Computing 56, 385–395 (1996)

Computing

A Parallel Algorithm for Solving Special Tridiagonal Systems on Ring Networks

K.-L. Chung, W.-M. Yan and **J.-G. Wu,** Taipei

Received September 26, 1994; revised October 9, 1995

Abstract — Zusammenfassung

A Parallel Algorithm for Solving Special Tridiagonal Systems on Ring Networks. The solution of special linear, circulant-tridiagonal systems is considered. In this paper, a fast parallel algorithm for solving the special tridiagonal systems, which includes the skew-symmetric and tridiagonal-Toeplitz systems, is presented. Employing the diagonally dominant property, our parallel solver does need only local communications between adjacent processors on a ring network. An error analysis is also given. On the nCUBE/2E multiprocessors, some experimental results demonstrate the good performance of our stable parallel solver.

AMS Subject Classifications: 65F05, 15A23

Key words: nCUBE/2E multiprocessors, matrix perturbation, parallel algorithm, performance, ring network, tridiagonal Toeplitz linear systems.

Ein parelleler Algorithmus zur Lösung spezieller Tridiagonalsysteme auf Ring-Netzwerken. Wir betrachten die Lösung einer Klasse von speziellen tridiagonalen Gleichungssystemen, die schiefsymmetrische und Töplitz-Systeme einschließt, und geben einen schnellen, parallelen, Algorithmus dafür an. Bei Vorliegen von Diagonal-Dominanz benötigt unser paralleler Solver nur Kommunikation zwischen benachbarten Prozessoren auf einem Ring-Netzwerk. Eine Fehleranalyse wird angegeben. Einige experimentelle Resultate, die auf einem nCUBE/2E Gerät gewonnen wurden, zeigen das gute Verhalten unseres stabilen, parallelen Solvers.

1. Introduction

Throughout this paper, matrices are represented by uppercase letters, vectors by bold lowercase letters, and scalars by plain lowercase letters. The superscripts T correspond to the transpose operation. Consider an $n \times n$ linear system

$$A_n \mathbf{x} = \mathbf{y}, \tag{1}$$

where

$$A_n = \begin{pmatrix} \beta_1 & \gamma & & & & \beta_2 \\ \alpha & \beta & \gamma & & & \\ & \cdot & \cdot & \cdot & & \\ & & & \alpha & \beta & \gamma \\ \beta_2' & & & & \alpha & \beta_1' \end{pmatrix}_{n \times n}$$

and $|\beta| > |\alpha + \gamma|$. Solving this special linear system arises in many applications, e.g., numerical methods for differential equations [10], [9], [16], [15]. When setting $\beta_1 = \beta/u (u = 1 \text{ or } 2)$, $\beta_1' = \beta/v (v = 1 \text{ or } 2)$, $\alpha = \gamma$, and $\beta_2 = \beta_2' = \rho\gamma (\rho = 0 \text{ or } 1$, if $\rho = 1$ then it implies $u = v = 1)$, Boisvert [1] gave a survey to describe many fast serial algorithms for solving this class of special tridiagonal systems. These algorithms are Gaussian elimination [5], cyclic reduction [10], special LU factorization [16], [11], reversed triangular factorization [6], [7], [8], [1], and Toeplitz factorization with Sherman–Morrison formula [9]. Recently, Yan and Chung [17] presented a new algorithm for solving (1) sequentially when setting $\alpha = \gamma = \beta_2 = \beta_2'$ and $\beta_1 = \beta_1' = \beta$. Based on cyclic reduction method by Golub and Hockney [10], Bondeli and Glander [2] presented a parallel cyclic reduction for solving (1) when setting $\alpha = \gamma = 1$, $\beta_1 = \beta_1' = \beta$, and $\beta_2 = \beta_2' = 0$.

Following the approach in [17], this paper presents a new and fast parallel algorithm for solving the special tridiagonal system of (1). Employing diagonally dominant property, our parallel solver does need local communications between adjacent processors on a ring network. An error analysis for our parallel algorithm is also given. On the nCUBE/2E multiprocessors, some experimental results are illustrated to demonstrate the good performance of our stable parallel solver.

In Section 2 of this paper, we first describe the three-phase parallel solver on the ring network and the corresponding error analysis. Then, the parallel pseudo code is given. Section 3 contains performance results on the nCUBE/2E multiprocessors. Section 4 concludes the paper.

2. The Parallel Algorithm and its Analysis

Our parallel solver consists of three phases called the Toeplitz factorization, the substitution procedure, and the update procedure, respectively. Suppose that we have a ring network with p processors. In what follows, we will show how each phase can be accomplished on the ring network in a highly parallel way.

