ELSEVIER Applied Mathematics and Computation 133 (2002) 547–557

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

PCR algorithm for parallel computing minimum-norm (T) least-squares (S) solution of inconsistent linear equations ☆

Yimin Wei [a,*], Guorong Wang [b]

[a] Department of Mathematics, Fudan University, Shanghai 200433, People's Republic of China

[b] Department of Mathematics, Shanghai Normal University, Shanghai, People's Republic of China

Abstract

This paper presents a new highly parallel algorithm for computing the minimum-norm (T) least-squares (S) solution of inconsistent linear equations $Ax = b(A \in \mathbb{R}_r^{m\times n}$, $b \notin \mathbb{R}(A))$. By this algorithm the solution $x = A_{S,T}^{+}b$ is obtained in $T = (1+m)(1+\log_2 m) + n(6+\log_2(n-r+1)+\log_2 m+\log_2 n) - r(1+\log_2 n)$ steps with $P = mn$ processors when $m \geqslant 2(n-1)$ and with $P = 2n(n-1)$ processors otherwise.

Keywords: Parallel algorithm; The minimum-norm (T) least-squares (S) solution; Inconsistent linear equations; Weighted Moore–Penrose inverse; Time complexity

1. Introduction

Let $A \in \mathbb{R}_n^{n\times n},\ b \in \mathbb{R}^n$. Then the unique solution $x = A^{-1}b$ of the nonsingular equations

$$Ax = b \tag{1.1}$$

is given, componentwise, by

$$x_i = \det A_i / \det A_{n+1} \quad (i = 1, 2, \ldots, n), \tag{1.2}$$

☆ Project 19901006 and 19971057 supported by National Natural Science Foundation of China. The first author is also supported by Science Foundation of Computational Physics.

* Corresponding author.

E-mail address: ymwei@fudan.edu.cn (Y. Wei).

PII: S0096-3003(01)00256-9

where

$$A_{n+1} = A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},$$

$$A_i = \begin{bmatrix} a_{11} & \cdots & a_{1,i-1} & b_1 & a_{1,i+1} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2,i-1} & b_2 & a_{2,i+1} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & \cdots & a_{n,i-1} & b_n & a_{n,i+1} & \cdots & a_{nn} \end{bmatrix}. \tag{1.3}$$

The A_i $(i = 1, 2, \ldots, n+1)$ are all identical, except in one column. Eq. (1.2) is called Cramer's rule [2].

Cramer's rule is abandoned due to its inefficiency on serial processors. A highly parallel algorithm for the solution $x = A^{-1}b$ of nonsingular equations (1.1) is presented in [4] and is called the Parallel Cramer's Rule (PCR). An elimination method akin to that used in Gaussian Elimination (GE) is used in PCR algorithm.

Sridhar [4] shows that the solution is obtained by PCR algorithm in n steps with no more than $2n(n-1)$ processors. The parallelism in PCR algorithm is analysed under the assumption of an unbounded parallel computational model and issues relating to interprocessor communication and task scheduling have not been considered.

PCR algorithm has been tested on a number of systems of linear equations and was found to exhibit stability and accuracy identical to Parallel Gaussian Elimination (PGE) [5,7]. Since PCR algorithm provides approximately twice the speedup over PGE with only twice the number of processors, it offers exciting possibilities for VLSI implementation as well as MIMD parallel processing structures.

Wang [9] gave a parallel algorithm for computing the minimum-norm least-squares solution of inconsistent linear equations $Ax = b$, where $A \in \mathbb{R}_r^{m\times n}$, $b \notin \mathbb{R}(A)$.

The concept of the weighted M–P inverse is defined as follows.

Let $A \in \mathbb{R}_r^{m\times n}$, and S, T be positive definite matrices of order m and n, respectively. Then there is a unique matrix $X \in \mathbb{R}_r^{m\times n}$ satisfying

$$AXA = A, \quad XAX = X, \quad (SAX)^{\mathrm{T}} = SAX, \quad (TXA)^{\mathrm{T}} = TXA, \tag{1.4}$$

where "T" denotes the transpose. This X is called the weighted M–P inverse of A, and is denoted by $X = A_{S,T}^{+}$ [12–16].

