American Journal of Applied Sciences 4 (5): 300-306, 2007
ISSN 1546-9239
© 2007 Science Publications

Parallel Approaches for Intervals Analysis of Variable Statistics in Large and Sparse Linear Equations with RHS Ranges

Peerayuth Charnsethikul
Operations Research and Management Science Units, Industrial Engineering Department
Kasetsart University, Bangkok 10903, Thailand

Abstract: This study proposes an algorithm capable of working in parallel for solving large and sparse linear equations under given right hand side (RHS) ranges. A comparative study to the direct linear programming method is reported theoretically, computationally and discussed. Moreover, the approach can be adapted for the system under domain decompositions structure leading to a better efficiency experimentally.

Key words: RHS ranges, large and sparse linear equations, parallel approaches, domain decompositions

INTRODUCTION

This paper considers a system of linear equations, $AX = b$ with $\det(A) \neq 0$ and $l \leq b \leq u$ where $A = [a_{ij}]$ is an $n \times n$ matrix, $b = [b_i]$ is an $n \times 1$ uncertain vector lying between vectors $l = [l_j]$ and $u = [u_j]$. The problem is to solve for the possible range of $X = [x_j]$. Mathematically, it can be formulated as the following linear programming (LP) problems.

$$\begin{array}{ll} \text{Min (Max)} & x_k,\ k = 1,2,\ldots,n \qquad \text{(P1)} \\ \text{Subject to} & \sum_{j=1}^{n} a_{ij}x_j - b_i = 0 \quad,\ i = 1,2,\ldots,n \\ & x_j \text{ unrestricted},\ l_j \leq b_j \leq u_j\ ,\ j = 1,2,\ldots,n \end{array}$$

The above model becomes more complex when n is large. Usually, this situation arises in approximately solving linear boundary partial differential equations. Applying finite approximation schemes normally leads to the result of very large and sparse linear equations. By representing uncertainties of equation right hand sides and boundary conditions as a set of statistical confidence intervals, the solution as a set of related variable ranges can be solved using the above formulation. Directly, the problem can be solved as a set of independent 2n LP problems. However, this paper will present another approach theoretically equivalent but computationally much faster as n grows and the proposed method is suitable for use in parallel architecture. Additionally, the approach is illustrated for further applications on obtaining confidence intervals of variable variance/covariance matrix for a given ranges of right hand side variance/covariance.

Linear programming (LP) has been one of the major tools in solving real world resource allocation problems. Its origin and early development historically were described and presented extensively by Dantzig[1]. The problem of determining the confidence interval of statistics also has long been studied by statisticians[2]. A large and sparse system of linear equations normally arises in finite approximations scheme for solving boundary value ordinary and partial differential equation basically found in engineering analysis and computational physics. The resulting system is sometimes too large to be handled by the direct approach and the iterative indirect methods are frequently more appropriate[3,4]. The integration of these three aspects has not yet been studied systematically but its application is motivated by industrial nature of dynamic diffusion or transport phenomena where repetitive operations cause a chance effect leading to a stationary uncertainty of system input or initial/ boundary conditions.

For a more complex system where the resulting equations size can be very large, parallel computations can be more efficient. Some early works on parallel and vector solutions for large linear systems were presented in Heller[5], Ortega[6] and Stone[7]. Distributed-memory based parallel algorithms on basic matrix-vector multiplication especially in case of large block-distributed matrices were initially studied in Dekel *et al.*[8]. This research area has been received much attention and several new approaches on different

Corresponding Author: Peerayuth Charnsethikul, Operations Research and Management Science Units, Industrial Engineering Department, Kasetsart University, Bangkok 10903, Thailand

applications based on new software and hardware advancements have been proposed as found in Ben-Asher and Haber[9], Catalyurek and Aykanat[10], Hendrickson *et al.*[11], Ogielski and Aiello[12], Romero and Zapata[13] and Ujaldon *et al.*[14]. Extensively, these basic parallel operations plays a major role on further developments of parallel iterative algorithms for large and sparse linear systems as illustrated in Basserman[15] and Ortigosa *et al.*[16]. For a much larger scale system such as time varying three dimensional partial differential equations as found in the fields of solid mechanics and diffusion processes, domain decompositions[17,18] have been a major support for developments of parallel and distributed computing. Their main idea is to partition the original domain to several smaller domains resulting in a set of much less dependent smaller systems of linear equations that can be handled by each processor in parallel. Then, their solutions are adjusted and integrated with a set of coupling equations under manageable sizes related to common variables among sub-problems. A well illustrated example can be found in Saad[4]. Recently, with the advancements of parallel hardware, parallel and distributed algorithms for the direct Gauss elimination approach have been proposed as other alternatives expected theoretically that their efficiency should be better than the iterative algorithms as presented in Balasubramanya *et al.*[19], Chandra and Siva Ram Murthy[20], Gallivan *et al.*[21] and Tufo and Fischer[22]. A special case of applications on both parallel matrix-vector multiplication and parallel-distributing for solutions of large and sparse linear system is due to solving the steady state discrete and continuous Markov chain model as presented recently in Benzi and Tuma[23].