2.1 Phase 1: Toeplitz factorization

Let

$$A_q' = \begin{pmatrix} a & \gamma & & & & & \\ \alpha & \beta & \gamma & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & & \alpha & \beta & \gamma & \\ & & & & \alpha & \beta & \end{pmatrix}_{q \times q} = L'U', \tag{2}$$

$$L' = \begin{pmatrix} 1 & & & & & \\ -b & 1 & & & & \\ & \cdot & \cdot & & & \\ & & & -b & 1 & \\ & & & & -b & 1 \end{pmatrix}, \text{ and } U' = \begin{pmatrix} a & \gamma & & & & \\ & a & \gamma & & & \\ & & \cdot & \cdot & & \\ & & & & a & \gamma \\ & & & & & a \end{pmatrix}, \quad (3)$$

which implies that $a - \gamma b = \beta$ and $-ab = \alpha$. This implies that $b = -\alpha/a$ and $a = (\beta \pm \sqrt{(\beta^2 - 4\gamma\alpha)})/2$. Since we wish the matrices L' and U' to be diagonally dominant, we will select the sign so that the absolute value of a is greater than $\max(|\alpha|, |\gamma|)$. That is, when $\beta > |\alpha + \gamma|$, we chose $a = (\beta + \sqrt{(\beta^2 - 4\gamma\alpha)})/2$, from which $a > (|\alpha + \gamma| + \sqrt{((\alpha + \gamma)^2 - 4\gamma\alpha)})/2 = (|\alpha + \gamma| + |\alpha - \gamma|)/2 = \max(|\alpha|, |\gamma|)$. When $\beta < -|\alpha + \gamma|$, we chose $a = (\beta - \sqrt{(\beta^2 - 4\gamma\alpha)})/2$, then $a < [-|\alpha + \gamma| - \sqrt{((\alpha + \gamma)^2 - 4\gamma\alpha)}]/2 = (-|\alpha + \gamma| - |\alpha - \gamma|)/2 = -\max(|\alpha|, |\gamma|)$.

Since one of our choices always makes $|a| > \max(|\alpha|, |\gamma|)$, hence the bidiagonal Toeplitz matrices L' and U' are diagonally dominant. The computation of a and b provides the Toeplitz factorization of the matrix A' (phase 1 in our parallel solver), which can be done by each processor in $O(1)$ time on the ring network.

2.2 Phase 2: Substitution Procedure

For convenience, suppose n is a multiple of the number of processors used, i.e., $n = pq$. Initially, we partition $\mathbf{y}$ into p parts and $\mathbf{y} = (\mathbf{y}^{1^T}, \mathbf{y}^{2^T}, \ldots, \mathbf{y}^{p^T})^T$, where $\mathbf{y}^i = (y_1^{(i)}, y_2^{(i)}, \ldots, y_q^{(i)})^T$ for $1 \le i \le p$. The ith processor first solves $A'_q \mathbf{z}^i = \mathbf{y}^i$, $1 \le i \le p$, where $\mathbf{z}^i = (z_1^{(i)}, z_2^{(i)}, \ldots, z_q^{(i)})^T$. That is, the ith processor solves $L'_q U'_q \mathbf{z}^i = \mathbf{y}^i$. It can be verified that each processor in the ring network needs about $5q$ time to perform the forward and backward substitution procedure for solving $L'_q U'_q \mathbf{z}^i = \mathbf{y}^i$ (second phase in our algorithm) sequentially. Let $\mathbf{w} = A_n \mathbf{z} - \mathbf{y}$, where $\mathbf{z} = (\mathbf{z}^{1^T}, \mathbf{z}^{2^T}, \ldots, \mathbf{z}^{p^T})^T$ and $\mathbf{w} = (w_1, w_2, \ldots, w_n)^T$, then we have

$$w_1 = \left(\beta_1 z_1^{(1)} + \gamma z_2^{(1)} + \beta_2 z_q^{(p)}\right) - \left(a z_1^{(1)} + \gamma z_2^{(1)}\right) = (\beta_1 - a) z_1^{(1)} + \beta_2 z_q^{(p)},$$

$$w_n = \left(\beta'_2 z_1^{(1)} + \alpha z_{q-1}^{(p)} + \beta'_1 z_q^{(p)}\right) - \left(\alpha z_{q-1}^{(p)} + \beta z_q^{(p)}\right) = \beta'_2 z_1^{(1)} + (\beta'_1 - \beta) z_q^{(p)},$$

