots I_k\,0 \\ \star\,0 \dots 0\,I_k \end{bmatrix}$$

In (12) unity matrices $r \times r$ are marked with I_r. Each of the formulae (9) consists of 2^{s-1} independent multiplications of $k \times k$ matrices. If we have sufficient resource of processors this multiplications can be done simultaneously. The cost for performing the formulae (11) is the same.

For the formulae (5) to be applied it is necessary to compute

$$(I_{2^{s-1}} - M_m^s N_m^s)^{-1} \text{ and } (I_{2^{s-1}} - N_m^s M_m^s)^{-1} . \tag{13}$$

In view of the specific structure of these matrices (see (12)) and by using the fact that:

$$\begin{bmatrix} I_k & 0 & 0\dots 0 & B_1 \\ 0 & I_k & 0\dots 0 & B_2 \\ \vdots & & & \\ 0 & 0 & 0\dots I_k & B_{q-1} \\ 0 & 0 & 0\dots 0 & I_k - B_q \end{bmatrix}^{-1} = \begin{bmatrix} I_k & 0 & 0\dots 0 & -B_1(I_k - B_q)^{-1} \\ 0 & I_k & 0\dots 0 & -B_2(I_k - B_q)^{-1} \\ \vdots & & & \\ 0 & 0 & 0\dots I_k & -B_{q-1}(I_k - B_q)^{-1} \\ 0 & 0 & 0\dots 0 & (I_k - B_q)^{-1} \end{bmatrix}$$

(14)

$$\begin{bmatrix} I_k - B_1 & 0 & 0\dots 0 & 0 \\ B_2 & I_k & 0\dots 0 & 0 \\ \vdots & & & \\ B_{q-1} & 0 & 0\dots I_k & 0 \\ B_q & 0 & 0\dots 0 & I_k \end{bmatrix}^{-1} = \begin{bmatrix} (I_k - B_1)^{-1} & 0 & 0\dots 0 & 0 \\ -B_2(I_k - B_1)^{-1} & I_k & 0\dots 0 & 0 \\ \vdots & & & \\ -B_{q-1}(I_k - B_1)^{-1} & 0 & 0\dots I_k & 0 \\ -B_q(I_k - B_1)^{-1} & 0 & 0\dots 0 & I_k \end{bmatrix}$$

(B_i are $k \times k$ matrices, $i = 1, 2, \dots, q$, $1 \le q \le n/2$) it is clear that the cost of each of computations (13) is equal to the time which is necessary to invert one $k \times k$ matrix and to perform $2^{s-1} - 1$ independent multiplications of two $k \times k$ matrices.

For final computation of the elements of $(A_m^s)^{-1}$ by using formulae (5) it is necessary to compute expressions of following type:

$$\begin{aligned} P_m^s &= (I_{2^{s-1}} - M_m^s N_m^s)^{-1} \, (A_{m-1\; m-1}^{s-1})^{-1} \\ Q_m^s &= (I_{2^{s-1}} - N_m^s M_m^s)^{-1} \, (A_{m\; m}^{s-1})^{-1} \end{aligned} \tag{15}$$

for $m = 1, 2, \dots, n/2^s$ and

$$N_m^s P_m^s \text{ and } M_m^s Q_m^s \tag{16}$$

for $m = 1, 2, \dots, n/2^s$.

Each of the formulae (15) consists of 2^{2s-2} independent expressions of the type $CD + F$, where C, D and F are $k \times k$ matrices.These expressions can be compute simultaneously.

Finally, each of the formulae (16) consists of 2^{s-1} independent multiplications of $k \times k$ matrices, which can be done simultaneously, too.

4. Estimate of the time and resources which are necessary for realization of the algorithm

The described algorithm allows us to perform operations inparallel on different levels depending on available resource of processors. This problem will be considered in details below. Let us note the time which is necessary to perform the basic used operations on $k \times k$ matrices with the assistance of p processors in following way:

- $R(k,p)$ for a matrix to be inverted
- $M(k,p)$ for multiplying 2 matrices
- $S(k,p)$ for adding 2 matrices

In particular, $S(k,p)$ is also the time which is necessary to invert the sign of $k \times k$ matrix, i.e. for the operation $-D$ to be done,where D is a $k \times k$ matrix.

At the first step we obtain (in parallel, if there is sufficient number of processors) the inverses of $n/2$ matrices by using formulae (2). Each of this inversions needs

$$t_1 = 2\,R(k,p) + 6\,M(k,p) + 4\,S(k,p) \tag{17}$$

units of time.

At the S-th step of the algorithm ($S = 2, 3, \ldots, l$) we obtain (in parallel, if there is sufficient number of processors) the inverses of $n/2^s$ matrices by using in consecutive order formulae (9), (11), (12), (13), (15) and (16). Each of this inversions needs time which is given below. To estimate the time and resources of processors which we need to perform the described operations we shall use the well-known estimates:

$$\begin{aligned} M(k,k^3) &= O(\log k) \\ S(k,k^2) &= O(1) \text{ or } S(k,k) = O(\;pgk) \end{aligned} \tag{18}$$

and, for example

$$R(k,k^4) = O(\log^2 k). \tag{19}$$

Here and hereafter, as it is usual, $\log a$ means $\log_2 a$. The estimate (19) holds,when the Csanky's algorithm is applied to invert a $k \times k$ matrix. If any other algorithm is applied, then we shall obtain different estimates for $R(k,p)$.

