Wuhan University Journal of Natural Sciences Vol.1 No.3/4 1996, 292~308

ON INCREASING THE PARALLELISM IN NUMERICAL ALGORITHMS

D.J. Evans
Parallel Algorithms and Architectures Research Centre
Department of Computer Studies
University of Technology
Loughborough, Leics., U.K.

and

Institute of Software Engineering
Wuhan University
Wuhan
P.R. China.

ABSTRACT

Parallel algorithms have been designed for the past 20 years initially by parallelising existing sequential algorithms for many different parallel architectures. More recently parallel strategies have been identified and utilised resulting in many new parallel algorithms. However the analysis of such algorithms reveals that further strategies can be applied to increase the parallelism. One of these, i.e., increasing the computational capacity in each processing node can reduce the congestion/communication for shared memory/distributed memory multiprocessor systems and dramatically improve the performance of the algorithm.

Two algorithms are identified and studied, i.e., the cyclic reduction method for solving large tridiagonal linear systems in which the odd/even sequence is increased to a 'stride of 3' or more resulting in an improved algorithm. Similarly the Gaussian Elimination method for solving linear systems in which one element is eliminated at a time can be adapted to parallel form in which two elements are simultaneously eliminated resulting in the Parallel Implicit Elimination (P.I.E.) method. Numerical results are presented to support the analyses.

KEYWORDS: Granularity, cyclic and stride reduction, Gaussian and Parallel Implicit Elimination methods.

1. INTRODUCTION

The principal aim of exploiting parallelism in solving large scale scientific and engineering problems is to increase the throughput of the computing system by making it do more than one operation at the same time. To take advantage of multiple processors on a parallel computer it is necessary to restructure the sequential algorithm developed for uni-processor computers and schedule as many computations in parallel as possible. Because of these structural changes and rearrangement of computations the best sequential algorithm and the parallel algorithm may require a different amount of computation for solving the same problem.

For example the sequential algorithm for computing the inner product of 2 vectors $|a_1\ a_2|$ and $|b_1\ b_2|$ i.e.,

$$IP = \underline{a}^T\underline{b} = |a_1\ a_2|\ \begin{vmatrix} b_1 \\ b_2 \end{vmatrix} = a_1 b_1 + a_2 b_2.$$

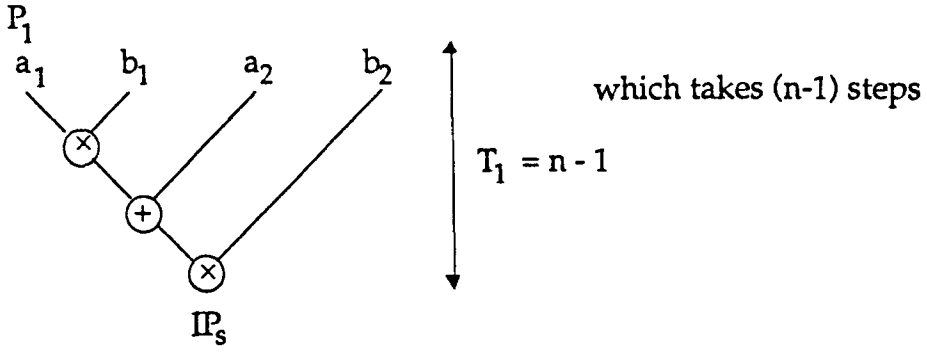

FIGURE 1

This algorithm can be parallelised by the use of the fan-in algorithm as follows:

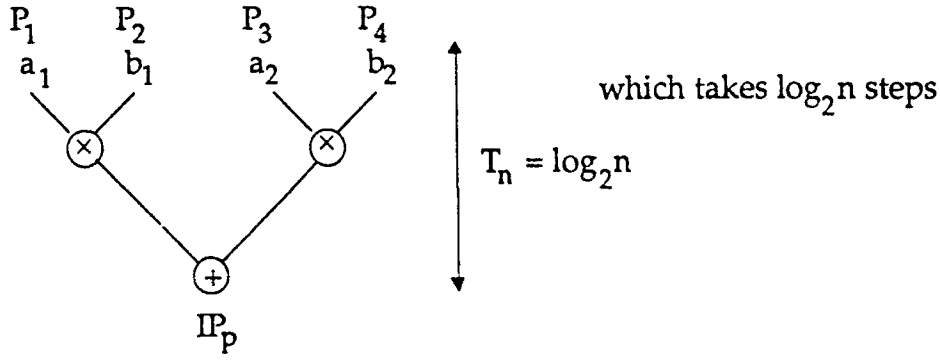

FIGURE 2

The ratio T_1/T_p which gives the speed-up $S_p = n/\log_2 n$ certainly looks impressive. However on further close examination we can notice a number of disadvantages. For instance, as we move down the tree the degree of parallelism is halved, hence the fan-in algorithm does not fully exploit all the processors. Also the solution is formed only in 1 processor which then has to be communicated to the other processors. A better solution is to calculate the partial solutions on all processors thus obviating the need to return the final result.