$$w_{iq} = \left(\alpha z_{q-1}^{(i)} + \beta z_q^{(i)} + \gamma z_1^{(i+1)}\right) - \left(\alpha z_{q-1}^{(i)} + \beta z_q^{(i)}\right) = \gamma z_1^{(i+1)} \quad \text{for } 1 \le i \le p-1$$

$$w_{iq+1} = \left(\alpha z_q^{(i)} + \beta z_1^{(i+1)} + \gamma z_2^{(i+1)}\right) - \left(a z_1^{(i+1)} + \gamma z_2^{(i+1)}\right) = \alpha z_q^{(i)} + (\beta - a) z_1^{(i+1)}$$
$$\text{for } 1 \le i \le p-1,$$

$$w_{iq+j} = 0, \; 0 \le i \le p-1, \; 2 \le j \le q-1.$$

Consequently, it follows that

$$A_n \mathbf{z} - \mathbf{y} = g_p \mathbf{e}_n + h_p \mathbf{e}_1 + \sum_{i=1}^{p-1} \left(g_i \mathbf{e}_{iq} + h_i \mathbf{e}_{iq+1}\right), \quad (4)$$

where

$$g_i = \gamma z_1^{(i+1)},\ h_i = \alpha z_q^{(i)} + (\beta - a) z_1^{(i+1)},\ 1 \le i \le p-1, \tag{5}$$

$$g_p = (\beta_1' - \beta) z_q^{(p)} + \beta_2' z_1^{(1)},\ h_p = (\beta_1 - a) z_1^{(1)} + \beta_2 z_q^{(p)}. \tag{6}$$

By (4), the solution of $\mathbf{x}$ in (1) will be determined approximately, say, $\mathbf{z} - \mathbf{p}$, in Section 2.3 later, where $A\mathbf{p} \approx g_p \mathbf{e}_n + h_p \mathbf{e}_1 + \sum_{i=1}^{p-1} (g_i \mathbf{e}_{iq} + h_i \mathbf{e}_{iq+1})$. To estimate the bound of $\|A\mathbf{x} - \mathbf{y}\|/\|\mathbf{y}\|$, we need the following lemma. Here $\|\bullet\|$ denotes the infinite norm of a vector.

Lemma 1. [4] *if*

$$L = \begin{pmatrix} l_0 & & & & & \\ l_1 & l_0 & & & & \\ & \cdot & \cdot & & & \\ & & & l_1 & l_0 & \\ & & & & l_1 & l_0 \end{pmatrix},\ |l_0| > |l_1|, \quad \text{then } \|L^{-1}\mathbf{y}\| \le \frac{1}{|l_0| - |l_1|}\|\mathbf{y}\|.$$

Similarly, we have the inequality

$$\|U^{-1}\mathbf{y}\| \le \frac{1}{|u_0| - |u_1|}\|\mathbf{y}\|$$

for the upper bidiagonal matrix

$$U = \begin{pmatrix} u_0 & u_1 & & & & \\ & u_0 & u_1 & & & \\ & & \cdot & \cdot & & \\ & & & & u_0 & u_1 \\ & & & & & u_0 \end{pmatrix},\ |u_0| > |u_1|.$$

From $A_q' \mathbf{z}^i = \mathbf{y}^i$, $1 \le i \le p$, Lemma 1, and the above inequality, it gives

$$\|\mathbf{z}^i\| = \|(L_q' U_q')^{-1} \mathbf{y}^i\| = \|U_q'^{-1}(L_q'^{-1} \mathbf{y}^i)\| \le \frac{1}{|a| - |\gamma|}\|L_q'^{-1}\mathbf{y}^i\|$$

$$\le \frac{1}{|a| - |\gamma|(1 - |b|)}\|\mathbf{y}^i\| \le \frac{1}{(|a| - |\gamma|)(1 - |b|)}\|\mathbf{y}\| \quad \text{for } 1 \le i \le p. \tag{7}$$

2.3 Phase 3: Update Procedure

By (4), in what follows, the solution of $\mathbf{x}$ in (1) will be determined approximately and only local communications between adjacent processors on the ring network are needed. Then, the bound of $\|A\mathbf{x} - \mathbf{y}\|/\|\mathbf{y}\|$ will be derived.

