SIAM J. NUMER. ANAL.
Vol. 14, No. 6, December 1977

SOLVING TRIANGULAR SYSTEMS ON A PARALLEL COMPUTER*

AHMED H. SAMEH† AND RICHARD P. BRENT‡

Abstract. In this paper we present alternative formulations of the algorithms of Chen and Kuck [IEEE Trans. Computers (1975)]. We also give a detailed error analysis, showing that if $\tilde{x}$ is the computed solution of the triangular system $Lx = f$, then it satisfies the equation $(L + \delta L)\tilde{x} = f$ where $\|\delta L\| \leqq O(n^2 \log n)\varepsilon\kappa^2(L)\|L\|$. Here $\kappa(L)$ is the condition number of L, $\|\cdot\|$ denotes the ∞-norm, and ε is the unit roundoff.

1. Introduction. Chen and Kuck [3], Heller [4], Borodin and Munro [1], and Orcutt [9] have all shown that, given $O(n^3)$ processors, a triangular system of n equations $Lx = f$ may be solved in $O(\log^2 n)$ steps.[1] Chen and Kuck [3], and Orcutt [9], also showed that if L is a banded lower triangular matrix of bandwidth $(m+1)$, i.e., its elements $\lambda_{ij} = 0$ for $i - j > m$,[2] then the time required for solving $Lx = f$ is $O(\log m \log n)$ using $O(m^2 n)$ processors. Hyafil and Kung [6] discussed the same problems for a limited number of processors. Kogge [7] analyzed the numerical stability of a parallel program for solving banded lower triangular systems with $m \leqq 2$.

In this paper we present two algorithms, expressed in matrix notation, for solving the system $Lx = f$ for both the dense and banded cases, and establish bounds on the time and number of required processors. With the exception of Chen and Kuck [3], none of the above references computed the actual number of processors required. For example, we can show that the algorithm discussed by Heller [4] requires $\frac{1}{6}n^3 + O(n^2)$ processors for the dense case compared to Chen and Kuck's result, and ours, of $\frac{1}{68}n^3 + O(n^2)$. Also, using our new formulation we show that for the banded case only $\frac{1}{2}m^2 n + O(mn)$ processors are necessary compared to Chen and Kuck's result of $\frac{3}{4}m^2 n + O(mn)$. Our main result, however, is an error analysis of the dense case. We show that if $\tilde{x}$ is the computed solution then $(L + \delta L_p)\tilde{x} = f$, where δL_p is bounded by, $\|\delta L_p\| \leqq \alpha(n)\varepsilon\kappa^2(L)\|L\|$. Here, $\|\cdot\|$ stands for the ∞-norm, $\alpha(n) = O(n^2 \log n)$, ε is the unit roundoff, $\kappa(L)$ is the condition number of L, and we assume that terms of $O(\varepsilon^2)$ are negligible. This bound can be very large compared to that of the sequential algorithm, Wilkinson [10], where $\|\delta L_s\| \leqq n\varepsilon\|L\|$.

Our machine assumptions are as follows:

(i) any number of processors can be used at any time (but we will give bounds on this number),

(ii) each processor can perform any of the four arithmetic operations in one time step, and

(iii) there are no memory or data alignment penalties.

* Received by the editors November 14, 1975, and in final revised form November 9, 1976. This work was supported by the National Science Foundation under Grants USNSF DCR73-07980 A02 and MCS-7521758.

† Department of Computer Science and the Center for Advanced Computation, University of Illinois, Urbana, Illinois 61801

‡ Computer Centre, The Australian National University, Canberra, Australia.

[1] Throughout this paper $\log n \equiv \lceil \log_2 n \rceil$, and time is measured in steps.

[2] Without loss of generality, we assume $n = 2^\mu$, $m = 2^\nu$.

Even though assumptions (i) and (iii) may be unrealistic, we believe that the algorithms presented here will be influential in creating parallel algorithms for specific realizable computers.

Throughout the paper we adopt the notational conventions of Householder [5]. So, except for dimensions and indices, lower case Greek letters represent scalars, lower case *italic* letters represent column vectors, and upper case Greek or *italic* letters represent matrices.

2. Algorithm I. The inverse of $L=[\lambda_{ij}]$ can be expressed as, Householder [5],

$$L^{-1}=M_nM_{n-1}\cdots M_1 \tag{2.1}$$

where

$$M_i=\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & \text{ith col.} & & \\ & & & 1 & & & \\ & & & & 1/\lambda_{ii} & & \\ & & & & -\lambda_{i+1,i}/\lambda_{ii} & 1 & \\ & & & & \vdots & & \ddots \\ & & & & -\lambda_{ni}/\lambda_{ii} & & & 1 \end{bmatrix} \tag{2.2}$$

and the solution x is given by

$$x=M_nM_{n-1}\cdots M_1f. \tag{2.3}$$

Evaluating the product $(M_{n-1}\cdots M_2M_1f)$ in parallel requires $\mu\equiv\log n$ stages. Let $M_i^{(0)}\equiv M_i$, $f^{(0)}\equiv f$, and $s\equiv 2^j$. Then at each stage $(j+1)$ we evaluate matrices $M_i^{(j+1)}=M_{2i+1}^{(j)}M_{2i}^{(j)}$ for $j=0,1,\cdots,\mu-2$ and $i=1,2,\cdots,n/(2s)-1$, and the vector $f^{(j+1)}=M_1^{(j)}f^{(j)}$ for $j=0,1,\cdots,\mu-1$. This process is shown in Fig. 1 for $n=8$.