It is well known that solving an LP using the classical simplex method is an exponentially bounded[24] algorithm but its average performance was derived by Borgwardt[25] resulting the complexity of $O(m)$ iterations with at most $O(m^2)$ operations per iterations where m is the number of constraints in the model. In this study as shown in problem P1, regardless of the bounded variables constraint, $m=n$. Therefore, solving the problem directly is exponentially bounded in the worst case and polynomial bounded as $O(n^4)$ on average. In this work, an alternative algorithm will be proposed. Its main computation complexity deals with computing A^{-1} leading to a better bound of $O(n^3)$. To design basically a parallel scheme for the algorithm, most works reviewed in the previous paragraph can be adapted and applied. An example in case of domain decomposition will be preliminary studied.

In summary, there has been a tremendous amount of works on solving large and sparse linear equations in parallel due to its applications and computer hardware advancements. These works can be used as basic tools to support the purpose of solving problem P1 by the upcoming proposed algorithm. In the next section, a set of parallel approaches will be proposed and verified theoretically and experimentally.

MATERIALS AND METHODS

Let $b'_i = b_i - l_i$. Therefore, $0 \le b'_i \le u_i - l_i$ (u'_i) for all $i = 1, 2, \ldots, n$ and

$$\sum_{j=1}^{n} a_{ij}x_j - b'_i = l_i \quad , i = 1,2,\ldots\ldots,n$$

Since $\det(A) \neq 0$, let $A^{-1} = [a'_{ij}]$, then

$$x_i - \sum_{j=1}^{n} a'_{ij}b'_j = \sum_{j=1}^{n} a'_{ij}l_i \quad , i = 1,2,\ldots\ldots,n$$

and

$$x_k = \sum_{j=1}^{n} a'_{kj}b'_j + \sum_{j=1}^{n} a'_{kj}l_j , \ k = 1,2,\ldots\ldots,n \qquad (1)$$

In order to solve the corresponding LP problem, the following theorem can be established.

Theorem: The solution vector X for the problem (P1) is optimal if and only if (1) is satisfied and:
for each k such that x_k is maximized, for all j, $b'_j = u'_j$ if $a'_{kj} > 0$, otherwise, $b'_j = 0$; for each k such that x_k is minimized, for all j, $b'_j = u'_j$ if $a'_{kj} < 0$, otherwise, $b'_j = 0$;

Proof: By considering equation (1), substitute b'_j by u'_j when $a'_{kl} > 0$ and by 0 otherwise. The result is

$$x_k^* = \sum_{j=1\ (a'_{kj}>0)}^{n} a'_{kj}u'_j + \sum_{j=1}^{n} a'_{kj}l_j \qquad (2)$$

Comparing to equation (1), $x_k^* \ge x_k$ since $b'_j \le u'_j$ for all j. Similarly, in case of the substitutions when $a'_{kj} < 0$, $x_k^* \le x_k$. Thus, the forward parts of the theorem hold.

To prove the backward statement, suppose that the solution obtained from equation (2) is not optimal and there exists another solution for $b'_j = c'_j$ for all j that maximizes x_k. The only alternative that can produce x_k larger than x_k^* is to assign at least one $c'_j > u'_j$. This is infeasible since $b'_j \le u'_j$. Therefore, the reverse condition holds in case of the maximization objective and again, similarly for the minimization case.

