Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2022 doi:10.20944/preprints202203.0036.v1

Vedic Mathematics Approach to Speedup Parallel Computations

Urmila Shrawankar and Chaitreya Shrawankar
RTM Nagpur University, Nagpur (MS), India
urmila@ieee.org

***Abstract*—** Solving Linear equations with large number of variable contains many computations to be performed either iteratively or recursively. Thus it consumes more time when implemented in a sequential manner. There are many ways to solve the linear equations such as Gaussian elimination, Cholesky factorization, LU factorization, QR factorization. But even these methods when implemented on a sequential platform yield slower results as compared to a parallel platform where the time consumption is reduced considerably due to concurrent execution of instructions. The above mentioned linear equation solving methods can be implemented on the parallel platform using the direct approaches such as pipelining or 1D and 2D Partitioning approach. Vedic mathematics is a very ancient approach for solving mathematical problems. These Vedic mathematical approaches are well known for quicker and faster computation of mathematical problems. Vedic Mathematics provides a very different outlook towards the approach of solving linear Equation on parallel platform. It could be considered as a better approach for reducing space consumption and minimizing the number of algebraic operations involved in solving linear equation. In future Vedic Mathematics might serve as a viable solution for solving linear equation on parallel platform.

Keywords : *Vedic mathematics; parallel computation; parallelism; Multicore Systems; pipelining; 1D and 2D partitioning; linear equations solving; Paravartya Yojayet method; Sunyam Anyat method; Sankalana Vyavakalanabhyam method; Sopantyadvayamantyam method*

1. INTRODUCTION

Mathematics is the mother of all sciences as it holds the solution to all the problems that are faced by the human race. But as the technology advances the complexity of the problems also increases and it is desirable that a faster and more accurate solution is obtained. Linear Equations form an integral part of the problem solving techniques [1],[2],[3],[4],[5] provided by mathematics which helps predict the unknown variables. But as the size of the problem accents the time consumed to compute the problems also increases. The uniprocessor system consumes a lot of time for computing the solution to linear equations with a large variable size increasing the energy consumption. Parallel computation of the problem on a multicore system however should faster results, as parallel computing is considered popular way of achieving high energy efficiency. In multi-core systems energy efficiency is a question of both the time and space, sharing of resources, and is highly dependent on the application characteristics such as its level of parallelism. Thus efficiency can be achieved by parallel applications mapping the program on many cores and the clock frequency can be lowered. But this not the case when implemented on actual parallel machine as there are many factors that hamper the efficiency of the system by causing overheads [6] such as-

- Inter-process Interaction
- Idling
- Excess Computation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2022 doi:10.20944/preprints202203.0036.v1

There are many ways to tackle this problem but main propaganda of this work is to survey and form a comparative analysis of all the existing techniques to solve linear equation on parallel platform [7]. It can be also noted during the further analysis of the previous works that after certain level the ordinary mathematics laws fail to give the desired faster and accurate results. Hence the need arises for the use of Vedic mathematical principles that satisfy void left by conventional mathematics. Vedic mathematics is acclaimed for yielding accurate results in minimum time [8], [9], [10]. These mathematical principles involve the use of simple arithmetic operations to compute a very large problem in most efficient manner.

2. MATHEMATICAL METHODS FOR SOLVING LINEAR EQUATIONS

2.1. Survey of methods for solving linear equations: will enlist all the methods that are available for solving linear equations irrespective of the platform [1],[2],[3],[4],[5] .

There are many methods available in conventional mathematics for solving linear equations. These methods mainly involve the use of matrices for computational purpose [11],[12],[13],[14],[15],[16]. The basic considerations for solving linear equations enlisted under this approach follow the following course of action-

Step 1: The system of linear equations

$$
\begin{aligned}
a_{0,0}x_0 + a_{0,1}x_1 + \ldots + a_{0,n-1}x_{n-1} &= b_0,\\
a_{1,0}x_0 + a_{1,1}x_1 + \ldots + a_{1,n-1}x_{n-1} &= b_1,\\
\vdots \quad\quad \vdots \quad\quad \vdots \quad\quad &\vdots\\
a_{n-1,0}x_0 + a_{n,1}x_1 + \ldots + a_{n-1,n-1}x_{n-1} &= b_{n-1}
\end{aligned}
$$

This can be represented in the form of Ax = B.
Where A is matrix containing the values (a i, j, x j), x is the variable vector
B is the vector containing independent terms.