Each matrix $M_i^{(j)}$ is of the form,

$$M_i^{(j)}=\begin{bmatrix} I_i^{(j)} & 0 & 0 \\ 0 & \hat{L}_i^{(j)} & 0 \\ 0 & S_i^{(j)} & \hat{I}_i^{(j)} \end{bmatrix} \tag{2.4}$$

where $\hat{L}_i^{(j)}$ is a lower triangular matrix of order s, $I_i^{(j)}$ and $\hat{I}_i^{(j)}$ are identity matrices of orders $q_i^{(j)}=is-1$ and $r_i^{(j)}=(n+1)-(i+1)s$, respectively. Clearly, $\hat{L}_i^{(0)}\equiv 1/\lambda_{ii}$, and $S_i^{(0)}=(-1/\lambda_{ii})(\lambda_{i+1,i},\lambda_{i+2,i},\cdots,\lambda_{n,i})^t$. Let

$$S_i^{(j)}=\begin{bmatrix} U_i^{(j)} \\ V_i^{(j)} \end{bmatrix} \tag{2.5}$$

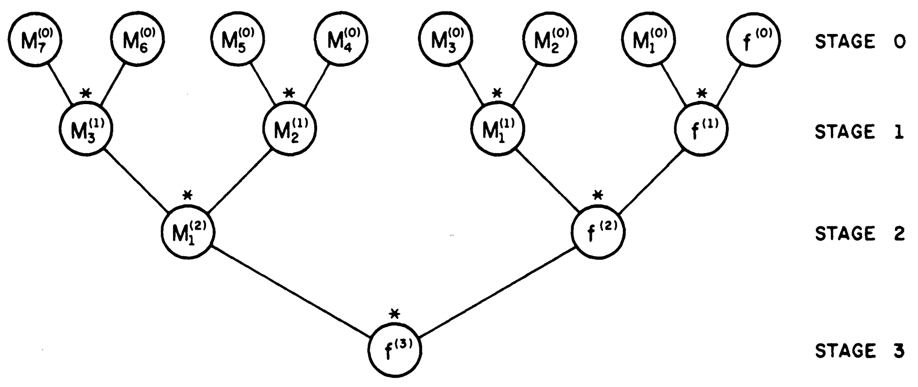

FIG. 1.

where, $U_i^{(j)}$ is a square matrix of order s. Thus,

$$\hat{L}_i^{(j+1)} = \begin{bmatrix} \hat{L}_{2i}^{(j)} & 0 \\ \hat{L}_{2i+1}^{(j)} U_{2i}^{(j)} & \hat{L}_{2i+1}^{(j)} \end{bmatrix} \tag{2.6}$$

and

$$S_i^{(j+1)} = [S_{2i+1}^{(j)} U_{2i}^{(j)} + V_{2i}^{(j)}, S_{2i+1}^{(j)}]. \tag{2.7}$$

In Fig. 2 we show the shapes of $M_{2i}^{(j)}$, $M_{2i+1}^{(j)}$, and $M_i^{(j+1)}$. Furthermore, if $f^{(j)t} = (g_1^{(j)}, g_2^{(j)}, g_3^{(j)})^t$ where the vectors $g_1^{(j)}$ and $g_2^{(j)}$ are of orders $s-1$ and s, respectively, then the three subvectors constituting $f^{(j+1)}$ are given by

$$\begin{aligned} f_1^{(j+1)} &= g_1^{(j)}, \\ f_2^{(j+1)} &= \hat{L}_1^{(j)} g_2^{(j)}, \\ f_3^{(j+1)} &= S_1^{(j)} g_2^{(j)} + g_3^{(j)} \end{aligned} \tag{2.8}$$

where the first two constitute the first $2s-1$ elements of the solution vector x.

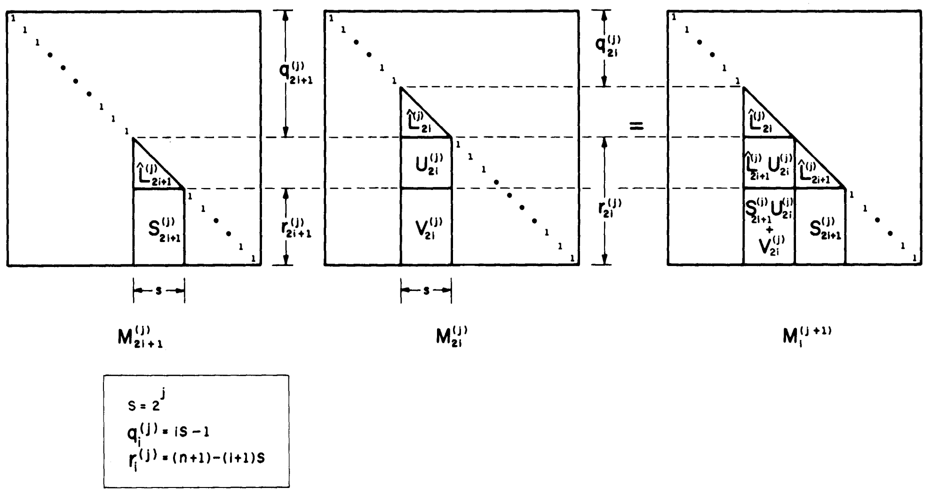

FIG. 2.