Additionally, it is obvious to conclude that the above procedure is a polynomial time algorithm with the complexity of $O(n^3)$ because the computation of A^{-1} is the method bottleneck. In summary, a general method for solving problem P1 is to compute A^{-1} and use substitute b'_j in equation (1) according to the theorem. However, when A is very large and sparse, in case of bandwidth matrix normally found in finite approximation for solving boundary value ordinary and partial differential equation, A^{-1} is fully dense. For an example, a partial differential equation in two dimensions with confidence interval of boundary value is approximated by subdividing 1000 discrete points in each domain leading to the system of linear equations with bandwidth matrix A and one million variables/equations ($n = 1000\times1000 = 1{,}000{,}000$). Solving for all possible variable ranges requires a high performance computing hardware facility with an appropriate parallel algorithm. One obvious direct approach is to distribute 2n independent LP problems of P1 among parallel processors in order to minimize the maximum allocated processing time. Nevertheless, a more efficient method adopted from the theorem can be modified to work in parallel by solving independently for each row elements of A^{-1} as follows.

Procedure P1:

```
Identify k (k=1,2,..,n)
Solve $A^Tz = e_k = [0,0,..,1\ (k^{th}\ position),\ldots,0]^T$ and compute
$u_k = \sum_{j=1\,(z_j>0)}^{n} z_j u'_j + \sum_{j=1}^{n} z_j l_j$
$l_k = \sum_{j=1\,(z_j<0)}^{n} z_j u'_j + \sum_{j=1}^{n} z_j l_j$
End.
```

The above algorithm is designed to break the whole operations into n independent and almost identical jobs. Each job is concerned with solving for a'_{kj}, $j = 1,2,..,n$ representing by the vector z and can be distributed to each available computer and the result of u_k and l_k can be integrated at the end. Therefore, the decision problem is how to allocate n jobs among all available m parallel processors in order to minimize the total makespan. The difference from the original scheduling research is that in this case, the exact processing time of each job is not known in advance but each job has the same instructions and the effect resulting different processing time is the different right hand side and the number of nonzero z_j for all j in each iteration k.

Comparing with the direct approach of distributing 2n LP problems, firstly, the complexity of P1 is polynomial bounded ($O(n^3)$ for the sequential processing and $O(n^3/p)$ for the parallel computations with *p* processors) while the complexity of LP is exponentially bounded in the worst case and $O(n^4)$ in the average case.

Applications in statistics: The proposed procedure can be applied to solve the following linear statistical model. Consider the matrix system of linear equation, AX = b with random vector b representing by expected values vector E[b] and variances/co-variances matrix $K[bb^T]$. In general, obtaining the exact values of E[b] and $K[bb^T]$ may not be possible especially in the case that b is measured from some experiments or randomly generated to intimate the real situation. If a steady state does exist, both statistical parameters can be statistically estimated as confidence intervals (Chapters 7 and 10 of Lindgren[2]). Therefore, the decision problem is to determine the confidence interval of statistical parameters of X ($E[X]$, $K[XX^T]$) for a given confidence interval E[b], $K[bb^T]$.

The problem of determining the confidence interval of E[X] can be solved directly using the method described in the previous section. However, the problem concerning $K[XX^T]$ is not obviously linked. Next, consider the following algebraic analysis.

First, multiply equations ith and kth of an n by n linear system and apply the expectation leading to the equation as follows.

$$\sum_{j=1}^{n}\sum_{l=1}^{n} E[a_{ij}a_{kl}x_jx_l] = E[b_i,b_k] \quad i, k= 1,2,\ldots\ldots,n$$

By the symmetry property of covariance elements, the equation can be adapted as follows.

$$\sum_{j=1}^{n} E[a_{ij}a_{kj}x_j^2]+2\{\sum_{j=1}^{n}\sum_{l=j+1}^{n} E[a_{ij}a_{kl}x_jx_l]\} = E[b_i,b_k] \quad i,=1,2,\ldots,n \quad k=i,i+1,..,n$$

Then, expand the expected value of each term in the equation and obtain the following equation with variance/co-variance as variables.

$$\sum_{j=1}^{n} a_{ij}a_{kj}\mathrm{Var}[x_j]+2\{\sum_{j=1}^{n}\sum_{l=j+1}^{n} a_{ij}a_{kl}\,K[x_j,x_l]\} = K[b_i,b_k] \quad i,=1,2,\ldots\ldots,n \quad k=i,i+1,..,n \qquad (3)$$

Therefore, the above resulting equations system is linear and for a given range or confidence interval of $K[b_i,b_k]$ for all i and k, the proposed method in the previous section can be used to solve for the range of $\mathrm{Var}[x_j]$ and $K[x_j, x_l]$ for all j and l as well. However, in case of large and sparse linear system, n can be very large with the total number of equations/variables of n(n+1)/2. The system of corresponding linear system is so large that a single processor computing system is insufficient for handling this matter. Hence, a parallel algorithm is proposed as follows.

