Complexity of Parallel Block Gauss Jordan Algorithm

Salman H. Abbas

Department of Mathematics University of Bahrain P O Box 32038, Bahrain

ABSTRACT

In this paper the block Gauss-Jordan algorithm for solving linear systems of equations is presented in the proposed for multitasking. The paper indicates how the availability of more than one processor must change our approach to the problem of computing solutions of linear systems.

KEYWORDS

Block Gauss-Jordan algorithm; multitasking; multiprocessor Sequent Balance, predicted time.

1. INTRODUCTION.

The problem we want to consider is how to compute the numerical solution of the linear system.

$$A\underline{x} = \underline{b} \qquad (1)$$

Where A is a dense matrix of size $n \times n$ and $\underline{x}$, $\underline{b}$ are vectors of size $n \times 1$. We consider here parallel solution techniques, which are suitable for MIMD local memory, architectures. On sequential machine the method requires $\frac{n^3}{2} + O(n^2)$ multiplication and divisions. On parallel machine with $(N-1)(N+M)$ processors [Heller, 1978] shows that the method required $3N+1$ steps, where A is of size $M \times N$. If only N processors are available [Quinn, 1988] shows that the Gauss – Jordan algorithm without pivoting requires $N^2 + 2NM + M$ arithmetic steps.

In the present paper, we show that parallel block Gauss – Jordan algorithm is much faster than parallel LU-decomposition and if the number of processors matches the number of block rows q, then block Gauss – Jordan algorithm is better suited to parallel implementation than any methods of Gaussian type in computing numerical solutions of linear systems.

A comprehensive list of references is available in the recent study, see for example [Abbas, (1990); 2000,2001; Bader, Gehrke (1991); Barrodale, Stuart, (1977), Charmberlain (1987); Geist, Romine (1988); Heath, Romine (1988); Heller, (1978); Lord, Kowalik, Kumar, (1983); Ortega et al (1988); Pease (1976); Purushotam et al, (1992); Quinn, (1988); Rivers et al, (1990); Sameh, Kuck, (1978)].

2. THE ALGORITHM

Our purpose in this section is to explain the idea of parallel algorithm for solving (1) on multiprocessor computer. If we consider the above system (1), partition the matrix A into blocks of size $w \times w$ and the vectors $\underline{x}$, $\underline{b}$ into blocks of $w \times 1$, where $1 \le i \le q$ and q the number of blocks rows $\left(= \frac{n}{w}\right)$, then the system becomes:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1q} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2q} \\ A_{31} & A_{32} & & & \\ . & . & & & . \\ . & . & & & . \\ . & . & & & . \\ A_{q1} & A_{q2} & & & A_{qq} \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ . \\ . \\ . \\ \underline{x}_q \end{bmatrix} = \begin{bmatrix} \underline{B}_1 \\ \underline{B}_2 \\ . \\ . \\ . \\ \underline{B}_q \end{bmatrix}$$

In Gauss – Jordan algorithm every block is eliminated from each row of the matrix except the diagonal block, which is made equal to unity. The algorithm consists of four stages:

Stage 1: Reduce the blocks below the diagonal to zero,

For $i = 1...q-1$ loop
  Compute A_{ii}^{-1};
  For $j = i+1...q$ loop
    $M_{ji} := A_{ji} * A_{ii}^{-1}$;
    For $k = i+1...q$ loop
      $A_{jk} := A_{jk} - M_{ji} \times A_{ik}$;
      $\underline{B}_k := \underline{B}_k - M_{ji} \times A_{ik}$;
    end loop;
  end loop;
end loop;

This involves getting zeros below the diagonal and updating the right hand side vector.