Further if we now consider the outer product of 2 vectors $|a_1\ a_2|$ and $|b_1\ b_2|$, i.e.,

$$OP = \underline{a}.\underline{b}^T = \begin{vmatrix} a_1 \\ a_2 \end{vmatrix} |b_1\ b_2| = \begin{vmatrix} a_1b_1 & a_1b_2 \\ a_2b_1 & a_2b_2 \end{vmatrix}$$

we see that the amount of computation at each stage increases.

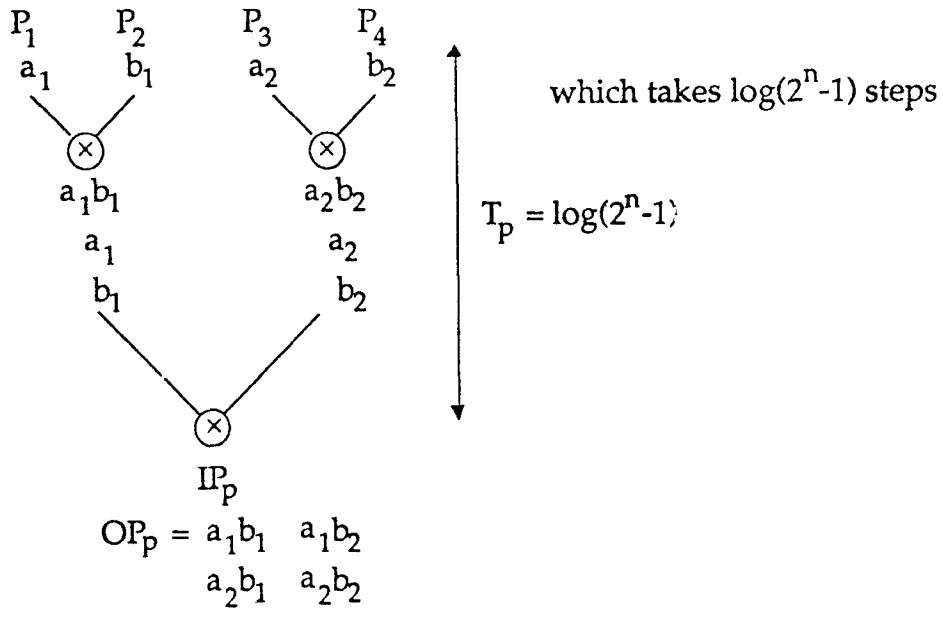

FIGURE 3

Thus the amount of computational work increases as we move down the tree to be computed on less processors which makes the $S_p \sim \frac{n-1}{\log(2^n-1)} \sim 1$ which is not impressive at all!!

Fortunately on todays distributed machines, the processors are connected by by-directional channels which makes it possible for 2 processors to exchange messages simultaneously. Thus, the situation can be restored as shown in Figure 4.

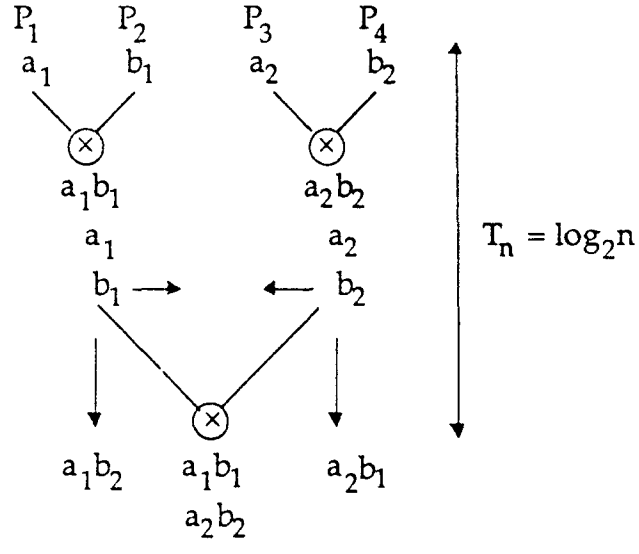

FIGURE 4

This suggests that there is an optimal granularity size for each sub-problem which must be exploited to improve the performance of the parallel algorithm.

2. SOLUTION OF TRIDIAGONAL LINEAR SYSTEMS

The cyclic odd-even reduction method [1] is a well known algorithm for the parallel solution of tridiagonal linear systems. We briefly describe the algorithm for the normalised constant term symmetric system $A\underline{u} = \underline{d}$ of the form,

$$\begin{bmatrix} b & 1 & & & & & \\ 1 & b & 1 & & & & \\ & 1 & b & 1 & & & \\ & & 1 & b & 1 & & \\ & & & 1 & b & 1 & \\ & & & & 1 & b & 1 \\ & & & & & 1 & b \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} . \qquad (2.1)$$

We now multiply equations 2, 4, 6 by -b, adding the two adjacent rows to each of them. Then the system (2.1) becomes

$$\begin{bmatrix} b & 1 & & & & & \\ 0 & b^{[2]} & 0 & 1 & & & \\ & 1 & b & 1 & & & \\ & 1 & 0 & b^{[2]} & 0 & 1 & \\ & & & 1 & b & 1 & \\ & & & 1 & 0 & b^{[2]} & 0 \\ & & & & & 1 & b \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2^{[2]} \\ d_3 \\ d_4^{[2]} \\ d_5 \\ d_6^{[2]} \\ d_7 \end{bmatrix} , \qquad (2.2)$$

where $b^{[2]} = 2 - b^2$, $d_j^{[2]} = d_{j-1} - b\, d_j + d_{j+1}$, for j=2,4,6.