2. Algorithm

Let $A \in \mathbb{R}_r^{m\times n}$, $m \geqslant n$, $b \in \mathbb{R}^m$ and S, T be positive definite matrices of order m and n, respectively. If the linear equation

$$Ax = b \tag{2.1}$$

is inconsistent, then the minimum-norm (T) least-squares (S) solution of (2.1) is

$$x = A_{S,T}^{+} b. \tag{2.2}$$

For any $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^m$, $A(i \to x)$ denotes the matrix obtained by replacing ith column of A with x, $\mathbb{R}(A)$ and $\mathbb{N}(A)$ denote the range and the null space of A, respectively.

Cramer's rule for computing the minimum-norm (T) least-squares (S) solution of (2.1) is given in [6,10].

A condensed Cramer's rule for the minimum-norm (T) least-squares (S) solution of (2.1) is presented as follows.

Theorem 1. *Let $A \in \mathbb{R}_r^{m\times n}$, $b \in \mathbb{R}^m$, $b \notin \mathbb{R}(A)$, and S, T be positive definite matrices of order m and n, respectively, and $V \in \mathbb{R}_{n-r}^{n\times(n-r)}$ be a matrix whose columns form the basis of $T\mathbb{N}(A)$. We define*

$$B = VV^{\mathrm{T}}. \tag{2.3}$$

Then B satisfies

$$\mathbb{R}(B) = T\mathbb{N}(A) = T\mathbb{N}(A^{\mathrm{T}}SA), \quad \mathbb{N}(B) = T^{-1}\mathbb{R}(A^{\mathrm{T}}) = T^{-1}\mathbb{R}(A^{\mathrm{T}}SA), \tag{2.4}$$

$$(A^{\mathrm{T}}SA + B)^{-1} = (A^{\mathrm{T}}SA)_{T^{-1},T}^{+} + B_{T^{-1},T}^{+} \tag{2.5}$$

and the solution x of the nonsingular equations

$$(A^{\mathrm{T}}SA + B)x = A^{\mathrm{T}}Sb \tag{2.6}$$

is the minimum-norm (T) least-squares (S) solution of (2.1). *If we denote*

$$C = A^{\mathrm{T}}SA + B \in \mathbb{R}^{n\times n}, \quad d = A^{\mathrm{T}}Sb \in \mathbb{R}^n \tag{2.7}$$

then x is given, componentwise, by

$$x_i = \det\ C_i / \det\ C_{n+1} \quad (i = 1, 2, \ldots, n), \tag{2.8}$$

where

$$C_{n+1} = C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix},$$

$$C_i = \begin{bmatrix} c_{11} & \cdots & c_{1,i-1} & d_1 & c_{1,i+1} & \cdots & c_{1n} \\ c_{21} & \cdots & c_{2,i-1} & d_2 & c_{2,i+1} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{n1} & \cdots & c_{n,i-1} & d_n & c_{n,i+1} & \cdots & c_{nn} \end{bmatrix}. \tag{2.9}$$

Proof. By assumption,

$$\begin{aligned} \mathbb{R}(B) &= \mathbb{R}(VV^{\mathrm{T}}) = \mathbb{R}(V) = T\mathbb{N}(A) = T\mathbb{N}(S^{1/2}A) = T\mathbb{N}[(S^{1/2}A)^{\mathrm{T}}(S^{1/2}A)] \\ &= T\mathbb{N}(A^{\mathrm{T}}SA), \\ \mathbb{N}(B) &= \mathbb{N}(VV^{\mathrm{T}}) = \mathbb{N}(V^{\mathrm{T}}) = [\mathbb{R}(V)]^{\perp} = [T\mathbb{N}(A)]^{\perp} = [\mathbb{N}(AT^{-1})]^{\perp} \\ &= \mathbb{R}[(AT^{-1})^{\mathrm{T}}] = \mathbb{R}(T^{-1}A^{\mathrm{T}}) = T^{-1}\mathbb{R}(A^{\mathrm{T}}) = T^{-1}\mathbb{R}(A^{\mathrm{T}}S^{1/2}) \\ &= T^{-1}\mathbb{R}[(A^{\mathrm{T}}S^{1/2})(A^{\mathrm{T}}S^{1/2})^{\mathrm{T}}] = T^{-1}\mathbb{R}(A^{\mathrm{T}}SA). \end{aligned}$$

