With parallel algorithm solved tridiagonal system

Liang-Jung Shen
Department of Electronic Engineering
Nan Jeon Institute of Technology
Email: altus@mail.njtc.edu.tw

***Abstract* —In recent years there has been increasing interest in the study and research of parallel algorithms. Parallel programming uses multiple computers or distributed computing system to solve the problem at simultaneous and computational speed is faster than a single computer. This paper utilized divide-and-conquer concept to present brand-new method of parallel process in order to solve tridiagonal system. This skill is called "Recursive-Doubling-Elimination" algorithm. This approach allows a tridiagonal system to be solved in O($\log_2 n$) times on a barrel shifter network of O($n/\log_2 n$) processors. It is also cost-optimal in the sense that the product of the execution time and the number of processors is minimal.**

***Index Terms*—tridiagonal system, Barrel shifter, Recursive-Doubling-Elimination.**

I. INTORDUCTION

Solving tridiagonal system has the following several kinds of important application on science

1. Solving of differential equation or partial differential equations for example FACR (Fourier Analysis-Cyclic Reduction) Successive Line. [1][2]

2. Cubic polynomials spline method. [3][4]

3. Using finite difference method will solve orthogonal curvilinear coordinate system to get coupled finite difference equations. [5]

A tridiagonal matrix has nonzero elements that only in the main diagonal and it neighboring elements. A diagonal is below main diagonal and another one is above main diagonal, as show in Figure 1. For example a tridiagonal system for n unknowns may be written as:

$$c_i x_{i-1} + a_i x_i + b_i x_{i+1} = r_i$$
$$where \quad 1 \le i \le n, \quad i, n \in N$$

The ones that because exist very high among data are interdependent, it is very difficult to use parallel method to deal with the matrix. If use the general linear algebra method to solve, no matter how many processors are used, complexity of time needs for O(n). Because of the above-mentioned reasons, this is really a difficult parallel problem.

$$\mathbf{A}\boldsymbol{x} \equiv \begin{bmatrix} a_1 & b_1 & & & & & & \\ c_2 & a_2 & b_2 & & & & & \\ & c_3 & a_3 & b_3 & & & & \\ & & \cdots & \cdots & \cdots & & & \\ & & & \cdots & \cdots & \cdots & & \\ & & & & \cdots & \cdots & \cdots & \\ & & & & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & & & & & c_n & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ .. \\ .. \\ .. \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ .. \\ .. \\ .. \\ r_{n-1} \\ r_n \end{bmatrix} \equiv \mathbf{r}$$

Figure 1 Linear tridiagonal system

Literature has proposed a lot of method with parallel calculations for tridiagonal system in the past, as shown in Table 1. This paper presents the algorithm that is applied to solve a tridiagonal system. Based on the recursive doubling elimination method is parallel computed by a barrel shifter network with O($n/\log_2 n$) processors taking O($\log_2 n$) times.

Table 1 Parallel algorithm solves performance of tridiagonal system.

Method	Time Complexity	Processor Complexity
LU decomposition [6]	$O(\log_2 n)$	$O(n)$
QR decomposition [7]	$O(\log_2 n)$	$O(n)$
cyclic reduction [8]	$O(\log_2 n)$	$O(n)$
divide and conquer[9]	$O(\sqrt{n})$	$O(\sqrt{n})$
Continued fraction [10]	$O(\log_2 n)$	$O(n/\log_2 n)$

II. RECURSIVE DOUBLING ELIMINATION METHOD

This method emphasized the skill operation of matrix element of simultaneous. If tridiagonal matrix size is n by n (coefficient matrix as shown in Figure 2), calculation can be divided into upper and lower parts parallel process and simultaneous calculation every part, then the result can be finished in ($\log_2 n$) steps.