Stage 2: Reduce the blocks above the diagonal to zeros:

For $i = q...2$ loop
  For $j = i-1..1$ loop
    $M_{ji} = A_{ji} \times A_{ii}^{-1}$;
  For $k = i..1$ loop
    $A_{jk} = A_{jk} - M_{ji} \times A_{ik}$
    $\underline{B}_k = \underline{B}_k - M_{ji} \times \underline{B}_i$

Stage 3: Reduce the blocks in the diagonal into unit blocks:

For $i = 1..q$ loop
  $A_{ii} = A_{ii}^{-1} \times A_{ii}$
  $\underline{B}_i = A_{ii}^{-1} \times \underline{B}_i$
end loop

Stage 4: Solve the full set
Solve for $i = 1,2,...,q$
$\underline{x}_i = \underline{B}_i$

3. PARALLEL ALGORITHM

In this section we consider the parallel implementation of block Gauss-Jordan algorithm using multitasking. Assuming that P processors are available and $p < q$, then the updated blocks in each block row can be carried in parallel. Therefore, the following procedures are needed:

1- procedure to compute the inverse block A;
2- procedure to multiply two blocks;
3- procedure to multiply a block vector by block;
4- procedure to subtract two blocks vectors.

On this basis the major tasks are as follows:

(I) The first task is to reduce the blocks below the diagonal to zero;
Accept block row (i), block row (j), i;

Call procedure to compute A_{ii}^{-1};

Evaluate the multipliers $M_{ji} - A_{ji} \times A_{ii}^{-1}$ in parallel;

Call a task to update the blocks;

Compute $A_{jk} = A_{jk} - M_{ji} \times A_{jk}$; $\underline{B}_k = \underline{B}_k - M_{ji} \times \underline{B}_i$.

Collect the results.

(II) The second task is to reduce the blocks above the diagonal to zero,
compute the multipliers $M_{ji} = A_{ji} \times A_{ii}^{-1}$:

Update the blocks and the right hand side vector using the following formula:

$A_{jk} = A_{jk} - M_{ji} \times A_{ik}$;
$\underline{B}_k = \underline{B}_k - M_{ji} \times \underline{B}_i$.

(III) The third task to transfer the diagonal into identity blocks;
Compute $A_{ii}^{-1} A_{ii}$
And $\underline{B}_i = A_{ii}^{-1} \times \underline{B}_i$
Finally solve $\underline{x}_i = \underline{B}_i$; $i = 1,...,q$.

For a Multiprocessor Sequent Balance having at least 10 processors, the sequence of computation in case of reducing the blocks below the diagonal to zeros taking place in each processor is described below. The computation can go in parallel in all processors.

Processor 1: Compute M_{ji}

Eliminate the blocks in first block row. Communication of the update blocks between processors.

Processor 2: Eliminate the blocks in second block row, communication of the update blocks between processors, and soon until the 10^{th} processors.

Processor 10: Eliminate the blocks in the 10^{th} block row communication of the update blocks between processors.

In our case $p < q,$ so the remaining blocks are waiting until the processors are free.

4. PREDICTED COSTS

In this section we give the sequential and parallel costs of the block Gauss – Jordan algorithm.

4.1 Sequential Algorithm Cost. In the sequential mode the cost depends only on the number of arithmetic operations required for the algorithm. So, reducing the blocks below the diagonal to zeros requires:

$$t_1 = \sum_{i=1}^{q-1}\left\{\frac{w^3}{2} + \sum_{j=i+1}^{q}\left[w^3 + \sum_{k=i+1}^{q}\left(w^3 + w^2\right)\right]\right\}$$

$$= \sum_{i=1}^{q-1}\left\{\frac{w^3}{2} + \sum_{j=i+1}^{q}\left[w^3 + \left(w^3 + w^2\right)(q-i)\right]\right\}$$

$$= \sum_{i=1}^{q-1}\left[\frac{w^3}{2} + w^3(q-i+1)(q-i+1) + w^2(q-i)^2\right]$$

$$= \frac{w^3}{2}(q-1) + w^3q^2(q-1) - 2w^3q^2\frac{(q-1)}{2} + w^3q(q-1)$$

$$- w^3\frac{q(q-1)}{2} + w^3q\frac{(q-1)(2q-1)}{6} + w^2q^2(q-1)$$

$$- 2w^2\frac{q^2(q-1)}{2} + w^2\frac{q(q-1)(2q-1)}{-6}$$

$$= \frac{w^3}{2}(q-1) + \frac{w^3}{6}(2q^3 - 3q^2 + q + 3q^2 - 3q) + \frac{w^2}{6}(2q^3 - 3q^2 + q)$$