Now since the even rows are independent of the odd rows they may be separated as follows

$$\begin{bmatrix} b^{[2]} & 1 & \\ 1 & b^{[2]} & 1 \\ & 1 & b^{[2]} \end{bmatrix} \begin{bmatrix} u_2 \\ u_4 \\ u_6 \end{bmatrix} = \begin{bmatrix} d_2^{[2]} \\ d_4^{[2]} \\ d_6^{[2]} \end{bmatrix} \qquad (2.3)$$

Applying the above process once more to the system (2.3), i.e. multiplying the second row of (2.3) and adding the first and third rows, the system (2.3) becomes

$$\begin{bmatrix} b^{[2]} & 1 & \\ 0 & b^{[3]} & 0 \\ & 1 & b^{[2]} \end{bmatrix} \begin{bmatrix} u_2 \\ u_4 \\ u_6 \end{bmatrix} = \begin{bmatrix} d_2^{[2]} \\ d_4^{[3]} \\ d_6^{[2]} \end{bmatrix} \tag{2.4}$$

where $b^{[3]} = (2-b^{[2]})^2$, $d_4^{[3]} = d_2^{[2]} - b^{[2]} d_4^{[2]} + d_6^{[2]}$.

Separating again the second row of system (2.4) we obtain

$$b^{[3]} u_4 = d_4^{[3]} , \tag{2.5}$$

which can be easily solved to obtain the solution u_4.

By a process of backsubstitution and in terms of u_4, the first and third rows of system (2.4) gives the values of u_2 and u_6 in the form,

$$b^{[2]} u_2 = d_2^{[2]} - u_4 \text{ and } b^{[2]} u_6 = d_6^{[2]} - u_4 , \tag{2.6}$$

to give the values of u_2 and u_6.

Continuing the backsubstitution in the same way to system (2.1) we can calculate the values of u_1, u_3, u_5 and u_7 to give the complete solution of (2.1).

The above solution process can be applied to any of the u values for $n=2^{m-1}$, 2^m or 2^{m+1} (m integer) choosing each time the even or odd rows of the system. Further the strategy has also been extended to a wider range of generalised tridiagonal systems.

The solution stages of the Cyclic Reduction algorithm are illustrated in Figure 5 and resemble the well known fan-in algorithm.

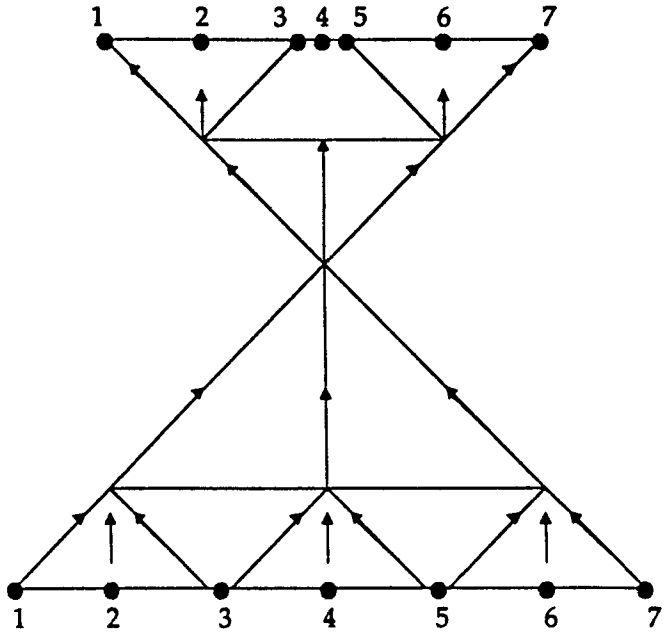

FIGURE 5

2.1 Stride of 3 Reduction Algorithm

It has been shown that the cyclic reduction algorithm consists of successive reductions of the system (2.1) to similar reduced systems in a 'stride of 2'. Similarly, we can consider forming a stride algorithm with reduced systems in a 'stride of 3' manner by increasing the complexity at each node.