Regarding main diagonal as the centre will carry on procedure of elimination to upper and lower element base on matrix elementary row operation. For example ith row, we multiply the row through by a multiple k_{di} ($k_{di} = -c_i + 1/a_i$) and elimination element below main diagonal. The same situation multiply the row through by a multiple k_{ui} ($k_{ui} = -b_{i-1} + 1/a_i$) and elimination element above main diagonal, as shown in Figure 3. By the same method, its result that does operation once again is as shown in Figure 4.

$$
\begin{bmatrix}
a_1 & b_1 & & & & & & & \\
c_2 & a_2 & b_2 & & & & & & \\
 & \ddots & \ddots & \ddots & & & & & \\
 & & c_{i-1} & a_{i-1} & b_{i-1} & & & & \\
 & & & c_i & a_i & b_i & & & \\
 & & & & c_{i+1} & a_{i+1} & & & \\
 & & & & & \ddots & \ddots & \ddots & \\
 & & & & & & c_{n-1} & a_{n-1} & b_{n-1} \\
 & & & & & & & c_n & a_n
\end{bmatrix}
$$

Figure 2 coefficient matrix

$$
\begin{bmatrix}
a_1 & 0 & b_1 & & & & & & & & & \\
0 & a_2 & 0 & b_2 & & & & & & & & \\
c_3 & 0 & a_3 & 0 & b_3 & & & & & & & \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & & & & & & \\
 & & c_{i-2} & 0 & a_{i-2} & 0 & b_{i-2} & & & & & \\
 & & & c_{i-1} & 0 & a_{i-1} & 0 & b_{i-1} & & & & \\
 & & & & c_i & 0 & a_i & 0 & b_i & & & \\
 & & & & & c_{i+1} & 0 & a_{i+1} & 0 & b_{i+1} & & \\
 & & & & & & c_{i+2} & 0 & a_{i+2} & 0 & b_{i+2} & \\
 & & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots \\
 & & & & & & & & c_{n-2} & 0 & a_{n-2} & 0 & b_{n-2} \\
 & & & & & & & & & c_{n-1} & 0 & a_{n-1} & 0 \\
 & & & & & & & & & & c_n & 0 & a_n
\end{bmatrix}
$$

Figure 3 first elimination

$$
\begin{bmatrix}
a_1 & 0 & 0 & 0 & b_1 & & & & & & & & \\
0 & a_2 & 0 & 0 & 0 & b_2 & & & & & & & \\
0 & 0 & a_3 & 0 & 0 & 0 & b_3 & & & & & & \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & & & & \\
c_{i-2} & 0 & 0 & 0 & a_{i-2} & 0 & 0 & 0 & b_{i-2} & & & & \\
 & c_{i-1} & 0 & 0 & 0 & a_{i-1} & 0 & 0 & 0 & b_{i-1} & & & \\
 & & c_i & 0 & 0 & 0 & a_i & 0 & 0 & 0 & b_i & & \\
 & & & c_{i+1} & 0 & 0 & 0 & a_{i+1} & 0 & 0 & 0 & b_{i+1} & \\
 & & & & c_{i+2} & 0 & 0 & 0 & a_{i+2} & 0 & 0 & 0 & b_{i+2} \\
 & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
 & & & & & & c_{n-2} & 0 & 0 & 0 & a_{n-2} & 0 & 0 \\
 & & & & & & & c_{n-1} & 0 & 0 & 0 & a_{n-1} & 0 \\
 & & & & & & & & c_n & 0 & 0 & 0 & a_n
\end{bmatrix}
$$

Figure 4 the second elimination

So matrix is after operation of m ($m = \log_2 n$) times, all value will be zero besides main diagonal element. That is to say this system has already been answered.

As to in the above mentioned elementary row operation course. The constant term(r) in the equation need to do relative operation too. If according to matrix of local:

$$
\begin{bmatrix}
c_u & a_u & b_u & & \\
 & c_i & a_i & b_i & \\
 & & c_d & a_d & b_d
\end{bmatrix}
$$

Assumption matrix size is n by n. Let $k_d = -(c_i / a_u)$, and $k_u = -(b_i / a_d)$, uth row is multiplied by k_d together, dth row is multiplied by k_u together, and then with ith 3 pieces of equation preface summation.