$$= \frac{w^3}{2}(q-1) + \frac{w^3}{6}(2q^3 - 2q) + \frac{w^2}{6}(2q^3 - 3q^2 + q)$$

$$= \frac{w^3}{6}(3q - 3 + 2q^3 - 2q) + \frac{w^2}{6}(2q^3 - 3q^2 + q)$$

$$\approx \frac{w^3q^3}{3} + \frac{w^2q^3}{3} \approx O\left(\frac{w^3q^3}{3}\right).$$

Similarly, reducing the blocks above the diagonal to zeros requires:

$$t_2 = \sum_{i=2}^{q}\sum_{j=1}^{i-1}\left[w^3 + \sum_{k=1}^{i}\left(w^3 + w^2\right)\right] = \sum_{i=2}^{q}\sum_{j=1}^{i-1}\left[w^3 + \left(w^3 + w^2\right)i)\right]$$

$$= \sum_{i=2}^{q}\left[w^3(i-1) + w^3i(i-1) + w^2i(i-1)\right]$$

$$= \sum_{i=2}^{q}\left[w^3(i-1)(i+1) + w^2i(i-1)\right]$$

$$= \frac{w^3}{6}\left(2q^3 + 3q^2 - 5q\right) + \frac{w^2}{6}\left(2q^3 + 3q^2 + q\right) - \frac{w^2q^2}{2} + \frac{w^2q}{2}$$

$$\approx O\left(\frac{w^3q^3}{3}\right)$$

and the cost of reducing the diagonal to identity blocks is:

$$t_3 = \sum_{i=1}^{q}\left[w^3 + w^2\right] = w^3q + w^2q = O(w^3q)$$

Hence, the total cost required in the sequential block Gauss-Jordan algorithm is

$$t = \sum_{i=1}^{3} t_i$$

$$\approx \frac{w^3}{6}\left(2q^3 + q - 3\right) + \frac{w^3}{6}\left(2q^3 + 3q^2 - 5q\right)$$

$$\approx O\left(\frac{2}{3}w^3q^3\right)$$

Hence, the sequential running time for block Gauss-Jordan algorithm is $O\left(\frac{2}{3}w^3q^3\right)$ obtained from the above formula (2) which is more expensive than sequential LU-decomposition obtained in [1].

4.2 Parallel Algorithm Cost. We give the predicted cost of the algorithm mentioned in the previous section in terms of arithmetic operation count, communication cost and data sending. A test program was written to measure these quantities. The estimate of the cost of one arithmetic operation, t_f, is 0.26 millisec, the time to set up one rendezvous, t_r, is 2 millisec and the time to send one data item, t_c, is 0.02 millisec.

The following table describes the number of multiplication and additions, and the number of steps required for each operation. It is assumed that there are at least q processors.

Table 1

Operation	Number of multiplications and additions	Number of steps
Inverse A_{ji}	$\frac{w^3}{2}$	$q-1$
Compute the multipliers	w^3	$q-1$
Reduce the blocks below the diagonal to zero	w^3	$q(q-1)/2$
Reduce the blocks above the diagonal to zero	w^3	$q(q-1)/2$
Reduce the blocks in the diagonal to identity blocks	w^3	$q(q-1)/2$

If we assume that the number of processors $p < q$, then the total number of arithmetic operations should be multiplied by q/p [18]. Therefore, the total cost of arithmetic operations is approximately $\left(\frac{3}{2}w^2q^2/\ p\right)t_f$.

At stage i, we have $2(q-i)$ rendezvous and $3q^2(q-1)w^2/2$ data sent. Therefore, $\sum_{i=1}^{q-1}(q-i)^2$ which is equal to $q(q-1)(2q-1)/3$ rendezvous and $2q(q-1)w^2/3$ data are sent.