Step 2: Convert the matrices in the upper triangular or lower triangular form or both as per the requirement of the method.

Step 3: The back substitution- The previously triangulated matrices are solved in reverse way to obtain the values of the variables.

The most appropriate methods for programming are noted below,

i. *Simple Gaussian Method* :
This method converts the Ax = B in the upper triangular matrix and solves the Ux = y and the back substitution to get the values of x.

ii. *Gaussian elimination with Partial Pivoting Method* :
The above Simple Gaussian method fails if the A [k, k] value is close or equal to zero. This flaw of the previous method is easily removed the use of partial pivoting in which a column is selected as a pivot and accordingly the adjustments are done to the matrices.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2022 doi:10.20944/preprints202203.0036.v1

iii. *LU Decomposition :*

Similar to the Gaussian elimination is the LU factorization technique [17], [18], [19], [20] that considers that

A = L * U where L and U are the Lower and Upper Triangular matrices respectively.

iv. *Cholesky Factorization :*

This method assumes that matrix A is real, symmetric and positive and is in the $A = L L^T$, here L is a lower triangular matrix with positive diagonal elements.

v. *QR Factorization* :

Here it can be assumed that A is real m x n matrix which can be decomposed as

A = QR,

Where Q is the m x m real orthogonal matrix

R is a real upper triangular matrix

The factorization here is performed using the Householders matrices of the form-

$H = I - \tau\upsilon\upsilon^T$,

Where τ is the scaling factor and υ is the column reflector.

2. 2. Survey of methods for parallel platform: will mention the methods in which the equations can be solved on parallel platform [21], [22], [23].

The solving of linear algebraic equations can be broadly categorized in two main divisions viz.

i. *Iterative method:* These methods are quoted to produce approximate results by repeatedly computing the same steps or iterations.

ii. *Direct method:* This method provides the exact solution based on finite number of arithmetic computations.

Based on the analysis of the above methods, by it can be roughly predicted that '*Direct method are best suited for Full or Dense matrices and iterative methods are best suited for very large or sparse matrices*'.

2.2.1 1D and 2D partitioning Approach

i. 1D Partitioning Approach : This approach divides the data that is required for computation. This data is considered as column or a row-wise vector of the matrix. For example, consider the matrix A that is given below

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

P_0

P_1

Here P0 is a row wise 1D partition whereas P1 is the column wise 1D partition of the matrix A.

ii. 2D Partitioning Approach – in the 2D partitioning approach the complete matrix is divide in an pxg matrix where the p,q are less than n,m.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2022 doi:10.20944/preprints202203.0036.v1

$$A = \overset{P^3}{\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}}$$

Here the P_3 partition can be considered as a 2D partition. Claims that the 2D partitioning is faster and more efficient than 1D partitioning approach.
The efficiency of the algorithms also differs based on the granularity of the decomposition of the problem. These results vary from problem to problem. This can be better analysed from the next section which gives the analysis of the complexities of the various linear equation solving methods based on the 1D and 2D partitioning approach.

2.2.2 Pipelining model - In this approach the processes that not having any type of dependencies (such as data dependence or control dependence) are divided in finite number of threads which each thread [21] then behaves as a producer and a consumer. The threads then consumes the input data that is provided to the model and produces the required or desired output data for the given process [27], [28]. This is the most direct approach for parallel implementation of algorithms [22], [23], [24] but there are various problems that need to be dealt with while utilizing this approach.

2.3. Analysis of performance based on Partitioning- the effect of matrix dimensions on the complexities of the Linear Equation solving methods.

The performance of the parallelizing algorithms [25], [26] are highly affected by the partitioning that is done for the problem to be solved. The finer the granularity of the problem the faster are the results is the general assumption, but when actual computation were performed on a multi core system the results seemed to vary.
Consider the Simple Gaussian Elimination, the complexity of the problem when approached sequentially is $O(n^3)$. The same problem when solved in a parallel way using the 1D partitioning algorithm yields the complexity of $\Theta(n^3 \log n)$ which is asymptotically higher than the sequential complexity of the problem thereby proving that the 1D partitioning is slower than sequential approach. The 2D partitioning with less than n^2 elements gives the complexity of $\Theta(n^3/p)$ where p is the number of processes, implies that 2D partitioning of data will give better results for the parallel implementation.