Under our machine assumptions, we can form each matrix M_i in (2.2) in 2 steps, one division and one subtraction, using $(n-i+1)$ processors (the number of nonzero elements in the ith column). Hence, forming all the matrices M_i, $i=1,2,\cdots,n$, requires 2 steps with $\sum_{i=1}^{n}(n-i+1)=n(n+1)/2$ processors. Similarly, the products $\hat{L}_{2i+1}^{(j)}U_{2i}^{(j)}$, $S_{2i+1}^{(j)}U_{2i}^{(j)}$, $\hat{L}_1^{(j)}g_2^{(j)}$, and $S_1^{(j)}g_2^{(j)}$ in (2.6)–(2.8) can be evaluated simultaneously in $1+\log s=(1+j)$ steps, the time required for evaluating the inner product of two vectors of order s. The sums $(S_{2i+1}^{(j)}U_{2i}^{(j)}+V_{2i}^{(j)})$ and $(S_1^{(j)}g_2^{(j)}+g_3^{(j)})$ in (2.7) and the last of (2.8) are evaluated in one additional step. Therefore, the time required for each stage $(j+1)$ is $\tau^{(j+1)}=(2+j)$. Consequently, the total time required for solving the system is given by,

$$\tau=3+\sum_{j=0}^{\mu-1}\tau^{(j+1)}=3+\mu(\mu+3)/2 \tag{2.9}$$

where we added one further step for the product $M_n f^{(\mu)}$.

Obtaining the corresponding number of required processors, however, is slightly more complicated than establishing the time bound (2.9). First, we establish the number of processors required in forming $M_i^{(j+1)}=M_{2i+1}^{(j)}M_{2i}^{(j)}$. Here, we form the products $\hat{L}_{2i+1}^{(j)}U_{2i}^{(j)}$ and $S_{2i+1}^{(j)}U_{2i}^{(j)}$ simultaneously. Forming each column of $\hat{L}_{2i+1}^{(j)}U_{2i}^{(j)}$ involves s inner products of s pairs of vectors of orders $1,2,\cdots,s$. Thus, each column requires $\sum_{k=1}^{s}k=s(s+1)/2$ processors, and the product $\hat{L}_{2i+1}^{(j)}U_{2i}^{(j)}$ requires $s^2(s+1)/2$ processors. In the meantime, $S_{2i+1}^{(j)}U_{2i}^{(j)}$ involves $sr_{2i+1}^{(j)}$ inner products of vectors of order s. Therefore, the number of processors required for this product is $s^2r_{2i+1}^{(j)}=s^2r_i^{(j+1)}$. Hence, the matrix products in (2.6) and (2.7) require

$$\pi'=\tfrac{1}{2}s^2(s+1)+s^2r_i^{(j+1)}=\tfrac{1}{2}(2n+3)s^2-\tfrac{1}{2}(4i+3)s^3 \tag{2.10}$$

processors. After the above products are formed, we obtain the sum $(S_{2i+1}^{(j)}U_{2i}^{(j)}+V_{2i}^{(j)})$. This requires $sr_{2i}^{(j)}$ processors, or

$$\pi''=(n+1)s-2(i+1)s^2 \tag{2.11}$$

processors. Recalling that $s=2^j$, one can easily verify that for $j=0,1,\cdots,\mu-2$ and $i=1,2,\cdots,n/(2s)-1$, the number of processors required for evaluating $M_i^{(j+1)}$ is given by

$$\pi_i^{(j+1)}=\max\{\pi',\pi''\}=\pi'. \tag{2.12}$$

Similarly, we can show that evaluating $f^{(j+1)}$ requires

$$\pi_0^{(j+1)}=\tfrac{1}{2}(2n+3)s-\tfrac{3}{2}s^2 \tag{2.13}$$

processors. Consequently, each stage $(j+1)$ for $j=0,1,\cdots,\mu-2$, requires

$$\pi^{(j+1)}=\sum_{k=0}^{n/(2s)-1}\pi_k^{(j+1)}=\tfrac{3}{2}s^3-\tfrac{1}{4}(5n+12)s^2+\tfrac{1}{4}(n^2+7n+6)s \tag{2.14}$$

and the whole algorithm requires,

$$\pi=\max\left[\max_{0\leqq j\leqq\mu-2}\{\pi^{(j+1)}\},\ n(n+1)/2\right]$$

processors. For $n \geqq 16$ the maximum is achieved for $s = n/8$, yielding

$$\pi = \frac{n}{64}\left(\frac{15}{16}n^2 + 11n + 12\right) \tag{2.15}$$

or $\pi = n^3/68 + O(n^2)$ processors. For $n \geqq 256$, $\pi \leqq n^3/64$.

Hence, we have established the following theorem.

THEOREM 2.1. *The triangular system of equations $Lx = f$, where L is a lower triangular matrix of order n, can be solved in $\tau = \frac{1}{2}\log^2 n + \frac{3}{2}\log n + 3$ steps using no more than $\pi = (15/1024)n^3 + O(n^2)$ processors.* □

LEMMA 2.2 *Let L be defined as in Theorem 2.1. Then L^{-1} can be obtained in the same time required for solving the system $Lx = f$, using no more than $(21n^3 + 60n^2)/128$ processors.* □

Proof. The time bound is trivially established by Theorem 2.1. However, the maximum number of processors used occurs at $s = n/4$, rather than $n/8$, yielding the above number of processors. In fact, we can show that for h right-hand sides, $h \geqq n$, the maximum number of processors is given by

$$\pi = \frac{1}{128}n^3 + \left(\frac{5h+3}{32}\right)n^2 + \left(\frac{3h}{8}\right)n \tag{2.16}$$

processors.