1. $a_i = a_i + b_u\left(\dfrac{-c_i}{a_u}\right) + c_d\left(\dfrac{-b_i}{a_d}\right) + 0$

2. $r_i = r \quad + r_u\left(\dfrac{-c_i}{a_u}\right) + r_d\left(\dfrac{-b_i}{a_d}\right) + 0$

3. $c_i = c_u\left(\dfrac{-c_i}{a_u}\right) + 0 \qquad b_i = b_u\left(\dfrac{-b_i}{a_d}\right) + 0$

$$u = i - 2^j \qquad , d = i + 2^j$$

$0 \le i \pm 2^j \le m \qquad i, j \in 0,1,2\ \ldots$ matrix size is m

III. Barrel Shifter

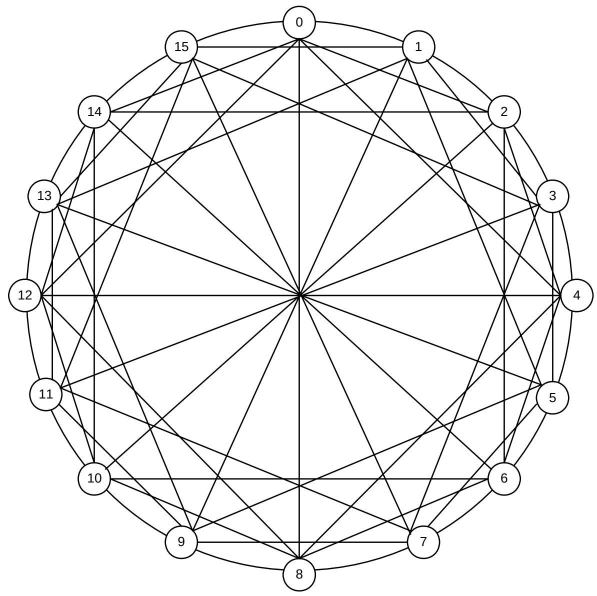

Figure 5 Barrel shifter

As shown in Fig.5 for a network of N=16 nodes, the *barrel shifter* is obtained from the ring by adding extra links from each node to those nodes having a distance equal to an integer power of 2. This implies that node i is connected to node j if $|j-i| = 2^r$ for some r=0,1,2…,n-1 and the network size is $N=2^n$. Barrel shifter has a node degree of d=2n-1 and a diameter D=n/2. Communication way of processor is based on the following transmission function:

$$B_{+i}(j) = (j + 2^i)\ (\bmod N)$$
$$B_{-i}(j) = (j - 2^i)\ (\bmod N)$$
$$0 \le j \le N-1,\ 0 \le i \le n-1,\ n = \log_2 N$$

IV. Recursive Doubling Elimination Algorithm

The algorithm adopts divide-and-conquer as the structure. In this method, the problem is divided into smaller subproblems. The solutions of these subproblems are found first. Then these are processed further, to get the solution of the complete problem. The algorithm is divided into three steps:

1. If matrix size is n by n, partition into $(n/\log_2 n)$ blocks and each one has $(\log_2 n)$ rows. Each block distribute to a processor responsible for calculating, as shown in Fig. 6. Simultaneity performs elementary row operation of each block, as shown in Fig. 7.

2. Take out the last row of every block, and make up smaller tridiagonal matrix, namely *eigenmatrix,* as show in Figure 8. And solving the eigenmatrix utilizes recursive doubling elimination method.

3. Solution calculated by *eigenmatrix,* backward substitution it in each processor. So can find every row of original matrix all only has one variable left. Solving job of original matrix just need utilize simple division, as shown in Figure 9.