From $a-\gamma b=\beta$ and $-ab=\alpha$, it derives to $\alpha+\beta b+\gamma b^2=0$. If we let $c=-\gamma/a$, then we have $\gamma+\beta c+\alpha c^2=0$. Let

$$\mathbf{p}_k=\Big(\underbrace{0,\ldots,0,1}_{k},\underbrace{b,\ldots,b^s,0,\ldots,0}_{n-k}\Big)^T$$

and

$$\mathbf{q}_k=\Big(\underbrace{0,\ldots,0,c^t,\ldots,c,1}_{k},\underbrace{0,\ldots,0}_{n-k}\Big)^T$$

for $q\le k\le n-q$. Here we assume $q>\max(s,t)$. In addition, we let $\mathbf{p}_1=(1,b,\ldots,b^{s-1},b^s,0,\ldots,0)^T$ and $\mathbf{q}_n=(0,\ldots,0,c^t,c^{t-1},\ldots,c,1)^T$. Accordingly, it gives

$$A\mathbf{p}_k=\Big(\underbrace{0,\ldots,0,\gamma,\beta+\gamma b}_{k},$$
$$\underbrace{\alpha+\beta b+\gamma b^2,\ldots,\alpha b^{s-2}+\beta b^{s-1}+\gamma b^s,\alpha b^{s-1}+\beta b^s,\alpha b^s,0,\ldots,0}_{n-k}\Big)$$

$$=\Big(\underbrace{0,\ldots,0,\gamma,a}_{k},\underbrace{0,\ldots,-\gamma b^{s+1},\alpha b^s,0,\ldots,0}_{n-k}\Big)$$
$$=\gamma\mathbf{e}_{k-1}+a\mathbf{e}_k+\alpha b^s\Big(-\frac{\gamma}{\alpha}b\mathbf{e}_{k+s}+\mathbf{e}_{k+s+1}\Big)$$
$$=\gamma\mathbf{e}_{k-1}+a\mathbf{e}_k+\alpha b^s(-c\mathbf{e}_{k+s}+\mathbf{e}_{k+s+1}),\tag{8}$$

$A\mathbf{q}_k$

$$=\Big(\underbrace{0,\ldots,0,\gamma c^t,\beta c^t+\gamma c^{t-1},\alpha c^t+\beta c^{t-1}+\gamma c^{t-2},\ldots,\alpha c^2+\beta c+\gamma,\alpha c+\beta}_{k},$$
$$\underbrace{\alpha,0,\ldots,0}_{n-k}\Big)$$
$$=\Big(\underbrace{0,\ldots,0,\gamma c^t,-\alpha c^{t+1},0,\ldots,0,a}_{k},\underbrace{\alpha,0,\ldots,0}_{n-k}\Big)$$
$$=\gamma c^t\Big(\mathbf{e}_{k-t-1}-\frac{\alpha}{\gamma}c\mathbf{e}_{k-t}\Big)+a\mathbf{e}_k+\alpha\mathbf{e}_{k+1}$$

$$=\gamma c^t(\mathbf{e}_{k-t-1}-b\mathbf{e}_{k-t})+a\mathbf{e}_k+\alpha\mathbf{e}_{k+1},\tag{9}$$

$$A\mathbf{p}_1=(\beta_1+\gamma b,\alpha+\beta b+\gamma b^2,\ldots,\alpha b^{s-2}+\beta b^{s-1}+\gamma b^2,\alpha b^{s-1}$$
$$+\beta b^s,\alpha b^s,0,\ldots,0,\beta_2')^T$$
$$=(\beta_1+\gamma b)\mathbf{e}_1-\gamma b^{s+1}\mathbf{e}_{s+1}+\alpha b^s\mathbf{e}_{s+2}+\beta_2'\mathbf{e}_n$$

$$= \gamma_2' \mathbf{e}_1 + \beta_2' \mathbf{e}_n + \alpha b^s(-c\mathbf{e}_{s+1} + \mathbf{e}_{s+2}), \tag{10}$$

$$A\mathbf{q}_n = \big(\beta_2, 0, \ldots, 0, \gamma c^t, \beta c^t + \gamma c^{t-1}, \alpha c^t + \beta c^{t-1} + \alpha c^{t-2}, \ldots, \alpha c^2 + \beta c + \gamma, \alpha c + \beta_1'\big)^T$$

$$= \beta_2 \mathbf{e}_1 + \gamma c^t \mathbf{e}_{n-t-1} - \alpha c^{t+1} \mathbf{e}_{n-t} + (\alpha c + \beta_1')\mathbf{e}_n$$

$$= \beta_2 \mathbf{e}_1 + \gamma_2 \mathbf{e}_n + \gamma c^t (e_{n-t-1} - b\mathbf{e}_{n-t}), \tag{11}$$