We now consider the reduced equations of (2.2), i.e.,

$$\begin{aligned} u_1 + b^{[2]} u_3 + u_5 &= d_3^{[2]}, \\ u_3 + b^{[2]} u_5 + u_7 &= d_5^{[2]}. \end{aligned} \tag{2.7}$$

In addition, we have from (2.1) the equation

$$u_3 + b\, u_4 + u_5 = d_4. \tag{2.8}$$

By adding equations (2.7) we obtain

$$u_1 + (1 + b^{[2]})\, u_3 + (1 + b^{[2]})\, u_5 + u_7 = d_3^{[2]} + d_5^{[2]}. \tag{2.9}$$

Then by multiplying (2.8) by $(1+b^{[2]})$ and subtracting from (2.9) we obtain the final results

$$u_1 - b\,(1 + b^{[2]})\, u_4 + u_7 = d_3^{[2]} + d_5^{[2]} - (1 + b^{[2]})\, d_4. \tag{2.10}$$

If we now rewrite equation (2.10) in the form

$$u_1 = \bar{b}^{[2]} u_4 + u_7 = \bar{d}_4^{[2]}, \tag{2.11}$$

which is the representative equation for a 'stride of 3' algorithm where

$$\bar{b}^{[2]} = -b\,(1 + b^{[2]}) = b\,(b^2 - 3),$$

and $$\bar{d}_4^{[2]} = d_3^{[2]} + d_5^{[2]} - (3 - b^2)\, d_4 = d_2 + d_6 - b\,(d_3 + d_5) - (1 - b^2)\, d_4 \,. \quad (2.12)$$

So the solution procedure for $n = 3^p - 1$ for $p=2$, i.e. $n=8$ is as follows. The system

$$A\,u = d$$

or

$$\begin{bmatrix} b & 1 & & & & & & \\ 1 & b & 1 & & & & & \\ & 1 & b & 1 & & & & \\ & & 1 & b & 1 & & & \\ & & & 1 & b & 1 & & \\ & & & & 1 & b & 1 & \\ & & & & & 1 & b & 1 \\ & & & & & & 1 & b \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \\ d_8 \end{bmatrix} , \quad (2.13)$$

is reduced to

$$\bar{A}^{[2]}\, u^{[2]} = \bar{d}^{[2]} ,$$

$$\begin{bmatrix} \bar{b}^{[2]} & 1 \\ 1 & \bar{b}^{[2]} \end{bmatrix} \begin{bmatrix} u_3 \\ u_6 \end{bmatrix} = \begin{bmatrix} \bar{d}_2^{[2]} \\ \bar{d}_6^{[2]} \end{bmatrix} , \quad (2.14)$$

which can be solved for u_3 and u_6, i.e.,

$$u_6 = [\, \bar{d}_3^{[2]} - \bar{b}^{[2]}\, \bar{d}_6^{[2]}] \,/\, (1 - \bar{b}^{[2]2}) \,, \quad u_3 = [\, \bar{d}_6^{[2]} - \bar{b}^{[2]}\, \bar{d}_3^{[2]}] \,/\, (1 - \bar{b}^{[2]})^2) \,. \quad (2.15)$$

Then, the remaining elements of the solution vector can be obtained by solving the 3 (2×2) subsystems,

$$\begin{bmatrix} b & 1 \\ 1 & b \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 - u_3 \end{bmatrix} , \begin{bmatrix} b & 1 \\ 1 & b \end{bmatrix} \begin{bmatrix} u_4 \\ u_5 \end{bmatrix} = \begin{bmatrix} d_4 - u_3 \\ d_5 - u_6 \end{bmatrix} , \begin{bmatrix} b & 1 \\ 1 & b \end{bmatrix} \begin{bmatrix} u_7 \\ u_8 \end{bmatrix} = \begin{bmatrix} d_7 - u_6 \\ d_9 \end{bmatrix} . \quad (2.16)$$

The solution stages of the 'Stride of 3' Reduction are illustrated in Figure 6 and can be seen to have been reduced when compared with Figure 5.

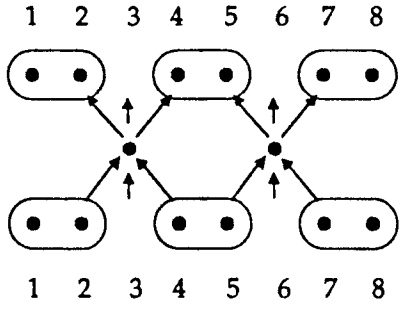

FIGURE 6: Stride of 3 reduction algorithm

Finally it can be shown that the computational complexity of the 'Stride of 3' reduction algorithm is 14 operations/reduction stage for $\log_3 n$ reductions in comparison to 9 operations/reduction stage for cyclic reduction, i.e. $\log_2 n$, which makes the stride reduction algorithm more efficient which has been achieved by increasing the granularity.

In a similar manner, a 'Stride of 4' or quadstride algorithm can be developed.

