A Fully Parallel Symmetric Matrix Transformation

Ayse Kiper

Department of Computer Engineering, Middle East Technical University
Ankara-Turkey

Abstract. A parallel algorithm for the transformation LAL^T of an nxn symmetric A through a nonsingular triangular L is presented. Speedup and efficiency successively are of $O((n^3+n^2)/(1+\lceil\log n\rceil))$ and $O(1/(1+\lceil\log n\rceil))$ with $O(1+\lceil\log n\rceil)$ parallel time.

1 Introduction

Similarity and orthogonal transformations have great significance in eigenvalue problems since the efficient numerical methods are based on these transformations. In this work we describe a parallel algorithm to generate the lower triangular part of the transformation

$$S = LAL^T \tag{1}$$

where A is an $n \times n$ symmetric and L is a non-singular lower triangular matrices. Method is heavily based on elementwise multiplication and addition of auxiliary matrices. Parallel assignment operations are also used but the time consumed for these is negligible compared to that required for the complete algorithm (hereafter referred to as PST). PST is particularly suitable for shared memory systems.

2 Parallel Symmetric Transformation Algorithm (PST)

The matrices A, L and S in (1) can be defined as

$$A = \left[a_{ij}\right],\ i = 1,2,\ldots,n\ ;\ j = 1,2,\ldots,n$$

$$L = \left[l_{ij}\right],\ j = 1,2,\ldots,i\ ;\ i = 1,2,\ldots,n\ \ and\ \ l_{ij} = 0 \quad for\ \ j \rangle i$$

$$S = \left[s_{ij}\right],\ i = 1,2,\ldots,n\ ;\ j = 1,2,\ldots,n.$$

Let the lower triangular part of S is defined by

$$S' = \left[\tilde{s}_1, \tilde{s}_2, \ldots\ldots, \tilde{s}_n\right] \tag{2}$$

where $\tilde{s}_k$ denotes the kth column of S' and can be expressed as

$$\tilde{s}_k = (\tilde{o}^T, \tilde{v}_k{}^T)^T,\ k = 1,2,\ldots,n$$

where $\tilde{o}$ is a null vector of size $(k-1)$ and $\tilde{v}_k$ is a $(n-k+1)$ vector whose *ith* element is denoted by v_{ki} and defined as

$$v_{ki} = \sum_{r=1}^{k} l_{kr} p_{ir}, \quad i = k, k+1, k+2, \ldots, n \tag{3}$$

where p_{ir} represents the sum of products

$$p_{ir} = \sum_{s=1}^{i} l_{is} a_{sr}, \quad i = r, r+1, r+2, \ldots, n. \tag{4}$$

PST algorithm can be described in the following six main steps:

Step 1.*(Matrix construction by parallel assignment)* Auxiliary matrices C and D of size $(n(n+1)/2) \times n$ are constructed by parallel assignment as

$$C = \begin{bmatrix} C_1^T & C_2^T & . & . & . & C_{n-1}^T & C_n^T \end{bmatrix}^T$$
$$D = \begin{bmatrix} D_1^T & D_2^T & . & . & . & D_{n-1}^T & D_n^T \end{bmatrix}^T$$

where C_i and D_i are submatrices of size $(n-i+1) \times n$ and obtained from the elements of L and A successively as

$$C_i = \begin{bmatrix} l_{i1} & \cdot & \cdot & \cdot & \cdot & l_{ii} & & & & \\ l_{i+1,1} & \cdot & \cdot & \cdot & \cdot & l_{i+1,i} & l_{i+1,i+1} & & 0 & \\ \cdot & & & & & \cdot & \cdot & \cdot & & \\ \cdot & & & & & \cdot & \cdot & & \cdot & \\ \cdot & & & & & \cdot & \cdot & & \cdot & \\ l_{n1} & \cdot & \cdot & \cdot & \cdot & l_{ni} & l_{n,i+1} & \cdot & \cdot & \cdot & l_{nn} \end{bmatrix}$$

$$D_i = \begin{bmatrix} a_{i1} & a_{i2} & \cdot & \cdot & \cdot & a_{ii} & & & & \\ a_{i1} & a_{i2} & \cdot & \cdot & \cdot & a_{ii} & a_{i,i+1} & & 0 & \\ \cdot & \cdot & & & & \cdot & & \cdot & & \\ \cdot & \cdot & & & & \cdot & & & \cdot & \\ \cdot & \cdot & & & & \cdot & & & \cdot & \\ a_{i1} & a_{i2} & \cdot & \cdot & \cdot & a_{ii} & a_{i,i+1} & \cdot & \cdot & \cdot & a_{in} \end{bmatrix}.$$

Step 2.*(Elementwise matrix-matrix multiplication)* Termwise multiplication of C and D yields

$$M = C * D$$

whose elements are the product terms $l_{is}a_{sr}(i = r, r+1, \ldots, n \; ; \; r = 1, 2, \ldots, n)$ in (4).