where

$$\gamma_2 = \alpha c + \beta_1', \qquad \gamma_2' = \beta_1 + \gamma b.$$

By (8), (9), (10), and (11) for $1 \le i \le p-1$, we have

$$A(u_i \mathbf{p}_{iq+1} + v_i \mathbf{q}_{iq}) = (\gamma u_i + a v_i)\mathbf{e}_{iq} + (a u_i + \alpha v_i)\mathbf{e}_{iq+1} + u_i \alpha b^s(-c\mathbf{e}_{iq+s+1} + \mathbf{e}_{iq+s+2}) + v_i \gamma c^t(\mathbf{e}_{iq-t-1} - b\mathbf{e}_{iq-t}), \tag{12}$$

$$A(u_p \mathbf{p}_1 + v_p \mathbf{q}_n) = (\beta_2' u_p + \gamma_2 v_p)\mathbf{e}_n + (\gamma_2' u_p + \beta_2 v_p)\mathbf{e}_1 + u_p \alpha b^s(-c\mathbf{e}_{s+1} + \mathbf{e}_{s+2}) + v_p \gamma c^t(\mathbf{e}_{n-t-1} - b\mathbf{e}_{n-t}). \tag{13}$$

Let $\gamma u_i + a v_i = g_i$ and $a u_i + \alpha v_i = h_i$ for $1 \le i \le p-1$; $\beta_2' u_p + \gamma_2 v_p = g_p$ and $\gamma_2' u_p + \beta_2 v_p = h_p$. Therefore, it gives

$$u_i = \frac{\alpha g_i - a h_i}{\alpha\gamma - a^2} \quad v_i = \frac{\gamma h_i - a g_i}{\alpha\gamma - a^2}, \quad 1 \le i \le p-1, \tag{14}$$

$$u_p = \frac{\gamma_2 h_p - \beta_2 g_p}{\gamma_2' \gamma_2 - \beta_2 \beta_2'}, \quad v_p = \frac{\gamma_2' g_p - \beta_2' h_p}{\gamma_2' \gamma_2 - \beta_2 \beta_2'}. \tag{15}$$

Let

$$\mathbf{p} = (u_p \mathbf{p}_1 + v_p \mathbf{q}_n) + \sum_{i=1}^{p-1} (u_i \mathbf{p}_{iq+1} + v_i \mathbf{q}_{iq}), \tag{16}$$

by (12), (13), (14), and (15), we have $A\mathbf{p} \approx g_p \mathbf{e}_n + h_p \mathbf{e}_1 + \sum_{i=1}^{p-1}(g_i \mathbf{e}_{iq} + h_i \mathbf{e}_{iq+1})$. Let $\mathbf{x} = \mathbf{z} - \mathbf{p}$, then we have the following theorem.

Theorem 2. [4]

$$\frac{\|A_n \mathbf{x} - \mathbf{y}\|}{\|\mathbf{y}\|} \le \max_{1 \le i \le p} \big(\max(|u_i \alpha b^s|, |v_i \gamma c^t|)\big)$$

$$\le \max(\eta_1 |\alpha b^s|, \eta_2 |\gamma c^t|, \eta_1' |\alpha b^s|, \eta_2' c^t|),$$

where

$$\eta_1 = \frac{|\gamma|\,|\alpha + ab| + |a\alpha|}{|\alpha\gamma - a^2|(|a| - |\gamma|)(1 - |b|)}, \quad \eta_2 = \frac{|\gamma|(|\alpha| + |\beta - 2a|)}{|\alpha\gamma - a^2|(|a| - |\gamma|)(1 - |b|)},$$

$$\eta_1' = \frac{|\gamma_2(\beta_1 - a) - \beta_2\beta_2'| + |\beta_2(\beta + \gamma_2 - \beta_1')|}{|\gamma_2'\gamma_2 - \beta_2\beta_2'|(|a| - |\gamma|)(1 - |b|)},$$

$$\eta_2' = \frac{|\gamma_2'\beta_2' - \beta_2'(\beta_1 - a)| + |\gamma_2'(\beta_1' - \beta) - \beta_2'\beta_2|}{|\gamma_2'\gamma_2 - \beta_2\beta_2'|(|a| - |\gamma|)(1 - |b|)}.$$