After the first reduction of the C.R. algorithm we have,

$$u_1 + b^{[2]} u_3 + u_5 = d_3^{[2]} \tag{2.17}$$

$$u_3 + b^{[2]} u_5 + u_7 = d_5^{[2]} \tag{2.18}$$

$$u_5 + b^{[2]} u_7 + u_9 = d_7^{[2]} \tag{2.19}$$

where, as before,

$$b^{[2]} = 2 - (b^{[1]})^2, \qquad b^{[1]} = b, \tag{2.20}$$

$$d_j^{[2]} = d_{j-1} - b^{[1]} d_j + d_{j+1}. \tag{2.21}$$

A further reduction stage is now carried out by multiplying equation (2.18) by $b^{[2]}$ and adding to equations (2.17) and (2.19) to obtain,

$$u_1 + b^{[3]} u_5 + u_9 = d_5^{[3]}, \tag{2.22}$$

and similar equations with a 'Stride of 4' where

$$b^{[3]} = 2 - (b^{[2]})^2 \tag{2.23}$$

$$d_j^{[3]} = d_{j-2}^{[2]} - b^{[2]} d_j^{[2]} + d_{j+2}^{[2]}. \tag{2.24}$$

Alternatively we can avoid the initial reduction stage and proceed to equation (2.22) by reformulating $b^{[3]}$ and $d_j^{[3]}$ as,

$$b^{[3]} = 2 - (2 - (b^{[1]})^2)^2 = 4b^2 - b^4 - 2, \tag{2.25}$$

and $d_j^{[3]} = d_{j-1} + d_{j-3} - b(d_{j-2} + d_{j+2}) - (1 - b^2)\, d_{j-1} + d_{j+1}) + b\,(2 - b^2)d_j$. (2.26)

2.4 The Stride of 5 or Quinstride Reduction Algorithm

The stride reduction algorithm can be extended to 'strides of 5' or more but gains achieved by reducing the number of reduction stages is offset by the increased complexity at each node which makes only the strides of 3 or 4 competitive. These issues as well as the extension to block tridiagonal systems are further discussed in [2].

2.5 Stability of the Stride Reduction Algorithm

It is well known that the cyclic reduction method is unstable and this instability occurs in the updates of the right hand sides of the system. Bunemann [3] presented an improved version of the algorithm in which he achieved greater stability by expressing the update in the following form,

$$d_i = b\,\alpha_i + \beta_i\,, \tag{2.27}$$

where α_i and β_i have a standard form for all i.

We now present a similar decomposition of the R.H.S. for the stride reduction algorithm.

Now we need to write $d^{[2]}$ so that it is of the form,

$$d_i^{[2]} = \bar{b}^{[2]}\, p_i^{[2]} + q_i^{[2]} = b(b^2 - 3)\, p_i^{[2]} + q_i^{[2]}\,. \tag{2.28}$$

From (2.12) we have,

$$d_i^{[2]} = d_{i-2} + d_{i+2} - b\,(d_{i-1} + d_{i+1}) - (1 - b^2)\, d_i\,. \tag{2.29}$$

Now if we let $q_i^{[1]} = d_i$, then (2.29) becomes,

$$\bar{d}_i^{[2]} = q_{i-2}^{[1]} + q_{i+2}^{[1]} - b(q_{i-1}^{[1]} + q_{i+1}^{[1]}) - (1 - b^2)\, q_i^{[1]}\,. \tag{2.30}$$

Combining (2.18) and (2.30) we obtain

$$b(b^2 - 3)\, p_i^{[2]} + q_i^{[2]} = q_{i-2}^{[1]} + q_{i+2}^{[1]} - b\,(q_{i-1}^{[1]} + q_{i+1}^{[1]}) - (1 - b^2)\, q_i^{[1]}\,. \tag{2.31}$$

If we let $p_i^{[2]} = b^{-1}\, q_i^{[1]}$ then (2.31) becomes,

$$p_i^{[2]} = b^{-1}\, q_i^{[1]}\,, \tag{2.32}$$

and $$q_i^{[2]} = q_{i-2}^{[1]} + q_{i+2}^{[1]} - b\,(q_{i-1}^{[1]} + q_{i+1}^{[1]}) - q_i^{[1]} + 3b\, p_i^{[2]}\,. \tag{2.33}$$

Similarly a recursive sequence can be set up for d_i as,

$$\bar{d}_i^{[j]} = \bar{b}^{[j]} p_i^{[j]} + q_i^{[j]}, \text{ for } j=3,4,\ldots, \tag{2.34}$$

which proves the stability of the Stride of 3 reduction algorithm.

3. PARALLEL SOLUTION OF LINEAR EQUATIONS

The application of finite element methods for solving partial differential equations generates a system of simultaneous linear equations of the form,

$$Au = b, \tag{3.1}$$

where A is a coefficient matrix of order n where n is the number of interior mesh points, b is a column vector containing the known sources and boundary terms and u is the unknown column vector.

The most commonly used strategy for the numerical solution is the Gaussian elimination method in which the variables u_i, i=1,2,...,n are successively eliminated one at a time resulting in a triangular system which is finally resolved by a process of backsubstitution in a sequential manner.

This method has remained practically unchanged since it was introduced by Gauss in 1830 apart from the introduction of pivoting to preserve numerical stability against the growth of rounding errors.