Step 3.*(Eleentwise vector-vector additions by fan-in) Columns of*

$$M = \left[\tilde{m}_1, \tilde{m}_2, \ldots\ldots, \tilde{m}_n\right]$$

are added in parallel by fan-in to yield

$$\tilde{p} = \sum_{i=1}^{n} \tilde{m}_i$$

in $\lceil \log n \rceil$ steps. Elements of $\tilde{p}$ are the sum of product terms p_{ir} given in (4). $\tilde{p}$ can be partitioned as

$$\tilde{p} = (\tilde{p}_1^T, \tilde{p}_2^T, \ldots\ldots, \tilde{p}_n^T)^T$$

with

$$\tilde{p}_r = (p_{ir}), \quad i = r, r+1, \ldots, n \; ; \; r = 1, 2, \ldots, n$$

Step 4.*(Matrix construction by parallel assignment)* E and F are of size $n \times (n(n+1)/2)$ and are constructed by assignment. Columns of

$$E = \left[\tilde{e}_1, \tilde{e}_2, \ldots\ldots, \tilde{e}_{n(n+1)/2}\right]$$

and

$$F = \left[\tilde{f}_1, \tilde{f}_2, \ldots\ldots, \tilde{f}_{n(n+1)/2}\right]$$

are obtained respectively by

$$\tilde{e}_{(i(i-1)/2)+r} \leftarrow (\tilde{o}_i^T, \tilde{p}_r^T)^T, \quad i = r, r+1, \ldots, n \; ; \; r = 1, 2, \ldots, n \tag{5}$$

and

$$\tilde{f}_{(i(i-1)/2)+r} \leftarrow (\tilde{o}_i^T, \tilde{l}_{ir}^T)^T, \quad r = 1, 2, \ldots, i \; ; \; i = 1, 2, \ldots, n. \tag{6}$$

In (5) and (6) $\tilde{o}$ denotes a null vector of size $(i-1)$ and in (6) $\tilde{l}_{ir}$ denotes a $(n-i+1)$ vector of equal elements defined as $\tilde{l}_{ir} = (l_{ir})$.

Step 5.(Elementwise matrix-matrix multiplication) Elementwise multiplication of E and F yields

$$R = E * F$$

whose elements are the sum of product terms p_{ir} when multiplied by appropriate constants l_{kr} as described in (3) and (4).

Step 6.(Elementwise vector-vector additions by fan-in) The appropriate columns of

$$R = \left[\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_{n(n+1)/2}\right]$$

are added by fan-in method using the formula

$$\tilde{s}_k = \sum_{i=1}^{k} \tilde{r}_{(k(k-1)/2)+i}, \quad k = 1, 2, \ldots, n$$

to yield the matrix S'. $\tilde{s}_n$ is obtained in $\lceil \log n \rceil$steps which determines the parallel time for Step 6.

3 Performance of PST

PST algorithm requires 2 multiplication steps which are independent of problem size and $2\lceil \log n \rceil$ addition steps. Memory or data alignment costs are negligible. It can easily be adapted to multiprocessor systems having limited number of processors with a minor effort by splitting the objects into subobjects. PST is equally suitable and adaptable for vector processors. These make the algorithm general.

Performance parameters speedup (T_1 / T_p) and efficiency $(S_p / p_{\max})$ are

$$S_p = \frac{n(3n^2 + 2n + 1)}{6(1 + \lceil \log n \rceil)}$$

and

$$E_p = \frac{3n^2 + 2n + 1}{3n(n+1)(1 + \lceil \log n \rceil)},$$

while the maximum number of processors used is

$$p_{\max} = n^2 (n+1) / 2 .$$

PST is an addition dominant algorithm particularly for large size of problems.