Hence, the predicted time for this algorithm is:

$$\left[2q(q-1)(q+1)\right]t_\tau/3+\left[q(q-1)(17q-4)w^2\right]t_c/6+\left[\frac{3}{2}w^3q^2\right]t_f/p \qquad (\ )$$

The above formula shows that the arithmetic operations cost is of $O(w^3q^2/p)$ where as the arithmetic operations cost of parallel LU-decomposition depends on $O(w^3q^3/p)$.

This indicates that the parallel block Gauss-Jordan is much faster than parallel LU-decomposition and if the number of processors matches the number of block rows, q, then block Gauss-Jordan algorithm is better suited to parallel implementation than any methods of Gaussian type.

REFERENCES

Abbas, S.H. *Parallel Algorithms of Linear Systems and Initial Value Problems*, Ph.D. Thesis, University of Liverpool (1990).

Abbas, S.H. (2000) On the cost of sequential and parallel algorithms for solving linear system of equations, *Inter. J. Computer Math*, **74**, 391-403

Abbas, S.H. (2001) Pacallel solution of dense linear equations, *Analele Universitatti din Timisora, Seria Mathematics-Information*, XXXXIX, No. 1, 3-12.

Bader G., Gehrke E. (1991) On the performance of transputer networks for solving linear systems of equations, *Parallel Computing*, **17**, 1397-1407.

Barrodale I., Stuart G. F. (1977) A new vaciant of Gaussian elimination, *J. Ins. Maths. Applics.*, **19**, 39-47.

Charmberlain R.M.*, An Alternative View of LU Factorization with Partial Pivoting on Hyper-cube Multiprocessor*, Hyper-cube Multiprocessor, Hypercube Multiprocessors, SIAM, Philadelphia (1987).

Geist G. A., Romine C.H., (1988) LU-factorization algorithms on distributed memory multiprocessor, *SIAM J. Sci. Stat. Comput*., **9**.

Heath M., Romine C., (1988) Parallel solution of triangular systems on distributed memory multiprocessors, *SIAM J. Sci. Stat. Comput*., **9**.

Heller, (1978) A survey algorithum in numerical linear algebra, *SIAM Review*, **20**, 740-776.

Lord R. E., Kowalik J.S., Kumar S.P., (1983) Solving linear algebric equation on MIMD computer, *J. Assoc. Comput. Mach*., **30**. 103-117.

Ortega J.M., Romine C.H., (1988)The IJK forma of factorization methods, II, *Parallel System, Parallel Computing*, **7**, 149-162.

Pease M. C., (1976) Matrix inversion using parallel processing, *J. Assoc. Complit. Mach*. , **14**, 757-764.

Purushotam B.V. et al, (1992) Performance Estimation of LU factorization on message passing multiprocessors, *Parallel Processing Letters*, **2**. No. 1, 51-60.

Quinn M.J., (1988) Designing *Efficient Algorithm for Parallel Computers*, McGraw Hill International Editions, Computer Science Series.

Rivers et al, (1990) *Gaussian elimination with pivoting on hypercubes,* Parallel Computing, 14, 51-60.

Pease M. C., (1976) Matrix inversion using parallel processing, *J. Assoc. Complit. Mach*. , **14**, 757-764.

Purushotam B.V. et al, (1992) Performance Estimation of LU factorization on message passing multiprocessors, *Parallel Processing Letters*, **2**. No. 1, 51-60.

Quinn M.J., (1988) Designing *Efficient Algorithm for Parallel Computers*, McGraw Hill International Editions, Computer Science Series.

Rivers et al, (1990) *Gaussian elimination with pivoting on hypercubes,* Parallel Computing, 14, 51-60.

Sameh, D.J. (1978), *On stable linear system,* J. Assoc. Comput, Mach., **25** 81-89.

تعقيدات القطاعات المتوازية الخوارزمية جاوس جوردن

سلمان عباس

قسم الرياضيات جامعة البحرين ، البحرين

الملخص

في هذا البحث ، تم استعراض خوارزمية جاوس جوردن لحل منظومة المعادلات الخطية باستخدام الحاسبات المتوازية (عند توفر اكثر من معالج واحد في البرمجة الخوارزمية قد غيرنظرتنا في إيجاد حلول للمنظومة الخطية).