Thus (2.4) is true. From [1] and (2.4), we have

$$(A^{\mathrm{T}}SA)^{+}_{T^{-1},T} = (A^{\mathrm{T}}SA)^{(1,2)}_{T^{-1}\mathbb{R}(A^{\mathrm{T}}SA),T\mathbb{N}(A^{\mathrm{T}}SA)} = (A^{\mathrm{T}}SA)^{(1,2)}_{\mathbb{N}(B),\mathbb{R}(B)}, \tag{2.10}$$

$$B^{+}_{T^{-1},T} = B^{(1,2)}_{T^{-1}\mathbb{R}(B),T\mathbb{N}(B)}. \tag{2.11}$$

It follows from (2.4), (2.10) and (2.11) that

$$(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}B = 0, \quad B^{+}_{T^{-1},T}(A^{\mathrm{T}}SA) = 0. \tag{2.12}$$

From [1,11], $(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}(A^{\mathrm{T}}SA)$ and $B^{+}_{T^{-1},T}B$ are the projectors,

$$(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}(A^{\mathrm{T}}SA) = P_{T^{-1}\mathbb{R}(A^{\mathrm{T}}SA),\mathbb{N}(A^{\mathrm{T}}SA)} = P_{\mathbb{N}(B),T^{-1}\mathbb{R}(B)}, \tag{2.13}$$

$$B^{+}_{T^{-1},T}B = P_{T^{-1}\mathbb{R}(B),\mathbb{N}(B)} \tag{2.14}$$

and

$$P_{\mathbb{N}(B),T^{-1}\mathbb{R}(B)} + P_{T^{-1}\mathbb{R}(B),\mathbb{N}(B)} = I. \tag{2.15}$$

From (2.12)–(2.15), we have

$$[(A^{\mathrm{T}}SA)^{+}_{T^{-1},T} + B^{+}_{T^{-1},T}](A^{\mathrm{T}}SA + B) = I.$$

Hence (2.5) is true.

From (2.4) and (2.11), we have $B^{+}_{T^{-1},T}A^{\mathrm{T}}Sb = 0$. Let

$$X = (A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}S.$$

Then

$$AXA = A(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}SA = AP_{T^{-1}\mathbb{R}(A^{\mathrm{T}}SA),\mathbb{N}(A^{\mathrm{T}}SA)} = AP_{T^{-1}\mathbb{R}(A^{\mathrm{T}}),\mathbb{N}(A)} = A, \tag{2.16}$$

$$XAX = (A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}SA(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}S = (A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}S = X, \tag{2.17}$$

$$(SAX)^{\mathrm{T}} = [SA(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}S]^{\mathrm{T}} = S^{\mathrm{T}}A(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}S^{\mathrm{T}} = SAX, \tag{2.18}$$

$$(TXA)^{\mathrm{T}} = [T(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}SA]^{\mathrm{T}} = T(A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}SA = TXA. \tag{2.19}$$

From (2.16)–(2.19), we have

$$X = A^{+}_{S,T}.$$

Therefore the unique solution of (2.6) is

$$\begin{aligned} x &= (A^{\mathrm{T}}SA + B)^{-1}(A^{\mathrm{T}}Sb) = (A^{\mathrm{T}}SA)^{+}_{T^{-1},T}A^{\mathrm{T}}Sb + B^{+}_{T^{-1},T}A^{\mathrm{T}}Sb \\ &= (A^{\mathrm{T}}SA)_{T^{-1},T}A^{\mathrm{T}}Sb = A^{+}_{S,T}b \end{aligned}$$

and it is just the minimum-norm (T) least-squares (S) solution of (2.1).

Using Cramer's rule to (2.6), we obtain (2.8) immediately. $\square$

By Theorem 1, a highly parallel algorithm for the minimum-norm (T) least-squares (S) solution of inconsistent linear equations (2.1) is given as follows.

We assume, for convenience of exposition, that the order of the linear equations is expressible as $n = 2^{l}$, where l is an integer. Later, we shall see that the algorithm is valid for arbitrary n.

First of all, we need an algorithm for computing the basis of $\mathbb{N}(A)$ [3].

A matrix $H \in \mathbb{R}^{n\times n}$ is said to be in Hermite echelon form if its elements h_{ij} satisfy the following conditions:

1. $h_{ij} = 0,\ i > j$.
2. h_{ii} is either 0 or 1.
3. if $h_{ii} = 0$ then $h_{ik} = 0$ for every $k,\ 1 \leqslant k \leqslant n$.
4. if $h_{ii} = 1$ then $h_{ki} = 0$ for every $k \neq i$.