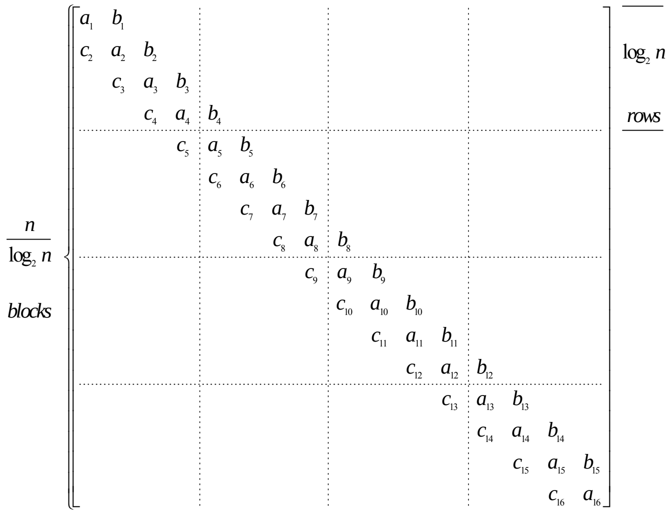

Figure 6 partition matrix

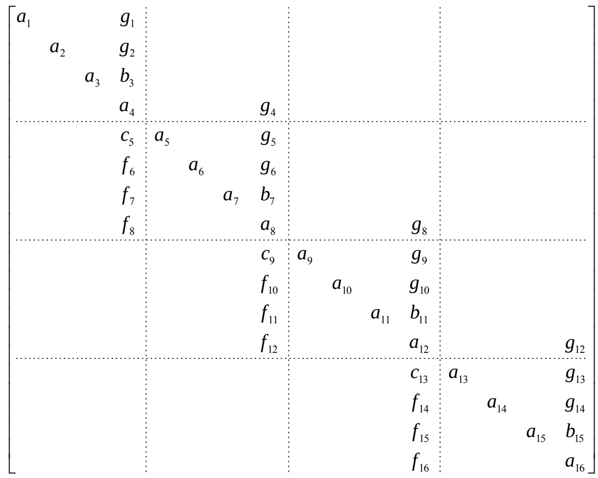
Figure 7 Simultaneity performs elementary row operation of each block

$$\begin{bmatrix} a_4 & g_4 & 0 & 0 \\ f_8 & a_8 & g_8 & 0 \\ 0 & f_{12} & a_{12} & g_{12} \\ 0 & 0 & f_{16} & a_{16} \end{bmatrix} \times \begin{bmatrix} x_4 \\ x_8 \\ x_{12} \\ x_{16} \end{bmatrix} = \begin{bmatrix} r_4 \\ r_8 \\ r_{12} \\ r_{16} \end{bmatrix}$$

Figure 8 eigenmatrix

$$
\begin{bmatrix}
a_1 & & & g_1 & & & & & & & & & & & & \\
& a_2 & & g_2 & & & & & & & & & & & & \\
& & a_3 & b_3 & & & & & & & & & & & & \\
& & & a_4 & & & & 0 & & & & & & & & \\
& & & c_5 & a_5 & & & g_5 & & & & & & & & \\
& & & f_6 & & a_6 & & g_6 & & & & & & & & \\
& & & f_7 & & & a_7 & b_7 & & & & & & & & \\
& & & 0 & & & & a_8 & & & & 0 & & & & \\
& & & & & & & c_9 & a_9 & & & g_9 & & & & \\
& & & & & & & f_{10} & & a_{10} & & g_{10} & & & & \\
& & & & & & & f_{11} & & & a_{11} & b_{11} & & & & \\
& & & & & & & 0 & & & & a_{12} & & & & 0 \\
& & & & & & & & & & & c_{13} & a_{13} & & & g_{13} \\
& & & & & & & & & & & f_{14} & & a_{14} & & g_{14} \\
& & & & & & & & & & & f_{15} & & & a_{15} & b_{15} \\
& & & & & & & & & & & 0 & & & & a_{16}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \mathbf{x}_4 \\ x_5 \\ x_6 \\ x_7 \\ \mathbf{x}_8 \\ x_9 \\ x_{10} \\ x_{11} \\ \mathbf{x}_{12} \\ x_{13} \\ x_{14} \\ x_{15} \\ \mathbf{x}_{16} \end{bmatrix}
=
\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \\ r_8 \\ r_9 \\ r_{10} \\ r_{11} \\ r_{12} \\ r_{13} \\ r_{14} \\ r_{15} \\ r_{16} \end{bmatrix}
$$