3.1 Parallel Implicit Elimination (PIE) Method

A new elimination scheme suitable for the parallel solution of the linear system (3.1) was introduced by Evans [4] who proposed the strategy of simultaneously eliminating 2 elements at the same time which when applied on a parallel computer produces improvements in efficiency.

Step 1: Parallel Implicit Elimination

Let the (n×n) matrix a_{ij} be partitioned in the form, i.e.

$$A = \left[\begin{array}{c|ccc|c} a_{11} & a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ \hline a_{21} & a_{22} & \cdots & & a_{2n} \\ \vdots & & & & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & & a_{n-1,n} \\ \hline a_{n1} & a_{n2} & \cdots & a_{n,n-1} & a_{nn} \end{array}\right] \tag{3.2}$$

be denoted in abbreviated form as,

$$A = \begin{bmatrix} a_{11} & \underline{a}_{1j}^T & a_{1n} \\ \underline{a}_{i1} & A_{ij} & \underline{a}_{in} \\ a_{n1} & \underline{a}_{nj}^T & a_{nn} \end{bmatrix} \qquad i,j = 2(1)n-1 , \tag{3.3}$$

where $\underline{a}_{i1}$ and $\underline{a}_{in}$, i=2,...,n-1 are (n-2) vectors. Similarly for the vectors $\underline{a}_{1j}^T$ and a_{nj}^T, j=2,...,n-1.

Now consider the matrix W consisting of the vector elements $\underline{w}_{i1}$ and $\underline{w}_{in}$, i=2,...,n-1, i.e.,

$$W = \begin{bmatrix} 1 & 0 & 0 \\ -\underline{w}_{i1} & I_{n-2} & -\underline{w}_{in} \\ 0 & 0 & 1 \end{bmatrix} , \tag{3.4}$$

where I_{n-2} is the unit matrix of order n-2.

Now consider the matrix product WA, i.e.,

$$WA = \begin{bmatrix} a_{11} & , & \underline{a}_{1j}^T & , & a_{1n} \\ -a_{11}\underline{w}_{i1}+\underline{a}_{i1}I_{n-2}-a_{n1}\underline{w}_{in} & , & -\underline{w}_{i1}\underline{a}_{1j}^T+A_{ij}-\underline{w}_{in}\underline{a}_{nj}^T & , & -a_{1n}\underline{w}_{i1}+\underline{a}_{in}I_{n-2}-a_{nn}\underline{w}_{in} \\ a_{n1} & , & \underline{a}_{nj}^T & , & a_{nn} \end{bmatrix} . \tag{3.5}$$

Now choose the values of vectors $\underline{w}_{i1}$ and $\underline{w}_{in}$ such that,

$$\begin{aligned} a_{11}\,\underline{w}_{i1} + a_{n1}\,\underline{w}_{in} &= \underline{a}_{i1} \\ a_{1n}\,\underline{w}_{i1} + a_{nn}\,\underline{w}_{in} &= \underline{a}_{in} , \end{aligned} \tag{3.6}$$

for i=2,...,n-1 to give,

$$\underline{w}_{i1} = \frac{-a_{n1}\,\underline{a}_{1n} + a_{nn}\,\underline{a}_{i1}}{a_{11}\,a_{nn} - a_{1n}\,a_{n1}} , \quad \underline{w}_{in} = \frac{a_{11}\,\underline{a}_{in} - a_{1n}\,\underline{a}_{i1}}{a_{11}\,a_{nn} - a_{1n}\,a_{n1}} . \tag{3.7}$$

Thus the simultaneous elimination of the two vector terms $\underline{a}_{i1}$ and $\underline{a}_{in}$ requires the determination of the $\underline{w}_{i1}$ and $\underline{w}_{in}$, for i=2,...,n-1 which requires the solution of (2×2) sets of equations.

These values will yield a Z matrix of the form,

$$Z = \begin{bmatrix} a_{11} & \underline{a}_{1j}^T & a_{1n} \\ 0 & A^1_{n-2,n-2} & 0 \\ a_{n1} & \underline{a}_{nj}^T & a_{nn} \end{bmatrix} \tag{3.8}$$

for j=2,...,n-1, where,

$$A^1_{n-2,n-2} = -\underline{w}_{i1}\,\underline{a}_{1j}^T + A_{n-2,n-2} - \underline{w}_{in}\,\underline{a}_{nj}^T, \tag{3.9}$$

is the reduced matrix of order n-2.

