병렬계산용 TTLS 알고리즘의 최적운용환경

김형중° 김용준 이장규
서울대학교 제어계측공학과

Optimized Operational Environment for Parallel TTLS Solver

H. J. Kim, Y. J. Kim and J. G. Lee
Dept. Control & Instrumentation Engineering, SNU

Abstract. A new tridiagonal Toeplitz linear system (TTLS) solver is proposed. The solver decomposes a strictly diagonally dominant TTLS equation into a number of subsystems using a limit convergent of an analytic solution of a continued fraction. Subsystem equations can be solved employing a modified Gaussian elimination method. The solver fully exploits parallelism. Optimized operational environment for the algorithm is discussed.

1. Introduction

An $n \times n$ matrix $A=(a_{ij})$ is Toeplitz if $a_{ij}=d_k$ for all i and j such that $i-j=k$, where d_k is an arbitrary constant. Furthur, if the elements of the matrix a_{ij} is zero for all $|i-j|\geq 2$, the matrix is called tridagonal. If

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \quad \text{for each} \quad i = 1, 2, \ldots, n,$$

then A is called a strictly diagonally dominant matrix.

The tridiagonal Toeplitz linear system (TTLS) plays a very important role in engineering problems. Specifically, they occur repeatedly in digital signal processing [7] and finite-difference approximation to various differential equations [1], [2], [5].

To speed up the computation time in solving a large-scale TTLS, recursive doubling [3], [8] and cyclic reduction [4] algorithms have been introduced. These algorithms are, by nature, all recursive during the decomposition phases, which are time-consuming procedures. Moreover, most of processors are left idle while a TTLS is decomposed. As the result, it is difficult to fully exploit parallelism with these algorithms.

A new parallel algorithm based on the modified Gaussian elimination method [6] has been proposed to solve a TTLS efficiently. The parallel algorithm is based on the modified Gaussian elimination method, a variant of the Gaussian elimination technique. This algorithm requires a continued fraction and its analytic solution during the decomposition phase to minimize the decomposition overhead. To minimize the interprocessor communication and maximize the degree of parallelism, the Gaussian elimination method is slightly modified using the limit convergent of the analytic solution of the continued fraction [6].

The efficiency of the algorithm is considered in this paper. Optimal environment for the minimum computation time for the proposed algorithm is discussed since the linear array is used. Compromise between computation and communication time should be required.

In Section 2, parallel algorithm for TTLS is developed. In Section 3, efficiency of the algorithm is shown based on the various indexes. Optimized operational environment, the measures to speed up the computation time, is also discussed. Section 3 summarizes the results.

2. Parallel Algorithm for TTLS

During the Gaussian forward elimination for a TTLS, a periodic continued fraction

$$\lambda_1 = \lambda,$$

$$\lambda_k = \lambda - \frac{a}{\lambda_{k-1}}, \quad k = 2, 3, \ldots, n.$$

appears. An analytic solution of the continued fraction [6] has been given. However, the analytic solution is time-consuming and prone to cause a serious numerical problem. Thus, when k is sufficiently large, we set $\lambda_k = \gamma$, where

$$\gamma = \frac{\lambda + \sqrt{\lambda^2 - 4a}}{2}.$$

A TTLS equation may be written in a form

$$\begin{bmatrix} \lambda & \beta & & & & \\ \alpha & \lambda & \beta & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \alpha & \lambda & \beta \\ & & & & \alpha & \lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ w_{n-1} \\ w_n \end{bmatrix}$$

or

$$A\underline{u} = \underline{w}.$$

The Gaussian elimination method is a sequential algorithm by nature. Thus, it should be modified for parallel computation. Hereafter, the modified Gaussian elimination method will be presented.

The Gaussian forward elimination is applied to eliminate α's in A as follows:

$$\lambda_1 = \lambda,$$

$$\lambda_k = \lambda - \frac{\alpha\beta}{\lambda_{k-1}}, \quad k = 2, 3, \ldots, n$$

and

$$\tilde{w}_1 = w_1,$$

$$\tilde{w}_k = w_k - \frac{\alpha}{\lambda_{k-1}} \tilde{w}_{k-1}, \quad k = 2, 3, \ldots, n.$$

Analytic solution of a continued fraction can be naturally involved in the parallel algorithm development. If A is strictly diagonally dominant, λ_k converges to γ for some finite integer $k<n$ when n is sufficiently large.