Figure 9 backward substitution

V. Complexity Analysis

Define parallel algorithm cost analysis as the product of execution time and processor quantity, as below:

$$c(n) = t(n) \times p(n)$$

Suppose the matrix size is n by n, based on the principle of partition matrix, so need $O(n/\log_2 n)$ processors altogether. That is to say the processor complexity is $O(n/\log_2 n)$.

Estimate the time complexity of algorithm and can be divided into three components:

1. Partition: Each block has $(\log_2 n)$ rows after partition matrix. Time complexity at this stage is :

$$O_1 = O(\log_2 n)$$

2. Recursive doubling elimination: Because matrix partition into $(n/\log_2 n)$ blocks and calculation with method of recursive doubling elimination, hence time complexity at the second stage is n:

$$O_2 = O\left(\log\left(\frac{n}{\log_2 n}\right)\right)$$

3. Backward substitution: Each block has $(\log_2 n)$ rows, so time complexity is:

$$O_3 = O(\log_2 n)$$

Whole time complexity is:

$$O_t = O_1 + O_2 + O_3$$

$$O_t = O(\log_2 n) + O\left(\log\left(\frac{n}{\log_2 n}\right)\right) + O(\log_2 n)$$
$$= O(\log_2 n)$$

And algorithm cost is:

$$\operatorname{cos} t(n) = O(\log_2 n) \times O\left(\frac{n}{\log_2 n}\right) = O(n)$$

VI. Conclusion

The algorithm is based on "divide-and-conquer" principle to present "recursive doubling elimination" method and to deal with tridiagonal system problem on the structure of barrel shifter. In cost analysis, utilize $O(n/\log_2 n)$ processors to finish within $O(\log_2 n)$ times. Its cost equals to $O(n)$, this is the best solution at present.

REFERENCE

[1] R. F. Boisvert, “Algorithms for special tridiagonal systems,” *SIAM J. Scientific Statist. Computat.*, vol.12, no.2, 1991, pp. 423-442.

[2] S. L. Johnsson, “Solving tridiagonal systems on ensemble architectures,” *SIAM J. Scientific Statist. Compt.*, vol.8, no.3, 1987, pp.354-392.

[3] H. Spath, *Spline Algorithms for Curves and Surfaces*. Utilitas Mathematica, 1974.

[4] Kuo-Liang Chung and Ferng-Ching Lin, “A cost-optimal parallel algorithm for B-spline surface fitting,” *Graphical models and image processing*, vol.53, no.6, November, 1991, pp. 601-605.

[5] Larry L. Schumaker, *Spline Function Basic Theoy*. JOHN WILEY & SONS, 1981.

[6] H. S. Stone, “Parallel tridiagonal equation solvers,” *ACM Trans. Math Software*, vol.1, no.4, 1975, pp. 289-307.

[7] A. H. Sameh and D. J. Kuck, “A parallel QR algorithm for symmetric tridiagonal matrices”, *IEEE Trans. Comp.* vol. C-26(2), 1977, pp.147-155.

[8] R. W. Hockney. “A first direct solution of Poisson’s equation using fourier analysis,” *Journal of ACM* vol.12, 1965, pp. 95-113.

[9] H. H. Wang. “A parallel method for tridiagonal equations,” *ACM Trans. Math. Software* vol.7, 1981, pp170-183.

[10] Ferng-Ching Lin and Kuo-Liang Chung. “A cost-optimal parallel tridiagonal system solver,” *Parallel Computing*, vol.15, 1990, pp. 189-199.