2.4 Parallel Psuedo Code

For simplicity, suppose the size of the linear system of (1) (= the size of **y** in (1)), n, is much greater than the number of processors, p, used in the ring network, and n is a multiple of p. Initially, the n data of **y** are partitioned into p parts, i.e., each processor stores n/p data. The processor with address 0 (node − id = 0) wants to compute $x_1, x_2, \ldots, x_{n/p}$; the processor with address 1 (node − id = 1) wants to compute $x_{n/p+1}, x_{n/p+2}, \ldots, x_{2n/p}$, and so on. In our parallel algorithm, each node performs the following pseudo-code.

```
/* initially the vector y is stores in x */
/* all the concerning variables are local */
q ← n/p
/* Phase 1: Toeplitz factorization */
if β > 0 then a ← [β + √(β² − 4γα)]/2 else a ← [β − √(β² − 4γα)]/2 endif
b ← −α/a; c ← −γ/a; d ← 1/a
/* Phase 2: Substitution procedure */
for i := 2 to q do x_i = x_i + b * x_{i−1} endfor /* forward substitution */
x_q = d * x_q
for i := q − 1 downto 1 do x_i = d * x_{i+1} + c * x_i endfor /* backward substitution
*/
/* Phase 3: Update procedure */
if p = 1 then
    t1 → x_q;   t2 → x_1

else
    send x_1 to node−[(id − 1) mod p];   send x_q to node−[(id + 1) mod p]
    receive the value from node−[(id − 1) mod p] and store it in t1
    receive the value from node−[(id + 1) mod p] and store it in t2
endif

if node−id = p − 1 then
  if node−id = 0 then
  g ← (β_1' − β)* x_q + β_2' * t2; h ← β_2 * x_q + (β_1 − a) * t2;
```

$\gamma_2 \leftarrow \alpha * c + \beta_1'$; $\gamma_2' \leftarrow \beta_1 + \gamma b$

det $\leftarrow (\gamma_2\gamma_2' - \beta_2\beta_2')$; $v \leftarrow (\gamma_2' * g - \beta_2' * h)/\text{det}$; $u \leftarrow (\gamma_2 * h - \beta_2 * g)/\text{det}$

else

$g \leftarrow \gamma * x_1$; $h \leftarrow \alpha * t_1 + (\beta - a) * x_1$; det $\leftarrow \alpha * \gamma - a^2$;

$u \leftarrow (\alpha * g - a * h)/\text{det}$

$g \leftarrow (\beta_1' - \beta) * x_q + \beta_2' * t2$; $h \rightarrow \beta_2 * x_q + (\beta_1 - a) * t2$;

$\gamma_2 \leftarrow \alpha * c + \beta_1'$; $\gamma_2' \leftarrow \beta_1 + \gamma b$

det $\leftarrow (\gamma_2\gamma_2' - \beta_2\beta_2')$; $v \leftarrow (\gamma_2' * g - \beta_2' * h)/\text{det}$

endif

else

if node-id = 0 **then**

$g \leftarrow \gamma * t2$; $h \leftarrow \alpha * x_q + (\beta - a) * t2$; det $\leftarrow \alpha * \gamma - a^2$;

$v \leftarrow (\gamma * h - a * g)/\text{det}$

$g \leftarrow (\beta_1' - \beta) * t_1 + \beta_2' * x_1$; $h \leftarrow \beta_2 * t_1 + (\beta_1 - a) * x_1$; $\gamma_2 \leftarrow \alpha * c + \beta_1'$;

$\gamma_2' \leftarrow \beta_1 + \gamma b$

det $\leftarrow (\gamma_2\gamma_2' - \beta_2\beta_2')$; $u \leftarrow (\gamma_2 * h - \beta_2 * g)/\text{det}$

else

$g \leftarrow \gamma * x_1$; $h \leftarrow \alpha * t_1 + (\beta - a) * x_1$; det $\leftarrow \alpha * \gamma - a^2$; $u \leftarrow (\alpha * g - a * h)/\text{det}$

$g \leftarrow \gamma * t2$; $h \leftarrow \alpha * x_q + (\beta - a) * t2$; det $\leftarrow \alpha * \gamma - a^2$; $v \leftarrow (\gamma * h - a * g)/\text{det}$

endif

endif

$t \leftarrow q - 1$; $s \leftarrow q - 1$; power $\leftarrow u$
for i := 1 **to** s **do** $x_i \leftarrow x_i -$ power; power $\leftarrow b *$ power **endfor**
power $\leftarrow v$
for i := 0 **to** t − 1 **do** $x_{q-i} \leftarrow x_{q-i}$ −power; power $\leftarrow c *$ power **endfor**

3. Experimental Results

The k-dimensional hypercube network, H_k, has 2^k nodes and $k2^{k-1}$ edges, where two nodes are linked with an edge if and only if their binary strings differ in one bit. Figure 1 illustrates an H_3. Based on the parallel pseudo code described in Section 2.4 and the embedding method to map a ring network into

the hypercube [13], we have implemented our parallel solver for solving (1) on a 16-node nCUBE/2E multiprocessors.