First, the linear system concerning variances and co-variances of X can be written alternatively as the following matrix equation.

$AK[XX^T]A^T = K[bb^T]$

Determining $K[XX^T]$ can be performed by solving the above linear system. A simple and direct approach is to find A^{-1} and substitute the following relation.

$$K[XX^T] = [A^{-1}]K[bb^T][A^{-1}]^T \quad (4)$$

For A as a dense matrix, the above matrix solution can be operated within $O(n^3)$ consisting of one n by n matrix inversion and two n by n matrix multiplications. Algebraically, each element in $K[XX^T]$ can be obtained by the following equation.

$$K[x_i,x_k] = \sum_{j=1}^{n}\sum_{l=1}^{n} a'_{ij}\, a'_{kl}\, K[b_j, b_l]$$
$$i= 1,2,\ldots,n \quad k= i,i+1,\ldots,n \quad (5)$$

Suppose that a given confidence range $K[b_j, b_l]$ is $[l_{ij}, u_{ij}]$ and A is large and sparse, a parallel computing scheme can be created and designed using the relationship (5). The total number of independent computational jobs is $N = n(n+1)/2$ where each job has the general description derived in a similar manner as the proposed procedure for the expected value case as follows.

Procedure P2:

```
Step 0: Identify the desired i and k ($i =1,2,..,n,\ k=i,i+1,i+2,..,n$)
Step 1: Let $y = [y_j]$ and $z = [z_l]$ and solve $A^Ty = e_i$ and $A^Tz = e_k$.
Step 2: Let Max = 0 and Min = 0
    For j = 1:n
        For l = 1,:n
                If $y_jz_l > 0$ Max = Max + $y_jz_l\, u_{jl}$
                            Min = Min + $y_jz_l\, l_{jl}$
                            Otherwise, Max = Max + $y_jz_ll_{jl}$
                                    Min = Min + $y_jz_lu_{jl}$
        End
    End
```

Step 3: $K[x_i,x_k]$ is within the range of [Min, Max]

Suppose that there are m available computers where $m < N$, a number of jobs can be assigned to each machine and the processing time in step 1 can be reduced when either i or k is fixed and another index is varied. For an example, when a computer is assigned a set of jobs with $i = 1$ and vary k from 1 to $p < n$, computing in step 1 is performed merely in the full form in case of i=1 and k=1. For another $k > 1$, the processing time in this step can be reduced in half since only solving $Az = e_k$ is needed.

An application in domain decompositions: In this section, both methods developed in the previous sections are applied to the case where A is formulated by the principle of domain decomposition. In general, the structure of the corresponding equations system is as follows.

$$B_i X_i + F_i Y = c_i \quad i=1,2,\ldots,s \quad (6)$$

$$\sum_{i=1}^{s} E_i X_i + CY = d \quad (7)$$

$$p_i \le c_i \le q_i\, ,\ r \le d \le t \quad (8)$$

where s is the number of sub-domain, B_i represents coefficients square matrix of sub-domain i with X_i , c_i , p_i and q_i as the associated variable, RHS, lower and upper bound vectors, respectively and E_i and F_i represent coefficients matrices of interaction between sub-domain i and the common domain. The matrix C represents the coefficients square matrix of the common domain with Y, d, r and t as the associated variable, RHS, lower and upper bound vectors, respectively. To solve for the ranges of mean vectors X_i, for all i and Y, the following model can be derived based on the similar principle of (1) in a matrix notation.

$$X_i = \sum_{j=1}^{s} A'_{ij}c_i + A'_{i0}d \quad , i =1,2,\ldots,s \quad (9)$$

$$Y = \sum_{j=1}^{s} A'_{0j}c_i + A'_{00}d \quad , i =1,2,\ldots,s \quad (10)$$

where,

$A'_{ij} = B_i^{-1}F_iG^{-1}E_jB_j^{-1}$, i,j = 1,2,…,s, $i \neq j$

$A'_{ii} = B_i^{-1} - B_i^{-1}F_iG^{-1}E_iB_i^{-1}$, i =1,2,..s

$$G = C - \sum_{i=1}^{s} E_iB_i^{-1}F_i$$

$A'_{i0} = -B_i^{-1}F_iG^{-1}$, i =1,2,..s

$A'_{0j} = -G^{-1}E_jB_j^{-1}$, j =1,2,..s

$A'_{00} = G^{-1}$

For confidence intervals of variable co-variance sub-matrix according to each domain interaction i and j, procedure P2 can be adapted in the following matrix form.