For a given matrix $A \in \mathbb{R}^{n\times n}$, the Hermite echelon form H_A obtained by row reducing A is unique; $\mathbb{N}(A) = \mathbb{N}(H_A) = \mathbb{R}(I - H_A)$ and a basis for $\mathbb{N}(A)$ is the set of nonzero columns of $I - H_A$.

Algorithm 1. Let $A \in \mathbb{R}^{n\times n}_{r}$. This algorithm is designed for computing $U \in \mathbb{R}^{n\times(n-r)}_{n-r}$ whose columns form the basis for $\mathbb{N}(A)$.

1. Row reduce A to its Hermite echelon form H_A.
2. Form $I - H_A$, and select the nonzero columns $u_1, u_2, \ldots, u_{n-r}$ from this matrix, $U = (u_1, u_2, \ldots, u_{n-r})$.

The following is PCR algorithm for computing minimum-norm (T) least-squares (S) solution $x = A^+_{S,T}b$ of inconsistent linear equations (2.1) in parallel.

Algorithm 2. Let $A \in \mathbb{R}^{m\times n}_r$, $b \in \mathbb{R}^m$, $b \notin \mathbb{R}(A)$, and S, T be positive definite matrices of order m and n respectively. This algorithm is designed for computing $x = A^+_{S,T}b$.

1. Compute $A^{\mathrm{T}}S$ in parallel.
2. Compute $d = (A^{\mathrm{T}}S)b$ and $C = (A^{\mathrm{T}}S)A$ in parallel.
3. By Algorithm 1 to compute $U = (u_1, u_2, \ldots, u_{n-r})$ whose columns form the basis for $\mathbb{N}(A) = \mathbb{N}(A^{\mathrm{T}}SA) = \mathbb{N}(C)$ in parallel.
4. Compute $V = TU$ in parallel, where $V = (v_1, v_2, \ldots, v_{n-r})$ whose columns form the basis for $T\mathbb{N}(A)$.
5. Compute $C \leftarrow C + VV^{\mathrm{T}}$ in parallel.
6. Form L-form and R-form matrices

$$\mathrm{LC} = (d \vdots C), \quad \mathrm{RC} = (C \vdots d).$$

LC and RC differ only in the position of the d vector, which appears on the left-hand side in LC and on the right-hand side in RC.

We shall triangulate LC in parallel by subtracting fractions of the pivotal column from c_{nn} as the first pivot, until all rows to the left of pivot $c_{kk}(k = n/2 + 1)$ have been reduced to zero.

$$\mathrm{LC} \to \left[\begin{array}{ccccccc}
d_1 & c_{11} & \cdots & c_{1,n/2} & c_{1,n/2+1} & \cdots & c_{1,n} \\
\vdots & \vdots & & \vdots & \vdots & & \vdots \\
d_{n/2} & c_{n/2,1} & \cdots & c_{n/2,n/2} & c_{n/2,n/2+1} & \cdots & c_{n/2,n} \\
0 & 0 & \cdots & 0 & c_{n/2+1,n/2+1} & \cdots & c_{n/2+1,n} \\
\vdots & \vdots & & & & & \vdots \\
0 & 0 & \cdots & \cdots & \cdots & \cdots\ 0 & c_{n,n}
\end{array}\right].$$

We shall triangulate RC in parallel by subtracting fractions of the pivotal row from rows below this row, starting from c_{11} as the first pivot until pivot $c_{kk}(k = n/2)$ is reached, leaving a submatrix of order $(n/2) \times (n/2 + 1)$ below this pivotal row.

$$
\mathrm{RC} \to \begin{bmatrix}
c_{11} & c_{12} & \dots & c_{1,n/2} & c_{1,n/2+1} & \dots & c_{1,n} & d_1 \\
0 & c_{22} & \dots & c_{2,n/2} & c_{2,n/2+1} & \dots & c_{2,n} & d_2 \\
0 & & & \vdots & \vdots & & \vdots & \vdots \\
\vdots & & & c_{n/2,n/2} & c_{n/2,n/2+1} & \dots & c_{n/2,n} & d_{n/2} \\
\vdots & & & 0 & c_{n/2+1,n/2+1} & \dots & c_{n/2+1,n} & d_{n/2+1} \\
\vdots & & & \vdots & \vdots & & \vdots & \vdots \\
0 & \dots & \dots & 0 & c_{n,n/2+1} & \dots & c_{n,n} & d_n
\end{bmatrix}.
$$