3. THE VEDIC MATHEMATICS APPROACHS

i. The analysis of *Paravartya Yojayet* method for solving linear equations
ii. The analysis of *Sunyam Anyat* method
iii. The analysis of *Sankalana Vyavakalanabhyam* method
iv. The analysis of *Sopantyadvayamantyam* method

Vedic mathematics is an ancient most mathematical catalogue of various approaches [8], [9], [10] to solve any numerical computation. Vedic mathematics is said to be descended from the *Atharva* Veda's *Upaveda* (sub section) named *Sthapathyaveda* that contains all the records of the engineering technology and computational strategies. The edge that Vedic mathematics has

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2022 doi:10.20944/preprints202203.0036.v1

over the conventional mathematical approaches is that these methods are way simpler, faster and accurate.

Vedic mathematics is nothing but a compilation of simpler arithmetic calculations to perform large and cumbersome problems in a very small amount of time. The linear equation solving in Vedic mathematics can be approached based on the various *sutras* that are mention in Vedas. Some of these approaches are enlisted as follows

3.1 The *Paravartya Yojayet* [8] method for solving linear equations

The normal system of linear equations of the form

$a_1x + b_1y = c_1$, $a_2x + b_2y = c_2$,

can be solved by using the cross multiplication rule named as *Paravartya Yojayet.*

This rule is divided in two main steps

i. *Step 1 :* Finding the denominator that is common for all the terms. This is obtained by the difference of the cross multiplications of the coefficient terms.
ii. *Step 2 :* Finding the individual numerators for each variable individually. The value of x can be obtained by forward cross multiplication of y coefficients to the independent terms.

3.2 The *Sunyam Anyat* [8] method:

This method is mainly used when there are very large values of the coefficient that seems to be difficult to compute. But using the *Sunyam Anyat* we can easily solve even these difficult problems. This sutra basically means that if one pair of coefficients is in a ration then the other pair equates to zero.

3.3 The *Sankalana Vyavakalanabhyam* [8] method

This the typical method that is used in the conventional mathematics for the computation of linear equation by the addition or subtraction of the two equations to obtain the values of the unknown variables.

3.4 The *Sopantyadvayamantyam* [8] method

This method is used to solve equations that are in the format of

$$\frac{1}{AB} + \frac{1}{AC} = \frac{1}{AD} + \frac{1}{BC}$$

and the factors (A, B, C and D) are in Arithmetic Progression.

This method gives the values of the variables just by the summation of the last and twice of the penultimate value of the progression.

These approaches are specifically used to simplify the calculation of the linear equations for normal person. But not all these methods can be used for parallel computation of linear equation.

The Paravartya Yojayet is the best fit for dealing with linear equations on a parallel platform. Based on the article [9] by DKR Babajee this principle is very lucidly explained for multiple variable values. The Paravartya Yojayet that was generalized in [9] is as follows

For two linear equations say

$a1,1x1 + a1,2x2 = b1$ ……. (a)

$a2,1x1 + a2,2x2 = b2$ ……. (b)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2022 doi:10.20944/preprints202203.0036.v1

Step 1 : Separate the coefficients of *x1* and *x2*

x1	x2	Independent terms
a1,1	a1,2	b1
a2,1	a2,2	b2

Step 2 : Find the numerator for *x1*

x1	x2	Independent terms
a1,1	a1,2	b1
a2,1	a2,2	b2

Nx1 = a1,2 b2 - b1 a2,2

Step 3 : Find the common denominator D

x1	x2	Independent terms
a1,1	a1,2	b1
a2,1	a2,2	b2

D = a1,2 a2,1 - a1,1 a2,2

Step 4: The numerator for x2

x1	x2	Independent terms
a1,1	a1,2	b1
a2,1	a2,2	b2

Nx2 = a2,1 b1 – b2 a1,1

And finally x1 and x2 will be

$$x1 = \frac{Nx1}{D} \qquad x2 = \frac{Nx2}{D}$$

Similarly for the equation that has *n* number of variables that are greater than 3 the two equations at a time are considered and their variables are equated.

This performed for the rest of the equations also. Once this is done the Paravartya Yojayet is applied n/2 times thereby yielding the values of the unknowns. Here it can be noted that there is no dependency in the process of equating the variables so that can be done concurrently on a multicore system thereby reducing the time consumed also the values to be found Paravartya Yojayet are also independent so they can be concurrently.

The two main stages of equating the variables and the application of Paravartya Yojayet are having dependence and so the parallel implementation needs to be done in two stages.