The Gaussian backward substitution procedure is now applied to obtain u_k's as follows:

$$u_n = \frac{1}{\lambda_n}\tilde{w}_n,$$

$$u_k = \frac{1}{\lambda_k}(\tilde{w}_k - \beta u_{k+1}), \quad k = n-1, n-2, \ldots, 1.$$

However, these procedures are all purely recursive. Although a recursive algorithm is desirable for sequential processing environment, it is the major obstacle to parallelism. To solve the TTLS with q processors simultaneously, it should be decomposed into q subsystem equations such that

$$A_i u_i = \tilde{w}_i, \quad i = 1, 2, \ldots, q$$

or, for ith processor,

$$\begin{bmatrix} \lambda_{j+1} & \beta & & & & \\ \alpha & \lambda & \beta & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \alpha & \lambda & \beta \\ & & & & \alpha & \lambda \end{bmatrix} \begin{bmatrix} u_{j+1} \\ u_{j+2} \\ \cdot \\ \cdot \\ u_{j+m-1} \\ u_{j+m} \end{bmatrix} = \begin{bmatrix} \tilde{w}_{j+1} \\ \tilde{w}_{j+2} \\ \cdot \\ \cdot \\ \tilde{w}_{j+m-1} \\ \tilde{w}_{j+m} \end{bmatrix}$$

with $j=m(i-1)$, where A_i is an $m\times m$ matrix. Now if A is strictly diagonally dominant, λ_{j+1} can be obtained from the analytic solution of the continued fraction immediately, where $a = \alpha\beta$. However, since the analytic solution is time-consuming and prone to cuase numerical problem, λ_{j+1} is replaced by γ of the limit convergent. Then,

$$\lambda_{j+1} = \gamma,$$

$$\lambda_{j+k} = \lambda - \frac{\alpha\beta}{\lambda_{j+k-1}}, \quad k = 2, 3, \ldots, m.$$

During the Gaussian forward elimination, $\tilde{w}_{j+k}$ cannot be obtained without knowing $\tilde{w}_j$. Thus, we define a new recursive equation pair

$$\hat{w}_{j+1} = \tilde{w}_{j+1},$$

$$\hat{w}_{j+k} = \tilde{w}_{j+k} - \frac{\alpha}{\lambda_{j+k-1}}\hat{w}_{j+k-1}, \quad k = 2, 3, \ldots, m.$$

and

$$m_j = 1,$$

$$m_{j+k} = -\frac{\alpha}{\lambda_{j+k-1}}m_{j+k-1}, \quad k = 1, 2, \ldots, m.$$

If $\tilde{w}_j$ is known, the Gaussian forward elimination can be as

$$\tilde{w}_{j+k} = \hat{w}_{j+k} + m_{j+k}\tilde{w}_j, \quad k = 1, 2, \ldots, m.$$

Similarly, we define a new recursive equation pair

$$\hat{u}_{j+m} = \frac{1}{\lambda_{j+m}},$$

$$\hat{u}_{j+k} = \frac{1}{\lambda_{j+k}}(\tilde{w}_{j+k} - \beta\hat{u}_{j+k+1}), \quad k = m-1, m-2, \ldots, 1.$$

and

$$n_{j+m+1} = 1,$$

$$n_{j+k} = -\frac{\beta}{\lambda_{j+k+1}}n_{j+k+1}, \quad k = m, m-1, \ldots, 1.$$

If u_{j+m+1} is known, the Gaussian backward substitution can be performed as

$$u_{j+m-k} = \hat{u}_{j+m-k} + n_{j+k+1}u_{j+m+1}, \quad k = 0, 1, \ldots, m-1.$$

Apparently, m_0, $\tilde{w}_0$, n_{n+1} and u_{n+1} are all zeroes. The modified Gaussian elimination algorithm requires additional procedures to compute m_k's and n_k's. Fig. 1 illustrates how to operate the proposed algorithm.

3. Optimal Environment for Parallel TTLS Solver

Since a TTLS equation is very simple, its computation time is directly proportional to n, dimension of a TTLS, for the sequential computer. During the Gaussian forward elimination, $3(n-1)$ multiplications/divisions and $2(n-1)$ additions/subtractions are required. During the Gaussian backward substitution, $2n-1$ multiplications/divisions and $n-1$ additions/subtractions are required. Thus, the straightforward Gaussian elimination method requires $5n-3$ multiplications/divisions and $3(n-1)$ additions/subtractions.