$$K_{ij} = \sum_{k=0}^{s}\sum_{l=0}^{s} A'_{ik}A'_{jl}B_{kl}\, , i,j = 0,1,2,..,s \quad (11)$$

where K_{ij} represents sub-matrix of co-variances between X_i and X_j when i,j = 1,2,..,s and between $X_{i(j)}$ and Y when j(i) = 0 and B_{kl} represents sub-matrix of co-variances between c_i and c_j when i,j = 1,2,..,s and between $c_{i(j)}$ and d when j(i) = 0. To verify the correctness of equations (6) – (11), consider AX = b formed by equations (6) and (7) as follows

$$\begin{pmatrix} B_1 & & & & & F_1 \\ & B_2 & & & & F_2 \\ & & B_3 & & & F_3 \\ & & & \cdots & & \cdots \\ & & & & \cdots & \cdots \\ & & & .. & B_s & F_s \\ E_1 & E_2 & E_3 & \ldots & E_s & C \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \cdots \\ \cdots \\ X_s \\ Y \end{pmatrix} \begin{matrix} = \\ = \\ = \\ \cdots \\ \\ = \\ = \end{matrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \cdots \\ \cdots \\ c_s \\ d \end{pmatrix}$$

Suppose that A^{-1} is in the following form.

$$\begin{pmatrix} A'_{11} & A'_{12} & A'_{13} & \ldots & A'_{1s} & A'_{10} \\ A'_{21} & A'_{22} & A'_{23} & \ldots & A'_{2s} & A'_{20} \\ A'_{31} & A'_{32} & A'_{33} & \ldots & A'_{3s} & A'_{30} \\ \ldots & & \cdots & & \cdots & \\ & & & \cdots & \cdots & \\ A'_{s1} & A'_{s2} & A'_{s3} & \ldots & A'_{ss} & A'_{s0} \\ A'_{01} & A'_{02} & A'_{03} & \ldots & A'_{0s} & A'_{00} \end{pmatrix}$$

By using the definition of A'_{ij}, it can be proven that the matrix property of AA^{-1} holds. Thus, A^{-1} is valid and can be used in both equations (9) and (10) as claimed. Also, as seen in the block structure described in equations (9), (10) and (11), firstly, the matrix G^{-1} must be computed and then, parallel and distributed computing can be independently allocated to each processor for determining X_i, Y and K_{ij} for all i and j.

Computational experience: In this study, the heat transfer equation ($\partial^2 Ø/\partial x^2 + \partial^2 Ø/\partial y^2 = f(x, y)$) approximated by 5 Points finite difference approach is used with randomly generated ranges of f(x, y) represented by l(x, y) and u(x, y), x = 0, Δ, 2Δ, …., $N_1Δ$ and y = 0, Δ, 2Δ, …., $N_2Δ$. Δ is varied under the constraint that $N_1Δ = N_2Δ = 1$. Excluding boundary elements, the total number of variables (n) for the corresponding set of linear equations is $(N_1- 1)(N_2 – 1)$ in case of the expected value. The proposed method was coded in Matlab[26] utilizing its special commands for sparse linear equations solver. For each problem size n, a sample of variables range containing 10 observations is obtained and its average is computed and summarized in the following table.

The Proposed Method

No. of Variables (n)	Average Time/Variable (sec.)
100,000	64
200,000	452
300,000	1,178
400,000	2,163
500,000	2,940
600,000	5,134
700,000	6,987
800,000	9,341
900,000	11,596
1,000,000	14,246

Simultaneously, the same set of problems is solved using the direct LP command, "Linprog[.]", of software Matlab[26]. The required computation times are reported in the following table.

The direct LP method

No. of Variables (n)	Average Time/Variable (sec.)
100,000	3,244
200,000	18,140

The results from both tables indicate that the proposed method is more efficient as the problem size grows. All computations are performed on a microcomputer system with CPU speed of 2.4 Ghz and 1 Gbytes RAM. Moreover, the proposed method is extended to determine a sample of covariance elements under various n and their average computing times are analyzed using 10 observations for each problem sizes and can be shown as follows.