When this procedure is continued for (n-1)/2 steps, then the final matrix Z is of the form,

$$Z = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ & a^{(1)}_{22} & \cdots & a^{(1)}_{2,n-1} & \\ & \ddots & & \ddots & \\ O & & a^{n-1/2}_{\frac{n+1}{2},\frac{n+1}{2}} & & O \\ & \ddots & & \ddots & \\ & a^{(1)}_{n-1,2} & \cdots & a^{(1)}_{n-1,n-1} & \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & a_{nn} \end{bmatrix} \quad \text{for } n = \text{odd} \tag{3.10a}$$

and

$$Z = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ & a^{(1)}_{22} & \cdots & a^{(1)}_{2,n-1} & \\ & \ddots & & \ddots & \\ & & a^{n-1/2}_{\frac{n}{2},\frac{n}{2}} \; a_{\frac{n}{2},\frac{n+1}{2}} & & \\ O & & a_{\frac{n}{2}+1,\frac{n}{2}} \; a_{\frac{n}{2}+1,\frac{n+1}{2}} & & O \\ & \ddots & & \ddots & \\ & a^{(1)}_{n-1,2} & \cdots & a^{(1)}_{n-1,n-1} & \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & a_{nn} \end{bmatrix} \quad \text{for } n = \text{even}\ , \tag{3.10b}$$

The solution of the reduced system is thus,

$$WAu = Wb \quad \text{or} \quad Zu = d\ , \tag{3.11}$$

where d = Wb and Z is given by eqns. (3.10a) and (3.10b) which requires a new solution process which will now be described.

Step 2: Bidirectional Substitution process

The evaluation of the R.H.S. in eqn.(3.12) can be determined after the evaluation of the w_{ij} by a bidirectional substitution process. Thus we have,

$$d = Wb = \begin{bmatrix} 1 & & & 0 \\ w_{21} & 1 & & w_{2n} \\ w_{31} & w_{32} & & w_{3n} \\ \vdots & & & \vdots \\ w_{n-1,1} & 0 & & w_{n-1,n} \\ 0 & & & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} \tag{3.12a}$$

which can be evaluated in the recursive form,

$$\begin{aligned} &d_1 = b_1; && d_n = b_n, \\ &d_2 = w_{21}\, b_1 + b_2 + w_{2n}\, b_n; && d_{n-1} = w_{n-1,1}\, b_1 + b_{n-1} + w_{n-1,n}\, b_n \\ &d_j = \sum_{1}^{j-1} w_{jk}\, b_k + b_j + \sum_{j+1}^{n} w_{jk}\, b_k; && d_{n-1} = \sum_{k=1}^{j-1} w_{n-j,k} + b_k + b_{n-j} + \Sigma\, w_{n-j}\, b_k, \end{aligned} \tag{3.12b}$$

for j=2(1)n-1/2.

After the evaluation of d, the solution process now requires a further stage which is described as follows:

Step 3: Bidirectional Solution Process.

The solution of the reduced systems (3.10) and (3.11) can be described as follows:

We commence with n=odd and equation (3.10a) and the solution process at the central element (n+1/2, n+1/2) immediately yields the solution

$$a_{n+1/2,n+1/2}\, u_{n+1/2} = d_{n+1/2}. \tag{3.13}$$

Then the n-1/2 and n+3/2 equations can be solved as a set of equations as described earlier

$$\begin{aligned} &a^{n-2/2}_{n-1/2,n-1/2}\, u_{n-1/2} + a^{n-2/2}_{n-1/2,n+3/2}\, u_{n+3/2} = d_{n-1/2} - a^{n-2/2}_{n-1/2,n+1/2}\, u_{n+1/2}, \\ &a^{n-2/2}_{n+3/2,n-1/2}\, u_{n-1/2} + a^{n-2/2}_{n+3/2,n+3/2}\, u_{n+3/2} = d_{n+3/2} - a^{n-2/2}_{n+3/2,n+1/2}\, u_{n+1/2}, \end{aligned} \tag{3.14}$$

to yield the solution values for $u_{n-1/2}$ and $u_{n+3/2}$.

This procedure continues bidirectionally from the centre until the final equations to be solved are:

$$a_{11}\,u_1 + a_{1n}\,u_n = d_1 - \sum_{i=2}^{n-1} a_{1i}\,u_i\,,$$

$$a_{n1}\,u_1 + a_{nn}\,u_n = d_n - \sum_{i=2}^{n-1} a_{ni}\,u_i\,, \qquad (3.15)$$

For n=even, we commence with eqn.(3.10b) and we see immediately that by omitting the initial stage (3.13) the procedure is identical with the case n=odd as described above.

3.2 Parallel Pivoting Scheme

For the first step of the parallel pivoting process we find a row i_0 such that

$$|a_{i_0 1}| = \max_{1\le i\le n} |a_{i1}|\,,$$

and exchange the places of row i_0 and row 1. Secondly, we now find a row number j_0 in the last column such that

$$a^{(1)}_{j_0 n} = \max_{2\le i\le n} |a^{(1)}_{jn}|$$

where the entries $a^{(1)}_{jn} = a_{jn} - \dfrac{a_{j1}\,a_{1n}}{a_{11}}$, j=2,...,n are obtained from the elimination of the first column but are not returned. This ensures that the maximum (2×2) pivot, i.e. $\begin{vmatrix} a_{11} & a_{1n} \\ a_{n1} & a_{nn} \end{vmatrix}$, is used in the parallel elimination. We now proceed to carry out the operations described in equation (3.7) where we have ensured that $|w_{in}| < 1$, for i=2,...,n-1, which implies numerical stability against rounding error growth in the parallel elimination process.