In either case these operations are of the general form

$$c_{ij} \leftarrow c_{ij} - c_{ik}c_{kj}/c_{kk}.$$

7. Form new L-form and R-form submatrices of order $(n/2) \times (n/2+1)$ in LC and RC that have not been triangulated

$$
\mathrm{LLC} = \begin{bmatrix}
d_1 & c_{11} & \cdots & c_{1,n/2} \\
\vdots & \vdots & & \vdots \\
d_{n/2} & c_{n/2,1} & \cdots & c_{n/2,n/2}
\end{bmatrix},
$$

$$
\mathrm{RRC} = \begin{bmatrix}
c_{n/2+1,n/2+1} & \cdots & c_{n/2+1,n} & d_{n/2+1} \\
\vdots & & \vdots & \vdots \\
c_{n,n/2+1} & \cdots & c_{n,n} & d_n
\end{bmatrix}
$$

and form two corresponding L-form and R-form matrices

$$
\mathrm{RLC} = \begin{bmatrix}
c_{11} & \cdots & c_{1,n/2} & d_1 \\
\vdots & & \vdots & \vdots \\
c_{n/2,1} & \cdots & c_{n/2,n/2} & d_{n/2}
\end{bmatrix},
$$

$$
\mathrm{LRC} = \begin{bmatrix}
d_{n/2+1} & c_{n/2+1,n/2+1} & \cdots & c_{n/2+1,n} \\
\vdots & \vdots & & \vdots \\
d_n & c_{n,n/2+1} & \cdots & c_{n,n}.
\end{bmatrix}.
$$

As before, LLC, LRC, RLC and RRC triangulated in parallel.

Form new L-form and R-form submatrices LLLC, LLRC, RRLC and RRRC of order $(n/4) \times (n/4+1)$ in LLC, LRC, RLC and RRC that have not been triangulated and form four corresponding L-form and R-form matrices RLLC, RLRC, LRLC and LRRC.

The algorithm therefore recursively doubles the number of submatrices, halves their order in each step. Since, by our assumptions $n = 2^l$, at the $l-1$ step, we form n submatrices of order 2×3.

8. n submatrices of order 2×3 are triangulated in parallel. We take c_{ii} and d_i from above n submatrices of order 2×3. Then

$$x_i = d_i / c_{ii} \quad (i = 1, 2, \ldots, n).$$

Example. Let

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}_3^{5\times 4}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \notin \mathbb{R}(A),$$

$$S = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 1 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 1 & 4 \end{bmatrix}.$$

1.

$$A^{\mathrm{T}}S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad d = A^{\mathrm{T}}Sb = \begin{bmatrix} 0 \\ 6 \\ 1 \\ 1 \end{bmatrix}.$$

2.

$$C = A^{\mathrm{T}}SA = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3.

$$H_{A^{\mathrm{T}}SA} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad I - H_{A^{\mathrm{T}}SA} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

4.

$$V = TU = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

5.

$$C \leftarrow C + VV^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 7 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

6.

$$\mathrm{LC} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 6 & 1 & 7 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 6 & 1 & 7 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\mathrm{RC} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 7 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

7.

$$\mathrm{LLC} = \begin{bmatrix} 0 & 1 & 1 \\ 6 & 1 & 7 \end{bmatrix} \rightarrow \begin{bmatrix} -6/7 & 6/7 & 1 \\ 0 & 0 & 7 \end{bmatrix},$$

$$\mathrm{LRC} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$$\mathrm{RLC} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 7 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 \\ 0 & 6 & 6 \end{bmatrix}, \quad \mathrm{RRC} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

8. $x_1 = (-6/7)/(6/7) = -1, \quad x_2 = 6/6 = 1, \quad x_3 = 1/1 = 1, \quad x_4 = 1/1 = 1.$

3. Complexity

In this section, we discuss the parallel arithmetic complexity of PCR algorithm for computing the minimum-norm (T) least-squares (S) solution of inconsistent linear equations (2.1).

Theorem 2. *By PCR algorithm, the minimum-norm* (T) *least-squares* (S) *solution of* (2.1) *is obtained in* $T = (1+m)(1+\log_2 m) + n(6+\log_2(n-r+1) + \log_2 m + \log_2 n) - r(1+\log_2 n)$ *steps with* $P = mn$ *processors when* $m \geqslant 2(n-1)$ *and with* $P = 2n(n-1)$ *processors otherwise.*

Proof. (1) Parallel computation of $A^{\mathrm{T}}S$ takes $T_1 = m(1+\log_2 m)$ steps and $P_1 = mn$ processors.