The number of *operations* (number of nodes in the computational graph), in the above parallel algorithm can be computed to be

$$\omega = \tfrac{2}{21}n^3 + O(n^2). \tag{2.17}$$

Now, a lemma in Brent [2] states that, if we have $\hat{\pi} < \pi$ processors then the time required for solving the triangular system is given by

$$\hat{\tau} \leqq \tau + (\omega - \tau)/\hat{\pi}. \tag{2.18}$$

If $\hat{\pi} = n^3/\sigma$, then

$$\hat{\tau} \leqq \tau + \tfrac{2}{21}\sigma(1+\delta) \tag{2.19}$$

where $\delta < 9/n$. For $n \geqq 256$

$$\hat{\tau} \leqq \tau + 0.10\,\sigma. \tag{2.20}$$

Thus, as long as $\sigma = O(\log^2 n)$, $\hat{\tau}$ remains $O(\log^2 n)$.

3. Algorithm II.

In this section we present an algorithm that requires the same time as Algorithm I but roughly twice the number of processors. The advantage of this algorithm is that it can be adapted to band systems. Let $L^{(0)} \equiv DL$ and $f^{(0)} \equiv Df$ where $D = \operatorname{diag}(\lambda_{11}^{-1}, \lambda_{22}^{-1}, \cdots, \lambda_{nn}^{-1})$. We form matrices $D^{(j)}$, $(j = 0, 1, \cdots, \mu - 1)$, such that if

$$L^{(j+1)} = D^{(j)}L^{(j)} \quad \text{and} \quad f^{(j+1)} = D^{(j)}f^{(j)} \tag{3.1}$$

then $L^{(\mu)} = I$ and $x = f^{(\mu)}$. Each matrix $L^{(j)}$ is of the form

$$L^{(j)} = \begin{bmatrix} I_s & & & & \\ G_{21}^{(j)} & I_s & & & \\ G_{31}^{(j)} & G_{32}^{(j)} & I_s & & \\ \hline & & & & \\ G_{n/s,1}^{(j)} & G_{n/s,2}^{(j)} & \cdots & G_{n/s,n/s-1}^{(j)} & I_s \end{bmatrix} \tag{3.2}$$

where $s = 2^j$, and $G_{ij}^{(0)} = \lambda_{ij}/\lambda_{ii}$. Therefore, $(D^{(j)})^{-1} = \operatorname{diag}(L_1^{(j)}, L_2^{(j)}, \cdots, L_{n/s}^{(j)})$, where

$$L_i^{(j)} = \begin{bmatrix} I_s & \\ G_{2i,2i-1}^{(j)} & I_s \end{bmatrix}.$$

Thus, for stage $(j+1)$ we have,

$$G_{ik}^{(j+1)} = \begin{bmatrix} I_s & \\ -G_{2i,2i-1}^{(j)} & I_s \end{bmatrix} \begin{bmatrix} G_{2i-1,2k-1}^{(j)} & G_{2i-1,2k}^{(j)} \\ G_{2i,2k-1}^{(j)} & G_{2i,2k}^{(j)} \end{bmatrix} \tag{3.3}$$

for $i, k+1 = 2, 3, \cdots, n/(2s)$, and

$$f_i^{(j+1)} = \begin{bmatrix} I_s & \\ -G_{2i,2i-1}^{(j)} & I_s \end{bmatrix} \begin{bmatrix} f_{2i-1}^{(j)} \\ f_{2i}^{(j)} \end{bmatrix} \tag{3.4}$$

for $i = 1, 2, \cdots, n/(2s)$. We can simultaneously evaluate the products $G_{2i,2i-1}^{(j)} G_{2i-1,2k-1}^{(j)}$, $G_{2i,2i-1}^{(j)} G_{2i-1,2k}^{(j)}$, and $G_{2i,2i-1}^{(j)} f_{2i-1}^{(j)}$ in $\tau' = 1 + \log s = (1+j)$ steps using $2s^3$ processors for the first two products and s^2 processors for the last one for a total of $\pi' = s^2(2s+1)$ processors. Performing the subtraction of the two pairs of matrices in (3.3) and the pair of vectors in (3.4) requires $\tau'' = 1$ step using $\pi'' = s(2s+1) < \pi'$ processors. Since we have $n(n-2s)/(8s^2)$ matrices $G_{ik}^{(j+1)}$, and $n/(2s)$ vectors $f_i^{(j+1)}$, then each stage $j+1$ requires $\tau^{(j+1)} = \tau' + \tau'' = (2+j)$ steps using $\pi^{(j+1)} = \frac{1}{4}sn^2 - \frac{1}{2}s^2 n + \frac{1}{2}sn$ processors. Forming $L^{(0)}$ and $f^{(0)}$ requires one step using $n(n+1)/2$ processors. Hence, the total time for solving the linear system $Lx = f$ is given by

$$\tau = 1 + \sum_{j=0}^{\mu-1} (2+j) = 1 + \mu(\mu+3)/2 \tag{3.5}$$

using $\pi = \max[\max_{0 \le j \le \mu-1}\{\pi^{(j+1)}\}, n(n+1)/2]$. For $n \geqq 16$ the maximum occurs at $s = n/4$, yielding

$$\pi = \frac{n^3}{32} + \frac{n^2}{8} \tag{3.6}$$

processors. Now we introduce the following theorem for solving banded lower triangular systems.