A comparison of the solution strategies of the Gaussian Elimination (GE) and Parallel Implicit Elimination (PIE) methods are given in Figure 7.

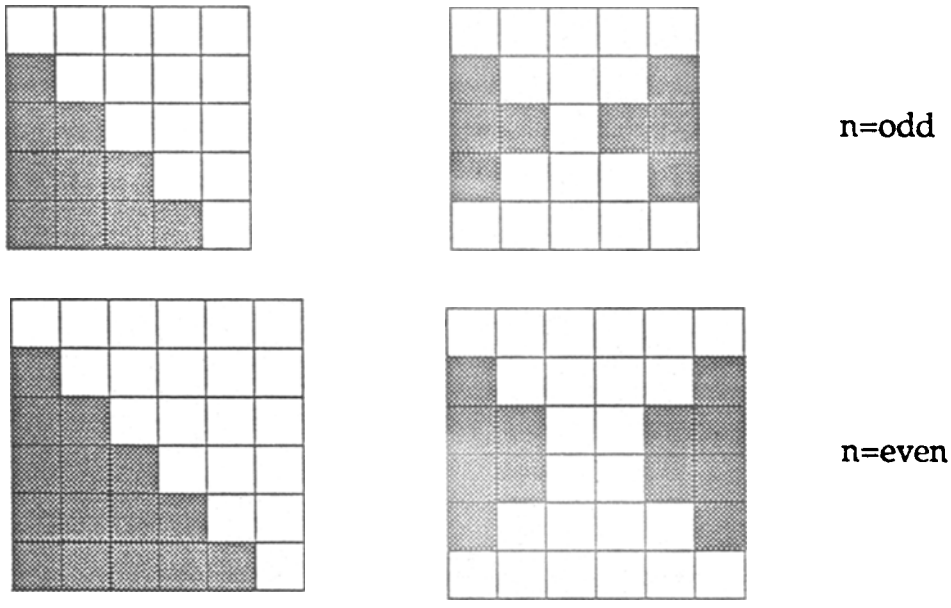

FIGURE 7

3.3 Numerical Results

The numerical tests were completed in single precision on the following matrix A

$$A = a_{ii} = 2N\ ;\ a_{ij} = |i - j|\ ,\ \text{for } i,j=1,2,\ldots,N \text{ for } N=50, 100, 200, 400.$$

The average timing results (in secs.) for the Gaussian Elimination method for P processors where P=1, 2, 5, 10 are tabulated in Table 1.

The parallelisation of the algorithm was achieved by suitably partitioning the rows among the available processors. Similarly, the timing results for the Parallel Implicit Elimination method are given in Table 2 for the same values of P.

Finally, the comparison of the speed-up values for the GE and PIE methods are given in Table 3.

Matrix	No. of Processors P			
Order N	1	2	5	10
50	3.99	2.32	1.49	1.49
100	28.59	15.07	6.91	5.48
200	223.16	111.92	47.53	26.6
400	1790.12	911.29	374.2	199.29

TABLE 1: Computation Time (in secs.) of the Gaussian Elimination Method

Matrix Order N	No. of Processors P			
	1	2	5	10
50	4.01	2.27	1.44	1.41
100	28.28	14.62	6.74	4.39
200	215.47	108.81	45.25	26.69
400	1697.92	858.94	349.69	179.67

TABLE 2: Computation Time (in secs.) of the Parallel Implicit Elimination Method

Matrix Order N	GE			PIE		
	No. of Processors P			No. of Processors P		
	2	5	10	2	5	10
50	1.720	2.678	2.678	1.767	2.785	2.844
100	1.897	4.137	5.217	1.934	4.196	6.442
200	1.994	4.695	8.389	1.980	4.762	8.073
400	1.964	4.784	8.982	1.977	4.856	9.450

TABLE 3: Comparison of Speed-Up Values for GE and PIE Methods

4. CONCLUSIONS

It can be seen from the results presented that greater parallelism results from increasing the granularity computation at each processor node, however this is offset by the increased complexity at each node. Thus, the determination of the optimal granularity size for each algorithm results in increased processor performance.

REFERENCES

[1] R. Hockney and G.H. Golub, A Fast Direct Solution of Poisson's Equation using Fourier Analysis, *J.A.C.M.*, 12 (1965), 95.

[2] D.J. Evans, Design of Parallel Numerical Algorithms, p.39-74 in *Parallel Computing 91*, edit. D.J. Evans, G.R. Joubert and H. Liddell, Elsevier Science Pubs., 1992.

[3] O. Bunemann, A Compact Non-iterative Poisson Solver, S.U.I.P.R., *Report 294*, Stanford Univ., Calif., (1969).

[4] D.J. Evans, Implicit Matrix Elimination (IME) Schemes, *Int.Jour.Comp.Maths.*, 48 (1993), 229-237.