(2) Parallel computation of $(A^{\mathrm{T}}S)b$ and $(A^{\mathrm{T}}S)A$ takes $T_2 = (1+n) \times (1+\log_2 m)$ steps and $P_2 = mn$ processors.

(3) Parallel computation of U takes $T_3 = 2n$ steps and no more than $P_3 = (n-1)^2$ processors [8].

(4) Parallel computation of $V = TU$ takes $T_4 = (n-r)(1+\log_2 n)$ steps and $P_4 = n^2$ processors.

(5) Parallel computation of $C \leftarrow C + VV^{\mathrm{T}}$ takes $T_5 = n(1+\log_2(n-r+1))$ steps and $P_5 = n(n-r+1)$ processors.

(6)–(7) Parallel triangulation of L-form and R-form matrices recursively takes $T_6 = n-1$ steps and $P_6 = 2n(n-1)$ processors [4].

(8) Parallel computation of $x_i = d_i/c_{ii}$ $(i = 1, 2, \ldots, n)$ takes $T_7 = 1$ steps and $P_7 = n$ processors.

Thus

$$T = \sum_{i=1}^{7} T_i = (1+n+m)(1+\log_2 m) + (n-r)(1+\log_2 n) + n(4+\log_2(1+n-r))$$

steps and

$$P = \max P_i = \begin{cases} mn, & m \geqslant 2(n-1), \\ 2n(n-1), & m \leqslant 2(n-1). \end{cases}$$

The case for arbitrary n and the problem relating to pivoting in the PCR algorithm are discussed in [4,9], and so are omitted here. □

References

[1] A. Ben-Isreal, T.N.E. Greville, Generalized Inverses: Theory and Applications, Wiley–Interscience, New York, 1974.

[2] D. Heller, A survey of parallel algorithms in numerical linear algebra, SIAM Rev. 20 (1978) 740–777.

[3] B. Noble, Applied Linear Algebra, Prentice-Hall, New Jersey, 1969.

[4] M.K. Sridhar, A new algorithm for parallel solution of linear equations, Inform. Process. Lett. 24 (1987) 407–412.

[5] M.A. Srinivas, Optimal parallel scheduling of Gaussian eliminations DAG's, IEEE Trans. Comput. C-32 (1983) 109–117.

[6] C.C. Verghese, A'Cramer Rule' for least-norm least-square-error solution of inconsistent linear equations, Linear Algebra Appl. 48 (1982) 315–316.
[7] O. Wing, J.W. Huang, A computational model for parallel solution of linear equations, IEEE Trans. Comput. C-29 (1980) 632–638.
[8] G. Wang, S. Lu, Fast parallel algorithm for computing generalized inverses A^+ and A^+_{MN}, J. Comput. Math. 6 (1988) 348–354.
[9] G. Wang, PCR algorithm for parallel computing munimum-norm least-squares solutions of inconsistent linear equations, Numer. Math. J. Chinese Univ., Ser. B 2 (1993) 1–10.
[10] G. Wang, A Cramer rule for minimum-norm (T) least-squares (S) solutions of inconsistent linear equations, Linear Algebra Appl. 74 (1986) 213–218.
[11] G. Wang, Weighted Moore–Penrose inverse, Drazin inverse and group inverse of the Kroneckor product $A \otimes B$ and some applications, Linear Algebra Appl. 250 (1997) 39–50.
[12] Y. Wei, Perturbation bound of singular linear system, Appl. Math. Comput. 105 (1999) 211–220.
[13] Y. Wei, Recurrent neural networks for computing weighted Moore–Penrose, Appl. Math. Comput. 116 (2000) 279–287.
[14] Y. Wei, H. Wu, J. Wei, Successive matrix squaring algorithm for parallel computing the weighted generalized inverse A^+_{MN}, Appl. Math. Comput. 116 (2000) 289–296.
[15] Y. Wei, H. Wu, Expression for the perturbation of the weighted Moore–Penrose inverse, Comput. Math. Appl. 39 (2000) 13–18.
[16] Y. Wei, H. Wu, The representation and approximation of the weighted Moore–Penrose inverse, Appl. Math. Comput. 121 (2001) 17–28.