4. IMPLIMENTATION DETAILS

Linear equation solving in Vedic mathematics has an alternative faster method for computation. This method is called the “Paravartya Yojan”. In this method the linear equations are solved by converting the complete equations in terms of one variable. This has been grasped in greater details by using the following generalization

As the method says that if the value of n is odd the procedure [9] of Paravartya Yojan needs to be applied (n-1)/2 times and if the number of equations is even i.e. if the value of n is even the procedure needs to be applied n/2 times.

 doi:10.20944/preprints202203.0036.v1

5. RESULTS AND DISCUSSION

This Paravartya Yojan is applied to the normal sequential format it has the time complexity of $O(n^2)$ which is similar to the previous methods such as the Cholesky, LU and QR Factorization. But the space complexity here is less than that of these methods as there is no use of matrices done here. Also the cache consumption is less. There is least fetching of data from the memory so, a considerable reduction in the latency caused due to the memory fetch operations.

From the sequential implementation it has been seen that the Paravartya Yojan needs to be applied n/2 times. But when the same is applied to a system of parallel threads consider only two here there is a considerable decrease in the number of time the rule needs to be applied there by reducing the complexity. Also there is need to wait for the constants that are derived from the previous equations. So there is a need to apply "critical" clause of OpenMP for synchronization. Using the critical clause will help get more accurate results for the same.

When the system of equations is divided among the threads there is a round robin distribution of the equations. There is no complexity or the need to keep track as there is no constraint on what equations to be used and should they be consecutive. As in case of the previous methods the sequence was of utmost importance because of the matrices being used. The computations required for the equations to be rearranged in terms of a variable are the most complex part and if done in sequential manner it consumes more time. This is where the need to parallelize the steps the dependency lies only when there is a need to insert the newly calculated coefficients to the system that not be done in the local thread, thus there is a need to follow up this value to the global thread.

Due to the reduction in the space consumption even in the parallel environment the delay caused by latency is very negligible for the memory fetch. The local values required for each thread is also decreased thereby reducing the space complexity of the whole system.

5. CONCLUSION

This paper enlightens about parallel computing or the use of multiprocessor system helps enhance the efficiency to solve any kind of problem. Given that there is need for efficiently design the algorithms and that the overhead that occurs at the start-up is duly compensated for by the amount of calculations that is needed to be done in order to neglect the overhead.
The Vedic mathematical calculations involve even lesser computations than the conventional LU, QR, Cholesky methods the method has been tested for the parallel implementation. The various Vedic mathematical approaches opens the avenues to have newer look towards solving linear equations. In future the use of Vedic Mathematics might serve a better purpose with respect to minimizing the cost required as it involves minimal calculations and with more number of variables only a few algebraic calculation might proof sufficient to solve linear equations.

REFERENCES

[1] S. Mou, J. Liu and A. S. Morse, A Distributed Algorithm for Solving a Linear Algebraic Equation, IEEE Transaction on Automatic Control, 2013. ISDN: 978-1-4799-3410-2/13/2013

[2] Chein-Shan Liu, An algorithm with m-step residual history for solving linear equations: Data interpolation by a multi-shape-factors RBF, Journal of Engineering Analysis with Boundary Elements 51, Elseiver, 123–135, 2015.

[3] M.H.B.M. Shariff, D.J. Grey, Parallel direct method for linear equations, Journal on Advances in Engineering Software 30, Elsevier, 839–845, 2010.

[4] Syed Atiqur Rahman and Mohd. Samar Ansari, A neural circuit with transcendental energy function for solving system of linear equations, Journal on Analog Integrated Circuit Signal Process 66, Elsevier,433440, 2011.

[5] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean-Luc Starck, Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit, IEEE Transactions On Information Theory, vol. 58, NO. 2, 1094-1121 FEBRUARY 2012

[6] Urmila Shrawankar, Mayuri Joshi, Overhead Management in Multi-Core Environment , arXiv preprint, arXiv: 2202.06744, 2022 - arxiv.org

[7] Urmila Shrawankar, Analysis of Factors of Parallelism and their Impact on Dense Algebra Problems, preprint, Research Square, DOI: 10.21203/rs.3.rs-1391459/v1, 2022,

[8] Krsna Tirathji Maharaja Jagadguru Swami Sri Bharati. Vedic Mathematics. Motilal Banarsidas Publishers Private Limited, Delhi, 13 edition, 2011.

[9] DKR Babajee, Solving System of Linear Equation using Paravatrya Yojana rule of Vedic mathematics, 2012.