Our parallel program has been executed on the machine with 1, 2, 4, 8, and 16 processors, respectively, and the nCUBE/2E parallel C source code for solving (1) is listed in [4]. Five sets of data corresponding to the five cases, namely, the symmetric Toeplitz tridiagonal case [17], the skew-symmetric Toeplitz tridiagonal case [8], the circulant-symmetric Toeplitz tridiagonal case [1], the symmetric near-Toeplitz tridiagonal case [1], and the circulant-skew-symmetric Toeplitz case [8], respectively, are taken to demonstrate the performance of our parallel solver.

In our implementations, we let $y_i = \text{rand}()$, $\alpha = 1$, and $\beta = 4$ for all cases, where the function 'rand()' denotes a random number generator. In the first set of data, we let $\gamma = \alpha = 1$, $\beta_1 = \beta_1' = \beta = 4$, and $\beta_2 = \beta_2' = 0$, representing the symmetric Toeplitz tridiagonal case. In the second set of data, $\gamma = -\alpha = -1$, $\beta_1 = \beta_1' = \beta = 4$, and $\beta_2 = \beta_2' = 0$, representing the skew-symmetric Toeplitz tridiagonal case. In the third set of data, $\gamma = \alpha = \beta_2 = \beta_2' = 1$, and $\beta_1 = \beta_1' = \beta = 4$, representing the circulant-symmetric Toeplitz trigonal case. In the fourth set of data, $\gamma = \alpha = 1$, $\beta_1 = \beta_1' = \beta/2 = 2$, and $\beta_2 = \beta_2' = 0$, representing the symmetric near-Toeplitz tridiagonal case. In the fifth set of data, $\gamma = -\alpha = \beta_2 = -\beta_2' = 1$, and $\beta_1 = \beta_1' = \beta = 4$, representing the circulant-skew-symmetric Toeplitz case.

Since the execution times for all these cases are identical, and the relative residuals, $\|A\mathbf{x} - \mathbf{y}\|/\|\mathbf{y}\|$, are of the same order. In Table 1, we only list the execution time ($T_p(n)$), $\|A\mathbf{x} - \mathbf{y}\|/\|\mathbf{y}\|$, and the speedup ($T_1(n)/T_p(n)$) for the first set of data, where n denotes the problem size, $T_1(n)$ denotes the time required when one processor is used, $T_p(n)$ denotes the time required when p processors are used. In the table, each entry has three values. The first value shows the execution time, the second value shows the sup-norm of the relative residual, and the third value denotes the speedup. It is shown that our parallel solver has good scalability and stability. The relative residual is small (less than 10^{-15}) when each processor processes more than 32 data.

We also reorder the sequence of data communication operations in order to

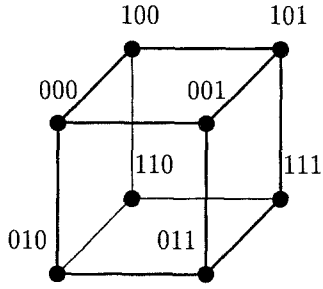

Figure 1. An H_3

Table 1. Execution time ($T_p(n)$, in milliseconds), ($\|A\mathbf{x} - \mathbf{y}\|/\|\mathbf{y}\|$), and speedup ($T_1(n)/T_p(n)$)

n	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$
64	0.751 0 1	0.764 0 0.98	0.616 2.3×10^{-10} 1.22	0.560 3.9×10^{-6} 1.34	0.517 4.7×10^{-3} 1.45
128	1.445 0 1	1.112 0 1.30	0.789 0 1.83	0.625 2.3×10^{-10} 2.31	0.531 3.9×10^{-6} 2.72
256	2.84 1.4×10^{-16} 1	1.836 0 1.55	1.114 0 2.55	0.79 0 3.59	0.648 2.3×10^{-10} 4.38
512	5.687 0 1	3.232 1.4×10^{-16} 1.76	1.809 0 3.14	1.137 0 5.00	0.821 0 6.93
1024	11.324 0 1	6.051 1.4×10^{-16} 1.87	3.204 1.4×10^{-16} 3.53	1.832 0 6.18	1.138 0 9.95
2048	22.622 0 1	11.719 0 1.93	6.051 1.4×10^{-16} 3.74	3.222 0 7.02	1.833 0 12.34

hide the long communication latency. We let each node send its x_q to its neighboring node at the begining of Phase 2, and read the value from another node in the receiving buffer after it sends its x_1 to another neighboring node in Phase 3. However, the experiment shows that the execution time makes no difference with that of using the original communication sequence. Since either one of the communication operations sends and receives one double precision number. The messages are very short. The total communication time of a message with length L between two nodes is dominated by the start-up time, which is about 150 μs [14]. This cannot be hidden by sending the message a long time before receiving it by another processor.