THEOREM 3.1. *Let L be a banded lower triangular matrix of order n and bandwidth $(m+1)$, $(\lambda_{ik} = 0$ for $i - k > m)$. Then the system $Lx = f$ can be solved in $\tau = (2 + \log m) \log n - \frac{1}{2}(\log^2 m + \log m) + 3$ steps using no more than $\pi = \frac{1}{2}m^2 n + O(mn)$ processors.* $\square$

Proof. The matrix L and the vector f can be written in the form

$$(3.7)\qquad L=\begin{bmatrix} L_1 & & & & \\ R_1 & L_2 & & & \\ & R_2 & L_3 & & \\ & & \cdots & \cdots & \\ & & & R_{n/m-1} & L_{n/m} \end{bmatrix},\qquad f=\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n/m} \end{bmatrix}$$

where L_i and R_i are $m\times m$ lower triangular and upper triangular matrices, respectively. Premultiplying both sides of $Lx=f$ by the matrix $D=\operatorname{diag}[L_i^{-1}]$, we obtain the system $L^{(0)}x=f^{(0)}$ where

$$(3.8)\qquad L^{(0)}=\begin{bmatrix} I_m & & & & \\ G_1^{(0)} & I_m & & & \\ & G_2^{(0)} & I_m & & \\ & & \cdots & \cdots & \\ & & & G_{n/m-1}^{(0)} & I_m \end{bmatrix}$$

and

$$(3.9)\qquad \begin{aligned} &L_1f_1^{(0)}=f_1,\\ &L_i[G_{i-1}^{(0)},f_i^{(0)}]=[R_{i-1},f_i],\qquad i=2,3,\cdots,n/m. \end{aligned}$$

From Theorem 2.1 and Lemma 2.2 we can show that solving the systems in (3.9) requires $\tau^{(0)}=\frac{1}{2}\log^2 m+\frac{3}{2}\log m+3$ steps, and by (2.16), using $\pi^{(0)}=(21/128)m^2n+O(mn)$ processors.

Now we follow Algorithm II and form the sequences $L^{(j+1)}=D^{(j)}L^{(j)}$, and $f^{(j+1)}=D^{(j)}f^{(j)}$ for $j=0,1,\cdots,\log(n/m)-1$. Each matrix $L^{(j)}$ is of the form,

$$(3.10)\qquad L^{(j)}=\begin{bmatrix} I_r & & & & \\ G_1^{(j)} & I_r & & & \\ & G_2^{(j)} & I_r & & \\ & & \cdots & \cdots & \\ & & & G_{n/r-1}^{(j)} & I_r \end{bmatrix}$$

where $r=2^j\cdot m$. Therefore, $D^{(j)}=\operatorname{diag}[(L_i^{(j)})^{-1}]$ $(i=1,2,\cdots,n/(2r))$ in which

$$(3.11)\qquad L_i^{(j)^{-1}}=\begin{bmatrix} I_r & \\ -G_{2i-1}^{(j)} & I_r \end{bmatrix}.$$

Hence, for the stage $j+1$, we have

$$(3.12)\qquad G_i^{(j+1)}=\begin{bmatrix} 0 & G_{2i}^{(j)} \\ 0 & -G_{2i+1}^{(j)}G_{2i}^{(j)} \end{bmatrix},\qquad i=1,2,\cdots,\frac{n}{2r}-1,$$

and

$$(3.13)\qquad f_i^{(j+1)}=\begin{bmatrix} f_{2i-1}^{(j)} \\ -G_{2i-1}^{(j)}f_{2i-1}^{(j)}+f_{2i}^{(j)} \end{bmatrix},\qquad i=1,2,\cdots,\frac{n}{2r}.$$

Observing that all except the last m columns of each matrix $G_i^{(j)}$ are zero, then we can evaluate simultaneously $G_{2i+1}^{(j)}G_{2i}^{(j)}$ and $G_{2i-1}^{(j)}f_{2i-1}^{(j)}$ for all i, in $1+\log m$ steps using $\pi' = \frac{1}{2}m(m+1)n - rm^2$ processors. Finally, we need one more subtraction to evaluate $f_i^{(j+1)}$ and $G_i^{(j+1)}$, for all i, using $\pi'' = \pi'/m$ processors. Therefore, $\tau^{(j+1)} = 2 + \log m$ steps and $\pi^{(j+1)} = \max\{\pi', \pi''\} = \frac{1}{2}m(m+1)n - rm^2$ processors. The total time is thus given by

$$\tau = \sum_{j=0}^{\log(n/m)} \tau^{(j)} = \mu(2+\nu) - \frac{\nu}{2}(\nu+1) + 3 \tag{3.14}$$

where $\mu = \log n$ and $\nu = \log m$, with $\pi = \max_j \{\pi^{(j)}\}$ processors. For $m = n/2$, $\pi \equiv \pi^{(0)}$, otherwise

$$\pi \equiv \pi^{(1)} = \tfrac{1}{2}m(m+1)n - m^3 \tag{3.15}$$

processors.

The algorithm of Theorem 3.1 requires $\omega = m^2 n \log(n/(2m)) + O(mn \log(n/(2m))$ operations. Again, we can show that for $\hat{\pi} = m^2 n/\sigma < \pi$ processors, the time required for the algorithm of Theorem 3.1 becomes (Brent [2]),

$$\hat{\tau} \leqq \tau + 2\sigma \log \frac{n}{2m}. \tag{3.16}$$

Hence, provided $\sigma = O(\log m)$, τ remains $O(\log m \log n)$.

4. Error analysis.

We present an error analysis of the algorithm in § 2. Let $*$ denote any of the four arithmetic operations; then a floating-point operation satisfies,

$$fl(\xi_1 * \xi_2) = (\xi_1 * \xi_2)(1+\delta)$$

where $|\delta| \leqq \varepsilon$, ε is the unit roundoff

$$\varepsilon = \begin{cases} \frac{1}{2}\beta^{1-t} & \text{(rounded operations)} \\ \beta^{1-t} & \text{(chopped operations)} \end{cases}$$

in which β is the radix of the machine and t is the length of the fraction.