[10] Urmila Shrawankar, Krutika Sapkal, Complexity analysis of Vedic mathematics algorithms for multicore environment, International Journal of Rough Sets and Data Analysis (IJRSDA), IGI Global, 4(4), 2017, Pages 31-47

[11] Tarek Nechma and Mark Zwolinski, Parallel Sparse Matrix Solution for Circuit Simulation on FPGAs, IEEE transactions on computers, VOL. 64, NO. 4, 1090-1103, APRIL 2015.

[12] Hatem Ltaief, Jakub Kurzak, and Jack Dongarra, Parallel Two-Sided Matrix Reduction to Band Bidiagonal Form on Multicore Architectures, IEEE Transactions on Parallel and Distributed Systems, VOL. 21, NO. 4,417-423, APRIL 2010

[13] Robert Armistead and Fangxing Li, Parallel Computing of Sparse Linear Systems using Matrix Condensation Algorithm, Conference on PowerTech, IEEE Trondheim , 1-6 ,2011. ISDN: 978-1-4244-8417-1/11/

[14] R. Oguz Selvitopi, Muhammet Mustafa Ozdal, and Cevdet Aykanat, A Novel Method for Scaling Iterative Solvers: Avoiding Latency Overhead of Parallel Sparse-Matrix Vector Multiplies, IEEE Transactions on parallel and distributed systems, VOL. 26, NO. 3, 632-645, MARCH 2015.

[15] Keqin Li, Fast and highly scalable parallel computations for fundamental matrix problems on distributed memory systems, Journal on Super computers 54:, Springer, 271–297, 2010.

[16] Florian Ries, Tommaso De Marco, and Roberto Guerrieri, Triangular Matrix Inversion on Heterogeneous Multicore Systems, IEEE transactions on parallel and distributed systems, VOL. 23, NO. 1,177184, JANUARY 2012.

[17] Guiming Wu, Yong Dou, Junqing Sun, and Gregory D. Peterson, A High Performance and Memory Efficient LU Decomposer on FPGAs, IEEE transactions on computers, VOL. 61, NO. 3,366-378, MARCH 2012.

[18] Heinecke and M. Bader, “Towards Many-Core Implementation of LU Decomposition Using Peano Curves,” Proc. Combined Workshops UnConventional High Performance Computing Workshop Plus Memory Access Workshop, pp. 21-30, 2009.

[19] Venetis and G. Gao, “Mapping the LU Decomposition on a Many-Core Architecture: Challenges and Solutions,” Proc. Sixth ACM Conference on Computing Frontiers, pp. 71-80, 2009.

[20] D. Maurer and C. Wieners, “A Parallel Block LU Decomposition Method for Distributed Finite Element Matrices,” IEEE Transaction on Parallel Computing, vol. 37, pp. 742-758, 2011.

[21] Michael M. Wolf, Michael A. Heroux, and Erik G. Boman, Factors Impacting Performance of Multithreaded Sparse Triangular Solve, J.M.L.M. Palma et al. (Eds.): VECPAR 2010, LNCS 6449, pp. 32–44, 2011.

[22] Ananth Grarma Ashul Gupta et al Introduction to Parallel Computing Second edition,Pearson, 2013.

[23] Scott Schneider, Martin Hirzel, Bug˘ ra Gedik, and Kun-Lung Wu, Safe Data Parallelism for General Streaming, IEEE transactions on computers, VOL. 64, NO. 2, 504-517, FEBRUARY 2015.

[24] Murat Manguoglu, A domain-decomposing parallel sparse linear system solver, Journal of Computational and Applied Mathematics 236, 319– 325, Elsevier, 2011.

[25] Ashish Raman, Anvesh Kumar and R.K.Sarin , High Speed Reconfigurable FFT Design by Vedic Mathematics, Journal of computer science and engineering, Volume 1,IEEE, issue 1 May 2010

[26] Girish Talmale, Urmila Shrawankar, Comparative Analysis of Different Techniques of Real Time Scheduling for Multi-Core Platform, arXiv preprint, arXiv:2112.13841, 2021 - arxiv.org

[27] Panagiotis D. Michailidis and Konstantinos G. Margaritis, Parallel direct methods for solving the system of linear equations with pipelining on a multicore using OpenMP, Journal of Computational and Applied Mathematics, Elsevier, 236, 326–341, 2011.

[28] Panagiotis D. Michailidis and Konstantinos G. Margaritis, Implementing Parallel LU Factorization with Pipelining on a MultiCore using OpenMP, 13th IEEE International Conference on Computational Science and Engineering, Pg. no253-260, 2010.