The Proposed Method

No. of Variables (n)	No. of Covariance Elements	Average Time/Element (sec.)
5,000	12,501,250	961
6,000	18,003,000	1,366
7,000	24,503,500	1,836
8,000	32,004,000	3,127

Since all test cases in the above table require full dense variance/co-variance matrices, computations of each variable variance/co-variance range consume up to *$O(n^2)$* limiting the much smaller upper bound of n as compared to the case of expected value with *O(n)*. The size of n > 8000 is prohibited due to available memory. Nevertheless, in practice, it is most likely to find that the resulting variance/co-variance matrix can be approximated as a sparse matrix. In this circumstance, a set of such large and sparse $K[bb^T]$ is randomly generated as banded matrices with k as the number of nonzero elements in each row. Then, ten samples of each n ranging from 10,000 to 500,000 are tested utilizing MATLAB powerful commands on sparse matrix operations[26] based on Procedure P2. The experimental results are as shown below.

No. of Variables (n)	k	Average Time/Element (sec.)
10,000	7	137
10,000	11	191
10,000	21	374
25,000	7	442
25,000	11	579
25,000	21	1,211
50,000	7	1,588
50,000	11	1,965
50,000	21	3,004
100,000	7	3,017
100,000	11	3,981
100,000	21	5,177
500,000	5	8,287

The result in the above table indicates some possibilities to conduct parallel computations in case of large and sparse variance/co-variance matrices. For a much larger scale problem, matrix partitioning and interpolations should be further developed. It should be noted that all computations have been conducted based on the assumption that solving the corresponding linear systems can be performed on a single processor and the proposed approach is designed to partition the whole problem to several independent linear systems capable of distributing one by one among parallel processor. However, when the matrix A becomes too large to be handled by a single processor, a set of clusters among processors has to be designed. Each cluster consists of a set of designed processors capable of solving each related system. To support this concept in case of Matlab, parallel programming is required and their basic study on matrix operations was experimented in[27].

Next, a preliminary study on the use of the direct approach P1 compared to using equations (9)-(10) when the problem is in the form of equations (6)-(8) is conducted as follows. First, the heat equation with L-shape boundary is decomposed to 200 domains with 2500 points (variables) in each domain and 5,000 common domain points with generated data of right hand side lower and upper limits. Then, the resulting equations in the non-decomposition mode consists of a sparse 500,000 variables system while the domain decomposition system is involved with sparse 2,500×2,500 B_i, 2,500×5,000 F_i, 5,000×2,500 E_i ($i = 1, 2,.., 200$) and 5,000×5,000 C. The experiment has been conducted using 4 identical processors as specified previously, working in parallel with the additional main server (2.4 GHz and 2 Gbytes RAM) working as jobs manager. For the direct approach, the average time out of ten samples for computing a variable interval is 2,917 sec. which is slightly less than the previous experiment case of 500,000 variables (2940 sec.). With five parallel processors, the expected computing time per element can be reduced by one-fifth resulting almost ten minutes (2917/5). To solve the problem using equations (9) for a specific i, G^{-1} is computed in the main server firstly. Then, computing A'_{ij}, $j = 0,1,2,..,s$ (=200 in this case) is distributed equally to each processor (=50) and A'_{0j} is computed in parallel by the main server. After that, the sub-vector interval for X_i is computed using equation (9) and algorithm P1 using the main server. At the end, 2,500 intervals are obtained. In this study, i =1 is conducted as an example resulting the total computation time of 57 hrs. 21 min. and 6 sec. Therefore, the average time for computing a variable interval is reduced to merely one and a half minute per element. This basic study initially illustrates a further direction for continuous improvement on implementing algorithm P1 using parallel processing. For algorithm P2, similar scheme can be conducted by using equations (11). It should be noted that the parallel strategy used in this case has not been fully developed and can be improved. For examples, computing G^{-1} and equation (9) with i=1 have not been parallelized yet. Further explorations should be investigated.

CONCLUSION

In this study, parallel approaches for determining confidence intervals of variables in the large and sparse linear equation system with RHS ranges is proposed and preliminary tested its application possibilities. The basic concept has been verified and validated mathematically. Comparisons with the direct linear programming approach are conducted in case of the mean confidence interval. The test results illustrate that the approach greatly improves outcome efficiency. Moreover, the proposed approach can be adapted to work in parallel using domain decomposition. An example is preliminary studied and the initial result indicates a further improvement.