We introduce the following lemma for establishing a bound on the absolute error of an inner product performed on a parallel computer. $O(\varepsilon^2)$ terms are neglected here and throughout this section.

LEMMA 4.1. *Let $a = \{\alpha_i\}$ and $b = \{\beta_i\}$ be two vectors of order n. Performing the inner product $a^t b$ in $(1+\log n)$ steps, then we obtain*

$$|fl(a^t b) - a^t b| \leqq (1+\log n)\varepsilon |a|^t |b| \tag{4.1}$$

where $|a|$ denotes the vector with elements $|\alpha_i|$, $i = 1, 2, \cdots, n$.

Proof. The proof is rather simple. Let $a^t = (\alpha_1, \alpha_2, \cdots, \alpha_n)$, and $b^t = (\beta_1, \beta_2, \cdots, \beta_n)$. The multiplications $\gamma_i^{(0)} = \alpha_i\beta_i$, $i = 1, 2, \cdots, n$, are performed simultaneously in one step. The extended addition $\sum_{i=1}^n \gamma_i^{(0)}$ is obtained in $\mu = \log n$ steps. At each step $(j+1)$, $j = 0, 1, \cdots, \mu - 1$, we have

$$\gamma_i^{(j+1)} = \gamma_{2i}^{(j)} + \gamma_{2i-1}^{(j)} \tag{4.2}$$

for $i=1,2,\cdots,n/2^j$, i.e., $\gamma_i^{(j)}=\sum_{k=k_1}^{k_2}\gamma_k^{(0)}$ where $k_1=1+(i-1)2^j$ and $k_2=i2^j$. If $\tilde{\gamma}_i^{(j)}\equiv fl(\gamma_i^{(j)})$, then

$$\tilde{\gamma}_i^{(0)}=\gamma_i^{(0)}+\theta_i^{(0)} \tag{4.3}$$

where $|\theta_i^{(0)}|\leqq\varepsilon|\gamma_i^{(0)}|$. Furthermore,

$$\tilde{\gamma}_i^{(j+1)}=\tilde{\gamma}_{2i}^{(j)}+\tilde{\gamma}_{2i-1}^{(j)}+\hat{\theta}_i^{(j+1)} \tag{4.4}$$

where, $|\hat{\theta}_i^{(j+1)}|\leqq\varepsilon\sum_{k=\hat{k}_1}^{\hat{k}_2}|\gamma_k^{(0)}|$ in which $\hat{k}_1=1+(i-1)2^{j+1}$, and $\hat{k}_2=i2^{j+1}$. Combining (4.3) and (4.4), we have

$$\tilde{\gamma}_1^{(\mu)}=a^tb+\theta_1^{(\mu)} \tag{4.5}$$

with

$$|\theta_1^{(\mu)}|\leqq(1+\mu)\varepsilon|a|^t|b| \tag{4.6}$$

proving the lemma. Note that since the exact value of an inner product can be very small, the computed inner product might have a very large relative error.

At each stage j of the algorithm $(j=1,2,\cdots,\mu)$, where $\mu=\log n$, we perform inner products of vectors u and v the maximum order of which is $(1+2^j)$. Performing each inner product in $1+\log(1+2^j)=(j+2)$ steps, we have from Lemma 4.1,

$$|fl(u^tv)-u^tv|\leqq(j+2)\varepsilon|u|^t|v|. \tag{4.7}$$

Without loss of generality, we assume that $\|L\|\leqq 1$. If $\tilde{M}_i^{(j)}\equiv fl(M_i^{(j)})$ and $\tilde{f}^{(j)}\equiv fl(f^{(j)})$, then the algorithm of Theorem 2.1 is given by

$$\begin{aligned}\tilde{M}_i^{(j+1)}&=\tilde{M}_{2i+1}^{(j)}\tilde{M}_{2i}^{(j)}+E_i^{(j+1)}, \qquad i=1,2,\cdots,\frac{n}{2^{j+1}}-1,\\ \tilde{f}^{(j+1)}&=\tilde{M}_1^{(j)}\tilde{f}^{(j)}+e^{(j+1)}\end{aligned} \tag{4.8}$$

for $j=0,1,\cdots,\mu-1$, where $E_i^{(j+1)}$ is lower triangular and

$$\begin{aligned}|E_i^{(j+1)}|&\leqq(j+2)\varepsilon|M_{2i+1}^{(j)}|\,|M_{2i}^{(j)}|,\\ |e^{(j+1)}|&\leqq(j+2)\varepsilon|M_1^{(j)}|\,|f^{(j)}|.\end{aligned} \tag{4.9}$$

Again, $|M|$ denotes the matrix with elements $|\mu_{kl}|$ and an inequality in matrix or vector form applies separately to corresponding elements on each side.

We introduce the following lemma in order to establish bounds on the norms of $E_i^{(j+1)}$ and $e^{(j+1)}$.

LEMMA 4.2. *Suppose* $L=\begin{bmatrix}L_{11}&0\\L_{21}&L_{22}\end{bmatrix}$ *with* $\|L\|\leqq 1$. *Then*

$$\|L_{22}^{-1}\|\leqq\|L^{-1}\|, \quad \textit{and} \quad \|\tilde{L}^{-1}\|\leqq\|L^{-1}\| \tag{4.10}$$

where $\tilde{L}=\begin{bmatrix}L_{11}&0\\L_{21}&I\end{bmatrix}$.