Formulae (9) require $t_s^1 = 2[2^{s-1}M(k,p)] = 2^s M(k,p)$ units of time. For all $n/2^s$ matrices, which are inverted at the S-th step, the operations (9) to be performed it needs $T_s^1 = nM(k,p)$ units of time. If this operations are done in parallel (i. e. if nk^3 processors are used) then the total time for performing (9) over all the $n/2^s$ matrices (in parallel) is equal to $O(\log k)$.

Thus at the stage of formulae (9) we obtain

(20)
$$\begin{aligned} T_s^1 &= n\,M(k,p) \\ &\quad \text{and with } nk^3 \text{ processors it takes} \\ O(\log k) &\quad \text{units of time.} \end{aligned}$$

In an analogous manner, when the formulae (11) are applied, we obtain the following values

(21)
$$\begin{aligned} T_s^2 &= n\,M(k,p) \\ &\quad \text{and with } nk^3 \text{ processors it takes} \\ &\quad O(\log k) \text{ units of time.} \end{aligned}$$

and for the formulae (12) to be performed

(22)
$$\begin{aligned} T_s^3 &= n\,S(k,p) \\ &\quad \text{and with } nk^2 \text{ processors } (nk \text{ processors}) \text{ it takes} \\ O(1) &\quad (\,O(\log k)\,) \text{ units of time.} \end{aligned}$$

The stage of the formulae (13) must be divided into 3 substages which can not be done simultaneously. The first substage consists of obtaining of the inverse matrices (denoted by $(I_k - B_q)^{-1}$ and $(I_k - B_1)^{-1}$ in the third section). For each of the $n/2^s$ matrices, which are work matrices at the S-th step, we must invert the corresponding two $k \times k$ matrices.

Hence we obtain

(23)
$$\begin{aligned} T_s^4 &= [n/2^{s-1}]\,R(k,p) \text{ and with } [nk^4/2^{s-1}] \text{ processors it takes} \\ &\quad O(\log k) \text{ units of time.} \end{aligned}$$

The second substage consists of matrix multiplications (denoted by $B_r(I_k - B_q)^{-1}$ and $B_r(I_k - B_1)^{-1}$ in the third section). For every of the $n/2^s$ matrices there are 2^{s-1} such multiplications. The corresponding values are:

$$T_s^5 = n\,M(k,p) \quad \text{and with } nk^3 \text{ processors it takes} \tag{24}$$
$$O(\log k) \quad \text{units of time.}$$

At the third substage inversions of sign are made. Here

$$T_s^6 = n\,S(k,p) \quad \text{and with } nk^2 \text{ processors } (nk \text{ processors}) \text{ it takes} \tag{25}$$
$$O(1) \quad (O(\log k)) \text{ units of time.}$$

For the operations (15) to be performed

$$t_s^7 + t_s^8 = 2\{2^{2s-2}[M(k,p) + S(k,p)]\} = 2^{2s-1}M(k,p) + 2^{2s-1}S(k,p)$$

units of time are needed for any of the $n/2^s$ matrices, which are inverted at the S-th step. Corresponding global values at this stage are

$$T_s^7 = 2^{s-1}n\,M(k,p) \text{ and with } 2^{s-1}nk^3 \text{ processors it needs } O(\log k) \text{ units of time.} \tag{26}$$

for the multiplications (denoted by CD in the third section).

Respectively

$$T_s^8 = 2^{s-1}n\,S(k,p) \quad \text{and with } 2^{s-1}nk^2 \text{ processors } (2^{s-1}nk \text{ processors}) \text{ it needs} \tag{27}$$
$$O(1) \quad (O(\log k)) \text{ units of time.}$$

for the additions (denoted by $[CD + F]$ in the third section).

At the end of the S-th step, formulae (16) are applied. At this last stage of the S-th step of the algorithm we obtain

$$T_s^9 = n\,M(k,p) \text{ and with } nk^3 \text{ processors it takes } O(\log k) \text{ units of time.} \tag{28}$$

We can obtain the global estimate now. As $S = 2, 3, \ldots, l$ and $l = \log n$, then it is clear that the total time for performing of the algorithm is

$$T = O(l \cdot \log^2 k) = O(\log n \cdot \log^2 k)$$

units of time when

$$P = \max\{n, k\} \cdot (nk^3)/2$$

processors are used.

References

1. Г. Корн, Т. Корн. Справочник по математике. Москва, Наука, 1973.
2. B.N. Parlett. The Symmetric Eigenvalue Problem. Englewood Clifs,N.Y., Prentice-Hall, 1980
3. J.H. Wilkinson, C.H. Reinsch. Handbook for Automatic Computation. Linear Algebra, vol. 2. New York, Springer-Verlag, 1971.
4. А. Андреев, Хр. Джиджев, Бл. Сендов, Н. Янев, Обзор по паралелно смятане, София, БАН, 1985.
5. R. Bevilacqua, F. Romani, G. Lotti. Parallel inversion of band matrices. *Comput. Artificial Inteligence*, **9**, 1990, 5, 493–501.
6. R. Bevilacqua, B. Codenotti, F. Romani. Parallel solution of block tridiagonal linear systems. *Linear Algebra Appl.*, **104**, 1988, 39–57.
7. W.D. Hoskins, G.E. McMaster. Properties of the inverses of a set of band matrices. *Linear and Multilinear Algebra*, **5**, 1987, 183–196.
8. E. Francomano, A. Pecorella, A. Tortorici Macaluso. Parallel experience on the inverse matrix computation, *Parallel Computing*, **17**, 1991, 907–912.

Center of Informatics and Computer Technology
Akad. G. Bonchev str. 25 A
Sofia 1113
BULGARIA

Received 29.12.1992