REFERENCES

1. Dantzig, G.B., 1963. Linear Programming and Extensions. Princeton University Press.
2. Lindgren, B.W., 1976. Statistical Theory. 3rd Edn., Macmillan Publishing Co., Inc.
3. Aykanat, C., F. Ozguner, F. Ercal and P. Sadayappan, 1988. Iterative algorithms for solution of large sparse systems of linear equations on hypercubes. IEEE Trans. Comput., 37: 1554-1567.
5. Heller, D., 1978. A survey of parallel algorithms in numerical linear algebra. SIAM Rev., 20: 740-777.
6. Ortega, J., 1988. Introduction to Parallel and Vector Solution of Linear System. Plenum Press.
4. Saad, Y., 1996. Iterative Methods for Sparse Linear Systems. PWS Publishing Co.
7. Stone, H.S., 1973. An efficient parallel algorithm for the solution of a tridiagonal linear system of equations. J. ACM, 20: 27-38.
8. Dekel, E., D. Nassimi and S. Sahni, 1981. Parallel matrix and graph algorithms. SIAM J. Comput., 10: 657-675.
9. Ben-Asher, Y. and G. Haber, 2004. Efficient parallel solutions of linear algebraic circuits. J. Parallel Distrib. Comput., 64: 163-172.

10. Catalyurek, U.V. and C. Aykanat, 1999. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. IEEE Trans. Parallel Distributed System, 10: 673-693.
11. Hendrickson, B., R. Leland and S. Plimpton, 1995. An efficient parallel algorithm for matrix-vector multiplication. Internat. J. High Speed Comput., 7: 73-88.
12. Ogielski, A.T. and W. Aiello, 1993. Sparse matrix computation on parallel processor arrays, SIAM J. Scient. Comput., 14: 519-530.
13. Romero, L.F. and E.L. Zapata, 1995. Data distribution for sparse matrix vector multiplication. J. Parallel Comput., 21: 583-605.
14. Ujaldon, M.U., E.L. Zapata, S.D. Sharma and J. Saltz, 1996. Parallelization techniques for sparse matrix applications. J. Parallel Distributed Comput., 38: 256-266.
15. Basserman, A., 1997. Parallel sparse matrix computations in iterative solvers on distributed memory machines. J. Parallel Distrib. Comput., 45: 46-52.
16. Ortigosa, E.M., L.F. Romero and J.I. Ramos, Parallel scheduling of the PCG method for banded matrices rising from FDM/FEM. J. Parallel Distrib. Comput., 63: 1243-1256.
17. Farhat, C. and P.S. Chen, 1994. Tailoring domain decomposition methods for efficient parallel coarse grid solution and for systems with many right hand sides. Contem. Math., 180: 401-406.
18. Smith, B., P. Bjorstad and W. Gropp, 1996. Domain Decomposition, Cambridge Univ. Press Cambridge, MA.
19. Balasubramanya, K.N.B., Murthy, C. and Siva Ram Murthy, 1996. Gaussian elimination based algorithm for solving linear equations on mesh connected processors. IEE Proc. Comp. Digit. Tech., 143: 407-412.
20. Chandra, R., C. Siva Ram Murthy, 2003. A faster algorithm for solving linear algebraic equations on the star graph. J. Parallel Distrib. Comput., 63: 465-480.
21. Gallivan, K., R.J. Plemmons and A.H. Sameh, 1990. Parallel algorithms for dense linear algebra computations. SIAM Rev., 32: 54-135.
22. Tufo, H.M. and P.F. Fischer, 2001. Fast parallel direct solvers for coarse grid problems. J. Parallel Distributed Comput., 61: 151-177.
23. Benzi, M. and M. Tuma, 2002. A parallel solver for large-scale Markov chain. Appl. Numer. Math., 41: 135-153.
24. Klee, V. and G. Minty, 1972. How good is the Simplex-Algorithm?, Inequalities III, O. Shisha (Ed.), Academic Press, New York, pp: 159-175.
25. Borgwardt, K.H., 1986. The Simplex Method: A Probabilistic Analysis. Springer-Verlag.
26. The Math Works, Inc. MATLAB manual and instruction sets, 2000.
27. Milosavljevic, I.Z. and M.A. Jabri, 2001. Experimental evaluation of automatic array alignment in parallelized Matlab, J. Parallel Distributed Comput., 61: 784-809.