Proof. Since

$$L^{-1}=\begin{bmatrix}L_{11}^{-1}&0\\-L_{22}^{-1}L_{21}L_{11}^{-1}&L_{22}^{-1}\end{bmatrix}$$

then, it is clear that the first of (4.10) is satisfied. Also,

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & I \end{bmatrix} = L \begin{bmatrix} I & 0 \\ 0 & L_{22}^{-1} \end{bmatrix}.$$

Consequently,

$$\left\| \begin{pmatrix} L_{11} & 0 \\ L_{21} & I \end{pmatrix}^{-1} \right\| \leqq \|L\| \, \|L^{-1}\| \leqq \|L^{-1}\|.$$

From the above lemma we see that

$$\|M_i^{(j)}\| \leqq \|L^{-1}\|. \tag{4.11}$$

Also, from the fact that $f^{(j)} = M_1^{(j-1)} f^{(j-1)}$, (2.1), and (2.3) we obtain

$$f^{(j)} = M_{-1+2^j} \cdots M_2 M_1 f = M_{2^j}^{-1} \cdots M_{n-2}^{-1} M_{n-1}^{-1} M_n^{-1} x.$$

Hence,

$$\|f^{(j)}\| \leqq \|L\| \, \|x\| \leqq \|x\|. \tag{4.12}$$

Consequently, from (4.9)

$$\begin{aligned} \|E_i^{(j+1)}\| &\leqq (j+2)(1+2^j)\varepsilon \|L^{-1}\|^2, \\ \|e^{(j+1)}\| &\leqq (j+2)(1+2^j)\varepsilon \|L^{-1}\| \, \|x\|. \end{aligned} \tag{4.13}$$

For a complete analysis we need to consider the structure of $E_i^{(j+1)}$. From (2.4)–(2.7) we see that the first $\rho_{j+1} = (2i+1)2^j - 1$ rows of $E_i^{(j+1)}$ are zero; hence $M_l E_i^{(j+1)} = E_i^{(j+1)}$ for $l \leqq \rho_{j+1}$. Furthermore, the last $(n - \rho_{j+1})$ columns of $E_i^{(j+1)}$ are zero; thus $E_i^{(j+1)} M_l = E_i^{(j+1)}$ for $l \gneqq \rho_{j+1}$. Let $L(k, l) = M_k^{-1} M_{k+1}^{-1} \cdots M_l^{-1}$, and $\gamma_{ij} = i2^j - 1$. We verify that

$$\tilde{M}_i^{(j)} = L^{-1}(1, \gamma_{i+1,j})(I + P_i^{(j)}) L(1, \gamma_{ij}) \tag{4.14}$$

where

$$P_i^{(j)} = \sum_{s=1}^{j} \sum_{k=i2^{j-s}}^{(i+1)2^{j-s}-1} L(1, \gamma_{k+1,s}) E_k^{(s)} L^{-1}(1, \gamma_{k,s}). \tag{4.15}$$

Unfortunately, however, we cannot take advantage of the above properties of $E_i^{(j+1)}$ to simplify the expression under the summation in (4.15). It is trivial to show that (4.14) holds for $j = 1$. From (4.14) and the first of (4.8) we obtain

$$\tilde{M}_i^{(j+1)} = L^{-1}(1, \gamma_{i+1,j+1})(I + P_{2i}^{(j)} + P_{2i+1}^{(j)}) L(1, \gamma_{i,j+1}) + E_i^{(j+1)} + O(\varepsilon^2).$$

Using (4.15) we easily show that

$$P_i^{(j+1)} = P_{2i}^{(j)} + P_{2i+1}^{(j)} + L(1, \gamma_{i+1,j+1}) E_i^{(j+1)} L^{-1}(1 + \gamma_{i,j+1}) + O(\varepsilon^2).$$

Neglecting terms of $O(\varepsilon^2)$ we establish (4.14).

From the second equation in (4.8), we have

$$\tilde{f}^{(\mu+1)} = (M_n \tilde{M}_1^{(\mu)} \cdots \tilde{M}_1^{(1)} M_1^{(0)}) f + \sum_{k=1}^{\mu+1} L^{-1}(2^k, n) e^{(k)} \tag{4.16}$$

where $L^{-1}(2n, n) \equiv I$. Also, from (4.14)

$$M_n \tilde{M}_1^{(\mu)} \cdots \tilde{M}_1^{(1)} M_1^{(0)} = L^{-1}(I+P)$$

where $P = \sum_{j=1}^{\mu-1} P_i^{(j)}$ is a lower triangular matrix. If $\tilde{x}$ is the computed solution then (4.16) reduces to

$$\tilde{x} - x = L^{-1}Pf + e \tag{4.17}$$

in which

$$e = \sum_{k=1}^{\mu+1} L^{-1}(2^k, n)e^{(k)}. \tag{4.18}$$

Let us define the constants α_1 and α_2 by

$$\alpha_1 = \tfrac{1}{4}n^2 \log n + O(n \log n),$$
$$\alpha_2 = \tfrac{1}{2}n . \log n + O(n).$$

Thus, from (4.12)–(4.15) and Lemma 4.2 we have

$$\begin{aligned} \|L^{-1}Pf\| &\leqq \alpha_1 \varepsilon \|L^{-1}\|^3 \|x\|, \\ \|e\| &\leqq \alpha_2 \varepsilon \|L^{-1}\|^2 \|x\|. \end{aligned} \tag{4.19}$$

Hence, from (4.17)

$$\|\tilde{x} - x\| / \|x\| \leqq \alpha_1 \varepsilon \|L^{-1}\|^3. \tag{4.20}$$