Consider a symmetric and strictly diagonally dominant TTLS. Assume that there are q processors such that $q = 2^p$, that is to say, at the pth phase, an $n\times n$ TTLS is decomposed into q subsystems of dimension m. Then, at least p decomposition phases are required to use q processors fully. At the pth decomposition phase, cyclic reduction algorithm [4] requires $m+3$ memory elements. Recursive doubling algorithm [3] requires $3m+2$ memory elements. All of them require $m+3$ memory elements during the linear equation solution phase. The required memory elements are used to store the minimum information, for example, α, β, λ, and w_k's. In our algorithm, $m+3$ memory elements are required during the linear equation solution phase only.

Consider that $2^p m=n$. Then, computation time complexity of cyclic reduction and recursive doubling algorithms amounts to $O(n)$ under the assumption that communication time complexity is negligible. That of our algorithm is $O(m)$. Since that of the sequential algorithm is $O(n)$, efficiency of our algorithm is evident. Total communication time complexities of recursive doubling and our algorithms are $O(\log_2 q)$ and $O(q)$, respectively, although they exchange 2 data between the neighboring processors. Total memory complexity of our algorithm is $O(n)$. However, others are all $O(pn)$ for minimum computation time. Table 1 summarizes the performances of the above-mentioned algorithms.

To minimize the computation time of the algorithm, compromise between computation and communication time should be required. Since the computation and communication time complexities are $O(m)$ and $O(n/m)$, assume that m and n/m unit operations are to be performed, respectively. Let unit communication time be ρ times slower than unit computation time. Then, it is easy to show that the maximum order of the decomposed subsystem is determined as

$$m = \sqrt{\rho n}$$

to minimize the total computation time. In this case, processor number q is determined as

$$q = \sqrt{n/\rho}.$$

4. Conclusion

To effectively solve a large-scale TTLS, a new parallel algorithm [6] has been proposed. The algorithm is based on the modified Gaussian elimination method, a variant of the Gaussian elimination technique. When the algorithm is employed, a strictly diagonally dominant TTLS can be decomposed into q subsystems at once. Only q TTLSs of dimension m are to be solved. Its efficiency has been demonstrated based on the quantitative indexes such as total computation time, communication time and memory complexities, $O(m)$, $O(n/m)$ and $O(n)$, respectively. Maximum order of the subsystem m can be determined as a function of n and ρ, under the assumption that unit communication time is ρ times slower than unit computation time. Thus, to minimize the total computation time, communication time between the processors should be minimized.

References

[1] R. L. Burden, J. D. Faires and A. C. Reynolds, *Numerical Analysis 2nd* ed. (Addison-Wesley, Reading, MA, 1982).

[2] B. L. Buzbee, A fast Poisson solver amenable to parallel computation, *IEEE Trans. Comput.* C-20 (1973) 793-796.

[3] D. J. Evans, Parallel numerical algorithms for linear systems, in: D. J. Evans, ed., *Parallel Processing Systems* (Cambridge University Press, 1982) 357-383.

[4] R. W. Hockney, A fast direct solution of Poisson's equation using Fourier analysis, *J. ACM* 12 (1965) 95-113.

[5] L. W. Johnson and R. D. Riess, *Numerical Analysis*, *2nd* ed. (Addison-Wesley, Reading, MA, 1982).

[6] H. J. Kim and J. G. Lee, A parallel algorithm solving a tridiagonal Toeplitz linear system, *Parallel Computing* (*to be appeared*).

[7] R. Roberts and C. T. Mullis, *Digital Signal Processing* (Addison-Wesley, Reading, MA, 1987).

[8] H. S. Stone, An efficient parallel algorithm for the solution of a tridiagonal system of equations, *J. ACM* 20 (1973) 27-38.

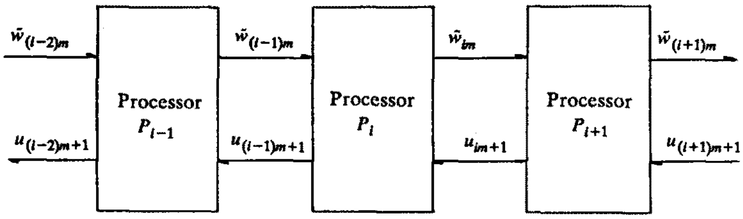

Figure 1. Operational scheme for the parallel TTLS solver.

Table 1. Performance comparison of the parallel TTLS solvers.

	Sequential	Cyclic Reduction	Recursive Doubling	Ours
Total Computation Counts	$O(n)$	$O(n)$	$O(n)$	$O(n/q)$
Total Communication Counts	-	$O(n)$	$O(\log_2 q)$	$O(q)$
Total Memory Counts	$O(n)$	$O(pn)$	$O(pn)$	$O(n)$