4. Concluding Remarks

The significance of solving special tridiagonal Toeplitz linear systems is due to its use in many areas such as using finite difference methods to solve linear constant-coefficient boundary-value problems and solving uniform B-spline curve/surface fitting problems. We have presented an efficient parallel algorithm for solving special tridiagonal Toeplitz systems on the ring network. Employing diagonally dominant property, our parallel solver does need local communications between adjacent processors on the ring network. An error analysis has also been provided. The corresponding implementation has been done on the nCUBE/2E multiprocessors.

Acknowledgements

The authors would like to thank the referees for several valuable comments. These comments improved the quality and presentation of this paper. The research was supported in part by the National Science Council of R.O.C. under contracts NSC85-2121-M011-002 and NSC-852213-E003-001.

References

[1] Boisvert, R. F.: Algorithms for special tridiagonal systems. SIAM J. Sci. Stat. Comput. *12*, 423–442 (1991).

[2] Bondeli, S., Gander, W.: Cyclic reduction for special tridiagonal systems. SIAM J. Matrix Anal. Appl. *15*, 321–330 (1994).

[3] Chung, K. L., Yan, W. M.: A fast algorithm for cubic B-spline curve fitting. Comput. Graphics *18*, 327–334 (1994).

[4] Chung, K. L., Yan, W. M., Wu, J. G.: A parallel algorithm for solving special tridiagonal systems on ring networks. Research Report, Dept. Inform. Mgmt., National Taiwan Inst. of Tech. (May 1994).

[5] Dongarra, J. J., Moler, C. B., Bunch, J. R., Stewart, G. W.: LINPACK User's Guide. New York: SIAM Press 1979.

[6] Evans D. J., Forrington, C. V. D.: Note on the solution of certain tridiagonal systems of linear equations. Comput. J. *5*, 327–328 (1963).

[7] Evans, D. J.: An algorithm for the solution of certain tridiagonal systems of linear equations. Comput. J. *15*, 356–359 (1972).

[8] Evans, D. J.: On the solution of certain Toeplitz tridiagonal linear systems. SIAM J. Numer. Anal. *17*, 675–680 (1980).

[9] Fischer, D., Golub, G., Hald, O., Levia, C., Winlund, O.: On Fourier-Toeplitz methods for separable elliptic problems. Math. Comput. *28*, 349–368 (1974).

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

[11] Malcolm, M. A., Palmer, J.: A fast method for solving a class of tridiagonal linear systems. Comm. ACM *17*, 14–17 (1974).

[12] Rojo, O.: A new method for solving symmetric circulant tridiagonal systems of linear equations. Comput. Math. Appl. *20*, 12, 61–67 (1990).

[13] Saad, Y., Schltz, M. H.: Topological properties of hypercubes. IEEE Trans. Comput. *37*, 867–872 (1988).

[14] Schmidt–Voigt, M.: Efficient parallel communication with nCUBE 2S processor. Parallel Comput. *20*, 509–530 (1994).

[15] Smith, G. D.: Numerical solution of partial differential equations: finite difference methods, 3rd ed. Oxford: Oxford University Press, 1985.

[16] Widlund, O. B.: On the use of fast methods for separable finite difference equations for the solution of general elliptic problems. In Sparse matrices and their applications (Rose, D. J., Willoughby, R. A., eds.), pp. 121–131. New York: Plenum Press 1972.

[17] Yan, W. M., Chung, K. L.: A fast algorithm for solving special tridiagonal systems. Computing *52*, 203–211 (1994).

Dr. K.-L. Chung
Department of Information Management
National Taiwan Institute of Technology
No. 43, Section 4, Keelung Road,
Taipei, Taiwan 10672, R.O.C

Mr. W.-M. Yan
Department of Computer Science
and Information Engineering
National Taiwan University,
Taipei, Taiwan 10764, R.O.C.

Dr. J.-G. Wu
Department of Information
and Computer Education
National Taiwan Normal University,
Taipei, Taiwan 10610, R.O.C.