Premultiplying both sides of (4.17) by L, we obtain

$$L\tilde{x} - f = Pf + Le. \tag{4.21}$$

From the fact that $f = Lx$, (4.9), (4.15), and (4.18), we have

$$|Pf + Le| \leqq |\tilde{L}|\,|x| \tag{4.22}$$

where $\tilde{L}$ is a lower triangular matrix. Furthermore, from (4.13),

$$\|\tilde{L}\| \leqq \alpha_1 \varepsilon \|L^{-1}\|^2.$$

Hence, there exists a perturbation δL such that

$$(L + \delta L)\tilde{x} = f \tag{4.23}$$

where

$$\|\delta L\| \leqq \alpha_1 \varepsilon \|L^{-1}\|^2. \tag{4.24}$$

If we remove the assumption that $\|L\| \leqq 1$, then (4.24) becomes

$$\|\delta L\| \leqq \alpha_1 \varepsilon \kappa^2(L) \|L\| \tag{4.25}$$

in which $\kappa(L)$ is the condition number of L, and (4.20) becomes

$$\|\tilde{x} - x\| / \|x\| \leqq \alpha_1 \varepsilon \kappa^3(L) \|L\|. \tag{4.26}$$

The presence of $\kappa^2(L)$ and $\kappa^3(L)$ in (4.25) and (4.26) certainly tarnishes the error bounds. We have simulated Algorithm I and performed many numerical experiments on an IBM 360 Model 75 in long-precision (machine precision $\simeq 2.22\times 10^{-16}$). In most of the cases tested, both the sequential and parallel solutions produced relative errors of the same magnitude; this, in spite of the fact that for some of these test cases $\kappa(L)$ was of the order 10^{18}. Thus, in general we consider the error bounds (4.25) and (4.26) rather pessimistic. It is not difficult, however, to find examples where the parallel solution is inferior to the sequential one.

Consider the system,

$$\begin{bmatrix} 1 & & & \\ 1.07 & 1 & & \\ 1.02 & 1.10 & 1 & \\ \alpha & -(\alpha+\delta) & \beta & 1 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2.07 \\ 3.12 \\ -4.00017 \end{bmatrix}$$

where $\alpha = 0.993\times 10^{14}$, $\beta = -0.34\times 10^{-3}$, and $\delta = 4.0$. The condition number of this matrix is $O(10^{28})$. The exact solution of this system is $\xi_1=\xi_2=\xi_3=1$, and $\xi_4 = 0.00017$. Both the sequential and parallel algorithms obtain ξ_1, ξ_2, and ξ_3 to full accuracy. While the sequential algorithm obtained ξ_4 correctly, the parallel algorithm obtained for ξ_4 the value -0.0039. Another example is provided by the banded lower triangular matrix L_{32}, of order 32, with elements: $\lambda_{ii} = 1$, $\lambda_{i+1,i} = -4$, $\lambda_{i+2,i} = 1$, and zero elsewhere. For a certain right-hand side, we know the exact solution to 12 significant digits. In Table 1 we compare the number of correct digits in some of the components of the sequential and parallel solutions $\xi_j^{(s)}$ and $\xi_j^{(p)}$, respectively. It may be of interest to note that $\kappa(L_4) = O(10^2)$, $\kappa(L_8) = O(10^4)$, $\kappa(L_{16}) = O(10^9)$, and $\kappa(L_{32}) = O(10^{16})$.

Moreover, we compute a number, $\omega_{P/S}$, which compares the effects of rounding errors upon the parallel and sequential algorithms; see Miller [8]. For several systems $Lx = f$ we are able to obtain values of $\omega_{P/S}$ of the order 10^4. This means that the parallel algorithm may produce a solution which cannot be computed by the sequential algorithm except with much larger rounding errors.

Finally, in view of the above, it seems to be advisable when implementing Algorithm I to have means for floating-point accumulation of inner-products using a double length accumulator; see Wilkinson [10]. This will have a favorable practical effect on stabilizing the algorithm and reducing the error bounds.

TABLE 1

j	# of correct digits in $\xi_j^{(s)}$	# of correct digits in $\xi_j^{(p)}$
4	12	12
8	12	12
16	11	7
28	4	1
32	3	0

Acknowledgment. We thank the referees for their comments and careful reading of the manuscript. We wish also to thank Mr. John Larson for performing the numerical experiments.

REFERENCES

[1] A. BORODIN AND I. MUNRO, *Computational Complexity of Algebraic and Numeric Problems*, American Elsevier, New York, 1975.

[2] R. P. BRENT, *The parallel evaluation of general arithmetic expressions*, J. Assoc. Comput. Mach., 21 (1974), pp. 201–206.

[3] S. C. CHEN AND D. J. KUCK, *Time and parallel processor bounds for linear recurrence systems*, IEEE Trans. Computers, C-24 (1975), pp. 701–717.

[4] D. HELLER, *On the efficient computation of recurrence relations*, Institute for Computer Applications in Science and Engineering Rep. (ICASE), NASA Langley Res. Center, Hampton, VA. June 1974.

[5] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Blaisdell, New York, 1964.

[6] L. HYAFIL AND H. KUNG, *Parallel algorithms for solving triangular linear systems with small parallelism*, CDS Rep., Carnegie-Mellon University, Pittsburgh, December 1974.

[7] P. M. KOGGE, *The numerical stability of parallel algorithms for solving recurrence problems*, Tech. Rep. 315, Digital Systems Lab., Stanford Electronics Labs., Stanford, California, 1972.

[8] W. MILLER, *Roundoff analysis by direct comparison of two algorithms*, this Journal, 13 (1976), pp. 382–392.

[9] S. E. ORCUTT, *Parallel solution methods for triangular linear systems of equations*, Tech. Rep. 77, Digital Systems Lab., Stanford Electronics Labs., Stanford, California, 1974.

[10] J. H. WILKINSON, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1963.